博士論文 （要約）

Computational and chemical barriers for counting-dependent characterization of biomolecular networks

(計数に基づく生体分子ネットワークの特性評価における計算論および化学論的困難)

バリッシュ ロバート ダニエル
Barish Robert Daniel

Dedicated to William Leo Barish

Contents

Abstract

In this dissertation we concern ourselves with the "hardness", in the sense of both computational complexity theory as well as biochemistry or chemistry formal, of a few difficult counting problems that have direct relevance to the characterization of biochemical systems, and in particular, metabolic networks. Along the way we also prove a number of theories that may be of independent interest for computer scientists and graph theorists.

In **(Chapter 2)** we consider a number of complexity theoretic question broadly concerning the hardness of counting and approximating directed as well as undirected Hamiltonian cycles, Hamiltonian paths, and general cycles in graphs with a range of severe vertex degree, partiteness, vertex connectivity, and planarity constraints. Here, we are motivated by the direct correspondence of these complexity results to the problem of counting and approximately counting simple cycles in "reaction-centric graphs" of chemical or biochemical networks, where vertices correspond to "reaction centers" (e.g. proteins) and where directed (resp. undirected) edges correspond to irreversible (resp. reversible) flows of metabolites between these reaction centers.

In **(Chapter 3)** we extend the results of the previous chapter to show that counting undirected simple cycles on ($k \geq 3$)-regular (bipartite or planar) graphs, which in the context of the previous chapter can be interpreted as substrate cycles embedded on a ($k \geq 3$)-regular reaction-centric graph at the limit where all reactions are reversible, remains $\#P$-complete $\{\forall(k \geq 3) \in \mathbb{N}\}$-regular (bipartite or planar) graphs. We also prove that counting "circuits" (i.e. walks on graphs where edges are traversed at most once and vertices may be traversed an arbitrary number of times) on ($k = \{3, 4, 5\}$)-regular (bipartite or planar) graphs is $\#P$-complete, and show that this result can be extended to show that counting circuits is $\#P$-complete $\{\forall(k \geq 3) \in \mathbb{N}\}$-regular (bipartite or planar) graphs if a polynomial-time formula for circuits on the complete bipartite graph, $K_{(n,n)}$, and a complete bipartite graph with one missing edge, $\{K_{(n,n)} - e\}$, can be found. However, as an important caveat here is that, as a consequence of of our method of proof, we are unable to say anything meaningful regarding the existence or non-existence of approximation algorithms for counting cycles and circuits outside of the $k = 3$ case. We also leave it as an open question to the biological community if there is value in counting or enumerating circuits in metabolic networks.

An additional point of biochemical motivation for this chapter comes from the field of protein folding where Hamiltonian paths and cycles arise in discretized models of proteins on lattices and graphs. Consider, for example, the Hydrophobic-Polar (HP) packing model originally proposed by Dill et. al. in circa 1985 [49], wherein a polypeptide is treated as a self-avoiding embedding of a bipartite linear chain of vertices (where vertex coloration indicates amino acid hydrophobicity) into a two- or three-dimensional integer lattice. Here, based on the hypothesis that a primary driver of protein folding was the "packing" of hydrophobic residues and the exclusion of solvent from the core of a protein, Dill et. al. [49] abstracted the Minimum Free Energy (MFE) protein folding problem - wherein one attempts to minimize the free energy, $\Delta(G) = \Delta(H) - T \times \Delta(S)$, of a given protein fold

based on some scoring function - as a functional optimization problem of finding a particular self-avoiding embedding that maximizes the number of pairwise adjacent vertices of one color type. Since this time, there have been numerous variations and extensions of the HP model, and of notable interest to us in light of our **(Chapter 2)** results, what appears to be an increasing focus on self-avoiding embeddings on cubic hexagonal (honeycomb) graphs to allow for better treatment of the natural rotamer configurations of polypeptide chains (see for example (Jiang & Zhu, 2005) [88]). Furthermore, the pairwise optimization problem proposed in the original HP model is now known to be NP-hard in a variety of settings, and refer the reader to the original (apparently simultaneous) proofs of this hardness result by (Crescenzi et. al., 1998) [44] for $\mathbb{Z}^2$ lattices (Berger & Leighton, 1998) [19] for $\mathbb{Z}^3$ lattices.

Our results in this chapter are relevant to the calculation of configurational entropy of such models. Wherever we prove that counting or even approximately counting cycles in graphs is difficult, the same can be said for counting embeddings of self-avoiding polymers (e.g. cyclical polypeptide chains). With the results in this chapter, we are potentially able to say something interesting in the sense that calculating configurational entropy fails to become easier in the limit of high vertex degrees which typically correspond to coordination numbers in discrete models of protein folding, and there is evidence that greater coordination numbers for discrete protein folding models imply higher modeling accuracy, albeit with diminishing returns (see e.g. [141]).

In **(Chapter 4)** we prove a number of results concerning the computational complexity of counting and approximately counting Hamiltonian cycles and paths on highly restricted variants of cubic graphs. Perhaps most notably, we prove that counting either Hamiltonian paths or cycles on cubic 3-connected bipartite planar graphs is $\#P$-complete and polynomial-time inapproximable unless $NP = RP$. This is interesting in the sense that the Hamiltonicity of this family of graphs is a major open problem known as Barnette's conjecture [77], and because it is known that deciding the existence of a Hamiltonian cycle on this family of graphs is NP-complete if and only if Barnette's conjecture is false [56]. Therefore, while it is famously the case that $\#P$-complete counting problems can correspond to easy decision problems like computing the 01-permanent [180] or counting linear extensions in an arbitrary poset [31], this is the first case we know of where the a $\#P$-completeness result has been achieved corresponding to a decision problem of unknown complexity.

The immediate "biochemically relevant" application of this chapter is that it establishes the strongest case of the **(Chapter 2)** (Theorem 2) $\#P$-completeness and inapproximability result for counting substrate cycles or Elementary Flux Modes (EFMs), at the limit where all reactions are reversible, on cubic 3-connected bipartite planar graphs. However, and in the manner of our "biochemically justification" for **(Chapter 3)**, the results in this chapter can also be understood to imply a hardness result for computing the configurational entropy of discrete state approximations of self-avoiding cyclical or linear polymers (e.g. polypeptide chains or ribonucleic acid polymers) at the limit where the polymer is in a compact phase. Here we abstract the polymer under consideration as a self-avoiding embedding of a cycle or path of vertices R into a graph G, which serves as an approximation for a confined volume, and we simply require that this embedding is "covering" (i.e. that the mapping between vertex sets $f : V(P) \mapsto V(G)$ is a bijection, or said differently, a 1-to-1 mapping) such that each mapping corresponds

to a Hamiltonian cycle or path in G. We therefore have that computing the configurational entropy of the polymer under such a model, is $\#P$-complete even on cubic 3-connected bipartite planar graphs and polynomial-time inapproximable on this family of graphs unless $NP = RP$. We remark that there is precedence for the successful use of such models in polymer physics (see e.g. [138] and [156]), and that we might expect a polypeptide chain to approximate a compact phase after encapsulation or some other manner of confinement by chaparone proteins assisting its folding.

In **(Chapter 5)** we attempt to strengthen the results of **(Chapter 2)** and **(Chapter 4)** by showing that the hardness of determining only the least significant bit (i.e. the parity) of the number of Hamiltonian cycles and simple cycles more generally on cubic weakly-3-connected bipartite planar digraphs and subcubic 2-connected bipartite planar undirected graphs, are complete for the class $\oplus P$, and are thus among the hardest known parity counting problems. While we leave it as an open question to the biochemical community as to whether these results have any direct physical significance, we argue that they provide a strong clue as to the difficulty of efficiently extracting information about the number of cycles in even very topologically restricted classes of graphs.

Finally, in **(Chapter 6)** we attempt a different sort of "worst case" hardness analysis, and consider the physical barriers to counting ultradilute RNA species from the perspective of fundamental nucleic acid chemistry.

To explain our motivation and to introduce this chapter, consider the challenge of recording the phenotype of a cell or section of tissue in an organism as it exists at some moment in time. Or alternatively, to make a naïve attempt at formality, consider the challenge of determining the elements of some set $S = Q \times M \times V$, where Q represents a set of labels for some appropriate subset of all RNA and protein species in the cell or tissue section, and where M and Q correspond to integer copy number counts $(m_1, m_2, ...) \in \mathbb{Z}$ and coordinate vectors $(v_1, v_2, ...) \in \mathbb{R}^3$, respectively, for each of the elements $(q_1, q_2, ...) \in Q$. Conceptually as well as practically speaking, this is essentially a problem of appropriately carving the volume of the cell or tissue sample into some set of partitions H, and then counting (or more accurately, approximately counting) the molecular species $q_i \in Q$ in each partition $h_i \in H$.

Here, we focus on a series of ultra-high sensitivity assays for the parameters Q and M [74, 128, 129], and in particular the photo-DEAN method of Yokomori et. al. [193–196], which can be used to identify the existence of RNA species and determine their individual counts, and look carefully at the barriers at the limit of low RNA molecular concentrations. Specifically, we look at the difficulty of diffusion-based search in the low target copy number limit; we look at challenges presented by spontaneous decomposition of (deoxy)ribonucleic acid (DNA) due to depurination or depyrimidination followed by intramolecular cleavage via β-elimination at abasic sites; we look at cytosine deamination via hydrolytic deamination; we look at spontaneous decomposition of RNA via transesterification; finally, we consider Cyclobutane Pyrimidine Dimer (CPD) formation and other undesired photochemistry as a result of UV irradiation for the photo-DEAN method [193–196].

Chapter 1

Introduction

1.1 Preamble

We do our utmost to elaborate upon any special terms or concepts as they arise *in situ* in the various parts of this dissertation. However, and while this seems to work decently well for concepts arising in biochemistry - see if you don't agree after reading section **(1.2)**, **(Chapter 2)**, and **(Chapter 6)** - we realize that more specialized concepts and terminology arising in computational complexity theory and/or graph theory may require a more explicit approach. We therefore include a broader, albeit less directly on topic, introduction to both of these areas. Specifically, starting with formal definitions for a few basic categorizations of functions (injections, surjections, and bijections) (**1.3**), we then define complexity classes, various reductions among complexity classes, randomized approximation schemes, elaborate to some extent on satisfiability problems (in particular $3SAT$ and its variants), and briefly introduce the resource bounded version of the Arithmetical Hierarchy (AH) known as the Polynomial Hierarchy (PH) (**1.4**). We next introduce a few terms and concepts in graph theory relevant to this dissertation (**1.5**), and finally put everything together to introduce the Hamiltonian cycle and path decision problems and elaborate on their importance in computational complexity theory (**1.6**).

We wish to stress that this is very much a "depth-first" explanation of only a small subset of relevant terms and terminology from complexity theory and graph theory. On the complexity theoretic side we certainly do not, for example, do justice to the concept of Turing machines, Cook reductions, or the Polynomial Hierarchy (PH). The same is likewise true on the graph theoretic side where we by no means do justice to e.g. Menger's theorem, Kuratowski's theorem [105], Wagner's theorem [186], or Steinitz's theorem [165]. Accordingly, we refer the interested reader to Papadimitriou [139] for a introduction to complexity theory (or to either Arora & Barak [9] or Sipser [160] for a slightly gentler introduction), and for graph theory to either the texts of Bondy & Murty [25] or Bollobás [24], and perhaps also Oxley's *Matroid Theory* [136] (this latter recommendation being for the reader with a particular interest in bridges between graph theory and other areas of mathematics). Finally, we remark that our use of complexity theory is very much focused on counting, approximately counting, or enumerating objects embedded in graphs. We therefore strongly recommend to the reader Jerrum's very accessible *Counting, Sampling, and Integrating: Algorithms and Complexity* [87] (based on a lecture series of his at ETH Zürich), as well as Welsh's *Complexity: Knots, Colourings and Counting* [188].

1.2 Substrate cycles, Elementary Flux Modes (EFMs), and Extreme Pathways (EPs) in biochemical networks

In **(Chapter 2)** we take a careful look at the computational complexity of counting objects known as "substrate cycles" in "reaction-centric graphs" of chemical or biochemical networks under a combination of stringent topological restrictions. Here, reaction-centric graphs are simply graph-based representations of the binary relations composing these networks, wherein we represent metabolite reaction centers as vertices and draw directed (resp. undirected) edges between reaction centers to represent irreversible (resp. reversible) flows of metabolites. Thus, in this context we can understand substrate cycles as simply being cycles embedded in reaction-centric graphs.

We remark that, depending on the nature of the system under consideration, substrate cycles may also correspond to objects of interest to biologists known as "Elementary Flux Modes" or EFMs (Schuster & Hilgetag; 1994) [155], and if all reactions are irreversible, to a subclass of these objects known as "extreme pathways" (Schilling et. al.; 2000) [154], where the former are minimal sets of reactions or enzymes that can maintain a particular steady-state reaction, and the latter are the extreme rays of a "flux cone" representing a given biochemical or metabolic network at steady state. We further remark that in a system where the aforementioned correspondence between substrate cycles and either of these objects exists, in particular in a "closed system" or subset of a network approximating a "closed system" (e.g. a cell or organelle), a characterization of the hardness of counting or approximating the number of substrate cycles will likewise correspond to the hardness of counting or approximating the number elementary flux modes, and in the case of directed graphs, to the hardness of counting or approximating the number of extreme pathways.

To more formally introduce elementary flux modes and extreme pathways, we first note that for a biochemical or metabolic network having only irreversible reactions, their definitions coincide. Here, both elementary flux modes and extreme pathways correspond to the generating vectors of an infinite pointed (containing the origin) convex polyhedral "steady-state" flux cone $\mathbb{K}$ where we have that the volume occupied by $\mathbb{K}$ on the positive orthant of $\mathbb{R}^n$ corresponds to linear combinations of reaction rates. In other words, we have that $\mathbb{K} = \sum_{i=1}^{n}(w_i * p_i) = 1$ where p_i are vectors encoding extreme pathways, which are equivalent to elementary flux modes in this case, and each $w_i \in R^+$ corresponds to a strictly non-activity "weight" or "activity level" for each p_i.

However, for biochemical or metabolic networks where some reactions are reversible, elementary flux modes take a bit more work to define. Here, consider a "stoichiometry matrix" S for some chemical reaction network (e.g. a metabolic network), where columns correspond to reactions, rows correspond to substrates, and where we populate entries in the matrix with (typically integer) values corresponding to how much of a substrate is used in a particular reaction. For example, if we have some reaction R of the form $r_a + 2 * r_b \longrightarrow r_c$, the positions in the stoichiometry matrix S on the column corresponding to the reaction R and along the rows in the matrix corresponding to reactants r_a, r_b, and r_c would have values -1, -2, and 1, respectively. Here, an elementary mode is simply a class of vectors v_i of reaction rates (i.e. a class of "flux vectors") that are: (1) interconvertable via multiplication by a positive real-valued scalar; (2) "physical" in the sense that the rate of an irreversible reaction in any flux vector v_i must be strictly non-negative; (3) satisfy the steady-state requirement that $\forall v_i$ we have that $Sv_i = 0$ (i.e. we have that all v_i are elements of the null space of the stoichiometry matrix S); and (4) satisfy an "elementarity" criterion requiring that,

roughly speaking, no proper subset of the reactions in a the vector itself constitutes an elementary mode. To elaborate on (4), this elementarity criterion implies that for a flux vector to be an elementary mode, a submatrix of the stoichiometry matrix S containing only the reactions with non-zero rates in the flux vector, must have a nullspace of dimension one, or equivalently, must have rank equal to the number of reactions it concerns minus one (Klamt et. al.; 2005) [101]. We can immediately see from this definition of elementary flux modes that, outside of the special case where all reactions in a system are reversible and unlike extreme pathways, nothing prevents one elementary mode from being the sum of two other elementary flux modes.

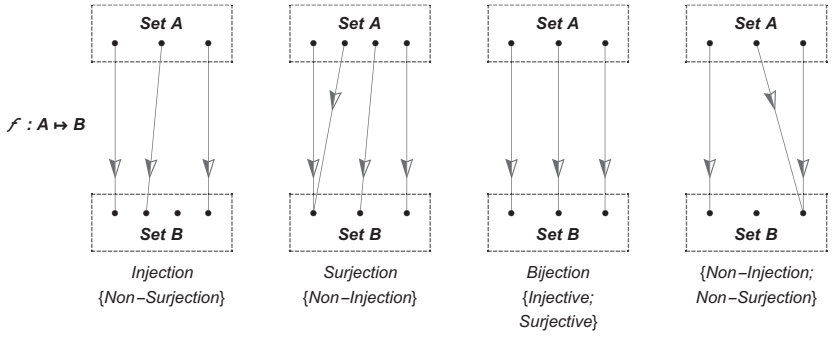
Finally, regarding precedence for our efforts in **(Chapter 2)**, we wish to briefly highlight (Acuña et. al.; 2009) [1] wherein the directed Hamiltonian cycle decision problem was utilized as a means of proving the NP-completeness of deciding if a particular subset of reactions in a biochemical network corresponds to an elementary mode. Briefly the strategy in (Acuña et. al.; 2009) [1] is to reduce the directed Hamiltonian cycle decision problem to the question of ("Does there exist an elementary mode containing exactly the specified reaction set $(r_1, r_2, ...) \in R$?") was to take an arbitrary digraph G with vertex set V_G and, in the case of no double edges (corresponding to reversible reactions), create a new graph Q by: (1) "splitting" every vertex $v_i \in V_G$ into two vertices, $\{v_{(i,1)}, v_{(i,2)}\}$, (2) adding a directed edge between the split vertices oriented towards $v_{(i,2)}$; (3) redirecting edges formerly oriented toward v_i to be oriented toward $v_{(i,1)}$ and edges formerly oriented away from v_i to be oriented away from $v_{(i,2)}$. Here, if one simply calls vertices in Q "reactants", directed edges "reactions", the set of directed edges or "reactions" between split vertices R, and the $\{-1, 0, 1\}$ incidence matrix for Q a "stoichiometry matrix", there is a 1-to-1 correspondence between directed cycles in G and elementary flux modes in the reaction network corresponding to Q. Therefore, we have that there exists a directed Hamiltonian cycle in G iff there exists a elementary mode involving all of the reactions R corresponding to directed edges between the "split" vertices in G.

1.3 Classification of functions according to their arguments and images

Injections (also called injective functions / injective maps / one-to-one functions / embeddings) :: A function f, defined on a set A and taking values on a set B ($f : A \mapsto B$), is an injection if it maps distinct elements to distinct elements, i.e. if $\forall (x, y) \in A$ we have that $f(x) = f(y)$ iff $x = y$, and correspondingly, that $f(x) \neq f(y)$ iff $x \neq y$.

Surjections (also called surjective functions / surjective maps / onto functions) :: A function f, defined on a set A and taking values on a set B ($f : A \mapsto B$), is a surjection if $\forall b_i \in B$ we have that there exists some $a_i \in A$ such that $f(a_i) = b_i$.

Bijection (also called bijective functions / one-to-one and onto functions) :: A function f, defined on a set A and taking values on a set B ($f : A \mapsto B$), is a bijection if it is both an injection (as defined above) and a surjection (also as defined above); i.e. f is a bijection if $\forall b_i \in B$ we have that there exists some unique $a_i \in A$ such that $f(a_i) = b_i$.



1.4 Complexity theoretic definitions and terminology

Complexity classes :: While definitions for individual complexity classes broadly vary, it is generally true that a complexity class will consist of a set of problems having related time $T(n)$ and/or space $S(n)$ requirements or complexities on a Turing machine with a write-only "input tape" of size n, a read-write "work tape" of size $S(n)$, and a write-only output tape.

Reductions in complexity theory :: A reduction is a formulation of an instance of some problem R (e.g. $3SAT$) in terms of different problem S (e.g. the existence of a Hamiltonian cycle in some graph).

Polynomial time Turing (Cook) reductions :: Here a problem R is polynomial-time Turing-reduced (this can be denoted as $R \leq_T^P S$ or simply $R \leq_T S$) to a problem S if R can be solved in polynomial time with multiple and/or adaptive calls to an oracle for problem S (here "adaptive" means that the nature of successive calls to an oracle can be influenced by the result(s) of previous call(s)); this is considered a "more powerful" form of reduction than a Karp many-one reduction.

Karp (polynomial-time many-one) reductions :: Here a problem R is Karp many-one reduced to a problem S if one has a function f that maps instances of some problem R to instances of S; i.e. for every instance x of problem R we have that $x \in R \leftrightarrow f(x) \in S$ (note: there are no requirements for f to be either injective / surjective / bijective); Karp many-one reductions may be denoted as: $R \leq_m^P S$, though sometimes these reductions are simply denoted as e.g. $R \leq_p S$ or $R \leq_m S$, where the subscript in these two instances stands for polynomial and mapping, respectively. We may intuitively distinguish polynomial-time Karp many-one reductions from so-called "1-Turing" or "Cook[1]" reductions, which are Turing reductions where only a single oracle call is permitted, by noting that polynomial-time Karp many-one reductions require that an explicit method is presented for translating the language of one type of problem instance into another. We can moreover note that many function- and language-type complexity classes known to be closed under Karp-type reductions may not be closed under "Cook[1]"-type reductions (see e.g. the end of "Section 2" of ref. [137] for a discussion concerning this latter point).

Karp (polynomial-time one-one) reductions :: Karp many-one reductions are one-one reductions if the reduction function f is injective.

Oracles :: Generally speaking, oracles are just "black boxes" capable of instantaneously yielding the answer to some predefined decision or function problem; if a function f has access to an oracle for some problem R (which may be complete for a given complexity class) this is typically written as f^R.

The complexity class P :: Problems that can be solved in polynomial time (i.e. time $T(n) = \mathcal{O}(n^k)$ where n is a string encoding the problem specification) on a deterministic Turing machine. Problems in this class are sometimes referred to as being "efficiently solvable", however this is misleading in the sense that the degree of the polynomial, k, may be an arbitrarily large integer.

The complexity class NP :: The set of all decision problems (i.e. problems with a "yes" or "no" answer in some formal system) with proofs verifiable in polynomial time on a deterministic Turing machine; equivalently, the class of problems where accepting instances can be accepted in polynomial time by a non-deterministic Turing machine. The complement class $coNP$ is defined with regards to polynomial-time verifiable counterexamples.

Valiant's counting class $\#P$ :: In 1979, Valiant defined a class $\#P$ [180,181] of counting problems as the set of integer function problems of the form $f : \Sigma^* \longrightarrow \mathbb{N}$ where one is tasked with determining the number of accepting pathways $f(x_i)$ for a nondeterministic Turing machine M running in polynomial time on all input strings x_i encoded over the alphabet Σ (where we typically have $\Sigma = \{0, 1\}$), and where $\forall x_i$ we have that $|x_i| = poly(|f(x_i)|)$. We can equivalently define the class $\#P$ as the set of function problems where one is tasked with determining the cardinality $f(x_i)$ of the witness set of some instance x_i of an NP set L, with respect to some binary witnessing relation $W_L \subseteq \Sigma^* \times \Sigma^*$ and some polynomial-time (sound and complete) NP verifier.

The parity polynomial-time complexity class $\oplus P$ :: The complexity class $\oplus P$ [72, 140, 180, 181] consists of the set of decision problems solvable on a nondeterministic Turing machine M running in polynomial time on binary encoded input, where a language L is in $\oplus P$ iff $\forall r \in L$, where r is a binary string encoding an instance of the language, we have that M has an odd number of accepting pathways. Alternatively, $\oplus P$ can be said to constitute the set of decision problems where, provided a string encoding a problem instance x_i in a language $L \in NP$, one is tasked with determining the value of the least significant bit for a binary encoding of the cardinal number of the witness set for x_i with respect to some witnessing relation W_L and some fixed polynomial-time NP-verifier for this relation. With regards to completeness for the class $\oplus P$, this is typically defined in terms of a form of polynomial-time many-one counting reductions [182]. Here, if f and h are two problems in $\oplus P$, we can reduce f to h via by finding a polynomial time function $R_1 : \Sigma^* \longrightarrow \Sigma^*$ where we have that $R_1(x) \in h$ iff $x \in f$ and where, in addition, we have that the parity of the number of accepting pathways is preserved.

Polynomial-time reductions among problems in the complexity class #P :: If $f : \Sigma^* \longrightarrow N$ is a is $\#P$-complete counting problem, $h : \Sigma^* \longrightarrow \mathbb{N}$ is a counting problem in the complexity class $\#P$, and we wish to reduce f to h in some manner to establish that $h \in \#P$ is also $\#P$-complete, one generally chooses to carry out either a polynomial-time Turing reduction or a weaker polynomial-time Karp many-one counting reduction. Roughly speaking, the essential difference between polynomial-time Turing and polynomial-time Karp many-one counting reductions is that Turing reductions allow for multiple adaptive calls to an oracle for h to solve f, while "weaker" polynomial-time Karp many-one reductions must consist entirely of operations on strings over some (typically binary) alphabet and do not involve oracle calls (and therefore may or may not be weaker than "1-Turing" or "Cook[1]" reductions which are Turing reductions where only a single oracle call is permitted). Therefore, it is generally speaking a stronger result to achieve a polynomial-time Karp many-one counting reduction as opposed to a polynomial-time Turing reduction, in particular one making use of multiple adaptive oracle calls.

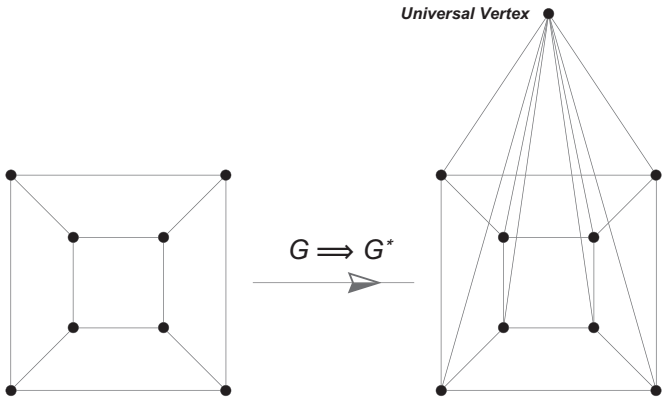
To elaborate on polynomial-time Karp many-one counting reductions (which are sometimes referred to as *weakly parsimonious* reductions), here, when reducing one integer function problem f to another integer function problem h, one has

two polynomial-time compatible functions $R_1 : \Sigma^* \longrightarrow \Sigma^*$ and $R_2 : \mathbb{N} \longrightarrow \mathbb{N}$, such that $f(x) = R_2(h(R_1(x)))$. If R_2 is the identity function we call the counting reduction a *parsimonious* reduction, and as a further special case, if R_2 is an integer division operation by an exponent of two, i.e. if $f(x) = \lfloor \left(\frac{h(R_1(x))}{2^{R_3(x)}} \right) \rfloor$ where $R_3 : \Sigma^* \longrightarrow \mathbb{N}_{>0}$, then we refer to the counting reduction as a *right-bit-shift* reduction [118]. Completeness for Valiant's $\#P$ counting class was originally defined in terms of polynomial-time Turing reductions [180, 181], however progress is increasingly being made to establish $\#P$-completeness with respect to polynomial-time Karp many-one counting reductions [34, 203].

{P, NP, #P, $\oplus P$,etc.}-completeness :: When a problem is "complete" for its complexity class, this implies that the problem is moreover reducible - via e.g. Turing or many-one counting reductions - to any other problem in its class in polynomial time. As an example, consider that the well-known P vs. NP problem, originally posed in (Cook; 1971) [40], is fundamentally a question of whether a polynomial-time algorithm can be found for any NP-complete problem, implying the existence of a polynomial-time solution, or lack thereof, for all NP-complete problems.

Randomized approximation schemes and Approximation Preserving reductions (AP-reductions) :: Let $f : \Sigma^* \longrightarrow \mathbb{N}$ be a counting problem in the complexity class $\#P$, let $x \in \Sigma^*$ be some appropriate input for f, and let $f(x) = N$. Following Karp & Luby [93], we define a Randomized Approximation Scheme (RAS) as a randomized algorithm which takes f and x as input and outputs some value $\hat{f}_{(\epsilon,\delta)}$ such that $Pr\left[\left(\frac{|\hat{f}_{(\epsilon,\delta)} - f(x)|}{f(x)} \right) > \epsilon \right] < \delta$, where we have some *error rate* parameter $0 < \epsilon < 1$ and some *accuracy parameter* $0 < \delta < 1$. A Fully Polynomial-time Randomized Approximation Scheme (FPRAS) is simply a RAS that has a running time polynomially bounded by $|x|$, ϵ^{-1}, and δ^{-1}. Now let $\{f, h\} : \Sigma^* \longrightarrow \mathbb{N}$ be two counting problems in the complexity class $\#P$. An *Approximation Preserving reduction* (AP-reduction) from f to h (denoted $f \leq_{AP} h$), as originally defined by Dyer et. al. [51], is a probabilistic oracle Turing machine M, taking as input a string $x \in \Sigma^*$ and error parameter $0 < \epsilon < 1$, and satisfying the following three conditions: (1) letting x be an instance of h and $0 < \delta < 1$ (where δ^{-1} is polynomially bounded by $|x|$ and ϵ^{-1}), we have that all calls to M specify an input of the form $\{x, \epsilon\}$; (2) we have that M is a RAS for f if it is a RAS for g; (3) the time complexity for M is polynomially bounded by $|x|$ and ϵ^{-1}. Here, if $f \leq_{AP} h$ and $h \leq_{AP} f$, we call f and h *AP-interreducible* and write $f \equiv_{AP} h$.

Example of a Karp-type (and parsimonious-type) reduction :: There exists a very simple reduction from the problem of detecting the existence of a Hamiltonian path in a graph to the problem of detecting the existence of a Hamiltonian cycle (i.e. a (Hamiltonian path) $\leq_m$ (Hamiltonian cycle) reduction). Here, if G is some graph where one wishes to decide the existence of a Hamiltonian path, one can construct a graph G^* in polynomial time where all vertices in G are connected to a "universal vertex". We then have that there is a Hamiltonian path in G iff there is a Hamiltonian cycle in G^*. We remark that this (Hamiltonian path) $\leq_m$ (Hamiltonian cycle) reduction is also parsimonious, i.e. we have that the number of Hamiltonian paths in G is equivalent to the number of Hamiltonian cycles in G^*.



Conjunctive Normal Form (CNF) :: A CNF formula Φ_{CNF}, which is a special form of a Quantified Boolean Formula (QBF), consists of a conjugation of $AND \leftrightarrow \wedge$ operations of conjucts, and conjucts consist of an arbitrary number of $OR \leftrightarrow \vee$ operations between literals – e.g. $(x_1 \vee x_2) \wedge x_3$ or $(x_1 \vee \neg x_2) \wedge x_3$ are in CNF form while e.g. neither $(\neg x_1 \vee x_2) \vee \neg x_3$ nor $(\neg x_1 \wedge x_2) \wedge x_3$ nor $(x_1 \wedge \neg x_2) \vee x_3$ nor $(x_1 \vee \neg x_2 \vee (x_3 \vee x_4)) \wedge x_5$ are in proper CNF form. Importantly, we have that every sentence in propositional logic can be expressed in CNF form.

3-Satisfiability ($3SAT$), Karp's 11th NP-complete decision problem :: An NP-complete Boolean satisfiability problem where one is asked to decide the truth value for a $kCNF$ expression (i.e. a CNF expression where conjucts uniformly have $k = 3$ literals) of the form, e.g.: $\Phi_{3SAT} = (\neg x_1 \vee x_2 \vee \neg x_{876}) \wedge (x_4 \vee x_1 \vee \neg x_3) \wedge ... \wedge (x_{943} \vee \neg x_2 \vee x_{876})$.

Variants of $3SAT$ :: With the proliferation of NP-completeness and NP-hardness as a metrics for defining the difficulty of solving problems arising in mathematics and increasingly the physical sciences - (see e.g. (Barahona; 1982) [14] regarding a proof of the NP-hardness of calculating the ground state of an Ising model on $\mathbb{Z}^3$ lattices or $\mathbb{Z}^2$ lattices in the presence of an external field; (Istrail; 2000) [84] regarding a proof of the NP-hardness for finding the ground state of Ising models on non-planar graphs) - there has been a concomitant effort to sharpen and extend the set of 21 NP-complete problems introduced by Richard Karp in 1972 [92], as well as their derivatives, in order to show hardness results for more restricted classes of inputs that might better correspond to "real world" instances of the problems. Consider the instance of the SAT problem with at most three literals per clause ($3SAT$), the 11th entry in Karp's list [92], and how over the years the NP-completeness for this decision problem was shown to hold in the case of e.g. ::

"Not-All-Equal(NAE)-$3SAT$" An NP-complete variant of the $3SAT$ decision problem where no satisfying instance of a of an $NAE - 3SAT$ formula has a clause with ≥ 2 "True" literals.

"Planar $3SAT$" formula where, representing clauses and literals as vertices in two distinct partite sets of a graph with edges indicating membership of literals in

clauses, the resulting bipartite graph can be embedded in the plane without edge crossings (Lichtenstein; 1982) [111];

"Planar exactly 1-in-3SAT", where in addition one requires exactly one literal in each clause is true (Dyer & Frieze; 1986) [52];

The case of "rectilinear planar 3SAT" formula where all vertices corresponding to literals are arranged in a line, all vertices corresponding to clauses are drawn as rectangles parallel to this line of literal vertices, and all edges must be perpendicular to the line of literal vertices (Knuth & Raghunathan; 1992) [103];

The case of "planar monotone exactly 1-in-3SAT" where one disallows clauses having a mixture of positive and negative literals (Hunt et. al.; 1998) [82];

The case of "rectilinear planar monotone 3SAT" with the added constraint one must position the rectangles corresponding to clause vertices above (below) the line of vertices if all of the literals in a clause are positive (negative) (de Berg & Khosravi; 2010) [48];

Finally, we remark that a few valleys in this landscape have also been found where constraints imposed upon 3SAT are sufficient to make the problem polynomial-time solvable. Consider for example the perhaps surprising proof that "planar Not-All-Equal (NAE) 3SAT", where all clauses must have either exactly two "True" and one "False" literal or two "True" and one "False" literal, is in P (Moret; 1988) [123].

Quantified Boolean Formula (QBF) :: A Quantified Boolean Formula (QBF) is an expression of the form: $\Phi_{QBF} = Q_1 x_1 Q_2 x_2 ... Q_n x_n \phi(x_1, x_2, ..., x_n)$ where $x_i \in \{0, 1\}$ and $Q_i \in \{\exists, \forall\}$. Here the example formula Φ_{QBF} is written in "prenex normal form" where one has a string of quantifiers called a "prefix" preceding a quantifier free "matrix" specifying an unquantified Boolean formula). We note that the individual x_i can also represent finite sets, $(b_1, b_2, ..., b_m) \in x_i$, of Boolean variables.

Polynomial Hierarchy (PH) :: The Polynomial Heirarchy PH, originally introduced by Stockmeyer in 1976 [166], is can be understood as a "resource bounded" variant of the Arithmetical Heirarchy (AH) from the field of mathematical logic.

For some intuition and grounding, consider the question of whether a Quantified Boolean Formula (QBF) is true, i.e. if some an expression of the form $\Phi_{QBF} = Q_1 x_1 Q_2 x_2 ... Q_n x_n \phi(x_1, x_2, ..., x_n)$, where $x_i \in \{0, 1\}$ and $Q_i \in \{\exists, \forall\}$, is an element in the language of True Quantified Boolean Formula ($TQBF$). For some intuition and grounding, we briefly remark that the NP-complete problem SAT – [Does there exist some $\{False, True\} = \{0, 1\}$ assignment to the set of variables x_i in a Boolean formula ϕ such that ϕ evaluates to $True$?] - is equivalent to: $\exists x_1 \exists x_2 ... \exists x_n \phi(x_1, x_2, ..., x_n)$, and consider that the $coNP$-complete problem $Tautology$ – [Do all possible $\{False, True\} = \{0, 1\}$ assignments for the variables x_i in given Boolean formula ϕ cause the formula to evaluate to the value $True$?] - is equivalent to: $\forall x_1 \forall x_2 ... \forall x_n \phi(x_1, x_2, ..., x_n)$.

$TQBF$ is, in the general case, $PSPACE$-complete under polynomial time and logarithmic space reductions (note that $PSPACE = \cup_{k \in \mathbb{N}} SPACE(n^k)$ where $SPACE(S(n))$ is the set of decision problem solvable on a deterministic Turing machine using a work tape of size $\mathcal{O}(S(n))$, i.e. $PSPACE$, is the set of decision problems solvable with a Turing machine having a polynomial bounded work tape of size $S(n) = poly(n)$ where n is the input problem size). However, if one bounds

the number of QBF quantifiers, $Q_i \in \{\exists, \forall\}$, i.e. bounds n in an expression of the form $Q_1 x_1 Q_2 x_2 ... Q_n x_n \phi(x_1, x_2, ..., x_n)$, then one is dealing with problems on the Polynomial Hierarchy $(PH) \implies PH \subseteq PSPACE$.

Concerning standard terminology for complexity classes in the Polynomial Hierarchy (PH) (assuming PH doesn't collapse), first we define $\sigma^P_{k=0} = \sum^P_{k=0} = \Delta^P_{k=0} = P$, where P is again the class of problems solvable in polynomial time (i.e. time $T(n) = \mathcal{O}(n^k)$) on a deterministic Turing machine and the superscript P just designates that we're considering the Polynomial Hierarchy (PH) and not e.g. the Arithmetic Heirarchy (AH)). Now, let $\sum^P_k$ (where $\sum^P_1 = NP$) be a formula f of the form: $Q_1 x_1 Q_2 x_2 ... Q_n x_n \phi(x_1, x_2, ..., x_n) = \exists x_1 \forall x_2 \exists x_3 ... Q_k x_k \phi(x_1, x_2, ..., x_k)$, where $Q_k = \forall$ if k is even and $Q_k = \exists$ if k is odd, and let $\prod^P_k$ (where $\prod^P_1 = coNP$ and $\forall (k \geq 0)$ where $co - \sum^p_{k+1} = \prod^p_{k+1}$) be a formula f of the form: $Q_1 x_1 Q_2 x_2 ... Q_n x_n \phi(x_1, x_2, ..., x_n) = \forall x_1 \exists x_2 \forall x_3 ... Q_k x_k \phi(x_1, x_2, ..., x_k)$, where $Q_k = \exists$ if k is even and $Q_k = \forall$ if k is odd. With these definitions in hand we can now write: $PH = \cup_{k \in \mathbb{N}} \sum^P_k = \cup_{k \in N} \prod^P_k$. Correspondingly, in terms of oracle-based definitions, we have that $\forall (k \geq 0)$ $\Delta^P_{k+1} = P^{\sum^P_k}$ and $\sum^P_{k+1} = NP^{\sum^P_k}$.

We remark that the following is "probably true" unless the Polynomial Hierarchy (PH) collapses, i.e. if we have $\sum^p_k = \sum^P_{k+1} \longleftrightarrow \sum^p_k = \prod^P_k$, if there exists some PH-complete problem, or if $PH = PSPACE$, etc.:

$$PH \subseteq P^{\#P} \subseteq PSPACE \subseteq EXPTIME \subseteq EXPSPACE$$

Finally, we note that Seinosuke Toda's landmark 1991 paper [172] established (among other things) that $PH \subseteq P^{\#SAT} \implies$ any problem $h \in PH$ is Cook reducible to a polynomial-time function $f \in P$ able to make a single call to an oracle for the $\#P$-complete problem $\#SAT$ (i.e. any problem $h \in PH$ is Cook reducible to $\#P$); for this proof, Toda won the 1998 Gödel prize.

1.5 Graph theoretic definitions and terminology

Vertex degree; (v_i) **::** The number of edges incident to a vertex v_i in a graph G; the maximum vertex degree can be denoted as $\Delta(G)$ and the minimum vertex degree as $\delta(G)$.

k-Regular graph :: A graph G is k-regular if all of its vertices $v_i \in V$ have uniform degree $\rho(v_i) = k$.

Cubic graph :: A graph is "cubic" if it is $(k = 3)$-regular.

Subcubic graph :: A graph is "subcubic" if the maximum vertex degree is $\Delta(G) = 3$.

k-Vertex-connected graph :: A graph G is a k-vertex-connected graph if the cardinality of the minimum vertex set whose removal disconnects G (i.e. minimum vertex cut) is of size at least k.

k-Edge-connected graph :: A graph G is a k-edge-connected graph if the cardinality of the minimum edge set whose removal disconnects G (i.e. minimum edge cut) is of size at least k.

Essentially-k-(vertex or edge)-connected graph :: A graph G is an essentially-k-vertex-connected graph (resp. essentially-k-edge-connected graph) if the cardinality of the minimum vertex set (resp. edge set) whose removal decomposes G into two connected components, each consisting of at least two vertices, is of size at least k.

Cyclically-k-(vertex or edge)-connected graph :: A graph G is a cyclically-k-vertex-connected graph (resp. cyclically-k-edge-connected graph) if the cardinality of the minimum vertex set (resp. edge set) whose removal decomposes G into two connected components, each containing at least one cycle, is of size at least k.

Bipartite graph :: The family of graphs where $\chi(G)$ (i.e. all $(k = 2)$–colorable graphs); every tree graph is bipartite, and every planar graph where all facial boundaries are cycles of even length is bipartite.

Chromatic number; $\chi(G)$ **::** The chromatic number of a graph G is the smallest set of "colors", or more generally, vertex labels necessary to paint the vertices of the graph such that no two vertices of the same color share an edge. Said differently, $\chi(G) = k$ implies the existence of a minimal k-coloring of G.

Chromatic polynomial; $\chi_G(z)$ **(alternatively written as** $\pi_G(z)$**,** $P_{(G,z)}$**, or** $C_{(G,z)}$**) ::** A polynomial and graph invariant which counts the number of distinct k-colorings for a graph G for all $k \leq z$, such that we have $\chi(G) = min\{z : \chi_G(z) > 0\}$ where $\chi(G)$ is the chromatic number of the graph G. Due to (Stanley; 1973) [163] we know that $\chi_G(-1)$ computes the number of acyclic orientations of G (note counting all acyclic orientations of an arbitrary simple

graph G is $\#P$ complete (Linial; 1986) [117]); the chromatic polynomial is a special case of and to some extent the inspiration for the Tutte polynomial, and we can write down the relationship between these two graph invariants as: $\chi_G(z) = (-1)^{(n-c)} z^c T_G(1-z, 0)$ (see "Theorem 3.1" in (Las Vergnas; 1980) [185]), where n is the cardinal number of the vertex set c is the number of connected components for an arbitrary graph G (in (Las Vergnas; 1980) [185] $\chi_G(-1) = T_G(2, 0)$ is explicitly stated).

Planar graph :: Planarity implies that the graph is $G \longrightarrow \mathbb{R}^2$ embeddable; by Wagner's theorem [186] a graph is planar iff it does not contain as minors the K_5 graph (i.e. the complete graph on $(n = 5)$ vertices) or $K_{(3,3)}$ graph (i.e. the complete bipartite graph where each partite set has $(n = 3)$ vertices, eq. the "utility" graph); we remark that Wagner's theorem [186] is equivalent to Kuratowski's theorem [105] (see e.g. "Theorem 17" and "Theorem 18" and the surrounding discussion on (pp. 24 – 25) of (Bollobás; 1998) [24]).

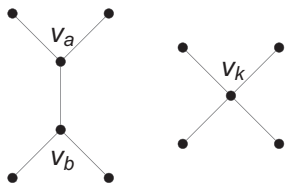
Plane graph :: An $\mathbb{R}^2$ embedded planar graph; i.e. a graph where there exists a mapping of all vertices to points on a plane, a mapping of all edges to disjoint (except for their extreme points) curves on a plane, and where the extreme points of two curves intersect iff they correspond to edges with a vertex in common.

Graph genus :: Minimum genus of a surface allowing for an embedding of a graph G without edge crossings; a planar graph is defined as having genus 0.

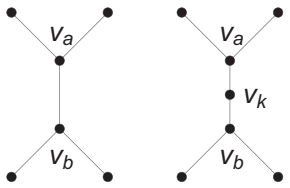
Vertex induced subgraph (sometimes just called an induced subgraph) :: Letting V_G and E_G be the set of vertices and edges, respectively, for an arbitrary graph G, a subgraph H "induced" by the vertices in some subset $R \subseteq V_G$ has vertex set R and an edge set consisting of all edges $e_i \in E_G$ between arbitrary vertices v_a and v_b where we have that $(v_a, v_b) \in R$.

k-Hop subgraph (sometimes called a k-neighborhood subgraph) :: A vertex induced subgraph (as defined immediately above) in an arbitrary graph G, here induced by the union of some selected vertex v_i and all vertices within k "edge-wise hops" of v_i.

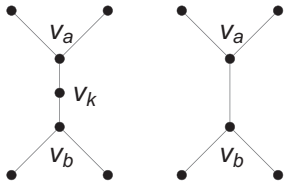
Edge contraction :: Letting $(v_a, v_b) \in V$ be a pair of vertices in an arbitrary graph G connected by an edge $e_{(v_a, v_b)} \in E$, edge contraction is an "operation" on G deletes the edge $e_{(v_a, v_b)}$ and merges v_a and v_b into a single vertex v_k.



Edge subdivision :: Letting $(v_a, v_b) \in V$ be a pair of vertices in an arbitrary graph G connected by an edge $e_{(v_a,v_b)} \in E$, edge subdivision is an "operation" on G that deletes the edge $e_{(v_a,v_b)}$, and replaces it with two new edges, $e_{(v_a,v_k)}$ and $e_{(v_k,v_b)}$, connecting v_a and v_b to a new vertex v_k.

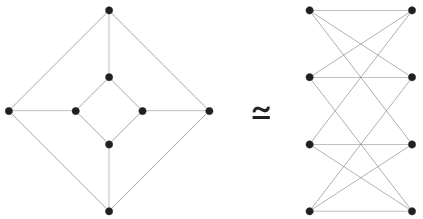


Edge smoothing (i.e. deletion of a degree $\rho(v_i) = 2$ vertex) :: In some sense the opposite of the (previously defined) edge subdivision operation wherein one has a vertex v_k of degree $\rho(v_k) = 2$ connected to some pair of vertices v_a and v_b by a pair of edges, $e_{(v_a,v_k)}$ and $e_{(v_k,v_b)}$, and one deletes $e_{(v_a,v_k)}$ and $e_{(v_k,v_b)}$ and adds a single edge $e_{(v_a,v_b)}$ between vertices v_a and v_b.



Graph minor :: A graph H is a minor of an arbitrary graph G if H can be generated from G by deleting vertices and/or deleting edges and/or contracting edges.

Isomorphic graphs :: Graphs G and H are isomorphic (denoted $G \simeq H$) if there is an edge-preserving bijection of the vertices $f : v_i \in V_G \longrightarrow v_k \in V_H$ for the two graphs.



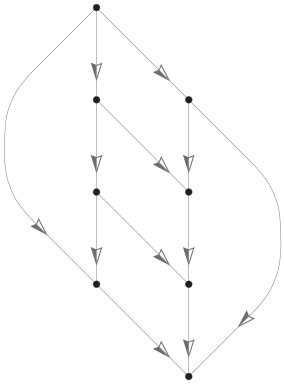
Homeomorphic graphs :: Graphs G and H are considered homeomorphic if some number of distinct and/or identical edge subdivision operations (as defined above) on G and H can generate an isomorphic pair of graphs; i.e. less rigorously, G and H are homeomorphic if they have the same the same "topological structure" (recall the coffee cup $\longleftrightarrow$ donut joke cited in classical topology).

Tree :: An undirected and connected graph where, for any pair of vertices, (v_a, v_b), there exists only a single path between v_a and v_b.

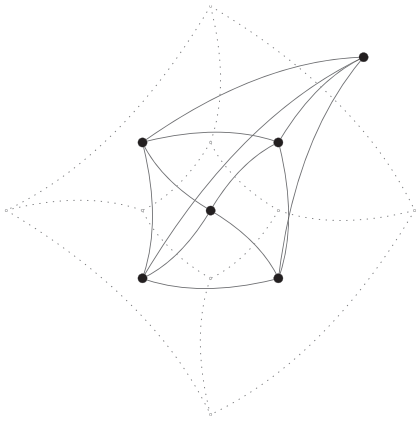
Arborescence :: A directed graph (digraph) variant of a tree where all edges are oriented to point away from some root vertex, v_r, and where we again have for any pair of vertices, (v_a, v_b), there exists only a single path between v_a and v_b.

Forest :: A graph where every connected component is a tree, i.e. a disjoint union of one or more tree graphs (a forest may also be the empty graph).

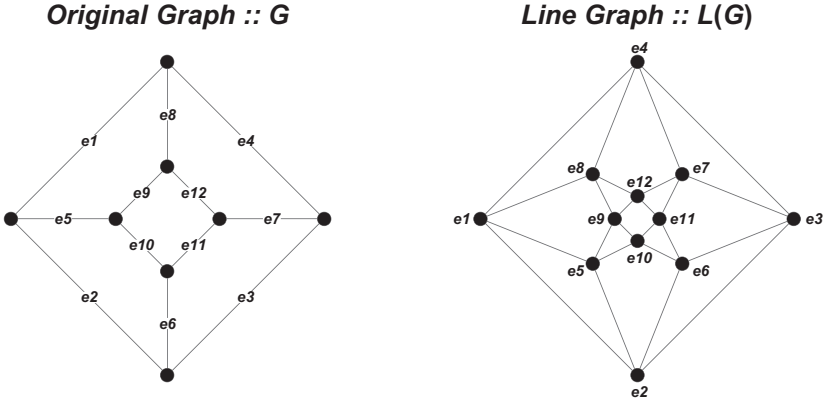
Directed Acyclic Graph (DAG) :: A cycle-free directed graph (digraph).



Dual graph :: Letting G be an arbitrary planar graph embedded in $\mathbb{R}^2$ (illustrated as the graph with dashed edges in the example below), we generate the geometric dual graph (equivalently, the combinatoric dual) graph G^* by positioning a vertex somewhere in each region defined by G and joining two vertices if and when the two regions defined by G share an edge (illustrated as the graph with solid edges in the example below).



Line graph; L_G :: Letting G be an arbitrary simple graph (i.e. a graph without multiple edges and/or loops), we generate the line graph, L_G, by placing a vertex somewhere along every edge in G, connecting these vertices iff their respective edges share a vertex in G, and deleting the original vertices in G.



1.6 Hamiltonian cycles, Hamiltonian paths, and decision problems concerning their existence in undirected and directed graphs

If G is an undirected connected graph on n vertices, a Hamiltonian cycle (resp. Hamiltonian path) "on", or equivalently, "in" G corresponds to any connected subgraph g' of G on n vertices and n edges (resp. $(n-1)$ edges) such that g' is an undirected cycle graph (resp. undirected path graph). If G is instead a directed graphs, or equivalently, a digraph, we have the additional requirement that g' is a directed cycle graph (resp. directed path graph) having vertex in-degrees and out-degrees at most one. Alternatively, we can define Hamiltonian cycles and paths as special cases of so-called spanning trees, where a spanning tree is a tree subgraph of some undirected graph or digraph G containing all of the vertices in G and having the minimum set of edges necessary to allow for reachability of all vertices from at least one vertex via an edge-wise walk respecting the orientation of any directed edges. Here, a Hamiltonian cycle (resp. Hamiltonian path) can be thought of as any spanning tree T of G where one has the additional constraint that every vertex in T has degree exactly two (resp. degree at most two).

Provided this context, we can now define the well-known Hamiltonian cycle (resp. directed Hamiltonian cycle) and Hamiltonian path (resp. directed Hamiltonian path) decision problems as problems where, provided some undirected graph (resp. digraph) G as input, one is tasked with correctly returning the answer "Yes" or "No" as to whether at least one copy of the relevant type of aforementioned subgraphs exists in G. When the answer to whether or not there exists at least one Hamiltonian cycle is "Yes" (resp. "No") we call G "Hamiltonian" (resp. "non-Hamiltonian"), and when the answer to whether or not there exists at least one Hamiltonian path is "Yes" (resp. "No") we refer to G as being "traceable" (resp. "non-traceable").

We note that both the concept of Hamiltonian cycles, and to some extent the concept of deciding if a Hamiltonian cycle exists in a graph, was first elaborated upon in a 1856 publication by Thomas Penyngton Kirkman concerning classification of polyedra (Kirkman; 1856) [99] wherein Kirkman discusses and possibility and impossibility, in certain cases, of "drawing" non-intersecting polygons along the "summits" (i.e. vertices) of polyhedra. The adjective "Hamiltonian" therefore postdates the graph theoretic concept to which it is now affixed, and is apparently due to William Rowan Hamilton's circa 1856 icosian game wherein one is challenged to complete a Hamiltonian cycle in on a dodecahedron after an opponent specifies five consecutive vertices in the graph (see e.g. (pg. 53) of (Bondy & Murty; 1976) [25]). We also note that, despite the fact that Hamiltonian cycles and paths were conceptualized ≈ 160 odd years prior to the current date, these seemingly abstract graph theoretic concepts have found a fundamental role in the field of computational complexity theory, which is a field very much having its origins in the mid- to late-20th century. Consider that the directed and undirected Hamiltonian cycle decision problems were, respectively, the 9th and 10th entries in (Karp; 1972)'s [92] list of 21 polynomial-time inter-reducible problems complete for the set NP. Karp prognosticated, correctly it would seem, that his 21 NP-complete problems would be useful extension of (Cook; 1971)'s [40] proof for the first known NP-complete quantified Boolean Satisfiability (SAT) problem, and that these problems would serve as a series of bridges or "layers of abstraction" between a variety of fields spanning, quoting (pg. 86) of (Karp; 1972) [92]: "...mathematical programming, graph theory, combinatorics, computational logic and switching theory...".

Chapter 2

Counting Substrate Cycles in Topologically Restricted Metabolic Networks

2.1 Chapter abstract

Substrate cycles in metabolic networks play a role in various forms of homeostatic regulation, ranging from thermogenesis to the buffering and redistribution of steady-state populations of metabolites. While the general problem of enumerating these cycles is $\#P$-hard, it is unclear if this result holds for realistic networks where e.g. pathological vertex degree distributions or minors may not exist. We attempt to address this gap by showing that the problem of counting directed substrate cycles ($\#DirectedCycle$) remains $\#P$-complete (implying $\#P$-hardness for enumeration) for any superclass of cubic weakly-3-connected bipartite planar digraphs, and at the limit where all reactions are reversible, that the problem of counting undirected substrate cycles ($\#UndirectedCycle$) is $\#P$-complete for any superclass of cubic 3-connected bipartite planar graphs where the problem of counting Hamiltonian cycles is $\#P$-complete. Lastly, we show that unless $NP = RP$, no $FPRAS$ can exist for either counting problem whenever the Hamiltonian cycle decision problem is NP-complete.

2.2 Introduction

In graph theoretic terms, substrate cycles can be thought of as cycles in *reaction-centric graphs* of metabolic networks, where vertices correspond to enzymes and directed (resp. undirected) edges correspond to irreversible (resp. reversible) flows of metabolite species between enzymes. Cycles in reaction-centric graphs are also known as cyclical *Elementary Flux Modes* (EFMs) [155], and may correspond to special cases of these objects denoted *extreme pathways* [154], where the former are minimal sets of reactions or enzymes that can maintain a particular steady-state reaction and the latter are the extreme rays of a *flux cone* for a given biochemical or metabolic network at steady state.

Substrate cycles have been linked to thermogenesis in the flight muscles of bumble bees [38] as well as in the brown adipose tissue of mammals [94,127], and have also been shown to have an important role in buffering steady-state populations [79], and in sensitizing regulatory mechanisms related to metabolism [3,127]. However, despite their apparent importance to biochemists, the only hardness results we are aware of for enumerating substrate cycles are an indirect consequence of proofs regarding the NP-hardness of counting all simple (not necessarily induced) cycles

in digraphs (see e.g. ("Theorem 17.4"; pp. 343 - 344) of Arora & Barak [9]) via reduction from the NP-complete Hamiltonian cycle decision problem [92], where digraphs can correspond to reaction-centric graphs where all reactions are irreversible, and the recent proof due to Yamamoto [191] that the problem of counting simple (not necessarily induced) cycles in arbitrary undirected graphs is $\#P$-complete as well as polynomial-time inapproximable unless $NP = RP$, where undirected graphs can correspond to reaction-centric graphs where all reactions are reversible. Moreover, the only complexity theoretic results we are aware of for enumerating EFMs is that counting EFMs in substrate-centric graphs, where metabolites are represented as vertices and reactions as edges, is $\#P$-complete [1] via reduction from the $\#P$-complete problem of counting perfect matchings [180], and that no *polynomial total-time* algorithm (see Johnson et. al. [89] for a definition of this term) can exist for enumerating EFMs containing a specified reaction unless $P = NP$ [2].

However, a question arises as to the practical relevance of these hardness results which were proven for general graphs. Consider substrate-centric graphs, where both metabolites are encoded as vertices which are connected by edges corresponding to enzymes. We can note that these graphs have been shown to have e.g. bounded diameters and to exhibit $P(k) \propto k^{-\gamma}$ power law scaling for their vertex degree distributions in a range of archaea, bacteria, and eukaryotic organisms [86]. It could therefore arguably be the case that the aforementioned hardness results are simply the consequence of pathological families or classes of graphs that have little relevance to actual metabolic networks.

Motivated by these concerns, we attempt to sharpen known hardness results for enumerating substrate cycles to apply to some family of hypothetical reaction-centric graphs, F, that is simultaneously more constrained than any realistic large scale metabolic network and "physical" in the sense that one would expect the constraints defining F to be satisfied individually by subgraphs composed of some reasonable fraction of enzymes in any given metabolic pathway. After examining the metabolic pathways in the Kyoto Encyclopedia of Genes and Genomes (KEGG) database [90], we decided that some form of vertex degree uniformity, vertex or edge connectivity, bipartiteness, and planarity constraints would reasonably meet these dual criteria (see e.g. (Figure 2.1)). We then proceeded to prove that the problem of counting substrate cycles remains $\#P$-complete ($\implies$ enumeration is $\#P$-hard) on reaction-centric graphs belonging to any superclass of the highly restricted class of cubic weakly-3-connected bipartite planar digraphs (Theorem 1), and on any superclass of cubic 3-connected bipartite planar undirected graphs where the problem of counting Hamiltonian cycles is $\#P$-complete (Theorem 2).

2.3 Hardness results for counting cycles on undirected graphs and digraphs

Theorem 1 *The problem of counting directed simple (not necessarily induced) cycles on cubic weakly-3-connected bipartite planar digraphs having vertex in-degree and out-degree at most two,* $\#DirectedCycle(C3BP :: In_2Out_2)$*, is* $\#P$*-complete under many-one right-bit-shift reductions.*

We note that (Theorem 1) is equivalent to the statement that counting substrate cycles in reaction-centric graphs corresponding to cubic 3-connected bipartite planar digraphs ($\implies$ all reactions are irreversible) is $\#P$-complete under many-one right-bit-shift reductions.

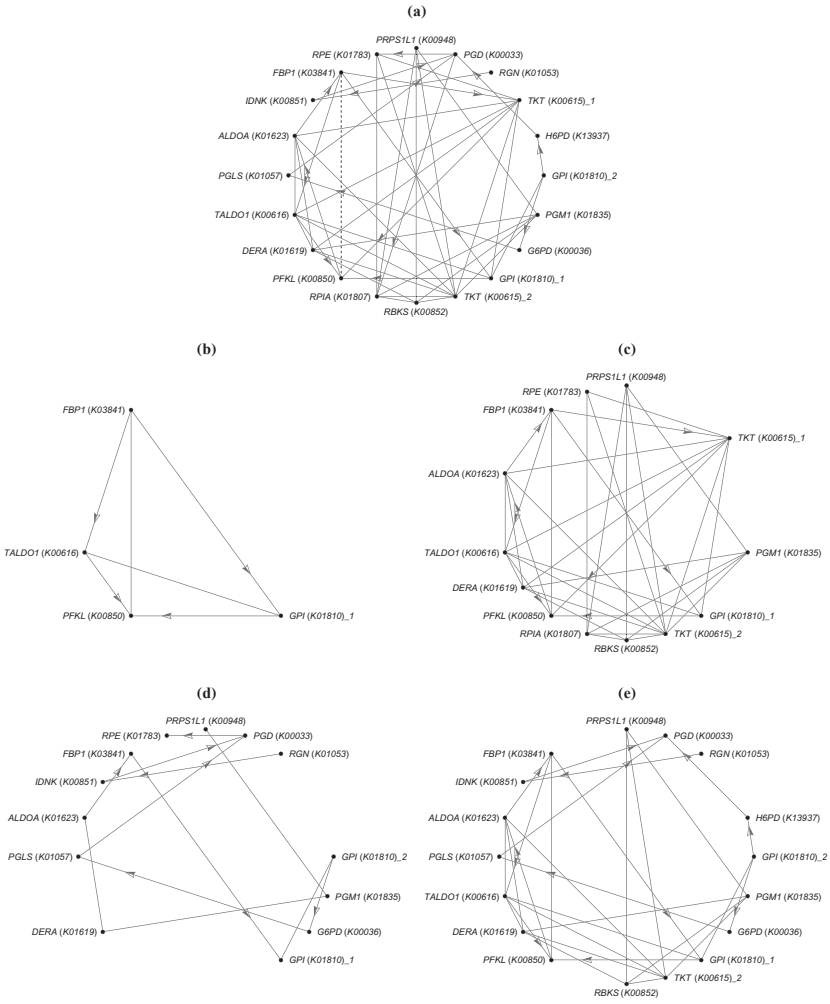


Figure 2.1: In (a) we show a reaction-centric graph (20 enzymes), simplified by not considering {ATP, $NADP^+$, NADPH} or other cofactors, for the largest set of *Homo sapiens* enzymes in the Kyoto Encyclopedia of Genes and Genomes (KEGG) database [90] (Release 81.0, January 1st, 2017) assigned to the pentose phosphate pathway, carrying out distinct reactions, and allowing for a weakly-connected reaction-centric graph. The names of enzymes are followed by their KEGG Orthology (KO) numbers, and may have a '_1' or '_2' postfix to indicate reactions involving distinct metabolite inputs. Edges that are directed (undirected) correspond to irreversible (reversible) flows of metabolites. Undirected edges constitute minimal substate cycles, e.g., the (dashed) undirected edge in (a) corresponds to a substrate cycle for a pair of enzymes that, if unchecked and provided a pool of ATP for to reduce to ADP, will continuously interconvert fructose-6-phosphate and fructose-1,6-bisphosphate, releasing heat in the process. In (a - e) we show maximal subgraphs (in terms of vertex / enzyme counts) of the reaction-centric graph from (a), which are: (b) cubic (4 enzymes); (c) weakly-3-vertex-connected (13 enzymes); (d) bipartite (13 enzymes); and (e) planar (17 enzymes).

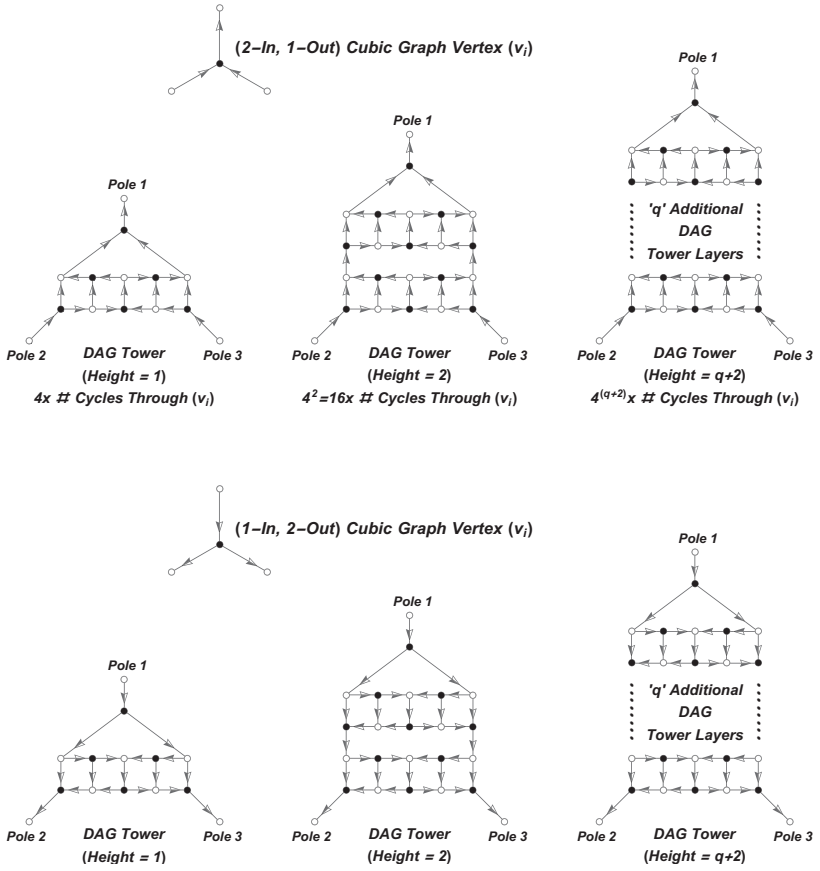


Figure 2.2: Scheme for generating a family of cubic 3-connected bipartite planar Directed Acyclic Graph (DAG) *vertex substitution gadgets* where the *height* q of the gadget can be specified as desired. If the appropriate (in terms of matching in-degree and out-degree counts) height q gadget is substituted in place of all vertices in some cubic 3-connected bipartite planar graph G, creating a new cubic 3-connected bipartite planar graph H, this will multiply the number of length L cycles in G by the factor $(4^q)^L$.

Claim 1.1 $\#DirectedCycle(C3BP :: In_2Out_2) \in \#P$.

Consider a nondeterministic Turing machine that accepts an input string encoding a cubic 3-connected bipartite planar digraph G and a directed edge-wise (or vertex-wise) path P in G, and accepts this input if and only if P is a simple directed cycle in G (i.e. a directed cycle which visits any given vertex at most once). Here, we define $\#DirectedCycle(C3BP :: In_2Out_2)$ as an integer function problem $f : \Sigma^* \longrightarrow \mathbb{N}$, where one is tasked with determining the number of accepting branches $f(x)$ for such a nondeterministic Turing machine $\implies \#DirectedCycle(C3BP :: In_2Out_2) \in \#P$.

Claim 1.2 $\#DirectedCycle(C3BP :: In_2Out_2)$ *is* $\#P$*-hard.*

We proceed with a strategy of creating some digraph H in polynomial time from an arbitrary simple cubic 3-connected bipartite planar digraph having in-degree and out-degree at most two, G, of order n, where via a gadget substitution scheme we amplify the number of simple length L cycles in the graph by a factor $(2^m)^L$ allowing us to calculate the number of Hamiltonian cycles via a simple integer division operation. However, in lieu of substituting this gadget in place of the edges of G in the manner of the proof for ("Theorem 17.4"; pp. 343 - 344) in Arora & Barak [9] and in the manner of Yamamoto [191], which precludes H from being 3-connected, we instead replace each of the n vertices in G with the gadget in such a manner as to ensure that H is a cubic weakly-3-connected bipartite planar digraph having in-degree and out-degree at most two iff G is cubic weakly-3-connected bipartite planar digraph having in-degree and out-degree at most two. To do so we define a surgery for cubic graphs which we denote *tripole substitution* wherein a selected vertex v_i in a cubic graph G is substituted with a target graph H having three degree $\rho(v_i) = 1$ *pole vertices* that are identified (via some bijection if necessary) as the vertices connected to the v_i vertex in G via an edge of identical orientation.

In (Figure 2.2) we illustrate the scheme for our *height* (using this term to satisfy the visual metaphor of a tower) q gadget used for tripole substitution at all vertices of G to create H. We can now make the following two observations: (1) that the gadget is a Directed Acyclic Graph (DAG), which implies that there will be no gadget internal cycles; (2) that the substitution of a height q gadget at every vertex in G to create H will amplify the number of simple length L cycles in G by the factor $(4^q)^L$. We set $q = \lceil n * log_4(n) \rceil$, which implies a rate of growth for q of $\mathcal{O}(n * ln(n))$, and which correspondingly implies that we can construct H in time polynomial in the number of vertices in G. Next, we note there can be at most $n^{(n-1)}$ directed cycles of length at most $(n-1)$ in a digraph as this term corresponds to the number of distinct length $(n-1)$ strings that can be created from an alphabet Σ of size $|\Sigma| = n$. This then implies that if the graph G is non-Hamiltonian we have an upperbound of $(4^q)^{(n-1)} * n^{(n-1)} = (4^{\lceil (n*log_4(n)) \rceil})^{(n-1)} * n^{(n-1)}$ directed cycles. However, if the graph G has at least one Hamiltonian cycle, H has at least $(4^q)^n = (4^{\lceil (n*log_4(n)) \rceil})^n$ directed cycles. Here, if we remove the ceiling function in the expression for q, the lowerbound number of cycles for Hamiltonian G is exactly a factor of n larger than the upperbound R for the number of cycles for non-Hamiltonian G, and with the ceiling function, at least a factor of n larger since $\forall \{n \in \mathbb{R}_{>0},\ q \in \mathbb{R}_{\geq 0}\}$ we have that $sgn\left[\frac{\partial}{\partial q}\left(\frac{(4^q)^n}{(4^q)^{(n-1)} * n^{(n-1)}}\right) = (4^q * n^{(1-n)} * ln(4))\right] = (+1)$.

Putting everything together, we have that the number of Hamiltonian cycles in G is equal to $\lfloor\left(\frac{(\#\ Directed\ Cycles\ in\ H)}{(4^{\lceil (n*log_4(n)) \rceil})^n}\right)\rfloor = \lfloor\left(\frac{(\#\ Directed\ Cycles\ in\ H)}{2^{(2n*\lceil (n*log_4(n)) \rceil)}}\right)\rfloor$. As this closed-form expression is a polynomial-time computable integer division operation

consistent with the requirements for a right-bit-shift reduction [118], and as the problem of counting Hamiltonian cycles on cubic 3-connected bipartite planar digraphs is $\#P$-complete under parsimonious reductions (though unremarked upon by Plesník, his construction in [146] is odd-cycle-free, and moreover becomes a parsimonious counting reduction from a variant of $\#3SAT$, and transitively $\#SAT$, if one reorients one edge in his "OR" gadget (see **(Chapter 3)**)), we have that $\#DirectedCycle(C3BP :: In_2Out_2)$ is $\#P$-hard.

Claim 1.3 *$\#DirectedCycle(C3BP :: In_2Out_2)$ is $\#P$-complete under right-bit-shift reductions.*

As we have that the counting problem is in $\#P$ (Claim 1.1), is $\#P$-hard (Claim 1.2), and because the reduction described in the proof argument for (Claim 1.2) is a right-bit-shift reduction [118], by definition we have that the counting problem is $\#P$-complete under right-bit-shift reductions.

Claim 1.4 *Unless we have that $NP = RP$, there does not exist a $FPRAS$ for $\#DirectedCycle(C3BP :: In_2Out_2)$.*

Let G be an arbitrary cubic 3-connected bipartite planar digraph of order n, let H be the gadget substituted graph constructed as described in (Claim 1.2), and let R represent an upperbound for the fraction of directed cycles in H not corresponding to Hamiltonian cycles. Proceeding now in much the same manner as Yamamoto [191], from the argument in (Claim 1.2) we have that $R = (\frac{(4^{\lceil (n*log_4(n))\rceil})^{(n-1)} * n^{(n-1)}}{(4^{\lceil (n*log_4(n))\rceil})^n}) \leq \frac{1}{n} \implies (\frac{(\#\ Directed\ Cycles\ in\ H)}{(4^{\lceil (n*log_4(n))\rceil})^n}) \leq \lfloor (\frac{(\#\ Directed\ Cycles\ in\ H)}{(4^{\lceil (n*log_4(n))\rceil})^n}) \rfloor + \frac{1}{4}$ for $n \geq 4$. As the Hamiltonian cycle decision problem is NP-complete for cubic 3-connected bipartite planar digraphs per (Claim 1.2), by the argument at the end of "Theorem 3" in Dyer et. al. [51], we have that $\#DirectedCycle(C3BP :: In_2Out_2) \equiv_{AP} \#SAT$. Finally, due to Valiant and Vazirani [183], we have that no $FPRAS$ can exist for approximating $\#SAT$ unless $NP = RP$, and that this holds for all counting problems in $\#P$ that are AP-interreducible in polynomial time. Therefore, there can be no $FPRAS$ for $\#DirectedCycle(C3BP :: In_2Out_2)$ unless $NP = RP$.

Theorem 2 *The problem of counting undirected simple (not necessarily induced) cycles on any superclass (superset) of cubic 3-connected bipartite planar graphs where the problem of counting Hamiltonian cycles is $\#P$-complete,* which we denote *$\#UndirectedCycle(Superclass\ C3BP :: \#HC = \#P)$, is $\#P$-complete under many-one counting reductions.*

We note that (Theorem 2) is equivalent to the statement that counting substrate cycles in reaction-centric graphs corresponding to any superclass of cubic 3-connected bipartite planar undirected graphs ($\implies$ all reactions are reversible) is $\#P$-complete whenever the problem of counting all Hamiltonian cycles is $\#P$-complete under many-one counting reductions.

Claim 2.1 *$\#UndirectedCycle(Superclass\ C3BP :: \#HC = \#P) \in \#P$.*

This follows straightforwardly from the method of argument in (Claim 1.1).

Claim 2.2 *$\#UndirectedCycle(Superclass\ C3BP :: \#HC = \#P)$ is $\#P$-hard.*

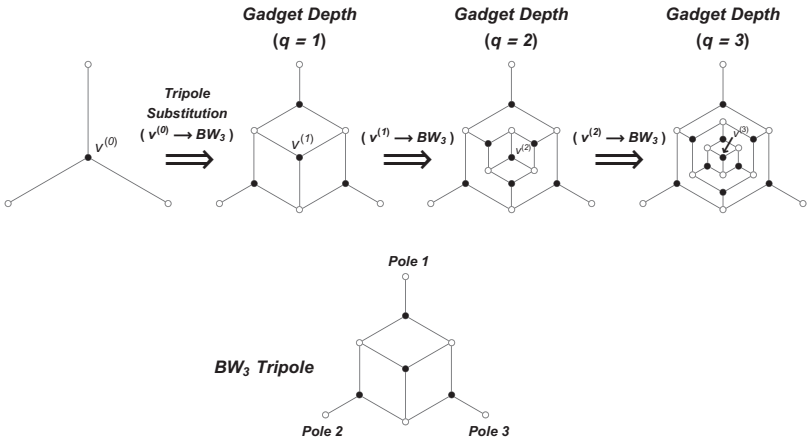


Figure 2.3: Let a "BW_3 tripole" be a BW_3 graph modified to have pole vertices {Pole 1, Pole 2, Pole 3} joined by an edge to the degree $\rho(v_i) = 2$ vertices in the original BW_3 graph. Here we show a recursive BW_3 tripole substitution scheme for generating a family of undirected cubic 3-connected bipartite planar *vertex substitution gadgets* where the *depth* q of the gadget can be specified as desired. If a depth q gadget is substituted in place of all vertices in some undirected cubic 3-connected bipartite planar graph G, creating a new cubic 3-connected bipartite planar graph H, this will multiply the number of length L cycles in G by $\Psi^L = (\frac{1}{2}(3 * 5^q - 1))^L$ and generate $n * \Phi_{cycles} = (n * (\frac{1}{4}(9 * 5^q - 8q - 9)))$ total gadget internal undirected cycles.

Let G be an arbitrary simple undirected cubic 3-connected bipartite planar graph of order n. In (Figure 2.3) we illustrate the scheme for our *depth* q gadget used for tripole substitution at all vertices of G to create H. We can now make the following three observations: (1) that H is an undirected cubic graph that is 3-connected iff G is 3-connected, bipartite iff G is bipartite, and planar iff G is planar; (2) that $n = |V_G|$ depth q gadgets will imply the existence of some number of undirected cycles internal to the set of gadgets; (3) that the substitution of a height q gadget at every vertex in G to create H will amplify the number of simple length L cycles in G by some *gadget amplification factor* denoted Ψ. Specifically, for cycles traversing a depth q gadget, we have the recurrence relation $a_{(q+1)} = 2 + 5 * a_q$ where $a_{(1)} = 7 \implies a_q = \Psi = \frac{1}{2}(3 * 5^q - 1)$, and for the number of cycles internal to a depth q gadget, we have the recurrence relation $b_{(q+1)} = (1 + 3 * 2 * \Psi + b_q) = (1 + 6 * \frac{1}{2}(3 * 5^q - 1) + b_q)$ where $b_{(1)} = 7 \implies b_q = \frac{1}{4}(9 * 5^q - 8q - 9)$. We set $q = \lceil log_5(\frac{1}{3}(1 + 2 * n^n))\rceil$, which implies a rate of growth for q of $\mathcal{O}(ln(n^n)) = \mathcal{O}(n * ln(n))$, and which correspondingly implies that we can construct H in time polynomial in the order of G. Noting as in (Claim 1.3) that there can be at most $n^{(n-1)}$ cycles of length at most $(n-1)$ in a graph, we have that if the graph G is non-Hamiltonian, post-amplification there are at most $((\frac{1}{2}(3 * 5^q - 1))^{(n-1)} * n^{(n-1)} + n * (\frac{1}{4}(9 * 5^q - 8q - 9)))$ undirected cycles. However, if the graph G has at least one Hamiltonian cycle, H has at least $(\frac{1}{2}(3 * 5^q - 1))^n + n * (\frac{1}{4}(9 * 5^q - 8q - 9))$ undirected cycles. Here, if we remove the ceiling function in the expression for q, the lowerbound number of cycles for Hamiltonian G is exactly a factor of n larger than the upperbound R for the number of cycles for non-Hamiltonian G, and with the ceiling function, at least a factor of n larger since $\forall\{n \in \mathbb{R}_{>0},\ q \in \mathbb{R}_{\geq 0}\}$ we have that:

(Exp. 2.1)

$$sgn\left[\frac{\partial}{\partial q}\left(\frac{(\frac{1}{2}(3 * 5^q - 1))^n + n * (\frac{1}{4}(9 * 5^q - 8q - 9))}{(\frac{1}{2}(3 * 5^q - 1))^{(n-1)} * n^{(n-1)} + n * (\frac{1}{4}(9 * 5^q - 8q - 9))}\right)\right] =$$

$$sgn\left[\frac{\partial}{\partial q}\left(\frac{(\frac{1}{2}(3 * 5^q - 1))^n}{(\frac{1}{2}(3 * 5^q - 1))^{(n-1)} * n^{(n-1)}}\right) = \left(\frac{3}{2}(5^q * n^{(1-n)} * ln(5))\right)\right] = (+1)$$

Now, setting $q \geq log_5(\frac{1}{3}(1 + 2 * n^n))$, letting $n * \Phi_{cycles} = (n * (\frac{1}{4}(9 * 5^q - 8q - 9)))$ be the number of cycles internal to the vertex substution gadgets, and letting $\Omega = (\frac{1}{2}(3 * 5^q - 1))^n$ be the number of undirected cycles in H per Hamiltonian cycle in G, we can write the relation:

(Exp. 2.2)

$$(\#\ Hamiltonian\ Cycles\ in\ G) = \lfloor\left(\frac{(\#\ Undirected\ Cycles\ in\ H) - n * \Phi_{cycles}}{\Omega}\right)\rfloor =$$

$$\lfloor\left(\frac{(\#\ Undirected\ Cycles\ in\ H) - (\frac{3n(n^n-1)}{2}) + 2n * log_5(\frac{2n^n+1}{3})}{(n^n)^n}\right)\rfloor$$

As this closed-form expression is polynomial-time computable, we have that $\#UndirectedCycle(Superclass\ C3BP :: \#HC = \#P)$ is $\#P$-hard.

We briefly remark that, letting R represent an upperbound for the fraction of undirected cycles in H that are not Hamiltonian cycles, and again setting $q \geq log_5(\frac{1}{3}(1 + 2 * n^n))$, we have the bound:

(Exp. 2.3)

$$R = \left(\frac{(\frac{1}{2}(3 * 5^q - 1))^{(n-1)} * n^{(n-1)} + n * (\frac{1}{4}(9 * 5^q - 8q - 9))}{(\frac{1}{2}(3 * 5^q - 1))^n} \right) =$$

$$\left(\frac{(\frac{(n^n)^n}{n}) + (\frac{3n^{(n+1)}}{2}) - (\frac{3n}{2}) - 2n * log_5(\frac{2n^n+1}{3})}{(n^n)^n} \right) \approx \left(\frac{(\frac{(n^n)^n}{n})}{(n^n)^n} \right) = \frac{1}{n}$$

Thus, letting $||x||$ be the nearest-integer function, for $n > 2$ (which holds without the approximation shown in (Exp. 2.3)) we have that the number of Hamiltonian cycles in G is equal to $|| \left(\frac{(\#\ Undirected\ Cycles\ in\ H)}{(n^n)^n} \right) ||$.

Claim 2.3 *$\#UndirectedCycle(Superclass\ C3BP :: \#HC = \#P)$ is $\#P$-complete under many-one counting reductions.*

As we have that the counting problem is in $\#P$ (Claim 2.1) and is $\#P$-hard (Claim 2.2), by definition we have that the counting problem is $\#P$-complete.

Claim 2.4 *Unless we have that $NP = RP$, there does not exist a $FPRAS$ for $\#UndirectedCycle(Superclass\ C3BP :: \#HC = \#P)$ whenever we have that the Hamiltonian cycle decision problem is NP-complete.*

This claim follows straightforwardly from (Exp. 2.2), (Exp. 2.3), and the method of argument in (Claim 1.4).

Corollary 2.1 *The problem of counting all undirected cycles on cubic 2-connected planar graphs, $\#UndirectedCycle(C2P)$, is $\#P$-complete.*

This corollary follows from the proof argument for (Theorem 2) and the proof by Liśkiewicz, Ogihara, and Toda [118] that the problem of counting all Hamiltonian cycles in an input graph is $\#P$-complete under right-bit-shift reductions on cubic 2-connected planar graphs.

Corollary 2.2 *The problem of counting all undirected cycles on cubic 3-connected planar graphs ($\#UndirectedCycle(C3P)$), and on cubic 2-connected bipartite planar graphs ($\#UndirectedCycle(C2BP)$), is NP-hard.*

This corollary follows from the proof argument for (Theorem 2) and the NP-completeness of the Hamiltonian cycle decision problem on cubic 3-connected planar graphs [67], cubic 3-connected bipartite graphs [4], and cubic 2-connected bipartite planar graphs [4].

2.4 Chapter concluding remarks

We note that substrate cycles in reaction-centric graphs corresponding to closed systems (or, more appropriately, approximations thereof) such as individual cells, organelles like the mitochondria, or assuming some appropriate generalization to arbitrary chemical reaction networks, the upper atmospheres of gas giants, are equivalent to EFMs (see for example the proof argument for "Theorem 6" in Acuña et. al. [1]). Furthermore, if all reactions are irreversible, there is also a bijective correspondence between substrate cycles and extreme pathways. Therefore, we have that our hardness results concerning substrate cycles have the same implications for counting and enumerating EFMs and, at the limit where all reactions are irreversible, extreme pathways in metabolic networks with the specified topological restrictions.

Chapter 3

Counting Simple Cycles and Circuits on Undirected Planar or Bipartite k-Regular Graphs

This chapter is in submission to a peer-reviewed journal. It will be published in the UT Repository after either the content of the chapter appears in a journal, or at the latest, by October of 2022.

Chapter 4

Counting and Approximately Counting Hamiltonian Cycles and Paths on a Class of Cubic Bipartite Polyhedral Graphs Conjectured to be Hamiltonian

This chapter is in submission to a peer-reviewed journal. It will be published in the UT Repository after either the content of the chapter appears in a journal, or at the latest, by October of 2022.

Chapter 5

The Hardness of Deciding the Parity Bit for the Number of Cycles on Subclasses of Directed Cubic and Undirected Subcubic Bipartite Planar Graphs

This chapter is in submission to a peer-reviewed journal. It will be published in the UT Repository after either the content of the chapter appears in a journal, or at the latest, by October of 2022.

Chapter 6

A Worst Case Analysis at the Level of Fundamental Nucleic Acid Chemistry for a Photocrosslinking-Based Gene Expression Profiling Method for the Quantitation of Ultradilute Ribonucleic Acids

6.1 Introductory discussion for DigiTag, GEP-DEAN, and photo-DEAN: embedding RNA expression into a library of uniformly amplifiable deoxyribonucleic acid polymers

At a sufficiently high level of abstraction, the DigiTag [128], DigiTag 2 [129], and GEP-DEAN [74] ("GEP-DEAN" = Gene Expression Profiling by DCN Encoding based Analysis, "DCN" = DNA Coded Numbers) assays are all variations on a "molecular version" of an injective function $f : S \rightarrowtail Z$, where S is a multiset of "signals" corresponding to polynucleotide strings of mRNA or cDNA (potentially having lengths running into the tens of kilobases), and where f maps each RNA signal to a specific integer label corresponding to a short deoxyribonucleic acid (DNA) oligonucleotide denoted a DNA Coded Number (DCN) tag.

More specifically, in each of these assays f consists of a procedure wherein a set of $\approx$ 15nt DNA probe pairs search for distinct $\approx$ 30nt target sequences on each of the $s_i \in S$, where target sequences are carefully selected to optimize for having optimal thermodynamic properties as well as uniqueness in a given mRNA or cDNA population. If two probes in a pair find the same $s_i \in S$, the $5'$ termini (resp. $3'$ termini) of the probe hybridized upstream (resp. downstream) of the other can be joined together via the application of some ligase enzyme of choice. Now, if the upstream probe is designed to encode a special DNA Coded Number (DCN) sequence on its $3'$ end, and the downstream probe encodes e.g. - a biotin (in the case of the original DigiTag [128] assay), a primer to directly allow for PCR amplification (in the case of the DigiTag 2 [129] assay), or a hybridizable capture sequence (in the case of the GEP-DEAN [74] assay) - this allows one to selectively isolate only the DCN "signal" sequences that participated in a successfully ligated probe pair complexes. Thus, we have that the procedure f has "embedded" (and we hasten to note that we use the term "embedded" in an informal sense) the signals in the multiset S into a library of known and (with appropriate sequence design) quantitatively amplifiable DCN oligonucleotides corresponding to a set of integer labels having distinct values for each of the elements in S.

At a lower level of abstraction, we can note that the selection of the Taq NAD^+ dependent DNA ligase (from *Thermus aquaticus*) for DigiTag [128] and DigiTag 2 [129] evidences the focus of these earlier studies on cDNA SNP detection and quantification. Consider here that this family of polymerases can exhibit on mismatch sensitivities, i.e. correct Watson-Crick hybridization stringencies, around two orders of magnitude beyond the fidelity of T4 DNA ligase [173],[h1] (not to discredit T4 DNA ligase's mismatch discrimination abilities [71]). Moreover, we have that the highest mismatch stringency for Taq is achieved if the mismatch is on the $3'$-OH side of the nick [15],[s7], and we can see that this is reflected in the encoding of the SNP Watson-Crick complement on $3'$ side of the nick between the probe pairs for the DigiTag [128] and DigiTag 2 [129] assays. However, for either DigiTag [128] or DigiTag 2 [129], an enzyme like Tth DNA ligase (from *Thermus thermophiles*) may have provided the high sensitivity of this enzyme to distal duplex fraying or mismatches, e.g. where mismatches 7 nts to 8 nts $5'$ of a nick region can mostly abrogate ligation [148]. The selection of the Taq enzyme for GEP-DEAN, where cDNA quantification rather than SNP detection appeared to be the focus of the study, can be understood from this context as arising from a desire for consistency with earlier methods.

We now shift our focus to Yokomori et. al.'s photo-DEAN [193–196] platform, which will be the primary focus for this chapter. Yokomori et. al.'s photo-DEAN [193–196] is a novel variant of the DigiTag [128], DigiTag 2 [129], and GEP-DEAN [74] DCN-based methods wherein Yokomori et. al. [193–196] attempts direct

detection and quantification of mRNA with DNA probes (as opposed to the reverse transcribed cDNA used by photo-DEAN's predecessors [74, 128, 129]), and in place of enzymatic ligation with a *Thermus* family enzyme (e.g. Taq as used in the aforementioned previous assays) [74, 128, 129], Yokomori et. al. [193–196] employs Fujimoto et. al.'s 5-R-vinyluracil ^{RV}U photoligation chemistry (where "R" = "CN", "COOH", etc.; "R" = "COOH" in the case of photo-DEAN [62–64, 121, 126, 131–135, 193–196, 199, 200],[s1].

Following photo-ligation, a biotin microbead-based affinity capture step (similar that used in the original DigiTag [128] and GEP-DEAN [74] methods) is then used to isolate DCN sequences on photo-ligated probes, and these DCN sequences are later amplified, fluorescently tagged, and annealed to a microarray for quantification. Concerning the initial bead purification step for isolating photoligated probe pairs, in light of Fujimoto et. al.'s demonstration of Ex Taq polymerase PCR amplification of an oligonucleotide with LHS and RHS primers joined by a ^{CV}U crosslink [133], there does not appear to be fundamental reason for this choice. In other words, it may also have been possible for Yokomori et. al. to directly PCR amplifying ^{CV}U photo-ligation products akin to the direct PCR amplification of properly ligated DCN-carrying sequences in the DigiTag 2 [129] assay.

On the one hand, the photo-DEAN [193–196] method of Yokomori et. al. appears to sacrifice the enzymatic fidelity check offered by *Thermus* NAD^+ dependent DNA ligase enzymes like Taq [96] or Tth [148] in exchange for a ^{CV}U ligation fidelity of $((ligation\ rate \mid WC\ hybridization)\ /\ (error\ rate)) \approx 10 - 100$, which can be contrasted with the $\approx 10^3$ fidelities achieved with the *Thermus* AK16D ligase [173]. Another sacrifice with this chemical ligation approach is that, beyond consideration of hybridization thermodynamics, mismatch discrimination will only occur in the immediate vicinity of a photo-ligation junction [63, 199, 200], which can be contrasted with the ability to achieve similar $\approx 10-100$ fidelities a distance 8 nts $5'$ of the point of ligation with the *Thermus* DNA ligase Tth [148]. Quickly, we arrived at our $\approx 10 - 100$ value for ^{CV}U ligation fidelity by first noting that the estimate seemed to correspond to the error rate for the ^{CV}U ligation of two 6-mers over an RNA template (see "Figure 4.6" in ref. [199]), and by taking into consideration Fujimoto et. al.'s observation that ^{CV}U will crosslink to a +1 cytidine in an overhang with a probability of $\approx 12\%$ [63] (though this could increase or decrease quite a bit depending on local stacking interactions at the nick site). And while it's true that almost negligible error rates were observed in Fujimoto et. al.'s study on the detection of indica and japonica rice strain SNPs [200], we point out that this is much more likely due to the fact that short 9-mer oligonucleotides were utilized as hybridization probes, making their ability to discriminate mismatches more a function of thermodynamics than the chemistry of the actual ligation method [h2].

On the other hand, Yokomori et. al.'s method [193–196] of ^{CV}U based photoligation and mRNA quantification has a potential very significant implication. While it's certainly true that if GEP-DEAN sensitivities and timescales can be achieved, photo-DEAN will allow for the detection and quantitation of cDNA, mRNA, or genomic DNA with rather impressive ≈ 18 zMol sensitivity, implying that target concentrations only amounting to $N_A * 18 * 10^{-21} \approx 6.02214129 * 10^2 3 mol^{-1} \times 18 * 10^{-21} \approx 1.1 * 10^4$ molecules will sit above the technique's noise floor. However, what is truly impressive about photo-DEAN is that it promises to allow us to do detect and quantify nucleic acids with this sensitivity strictly where AND when we shine UVA ($\approx$ 366 nm to $\approx$ 405 nm) light. Regarding the proper aiming of a diffraction limited beam of blue light, one can do with e.g. a laser spot illuminator operating at $\approx$ 20 Hz [109] or (c.2) a Digital Micromirror Display (DMD) array

with 10^5 to 10^6 or so individually programmable mirrors that allow for massively parallel spot illumination [109, 171]. Concerning uses for a DMD, one could, for example, simultaneously light up the mitochondria or some other organelle / foci / etc. in hundreds of thousands of cells in a fixed tissue sample (or perhaps a live tissue sample if photo-DEAN probes can be transfected into the cells via e.g. liposomes) [h3].

To the best of our knowledge, there is no precedent whatsoever for this kind of capability. For this reason, we focus our analysis in this chapter on Yokomori et. al.'s novel photo-DEAN method.

6.2 Time versus the number of Brownian "searcher" particles required to find a small number of target molecules: the unavoidable tradeoff between target dilution and the spontaneous generation of false-positive signals

For our analysis in this section, we primarily consider challenges associated with matching the sensitivity of Gotoh et. al.'s GEP-DEAN [74] assay using Yokomori et. al.'s photo-DEAN [193–196] method. And this is no modest objective as it requires that one achieve an extremely impressive lowerbound assay detection limit of ≈ 18 zmol or $(18 * 10^{-21} * N_A) \approx 10840$ molecules ($N_A \approx 6.02214129 * 10^{23} mol^{-1}$ being Avogadro's constant), in a $\approx 29.5\ \mu L$ reaction volume, which implies that the target sequences are at a concentration of ≈ 0.61 femtoMolar (fM) [76].

To begin, we note that for any method of detecting and quantitating ultralow nucleic acid target concentrations (e.g. a $\approx 0.61\ fM$ concentration of target molecules), there will always be a tradeoff between the concentration of Brownian "searcher" particles (i.e. molecular probes) and the time one is willing to afford for the detection step of the assay to reach a desired threshold degree of completion. To understand this point, consider that the time it takes to find a molecular target via diffusion is directly proportional to the number of molecules N searching for the target via Brownian motion.

More specifically, we can cite Condamin et. al.'s [39] derivation of molecular search time via simple diffusion in three-dimensions (more formally denoted the three-dimensional Mean First Passage Time (MFPT)), which is as follows:

(Exp. 6.1)

$$\langle \tau_{3D} \rangle \approx N^{-1} \left(\frac{V}{4\pi D(\frac{1}{a} - \frac{1}{R})} \right)$$

In the above expression: 'N' is the number of Brownian "searcher" particles (i.e. molecular probes); 'V' is the volume of solution in which the "searcher" particles diffuse (their targets are assumed to be immobilized in place); 'D' is the approximate diffusion coefficient of the Brownian "searcher" particles, which is assumed to be greater than or equal to that of their targets (allowing us to utilize the aforementioned target immobilization approximation); 'a' is the radius of the smallest sphere of sufficient size to, allowing arbitrary positioning of the sphere, encompass the Brownian "searcher" particles or their targets (imagine 'a' as a sort of one-dimensional analogue for the "barn" unit for area in classical nuclear physics); and finally, 'R' is the approximate initial distance between the Brownian "searcher" particle(s) and the target, i.e. the average molecular probe-to-target distance.

To provide an example for the use of Condamin et. al.'s [39] expression (i.e. Exp. 1), the MFPT for a lone Alexa 350 molecule colliding with a point-like target in a cell of volume $V \approx 4/3\pi(5\ \mu m)^3 \approx 0.524\ pL$, can be estimated by noting: (1) that Alexa 350 has dimensions of roughly $(13.0A \times 5.2A \times 3.2A)$ [187], allowing us to set $a \approx 5A$; (2) that the diffusion coefficient for Alexa 350 in water at $\approx 25^{\circ}C$ is $D_{(Alexa350, H_2O)} \approx 570\ \mu m^2/s$ [130] (or $D_{(Alexa\ 350,\ XO_{cyto})} \approx 185\ \mu m^2/s$ in *Xenopus* oocyte cytoplasm) [130]; and (3) that the mean distance between two points placed randomly in the unit ball S^2 is going be 36/35 (recall that this value is $4/\pi$ in the more familiar S^1 case) allowing us to set $R \approx 36/35(5\ \mu m) \approx (36/7)\ \mu m$. These values yield estimated molecular search times at $\approx 25^{\circ}C$ of $\approx 146\ s \approx 2.44\ min$ in water (or $\approx 450\ s \approx 7.51\ min$ in *Xenopus* oocyte cytoplasm) for Alexa 350 to find a point-like target in a volume approximately

$\approx (4/3\pi(5\ \mu m)^3)$ / $(13.0\ \mathring{A}\ \times\ 5.2\ \mathring{A}\ \times\ 3.2\ \mathring{A}) \approx 10^{12}$ fold larger than that of its own convex hull.

To write down some actual experimental parameters for molecular search processes relevant to photo-DEAN [193–196], we note that Gotoh et. al. [74] added $\approx 10 fM$ of each of his $5'$ and $3'$ Molecular Translation Table (MTT) probes to an aforementioned $\approx 29.5\ \mu L$ reaction volume (implying that each probe was at $\approx 339\ pM$ concentration), to achieve the quoted $\approx 0.61\ fM$ sensitivity value for the GEP-DEAN [74] assay. To briefly clarify, here "MTT" sequences are just the probes that hunt for a cDNA target and are ligated over this target to allow for capture and amplification of a DCN [74, 128, 129, 193–196] sequence encoded on one of the probes in the "MTT pair" [74]. Now, in terms of the required time for the MTT probes to find their cDNA targets via Brownian motion, probe hybridization was carried out by first raising the solution temperature to $\approx 95^{\circ}C$ for $\approx 3\ min \approx 180s$, then by ramping down the temperature at a rate of $\approx 0.1^{\circ}C$ per second to $\approx 45^{\circ}C$ (giving a total anneal time $\approx 8.33\ min \approx 500\ s$) where a ligation was finally performed for $\approx$ 1h using Taq ligase [76]. If we decide not to consider the initial denaturation ($\approx 3\ min$ at $\approx 95^{\circ}C$) and ligation ($\approx 1\ h$ at $\approx 45^{\circ}C$) steps as part of the molecular "search time", this implies that it took $\approx 8.33\ min \approx 500\ s$ for a $(10\ fmol/\ 18\ zmol) \approx 5.6 * 10^5$ fold excess of Brownian "searcher" particles to find what we estimate to be $>$ 50% of the $N \approx 10840$ targets at a concentration of ≈ 0.61 fM.

To make our point regarding the aforementioned tradeoff between the Brownian "searcher" particle concentration and reaction completion time, here assuming that probe $\langle\rangle$ target hybridization is a strongly thermodynamically downhill process, which is not precisely clear when one considers target secondary structure (of particular concern if one hybridizes DNA probes to RNA targets $\approx 30(+)$ nts in length), it would take $(500\ s) * (10\ fmol\ /\ 18\ zmol) \approx 8.8\ yrs$ to achieve the same detection rates if probes and their targets were incubated together at a 1-to-1 concentration ratio.

Another equally important point is that, no matter how one chooses to strike a balance between the concentration of probes relative to their targets and the time one affords an assay, if an assay requires probes to be chemically modified upon finding their targets, e.g. to allow for their isolation and amplification, and if the targets being hunted are at ultralow concentration, one needs to be sure that this chemical process does not occur spontaneously at an unacceptably high rate.

To understand this second point, imagine (perhaps quite reasonably) that the target molecule catalyzed chemical modification step of choice has second-order kinetics, and that the employed Brownian "searcher" particles or probes are present in at least a $\approx 10^2$ fold excess relative to their targets. This probe excess conveniently allows us to use a pseudo first-order rate approximation for the formation of target probe complexes, which yields an expression of the form [147, 175]: $[Target\langle\rangle Probe] \approx [Target_0]\ \times\ (1 - e^{(-[Probe_0]kt)})$, where $[Target_0]$ and $[Probe_0]$ correspond to the initial concentrations of the target and probe molecules, respectively. As previously discussed, notice that increasing the probe concentration relative to the target concentration by some factor k is roughly equivalent to decreasing the assay hybridization step time by the same factor k (and vice versa). We also imagine that the spontaneous occurrence of this chemical modification step, i.e. the rate of false-positive generation, is either a first-order process (e.g. spontaneous cleavage via depurination or depyrimidination followed by β-elimination) or a pseudo first-order process that occurs at a low enough rate as to not grossly perturb the probe concentrations (e.g. some form of spontaneous ligation / dimerization / multimerization / etc.). Notice now that

multiplying the probe concentration, or equivalently the assay time, by some factor k correspondingly multiplies the yield of spontaneously generated false-positive signals via the presumed first-order or pseudo first-order reaction.

We pause for a moment to consider the question: why do false-positive signals actually matter? Isn't it true that one can simply run a control experiment with a known amount of target species to quantitate the rate of false-positive generation, then adjust the experimental data accordingly?

The answer to this question is of course that for any chemical reaction with an exponentially probability of occurring (e.g. any first- or pseudo first-order process like depurination, depyrimidination, or β-elimination) the actual false-positive count will be Poisson distributed with some rate parameter $\lambda = (k_{(false\ positive\ rxn)} * t) = (k_{(f.p.\ rxn)} * t)$, e.g. where $k_{(f.p.\ rxn)}$ in expressed in terms of the unit s^{-1}, and t is the time in seconds over which the reaction takes place. Thus, there will necessarily be a point where control experiments can no longer help, which exists at the limit where the number of correct probe $\langle\rangle$ target hybridization reporting signals is on-order the square root of the rate parameter, i.e. $(\lambda)^{1/2} = (k_{(f.p.\ rxn)} * t)^{1/2}$.

A good way to see this "danger limit" is to note that, in accordance with the Central Limit Theorem (CLT), and in practice for $\lambda \geq 30$ or so, the discrete Probability Mass Function (PMF) of the Poisson distribution is actually well approximated by a Gaussian distribution with mean $\mu \approx \lambda$ and standard deviation $\sigma \approx \lambda^{1/2}$. More specifically, the expression for the discrete Probability Mass Function (PMF) of a Poisson distribution with rate parameter (λ) and support (i.e. the set of points where the PMF has non-zero value) $n \in \mathbb{N}$, can be written as:

$$PoissonPMF(\lambda) = \frac{\lambda^n}{n!} \times e^{-\lambda} = PoissonPMF(k_{rxn} * t)$$
$$= \frac{(k_{rxn} * t)^n}{n!} \times e^{-(k_{rxn} * t)}$$

And the expression for the continuous Probability Distribution Function (PDF) of a Gaussian distribution with mean (μ) and standard deviation (σ) can be written as:

$$NormalPDF(x, \mu, \sigma) = \frac{1}{\sigma(2\pi)^{1/2}} \times e^{\frac{(-(x-\mu))^2}{2\sigma^2}}$$

At the limit where $n \longrightarrow \inf$ (implying $n! \propto n^{(n+1/2)} * e^{(-n)} * (2\pi)^{1/2}$ via Stirling's approximation) there will be a convergence of the Poisson distribution PMF and the Gaussian distribution PDF.

Returning to our demonstration of the "danger limit", let's either remember the 68–95–99.7 rule of thumb or compute a simple integral (where "Erf" is the standard Gauss error function):

$$\int_{x=(\mu-\sigma)}^{x=(\mu+\sigma)} (\frac{1}{\sigma(2\pi)^{1/2}}) \times e^{\frac{(-(x-\mu))^2}{2\sigma^2}} dx = Erf(\frac{1}{2^{1/2}}) \approx 0.682689$$

Thus, there is a $\approx 31.73\%$ chance that the measured number of false-positives will be off by a factor of $\lambda^{1/2}$ with respect to the expected mean number of false-positive events, $\lambda = (k_{(f.p.\ rxn)} * t)$.

For a visual illustration of how the Gaussian distribution does a decent job of approximating the Poisson distribution (setting $\lambda \geq 30$) see (Figure 6.1) immediately below.

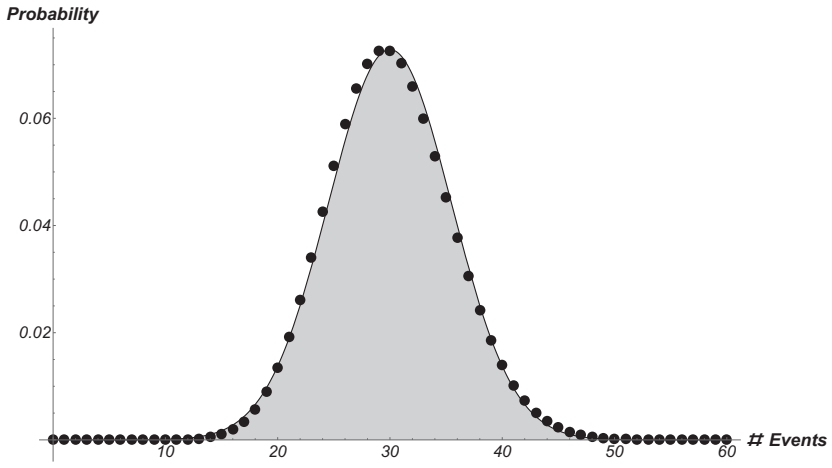


Figure 6.1: (Circles) Samples from the discrete Probability Mass Function (PMF) of a Poisson process with rate parameter $\lambda = 30$, where the PMF is given as $Pr[X = x] = (e^{(-\lambda)}\lambda^x \ / \ x!)$. (Curve) Plot of the continuous Probability Density Function (PDF) for a Normal distribution with mean μ and standard deviation $\sigma = (\mu)^{1/2} = 30^{1/2}$, where the PDF is given as $Pr[X = x] = (e^{(\frac{(x-\mu)^2}{2\mu})} \ / \ (2\pi\mu)^{1/2})$.

Regarding the argument that one can simply run the assay multiple times to better approximate $\mu = \lambda$, if you have the required amount of mRNA / cDNA / etc. targets to do this, or if you are willing to "blow up" (i.e. amplify) the target population (incurring various biases during the amplification procedure), what point does your ultrasensitive assay serve in the first place?

A general conclusion one can draw is that there are going to be unavoidable challenges in attempting to use probe molecules to find ultradilute targets via Brownian motion. For any practical assay, large concentrations of Brownian "searcher" particles (i.e. molecular probes) must be employed to find targets in an expedient fashion. Therefore, one must carefully analyze any first-, pseudo first-, or higher-order reactions that could yield products mimicking those produced by proper probe $\langle\rangle$ target complexes to report their existence (i.e. one must be careful of these processes generating false-positives).

Furthermore, in cases where the molecular target species one wishes to quantify are not only ultradilute, but also at ultralow copy number in the available sample volume (or sample volume that is practical to manipulate for the assay, e.g. the GEP-DEAN [74] method uses $\approx 29.5 \ \mu L$ reaction volumes where a $\approx 1 \ fM$ target concentration implies $N \approx 1.78 * 10^4$ target molecules), we note that one must correspondingly exercise care in avoiding assay conditions that would further deplete or destroy target molecules. The simple reason to worry about target copy number depletion, even if controlled and quantifiable, is that provided the assay's

sensitivity remains fixed, lower target copy numbers necessarily places one closer to the assay noise floor.

It's perhaps worth mentioning that there will always exist a fundamental noise floor arising from a combination of limitations in tool precision (e.g. pipette precision), instrument sensitivities, and (as just discussed) stochastic chemical processes (e.g. molecular first passage time distributions are not "sharp" in the sense that there is not a tight clustering of the first passage time distribution around the MFPT [39]), and which cannot be eliminated through the use of careful control experiments.

We note that processes which can depress target copy numbers include, however are no limited to: mechanical filtration steps (e.g. with nitrocellulose membranes), more generally processes that require repeated contact with hydrophobic surfaces that absorb nucleic acids non-specifically [6, 18] (e.g. transfer to a new polypropylene test tube will cause some percentage loss of nucleic acid molecules via non-specific hydrophobic absorption to the tube walls [18]), or steps that require temperatures of buffer conditions that degrade target molecules. For example, if target molecules are composed of RNA, high concentrations of divalent ions like Mg^{2+}, high pH, and high temperatures should be avoided to attenuate spontaneous polynucleotide backbone cleavage via transesterification. If the buffer ionic strength is decreased during the assay and/or the buffer pH is depressed causing base protonation [35, 206], the loss of nucleic acids via hydrophobic interactions should be of particular concern even in lieu of mechanical filtration or test tube transfer steps. Consider that low ionic strength buffer and base protonation both cause local (and in extreme cases global) duplex melting, exposing unstacked and/or poorly stacked hydrophobic nucleotide ring surfaces that can mediate hydrophobic interactions with surfaces or solvated particle contaminants [6].

In the following sections we attempt to analyze a number of false-positive signal generation pathways that are of particular concern for schemes where probes are cleaved or deaminated upon finding their targets. We also briefly turn our attention to identifying reasonable assay conditions, e.g. temperature ranges and buffer pH levels that sufficiently attenuate spontaneous transesterification-based decomposition (i.e. copy number depression) of RNA target molecules. Regarding this latter point, we note again that the current focus of Yokomori et. al. [193] appears to be the direct quantitation of mRNA via the photo-DEAN [193–196], and that all earlier DigiTag [128], DigiTag2 [129], and GEP-DEAN [74] methods have only been proven, thus far, to work for cDNA targets.

6.3 An overview of photo-DEAN buffer conditions and reaction temperatures up to and including the first round of PCR amplification for probe DCN sequences

We obtained an outline for the photo-DEAN [193–196] protocol from Yokomori et. al. [193]. In this section we use this outline to describe the buffer conditions and reaction temperatures employed, and the duration of time which they are employed, up to and including the first round of PCR amplification of DCN [74, 128, 129, 193–196] sequences.

(s.1) The "encoding" step where probe pairs are hybridized to target RNA molecules.

Buffer:

The stated buffer is $\approx 1x$ TE buffer: [$0.01M\ TE$, $0.02M DTT$, $0.150M\ NaCl$, $pH \approx 8$], which implies a $[Na^+] \approx 0.150M$ monovalent ion concentration and the absence of any divalent ions. Note that Invitrogen's "RNaseOUT Recombinant Ribonuclease Inhibitor" is also added to the reaction mixture, however we do not expect the addition of this reagent to effect the aforementioned buffer monovalent or divalent ionic strengths, or for that matter, noticeably perturb the buffer pH. More specifically, we can note that the storage buffer of "RNaseOUT" is: [$0.020M\ Tris - HCl$, $0.05M\ KCl$, $0.0005M\ EDTA$, $0.008M\ DTT$, 50% $glycerol$, $pH \approx 8$] (taken from Invitrogen's website: <http://tools.lifetechnologies.com/content/sfs/manuals/10777019.pdf>. Since the volume of added "RNaseOUT" storage solution is $\leq 20\%$ the total reaction buffer, and because there is no divalent metal in the storage solution, we do not expect the addition of this reagent to perturb the stated $[Na^+] \approx 0.150M$ ionic strength by more than $\approx 0.005M$ or so. We can also point out that this storage buffer has the same pH (i.e. $pH \approx 8$) as the $\approx 1x$ TE reaction buffer.

Reaction temperature(s) and times:

An anneal is performed by first holding the reaction volume at $\approx 95^\circ C$ for $\approx 180\ s$, then linearly ramping the temperature down to $\approx 40^\circ C$ at a rate of $\approx 0.1^\circ C\ /\ s$. This implies a total annealing time of $\approx (180\ s) + (95 - 40) * (10\ s) \approx (180\ s) + (550\ s) \approx 730\ s$.

(s.2) The "UV photoligation" step where a probe carrying a DCN [74, 128, 129, 193–196] sequence and a probe carrying a (5') biotin modification [128, 193–196] are photoligated together using Fujimoto et. al.'s ^{CV}U chemistry [62–64, 121, 126, 131–135, 199, 200],[s1].

Buffer:

It is our understanding that the buffer employed here is the same as that used for the probe $\langle\rangle$ mRNA target hybridization step (i.e. step **(s.1)**).

Reaction temperature(s):

During the ^{CV}U [62–64, 121, 126, 131–135, 199, 200],[s1] photoligation reaction the annealing reaction mixture from the probe $\langle\rangle$ mRNA target hybridization step (i.e. step **(s.1)**) is held at $\approx 40^\circ C$ for $\approx 10\ min \approx 600\ s$ while under irradiation by an

LED-based UV light source (Prizmatix Ltd.; EFICET 8332A UV Curing System) emitting at a peak intensity of $\lambda_{max} \approx 365\ nm$ and having a power density of $\approx 2.3\ W/cm^2$.

As a brief comment, in the context of photo-DEAN [193–196] where we've done away with the use of *Thermus* enzymes for the ligation reaction (e.g. Taq as used in the DigiTag [128], DigiTag2 [129], and GEP-DEAN [74] assays), the justification for the $\approx 40°C$ temperature used during this step is unclear to us. Perhaps this temperature step admits higher rates of target mRNA hybridization by DNA probes? We can also note here that we are unsure how temperature effects ^{CV}U [62–64,121,126,131–135,199,200],[s1] photoligation kinetics. On the one hand, lower temperatures will suppress duplex fraying / breathing, which could stabilize the stacking interactions of the ^{CV}U [62–64, 121, 126, 131–135, 199, 200],[s1] vinyl group. However, on the other hand, higher temperatures may allow for faster molecular configuration search times for a (perhaps energetically unfavorable) entry point to the desired $[2 + 2]$ cycloaddition reaction. We have not been able to find a treatment of this question in the literature.

(s.3) The λ $5'$-exonuclease "breakdown" step used for cleanup of unphotoligated DCN-encoding probe sequences with unprotected $5'$ termini.

Special note:

This step is intended to minimize false-positives during "magtration" with streptavidin coated magnetic beads, wherein one attempts to isolate properly photoligated photo-DEAN [193–196] probe pairs possessing both a $5'$ biotin a $3'$ DCN-encoding tail. Here, Yokomori et. al. [193] noticed that the magnetic beads employed for magtration had the side-effect of mediating hydrophobic interaction-based carryover of unphotoligated DCN-encoding probes. Therefore, in this step, Yokomori et. al. [193] is attempting to attenuate the magnitude of this problem by using a $5'$ exonuclease to digest the population the unphotoligated DCN encoding probes using the fact that these false-positive signals lack an "exonuclease shielding" $5'$ biotin (or at least shielding for some DNA $5'$ exonuclease enzymes).

It is reported by Yokomori et. al. [193] that this step reduces false-positive signals, in the form of unligated DCN probes, by a factor of $\approx 1/30$. See supplementary note (supp. note [s2]) for a brief discussion regarding the possibility of employing poly(T) or poly(A) polynucleotides to competitively inhibit DCN probe binding to magnetic bead surfaces and further reduce false-positive signals.

Buffer:

As this step involves a $0.5\ :\ 100$ dilution of the previous sample volume, we assume that the buffer in this step corresponds to New English Biolabs (NEB) λ exonuclease $\approx 1x$ reaction buffer [$0.067M\ Glycine - KOH$, $0.0025M\ MgCl_2$, $50\ \mu g/ml\ BSA$, $pH \approx 9.4$ at $\approx 25°C$], implying a monovalent salt concentration of $[K^+] \approx 0.067M$ and a divalent salt concentration of $[Mg^{2+}] \approx 0.0025M$ [w1].

For the purpose of contrast with other $5' \longrightarrow 3'$ DNA exonucleases, we note that the NEB T5 and T7 exonuclease $\approx 1x$ reaction buffers [$0.05M\ Potassium\ Acetate$, $0.02M\ Tris - acetate$, $0.01M\ Magnesium\ Acetate$, $0.001M\ DTT$, $pH \approx 7.9$ at $\approx 25°C$[have a monovalent ion concentration of $[K^+] \approx 0.05M$ and a divalent ion concentration of $[Mg^{2+}]$ $0.01M$ (see the links: [w2,w3]). As we can see from

these examples, while the NEB recommended reaction buffers for $5' \longrightarrow 3'$ DNA exonucleases having roughly the same ionic strength as the $\approx 0.5\ X\ TE$ buffer (with $[Na^+] \approx 0.075M$) used for the photo-DEAN [193–196] annealing step [193], they also have Mg^{2+} concentrations ranging from $[Mg^{2+}] \approx 0.0025M$ for the λ exonuclease to $[Mg^{2+}] \approx 0.01M$ for the T5 and T7 exonucleases.

Reaction temperature(s) and times:

The aforementioned buffer is held at $\approx 37°C$ for $\approx 30\ min \approx 1800\ s$, presumably for the DNA $5'$ exonuclease enzymatic digestion step. The temperature is then raised to $\approx 95°C$ for $\approx 3\ min \approx 180\ s$ for the purpose of λ exonuclease denaturation. Incidentally, we can note here that T5 exonuclease is poorly heat inactivatible [w2] and may, for this reason, be inappropriate as a substitute for λ exonuclease.

(s.4) Room temperature washing and NaOH-catalyzed RNA degradation (via transesterification).

Buffer:

Ignoring the initial wash step, which presumably occurs in the buffer from step **(s.3)** (where we guess monovalent and divalent ion concentrations of $[Na^+]$ or $[K^+] \approx 0.056M$ and $[Mg^{2+}] \approx 0.0075M$), the following alkaline treatment step is stated to occur in a buffer with $\approx 0.1M\ NaOH$ (implying the buffer is at $pH \approx 14 - (-log_{10}(0.1)) \approx 13$), a monovalent ionic strength of $[Na^+] \approx 0.05M$, and presumably no divalent metal.

Reaction temperature(s) and times:

Regarding the wash steps preceding and following $NaOH$ treatment, which we'll estimate occur $\approx 25°C$ (to guess a slight upperbound value for what could be considered "room temperature"), we unfortunately do not have an estimate for the total time to perform these steps. However, we do have that the alkaline treatment and wash step is performed for $\approx 2\ min \approx 120\ s$ at $\approx 45°C$. Provided the exponential dependence on temperature for most of the processes effecting nucleic acid probes and target molecules that we will discuss in the following sections - e.g. depurination, depyrimidination, β-elimination, RNA transesterification, cytosine deamination, etc. reactions should all show straight lines on Arrhenius plots for $ln(k_{rxn})$ vs. $1/(rxn\ temperature)$, and we have no reason to expect dominant rate-limiting processes at high temperatures $> 25°C$ (obviously much below this temperature the reaction buffer will begin to freeze and there will be problems) - we will simply consider the entirety of this step to occur in $\approx 120\ s$ at $\approx 45°C$.

Jumping briefly ahead (see section **6.6** and ref. [110]), we can calculate that the rate of RNA dinucleotide cleavage under these conditions ($pH \approx 13$, $\approx 45°C$) should be $k_{(RNA\ transesterification,\ (95°C, pH \approx 8),\ no\ Mg^{2+})} \approx (1.02 * 10^{-3}\ s^{-1})$, implying a fraction of cleaved RNA dinucleotides after $\approx 2\ min \approx 120\ s$ of
$\approx (1 - e^{(-(1.02*10^{-3}\ s^{-1})*120\ s)}) \approx 0.115 \approx 11.5\%$.

(s.5) The PCR amplification step (with KOD polymerase).

Buffer:

It's not precisely clear what buffer conditions are employed for PCR, however we do know that the KOD enzyme is used and $\approx 0.001M$ $MgSO_4$ is present in the final PCR reaction buffer ($\approx 2\ \mu L$ of $MgSO_4$ at $\approx 0.025M$ is stated to be added to a $\approx 50\ \mu L$ PCR reaction solution) [193]. We note that the amount of $MgSO_4$ added matches a recommendation from TOYOBO Life Sciences for their "KOD Plus" enzyme [w4]. As such we might guess that the buffer used for PCR has a divalent ion concentration of $]Mg^{2+}[\approx 0.001M$. Unfortunately, the monovalent ionic strength of the TOYOBO "KOD Plus" buffer appears to be either simply unstated or is perhaps proprietary.

Reaction temperature(s) and times:

To the best of our knowledge, Yokomori et. al.'s PCR procedure [193] consists an initial $\approx 2\ min \approx 120\ s$ incubation at $\approx 95^{\circ}C$ (step 1), followed by 25 cycles of steps 2 through 4. After these temperature cycling steps, our understanding is that the reaction volume is incubated at a temperature of $\approx 4^{\circ}C$ until the PCR amplified sample is collected:

(PCR temperature step 1): Hold at $\approx 94^{\circ}C$ ($\approx 2\ min \approx 120\ s$);

for ($iter = 1;\ iter <= 25;\ iter = iter + 1$) {

(PCR temperature step 2): Hold at $\approx 94^{\circ}C$ ($\approx 15\ s$);

(PCR temperature step 3): Hold at $\approx 64^{\circ}C$ ($\approx 30\ s$);

(PCR temperature step 4): Hold at $\approx 68^{\circ}C$ ($\approx 10\ s$);

};

(PCR temperature step 5): Hold at $\approx 4^{\circ}C$ (forever);

Considering the exponential amplification of target oligonucleotide sequences from PCR (specifically DCN sequences in the case of photo-DEAN [193–196]) we concern ourselves here only with the first cycle of PCR for our analysis of false-positives and false-negatives arising from fundamental chemical processes acting on oligonucleotide probes. Regarding any RNA in solution, as a consequence of **(s.4)**, which involves $pH \approx 13$ buffer conditions, "most" RNA polynucleotides should now be decomposed via OH^{-} facilitated transesterification (we earlier gave an estimate of $\approx 11.5\%$ RNA dinucleotide cleavage for step **(s.4)**, however please see section **6.6** for details).

So what can we conclude based on the previous brief descriptions (due to Yokomori et. al. [193]) of buffer conditions, reaction temperature(s), and reaction times up to and including the first photo-DEAN [193–196] PCR cycle?

Well, perhaps most importantly, it's clear that we spend a decent amount of time at $\approx 95^{\circ}C$. Specifically, we spend $\approx 180\ s$ at $\approx 95^{\circ}C$ in **(s.1)**, another $\approx 180\ s$ at

$\approx 95^{\circ}C$ for the $5' \longrightarrow 3'$ DNA exonuclease denaturation step in **(s.3)**, and finally, $\approx$ 135 s at $\approx 95^{\circ}C$ in the first round of PCR amplification during step **(s.5)**, implying a total time at $\approx 95^{\circ}C$ of $\tau_{95^{\circ}C} \approx 495$ s. We also need to consider the **(s.1)** thermal anneal, which ramps from $\approx 95^{\circ}C$ to $\approx 40^{\circ}C$ at a constant rate of $\approx 0.1^{\circ}C$ / s (for a total time of ≈ 550 s).

We note that it's largely pointless to consider an "average temperature" during the thermal anneal (which is $\approx 67.5^{\circ}C$ discounting the ≈ 3 $min \approx 180$ s at $\approx 95^{\circ}C$) since reaction kinetics for the processes we care about here (as discussed in **(s.4)**) scale exponentially rather than linearly with temperature. We suggest that the right thing to do to account for the thermal anneal, and something we can conveniently get away with because of the flat temperature ramp rate of $\approx 0.1^{\circ}C$ / s (again discounting the ≈ 180 s spent at $\approx 95^{\circ}C$ prior to the thermal ramping process), is to compute an integral of the form:

(Exp. 6.2)

$$k_{rxn}(mean\ rxn\ rate,\ T \in [t_0, t_1]) \approx \frac{1}{(t_1 - t_0)} \int_{t_0}^{t_1} (at\ T(^{\circ}C))\ dT$$

Other than these high temperature incubation and annealing steps, there isn't much else of concern for these photo-DEAN [193–196] steps (**(s.1)** to **(s.5)**) prior to exponential blowup of DCN sequences. While the addition of pseudo-physiological levels of magnesium ($[Mg^{2+}] \approx 0.0025M$ in **(s.3)** and $[Mg^{2+}] \approx 0.001M$ in **(s.5)**) would be cause for concern if the probes were composed of RNA, photo-DEAN [193–196] employs DNA probes and by step **(s.3)**, the first time where divalent ions appear, the RNA targets have already served their purpose. This is likewise true for step **(s.4)** where the buffer pH is raised to $\approx$ 13 via the addition of $\approx 0.1M$ $NaOH$. While this very high pH will certainly melt any nucleic acid duplexes and deprotonate the $2'$-OH group on RNA polynucleotides, catalyzing rapid transesterification-based decomposition, DNA probes missing the $2'$ hydroxyl group on their base sugars should be considerably more resilient to covalent alteration or breakdown at $pH \approx 11$ to $pH \approx 13$ or so. Regarding this last point, and once again jumping briefly ahead, under physiological ionic strength conditions (e.g. something like $[Na^{+}] \approx 0.1M$, $[Mg^{2+}] \approx 0.01M$, $pH \approx 7.4$) we note that DNA is approximately $\approx 10^5$ fold more stable to hydrolysis than RNA (see ref. [149]).

6.4 Spontaneous decomposition of DNA: rates and consequences for (deoxy)ribonucleic acid depurination, depyrimidination, and intramolecular cleavage via β-elimination at abasic sites

We now attempt to analyze the chemical stability of the single-stranded (deoxy)ribonucleic acid (DNA) probes, which, in the context of photo-DEAN [193–196], DigiTag [128], DigiTag2 [129], and GEP-DEAN [74] schemes, serve as Brownian "searcher" particles (i.e. molecules probes) for ultradilute cDNA or mRNA targets. We suggest that, with regards to the current version of photo-DEAN [193–196], this analysis may be of importance for quantification of false-negatives, i.e. the decay of signals produced when probes successfully interact with their targets. In particular, we note that the stability of the $\approx 10^2$ nucleotide DCN [74, 128, 129, 193–196] signal sequences, encoded on at least a subset of the probes, may be of concern prior to PCR or (hypothetically speaking) rolling circle amplification steps. Furthermore, we note that if the probe chemical modification step to report target discovery involves a ligation process, e.g. of a biotin modified oligonucleotide to allow for affinity purification as in the current photo-DEAN [193–196] scheme as well as the previous DigiTag [128] (though not DigiTag2 [129]) and GEP-DEAN [74] schemes, it may also make sense to quantitate the stability of the linkage between the aforementioned DCN [74,128,129,193–196] signals and this covalently attached label. And of course when probe $\langle\rangle$ target complexes can potentially have ultra-low $N \approx 10^3$ to $\approx 10^4$ copy numbers, every molecule matters in terms of rising above the inevitable assay noise floor.

We also suggest that this analysis is also of particular importance if alternative "cleavage-based" methods are utilized for chemically modifying probes to indicate target discovery or hybridization. Spontaneous probe cleavage can, in this case, directly imply the generation of false-positive signals.

-

To begin our analysis, we note that the primary means of chemical degradation of DNA is via depurination (primarily) or depyrimidination (secondarily, at a $\approx$ 20-fold reduced rate relative to depurination according to (Lindahl, 1993) [112]) followed by a fast abasic site cleaving (DNA backbone cutting) β-elimination process [12, 13, 167]. We can therefore say that, in and around $pH \approx 7$, at least in the absence of extremely high concentrations of divalent metal (e.g. $\gg 0.01M$ "physiological levels" of $[Mg^{2+}]$) [12, 13, 112, 115, 167], DNA should spontaneously cleave at a rate given by:

(Exp. 6.3)

$$k_{(DNA\ spontaneous\ cleavage,\ T^{\circ}C)} \approx (((Num.\ purine\ bases) * k_{(depurination,\ T^{\circ}C)}) + ((Num.\ pyrimidine\ bases) * k_{(depyrimidination,\ T^{\circ}C)})) * k_{(\beta-elimination,\ T^{\circ}C)}$$

We note that for the following sections we will repeatedly reference two buffers from (Lindahl & Nyberg, 1972) [115]:

"Lindahl Buffer A" [115]: [$0.1M$ $NaCl$, $0.01M$ *sodium phosphate*, $0.01M$ *sodium citrate*, $pH \approx 7.4$ at $\approx 80^{\circ}C$]. As noted in (Lindahl & Nyberg,

1972) [115], the pH and ionic strength of this buffer is only minimally perturbed by changes in temperature.

"Lindahl Buffer B" [115]: [$0.1M\ KCl$, $0.05M\ N-2-hydroxyethylpiperazine-N'-2-ethanesulfonicacid(Hepes)-KOH$, $0.01M\ MgCl_2$, $0.001M\ EDTA$, $pH \approx 7.4 \pm 0.15$ at $\approx 80°C$]. As noted in (Lindahl & Nyberg, 1972) [115], the pH of this buffer is more sensitive to changes in temperature than "Lindahl Buffer A" [115], and is only suggested for use in a temperature range of around $\approx 70°C$ to $\approx 80°C$ [115].

We focus our analysis (below) on rates in "Lindahl Buffer A" [115] which appears to closely mimic the $\approx 1x\ TE$ buffer: [$0.01M\ TE$, $0.02M\ DTT$, $0.150M\ NaCl$, $pH \approx 8$] used during the photo-DEAN [193–196] initial target "search process" or hybridization step (i.e. step **(s.1)**, with the anneal, described in section **6.3**). Consider that the monovalent ionic strength of the photo-DEAN [193–196] $\approx 1x\ TE$ buffer is roughly $[Na^+] \approx 0.150M$ (see section **6.3**), which is fairly close to the estimated "Lindahl Buffer A" [115] monovalent ionic strength of $[Na^+] \approx 0.12M$, and consider that neither of the two buffers contain any divalent metal. Regarding steps **(s.3)** and **(s.5)** where considerable time is also spent at high temperatures, for **(s.3)** we have a buffer with a monovalent salt concentration of $[K^+] \approx 0.067M$ and a divalent salt concentration of $[Mg^{2+}] \approx 0.0025M$ and for **(s.5)** the buffer is unknown though we are able to guess a divalent ion concentration of $[Mg^{2+}] \approx 0.001M$. Therefore, and while the unknown monovalent ion concentration in **(s.5)** is troubling, the roughly similar monovalent ion concentration in **(s.1)** and **(s.3)** and the presence of low $[Mg^{2+}] \leq 0.0025M\ Mg^{2+}$ concentrations in **(s.3)** and **(s.5)** should make the assumption of "Lindahl Buffer A" [115] a reasonable one [12, 13, 112, 115, 167].

Regarding the rate of depurination for ssDNA: observing the linear trend in the (Lindahl & Nyberg, 1972) [115] Arrhenius plots for DNA depurination (see "Figure 8" in ref. [115]), i.e. plots of $ln(k_{(depurination,\ T(°C))}$ vs. $1/T$, and noting their estimate of $E_{a(depurination)} \approx 31 \pm 2\ kcal/mol$ [115] for the activation energy of depurination measured in "Lindahl Buffer A" [115], we can take a ratio of Boltzmann factors to calculate the rate $k_{(depurination,T(°C))}$ for any temperature $T(°C)$ with the general expression:

(Exp. 6.4.1)

$$k_{(rxn,\ T°C)} \approx k_{(rxn,\ T_{ref}(°C))} \times e^{\left(\left(\frac{E_{a(rxn)}}{R\times(273.15+T_{ref}(°C))}\right)-\left(\frac{E_{a(rxn)}}{R\times(273.15+T(°C))}\right)\right)}$$

Where:

$$R = Universal\ Gas\ Constant \approx k_\beta * N_A \approx 1.9872041 * 10^{-3}\left(\frac{kcal}{mol*K}\right)$$

And for reference:

$$k_B = Boltzmann's\ Constant \approx 3.29983 * 10^{-27}\ (kcal/K)$$

$$N_A = Avogadro's\ Constant \approx 6.02214129 * 10^{23}\ mol^{-1}$$

Here, (Lindahl & Nyberg, 1972) [115] report a reference "Lindahl Buffer A" [115] value for native dsDNA depurination rates at $\approx 70°C$ of:

$$k_{(depurination,\ T_{ref}(70°C)} \approx (4 * 10^{-9} * 3.3)/(0.7)s^{-1} \approx (1.9 * 10^{-8}s^{-1})$$

Now, using **(Exp. 6.3)** and the experimentally measured value of $E_{a(depurination)} \approx 31 \pm 2\ kcal/mol$ [115], we can calculate a $\approx 95°C$ rate of $k_{(depurination,\ 95°C)} \approx (4.2 * 10^{-7}s^{-1})$. We can also use **(Exp. 6.2)** and **(Exp. 6.4.1)** together to calculate an average rate of depurination over the temperature range $\approx 40°C$ to $\approx 95°C$ (which would presumably match the average rate of depurination for the photo-DEAN [193–196] annealing reaction, i.e. step **(s.1)** in section **6.3**).

The calculation for this average rate of depurination over this interval is:

$$k_{(depurination,\ mean::T\in[40°C,\ 95°C])} \approx (\frac{1}{95-40}) \int_{T=40}^{T=95} k_{(depurination,\ T(°C))} dT \approx (6.3 * 10^{-8}\ s^{-1})$$

A direct value for $k_{(depurination,\ T_{ref}(70°C))}$ is only provided by (Lindahl & Nyberg, 1992) [115] for native DNA in "Lindahl Buffer B" [115], and as such, we resorted to estimating our "Lindahl Buffer A" [115] value by noting two statements in the paper (i.e. ref. [115]): (1) that the rate of depurination in "Lindahl Buffer B" [115] is $\approx 70\%$ the rate of depurination in "Lindahl Buffer A" [115], and (2) that denaturing the DNA prior to the experiment yields a ≈ 3.3 fold increase in the rate of depurination. We also note that the ≈ 3.3 fold increase in the rate of depurination for denatured vs. native DNA appears to be largely invariant in the (Lindahl & Nyberg, 1972) [115] ("Figure 8" in ref. [115]) Arrhenius plots, with data points taken in "Lindahl Buffer A" [115] up to a temperature of $\approx 80°C$. Above $\approx 80°C$, the "native DNA" primarily linear trend line expectedly converges on the "denatured DNA" primarily linear trend line [115].

Of course, the $k_{(depurination,\ T_{ref}(70°C))} \approx (4 * 10^{-9} * 3.3)/(0.7)\ s^{-1} \approx (1.9 * 10^{-8}\ s^{-1})$ estimate may be unsound for the reason that denaturation of DNA may alter the magnitude of the effects of magnesium on the depurination rate. On the other hand, the suggested adjustments for single-stranded DNA in Mg^{2+}-free buffer does not change the rate of depurination by more than an order of magnitude. Thus, we might argue that the worst-case consequences of this potentially unsafe assumption should be tolerable.

-

Regarding the rate of depyrimidination for ssDNA: (Lindahl & Karlstrom, 1973) [114] employed the aforementioned "Lindahl Buffer A" [115] to find reference measurements for the rate of thymine and cytosine depyrimidination at $\approx 95°C$ of $\approx 2.3 * 10^{-8}\ s^{-1}$ for (deoxy)thymine and $\approx 1.8 * 10^{-8}\ s^{-1}$ for (deoxy)cytosine, allowing us to estimate $k_{(depyrimidination,\ T_{ref}(95°C))} \approx 2.1 * 10^{-8}\ s^{-1}$. Note that

this value almost exactly corresponds to the estimate, cited in (Lindahl, 1993) [112], of $\approx$ 20-fold slower rates of depyrimidination relative to depurination, thus perhaps justifying the dsDNA $\longrightarrow$ ssDNA and $[Mg^{2+}] \approx 0.01M \longrightarrow [Mg^{2+}] \approx 0M$ adjustments we made to estimate the rate of depurination of ssDNA in "Lindahl Buffer A" [115] from a measurement on dsDNA in "Lindahl Buffer B" [115] (unless, of course, this is exactly how Lindahl made the estimate in ref. [112]).

Interestingly, (Lindahl & Karlstrom, 1973) [114] note that the addition of $\approx 0.01M$ $[Mg^{2+}]$ via the use of "Lindahl Buffer B" [115] does not perturb the depyrimidination reaction rate within experimental error. However, we also note that this latter "Lindahl Buffer B" [115] measurement by (Lindahl & Karlstrom, 1973) [114] was taken at $\approx 95^{\circ}C$ despite the warning by (Lindahl & Nyberg, 1972) [115] not to use their "Lindahl Buffer B" [115] outside of a $\approx 70^{\circ}C$ to $\approx 80^{\circ}C$ temperature range (due temperature-induced buffer pH changes), so perhaps some small contribution of Mg^{2+} is being disguised by a weak buffer pH dependence on temperature.

Finally, while (Lindahl & Karlstrom, 1973) [114] do not report activation energies for the depyrimidination of thymine and cytosine, we can find values of $E_{a(depyrimidination)} \approx 31\ kcal/mol$ to $\approx 34\ kcal/mol$ elsewhere in the literature [68, 157]. Using the value of $\approx 31\ kcal/mol$ to yield upperbound kinetics for depyrimidination, we can calculate average rate of depyrimidination over the temperature range $\approx 40^{\circ}C$ to $\approx 95^{\circ}C$ in the same manner as for the depurination reaction, which yields the value $k_{(depyrimidination,\ mean::T\in[40^{\circ}C,\ 95^{\circ}C])} \approx (3.2 * 10^{-9}\ s^{-1})$.

Regarding the rate of β-elimination at abasic sites in ssDNA: We note that (Sugiyama et. al., 1994) [167] found a rate for β-elimination of a centrally abasic site in an ssDNA trimer, at $pH \approx 7$ and a temperature of $\approx 90^{\circ}C$ in a buffer with about a $[Na^{+}] \approx 0.05M$ monovalent ion strength (and no divalent metal), of $k_{(\beta-elimination,\ 90^{\circ}C)} \approx 4.5 * 10^{-4}\ s^{-1}$. To elaborate, this is the rate of β-elimination for (Sugiyama et. al., 1994)'s [167] "COMPOUND 10", 5′-T(abasic site)T-3′, which was generated by heating the trinucleotide d(5′-TAT-3′) at $\approx 90^{\circ}C$ for $\approx 5\ min$ in $\approx 0.1M$ ($\approx 0.1N$) HCl ($pH \approx Log_{10}(0.1) \approx 1$). After neutralizing this solution, (Sugiyama et. al., 1994) [167] then heated the depurinated product d(5′-T_T-3′) at ($pH \approx 7,\ \approx 90^{\circ}C$), and collected data points via HPLC at $\{10, 20, 30, 45, 60, 90\}\ min$ time intervals (see "Figure 4" in (Sugiyama et. al., 1994) [167] for these data points).

For an activation energy for chain breakage at abasic sites, we turn to (Eigner et. al., 1961) [53], where $E_{a(\beta-elimination,\ ssDNA)} \approx 25 \pm 2\ kcal/mol$ was found. We note that this activation energy for ssDNA β-elimination is actually rather close to the value for dsDNA, $E_{a(\beta-elimination,\ dsDNA)} \approx 24.5 \pm 1.5\ kcal/mol$ found by (Lindahl & Andersson, 1972) [113].

Provided the closeness of these activation energies, we can test our ability to use **(Exp. 6.4.1)** and (Sugiyama et. al., 1994)'s [167] reference measurement of ssDNA β-elimination rates at $pH \approx 7$ and $\approx 90^{\circ}C$, $k_{(\beta-elimination,\ 90^{\circ}C)} \approx 4.5 * 10^{-4}\ s^{-1}$, to predict the results of experimental measurements for β-elimination in dsDNA at low temperatures (we were unable to find any similar measurements with ssDNA substrates having abasic sites). This works surprisingly well. According to (Laurence et. al., 1963) [107] and (Lawley et. al., 1969) [108], the half-life of a single abasic site internal to a dsDNA duplex is $\approx 2000\ h$ at $\approx 37^{\circ}C$ in $pH \approx 7$ phosphate buffer without divalent metal, implying a rate of β-elimination under these conditions of $\approx (ln(2)/((2000 * 60^2\ s))) \approx 9.63 * 10^{-8}\ s^{-1}$. Using (Sugiyama et. al., 1994)'s [167] measurement, (Eigner et. al., 1961)'s [53] estimate of a $\approx 25\ kcal/mol$ activation energy for abasic site cleavage in ssDNA, and **(Exp.**

6.4.1), we predict a value of $k_{(\beta-elimination,\ 37°C)} \approx 1.21 * 10^{-6}\ s^{-1}$, which is only a factor of $\approx (1.21 * 10^{-6}\ s^{-1})/(9.63 * 10^{-8}\ s^{-1}) \approx 12.6$ faster than the rate extrapolated from the experimental measurements of (Laurence et. al., 1963) [107] and (Lawley et. al., 1969) [108]. We feel that this is very reasonable considering the substrate change from ssDNA to dsDNA.

Regarding (Lindahl & Andersson, 1972)'s [113] measurement of $\approx 1.01 * 10^{-6}\ s^{-1}$ for the rate of β-elimination in the context of dsDNA at $pH \approx 7$ and $\approx 37°C$, we caution that, although this value seems more in line with our predicted estimate of $k_{(\beta-elimination,\ 37°C)} \approx 1.21 * 10^{-6}\ s^{-1}$, this is likely an artifact of their use of "Lindahl Buffer B" [115], which has a divalent ion concentration of $[Mg^{2+}] \approx 0.01M$. As (Lindahl & Andersson, 1972) [113] note, and at least in the case of dsDNA, the presence of Mg^{2+} appears to speed up β-elimination at abasic sites (the effects of Mg^{2+} on ssDNA β-elimination rates remains unclear). We do however note that (Lindahl & Andersson, 1972)'s [113] one data point in a variant of "Lindahl Buffer B" [115] without Mg^{2+}, where a β-elimination rate of $\approx 2.4 * 10^{-5}\ s^{-1}$ was observed, actually comes quite close to a value we can predict at $\approx 70°C$ of $k_{(\beta-elimination,\ 70°C)} \approx 6.0 * 10^{-5}\ s^{-1}$. Thus, it may be case that the protection a native DNA duplex offers an abasic site diminishes at higher temperatures, yielding only a ≈ 2.5 fold slowdown relative to our abasic site β-elimination rates in ssDNA at $\approx 70°C$.

For consistency with the previous discussions on ssDNA depurination and depyrimidination rates, let's calculate a rate of β-elimination at $pH \approx 7$ and $\approx 95°C$ of: $k_{(\beta-elimination,\ 95°C)} \approx 7.2 * 10^{-4}\ s^{-1}$. Using **(Exp. 6.2)** and **(Exp. 6.4.1)** we can also calculate an average rate of β-elimination during the photo-DEAN [193–196] step **(s.1)** anneal from $\approx 95°C$ to $\approx 40°C$ at a rate of $\approx 0.1°C/s$ (see (II.3)) of: $k_{(deamination,\ mean::T\in[40°C,\ 95°C])} \approx 1.3 * 10^{-4}\ s^{-1}$.

-

Putting all of this together, at $\approx 95°C$ in "Lindahl Buffer A" [115] we have rates for ssDNA depurination, depyrimidination, and polynucleotide chain-breaking β-elimination at abasic sites, of:

$$k_{(depurination,\ 95°C)} \approx 4.2 * 10^{-7}\ s^{-1}$$
$$k_{(depyrimidination,\ 95°C)} \approx 2.1 * 10^{-8}\ s^{-1}$$
$$k_{(\beta-elimination,\ 95°C)} \approx 7.2 * 10^{-4}\ s^{-1}$$

Which implies:

$$k_{(ssDNA\ chain\ breakage\ per\ pyrimidine,\ 95°C)} \approx 1.5 * 10^{-11}\ s^{-1}$$
$$k_{(ssDNA\ chain\ breakage\ per\ purine,\ 95°C)} \approx 3.0 * 10^{-10}\ s^{-1}$$
$$k_{(ssDNA\ chain\ breakage\ per\ base\ (ave),\ 95°C)} \approx 1.6 * 10^{-10}\ s^{-1}$$

At $\approx 37°C$ in "Lindahl Buffer A" [115] we have rates for ssDNA depurination, depyrimidination, and polynucleotide chain-breaking β-elimination at abasic sites, of:

$$k_{(depurination,\ 37°C)} \approx 1.5 * 10^{-10}\ s^{-1}$$

$$k_{(depyrimidination,\ 37°C)} \approx 7.6 * 10^{-12}\ s^{-1}$$
$$k_{(\beta-elimination,\ 37°C)} \approx 1.2 * 10^{-6}\ s^{(}-1)$$

Which implies:

$$k_{(ssDNA\ chain\ breakage\ per\ pyrimidine,\ 37°C)} \approx 1.8 * 10^{-16}\ s^{-1}$$
$$k_{(ssDNA\ chain\ breakage\ per\ purine,\ 37°C)} \approx 9.1 * 10^{-18}\ s^{-1}$$
$$k_{(ssDNA\ chain\ breakage\ per\ base\ (ave),\ 37°C)} \approx 9.5 * 10^{-17}\ s^{-1}$$

And for a flat temperature ramp anneal from $\approx 95°C$ to $\approx 40°C$ in "Lindahl Buffer A" [115], we have rates of:

$$k_{(depurination,\ mean::T\in[40°C,\ 95°C])} \approx 6.3 * 10^{-8}\ s^{(}-1)$$
$$k_{(depyrimidination,\ mean::T\in[40°C,\ 95°C])} \approx 3.2 * 10^{-9}\ s^{-1}$$
$$k_{(\beta-elimination,\ mean::T\in[40°C,\ 95°C])} \approx 1.3 * 10^{-4}\ s^{-1}$$

Which implies:

$$k_{(ssDNA\ chain\ breakage\ per\ pyrimidine,\ mean::T\in[40°C,\ 95°C])} \approx 4.2 * 10^{-13}\ s^{-1}$$
$$k_{(ssDNA\ chain\ breakage\ per\ purine,\ mean::T\in[40°C,\ 95°C])} \approx 8.2 * 10^{-12}\ s^{-1}$$
$$k_{(ssDNA\ chain\ breakage\ per\ base\ (ave),\ mean::T\in[40°C,\ 95°C])} \approx 4.3 * 10^{-12}\ s^{-1}$$

We can also state the experimental activation energies for these processes as (assuming a $\approx 31\ kcal/mol$ value for the activation energy of depyrimidination to yield an upperbound rate for the kinetics of this process and to match the estimated activation energy of depurination):

$$E_{a(depurination)} \approx 31 \pm 2\ kcal/mol$$

$$E_{a(depyrimidination)} \approx 31\ kcal/mol\ to\ \approx 34\ kcal/mol$$

$$E_{a(\beta-elimination,\ ssDNA)} \approx 25 \pm 2\ kcal/mol$$

Using **(Exp. 6.4.1)**, the above activation energies can be used to calculate rates of ssDNA depurination, depyrimidination, and β-elimination in at any temperature in "Lindahl Buffer A" [115] (or buffers with a similar monovalent ionic strength of $[Na^+] \approx 0.12M$ and no divalent metal).

-

Here, if we assume an upperbound probe length of $L = 100$ nts where $(Num.\ purine\ bases) \approx 50$ and $(Num.\ pyrimidine\ bases) \approx 50$, approximating a DNA oligonucleotide with "well mixed" $\{A, T, G, C\}$ sequence composition, we find $k_{(ssDNA\ chain\ breakage,\ 95^\circ C)} \approx 1.6 * 10^{-8}\ s^{-1}$. This implies that for the $\tau_{95^\circ C} \approx 495\ s$ spent at $\approx 95^\circ C$ up to and including the first round of PCR for the photo-DEAN [193–196] process (see steps **(s.1)**, **(s.3)**, and **(s.5)** in section **6.3**; note that we count the $\approx 94^\circ C$ (PCR temperature step 2) from **(s.5)** in the total for time spent at $\approx 95^\circ C$) we can expect that the fraction of cleaved probes will be:

$$frac[spontaneously\ cleaved\ well\ mixed\ \approx 100{-}mer\ DNA \mid (\approx 95^\circ C, pH \approx 8) \mid \tau_{95^\circ C} \approx 495\ s] = (1 - e^{(-k_{(ssDNA\ chain\ breakage, 95^\circ C)} * t)}) \approx (1 - e^{(-(1.6 * 10^{-8}\ s^{-1} * 495\ s)}) \approx 7.9 * 10^{-6}$$

Likewise, for the step **(s.1)** (again, see section **6.3**) anneal from $\approx 95^\circ C$ down to $\approx 40^\circ C$ at a constant $\approx 0.1^\circ C/s$ ramp rate, which takes $\tau_{(95^\circ C\ to\ 40^\circ C, 0.1^\circ C/s)} \approx 550\ s$, we have:

$$k_{(well\ mixed\ 100{-}mer\ ssDNA\ chain\ breakage, mean::T \in [40^\circ C, 95^\circ C])} \approx (4.3 * 10^{-10}\ s^{-1})$$

Which implies:

$$frac[spontaneously\ cleaved\ well\ mixed\ \approx 100{-}mer\ DNA \mid (\approx 95^\circ C, pH \approx 8) \mid \tau_{(95^\circ C\ to\ 40^\circ C, 0.1^\circ C/s)} \approx 550\ s] = (1 - e^{(-k_{(ssDNA\ chain\ breakage, mean::T \in [40^\circ C, 95^\circ C])} * t)}) \approx (1 - e^{(-(4.3 * 10^{-10}\ s^{-1} * 550\ s)}) \approx 2.4 * 10^{-7}$$

We draw the following conclusions from these calculations for the rate of ssDNA depurination, depyrimidination, and abasic site cleavage via β-elimination in a buffer similar to "Lindahl Buffer A" [115] ($[[Na]^+] \approx 0.12\ M$ and no divalent metal; $pH \approx 7.4$):

(Conclusion 1): Regarding the existing photo-DEAN [193–196] method, spontaneous cleavage of DNA probes appears to be of little consequence, even at the limit of single copy numbers of target species and arbitrary concentrations of probes.

For the photo-DEAN [193–196] steps up to and including the first round of PCR, as discussed above, we would expect a fraction: $\approx (1 - (1 - 7.9 * 10^{-6})(1 - 2.4 * 10^{-7})) \approx 8.1 * 10^{-6}$ of, for example, well-mixed $\approx 10^2$ nucleotide long DCN [74, 128, 129, 193–196] sequences to spontaneous cleave. This fraction should likewise hold (within an order-of-magnitude) for the specific population of probes that found their targets

and were successfully photoligated or otherwise chemically modified to indicate this event. The only conceivable danger from spontaneous probe degradation, via depurination or depyrimidination followed by β-elimination, would be in the limit where one is attempting to quantitate a large concentration of target molecules ($[target\ species] \gg 8.1 * 10^6$) with a precision of greater than ≈ 1 in $\approx 8.1 * 10^6$ or so.

The more subtle danger, of course, is that probes with uncleaved abasic sites may not exhibit the same amplification kinetics during initial rounds of PCR. Consider that abasic site generation will almost always precede DNA intramolecular cleavage, and will occur at a $(1/k_{(\beta-elimination)} \approx 10^3\ to\ 10^4)$ fold faster rate. For example, we would expect, after $\tau_{95^\circ C} \approx 495\ s$ at $\approx 95^\circ C$, under the aforementioned buffer conditions, that $\approx 1\%$ (i.e. a ≈ 0.01 fraction) of well mixed $\approx 10^2$ nucleotide DCN sequences will have a single abasic site. The influence of a signal abasic site would also be enormously compounded, for obvious reason, if rolling circle amplification is employed in place of a traditional PCR.

Thus, if one wishes to quantitate a population of target molecules at a sensitivity of ≈ 1 in ≈ 100 molecules or greater, any high temperature incubation steps or thermal anneals should be strictly avoided.

(Conclusion 2): In light of the previous comments, where we noted that $\approx 1\%$ (i.e. a ≈ 0.01 fraction) of $\approx 10^2$ nucleotide DCN [74, 128, 129, 193–196] sequences with well mixed sequence composition will have a single abasic site, care should be taken in deciding whether or not to expose pre-amplified photo-DEAN [193–196] probes to abasic site cleaving repair enzymes (e.g. endonuclease III (Nth), endonuclease VIII, FPG, etc.). On the one hand, this could lead to probe $\langle\rangle$ target signal loss (i.e. false-negatives). On the other hand, repair enzyme treatment of this sort could minimize variance in assay results by eliminating sequences that may amplify in an abnormal fashion, and if amplifiable, lead to the generation of products that exhibit abnormal microarray hybridization during the photo-DEAN [193–196] "decoding" step. However, we consider this latter possibility unlikely in light of the likely minimal thermodynamic contribution of one or two Single Nucleotide Polymorphisms (SNPs) in the context of a $\approx 10^2$ nucleotide DCN [74, 128, 129, 193–196] sequence.

A similar point can be made regarding exposure to Uracil DeGlycosylase (UDG) enzymes and derivatives in the next section (i.e. section **6.5**), where we will work out that cytosine deamination to uracil (leading to a $C \longrightarrow U$ transition mutation) has roughly the same kinetics as ssDNA depurination.

(Conclusion 3): For methods where probe site-specific cleavage is used to indicating probe $\langle\rangle$ target complex formation or hybridization, there is a real danger of false-positive signal general via spontaneous cleavage, as this may be fatal at the level of GEP-DEAN [74] excess concentrations of probes with respect to their targets.

More specifically, in the context of the GEP-DEAN [74] approach where a $((10\ fmol)/(18\ zmol)) \approx 5.6 * 10^5$ fold excess of probes is used relative to a population of cDNA targets at a $\approx 0.61\ fM$ concentration, the number of false-positive signals arising from spontaneous cleavage after $\tau_{95^\circ C} \approx 495\ s$ at $\approx 95^\circ C$ and $\tau_{(95^\circ C\ to\ 40^\circ C, 0.1^\circ C/s)} \approx 550\ s$ spent during step **(s.1)** anneal from $\approx 95^\circ C$ to $\approx 40^\circ C$ at a rate of $\approx 0.1^\circ C/s$ (see section **6.3**) can be calculated as:

$$frac[probes\ transformed\ into\ false\ positive\ signals] \approx$$
$$\#(signal\ nucleotides) * (1 - e^{(-(k_{(ssDNA\ chain\ breakage\ per\ base\ (ave),95^\circ C)})*495\ s)} * e^{(-(k_{(ssDNA\ chain\ breakage\ per\ base\ (ave),mean::T\in[40^\circ C,95^\circ C])})*550\ s)}$$
$$\approx \#(signal\ nucleotides) * (1 - e^{(-(1.6*10^{-10}\ s^{-1})*495\ s)} * e^{(-(4.3*10^{-12}\ s^{-1})*550\ s)})$$
$$\approx \#(signal\ nucleotides) * (8.2 * 10^{-8})$$

Thus, if we set $\approx \#(signal\ nucleotides)$ to $\approx 10^2$, which is about the length of a typical DCN [74, 128, 129, 193–196] sequence, we find:

$$frac[probes\ transformed\ into\ false\ positive\ signals] \approx 8.2 * 10^{-6}$$

Multiplying this be the excess of probes relative to cDNA, mRNA, etc. targets, we can calculate:

$$fold\ excess\ of\ false\ positives\ relative\ to\ proper\ signals$$
$$\approx (8.2 * 10^{-6}) * (5.6 * 10^5) \approx 4.6$$

As we found in section **6.2**, this fold excess of false-positives is a serious problem since it places us at the limit where the number of correct probe $\langle\rangle$ target hybridization reporting signals is on-order the square root of the rate parameter for false-positive signal generation: $\sqrt{\lambda} = \sqrt{k_{(f.p.rxn)} * t} \approx \sqrt{4.6} \approx 2.1$

The problem obviously becomes much more severe if an abasic site cleavage step needs to be considered, where we would have to multiply the aforementioned average rate of false-positive generation at $\approx 95^\circ C$ by $(k_{(\beta-elimination,95^\circ C)})^{-1} \approx (7.2 * 10^{-4}\ s^{-1})^{-1} \approx 1.4 * 10^3\ s$ and the aforementioned rate of false-positive generation during the anneal by $(k_{(\beta-elimination,mean::T\in[40^\circ C,95^\circ C])})^{-1} \approx (1.3 * 10^{-4}\ s^{-1})^{-1} \approx 7.7 * 10^3\ s$.

6.5 The rates and consequences of transition mutations due to hydrolytic deamination

From (Lindahl & Nyberg, 1974) [116] we find a $\approx 95^{\circ}C$ "Lindahl Buffer A" [115] rate of cytosine deamination of $k_{(deamination,95^{\circ}C)} \approx (2.0 * 10^{-7}\ s^{-1})$, which is very close to our previous estimated value of $k_{(depurination,95^{\circ}C)} \approx (4.2 * 10^{-7}\ s^{-1})$ in section **6.4**. We note that (Lindahl & Nyberg, 1974) [116] also approximate an activation energy for ssDNA cytosine deamination of $E_a \approx 29kcal/mol$. Using **(Exp. 6.4.1)** we can therefore estimate a general formula for cytosine deamination as a function of temperature in "Lindahl Buffer A" [115] as:

(Exp. 6.4.2)

$$k_{(deamination,T(^{\circ}C),ssDNA)} \approx (2.0 * 10^{-7}\ s^{-1}) * e^{\left(\left(\frac{29kcal/mol}{R*(273.15+95^{\circ}C}\right)-\left(\frac{29kcal/mol}{R*(273.15+T(^{\circ}C))}\right)\right)} \approx 3.28358 * 10^{1}0 * e^{\left(-14593.4273.15+T(^{\circ}C)\right)}\ s^{-1}$$

(Importantly, one should only trust the output of the above expression to the ≈ 2 significant figures of the experimentally measured reference kinetic rate of $\approx (2.0 * 10^{-7}\ s^{(}-1))$ at $\approx 95^{\circ}C$.)

For experimental measurement of depyrimidination at lower temperatures, allowing up to test a reaction rate extrapolation via the use of **(Exp. 6.4.1)**, (Frederico et. al., 1990) [57] performed direct measurements of cytosine deamination at $\approx 37^{\circ}C$ and $pH \approx 7.4$, in the context of "Lindahl Buffer B" [115], and found rates of $k_{(deamination,37^{\circ}C,ssDNA)} \approx (1 * 10^{-10}\ s^{-1})$ and $k_{(deamination,37^{\circ}C,dsDNA)} \approx (7 * 10^{-13}\ s^{-1})$ for ssDNA and dsDNA, respectively. Here, as (Lindahl & Nyberg, 1974) [116] discovered, "Lindahl Buffer B" [115] only shifts $k_{(deamination,95^{\circ}C,ssDNA)}$ by $+0.2$ units and the absence of the standard $\approx 0.01\ M\ MgCl_2$ in "Lindahl Buffer B" [115] does not appear to effect the deamination reaction rate. Therefore, it seems reasonable to use (Frederico et. al., 1990)'s [57] values to validate the usage of **(Exp. 6.4.2)**. Shockingly, using **(Exp. 6.4.2)** we find a rate constant for the deamination of ssDNA of $PREDICTED\left[k_{(deamination,37^{\circ}C,ssDNA)} \approx (1.2 * 10^{-10}\ s^{-1})\right]$, which almost exactly matches the experimental value of $PREDICTED\left[k_{(deamination,37^{\circ}C,ssDNA)} \approx (1 * 10^{-10}\ s^{-1})\right]$ found by (Frederico et. al., 1990) [57]. Regarding the ≈ 143 fold slowdown (Frederico et. al., 1990) [57] observed for cytosine deamination in dsDNA vs. ssDNA at $\approx 37^{\circ}C$ and $pH \approx 7.4$, this is perhaps unsurprising considering that, when stacked with other bases in the context of duplex DNA, the nucleophilicity of cytosine's C4 carbon and/or accessibility of the nucleobase's C5=C6 bond should be considerably reduced.

So what do these results imply? Perhaps a few things:

First, let's consider the fraction of deaminated cytosine bases, i.e. $C \longrightarrow U$ transition mutations, after completion of all photo-DEAN [193–196] pre-amplification steps and the first round of DCN [74, 128, 129, 193–196] PCR (described in section **6.3**). Here, after spending $\tau_{(95^{\circ}C)} \approx 495\ s$ at $\approx 95^{\circ}C$ where the rate of cytosine deamination was again experimentally measured by (Lindahl &

Nyberg, 1974) [116] to be $k_{(deamination, 95^{\circ}C)} \approx (2.0 * 10^{-7}\ s^{-1})$, we find a fraction cytosine deamination of:

$$frac[cytosine\ deamination;\ 1\ cytosine\ base \mid (pH \approx 7.4) \mid \tau_{95^{\circ}C} \approx 495\ s]$$
$$= 1 - e^{(-k_{(deamination,\ 95^{\circ}C)} * t)} = 1 - e^{(-(2.0 * 10^{-7}\ s^{-1}) * 495\ s)} \approx 9.9 * 10^{-5}$$

And after the annealing step where the temperature is ramped from $\approx 95^{\circ}C$ to $\approx 40^{\circ}C$ at a rate of $\approx 0.1^{\circ}C/s$ (implying a total annealing time of $\approx (180\ s) + (95 - 39) * (10\ s) \approx (95 - 40) * (10\ s) \approx (180\ s) + (550\ s) \approx 730\ s$, we can use **(Exp. 6.2)** and **(Exp. 6.4.2)** to calculate the average rate of cytosine deamination of:

$$k_{(deamination,\ mean::T \in [40^{\circ}C,\ 95^{\circ}C])}$$
$$\approx (\frac{1}{95 - 40}) \int_{T=40}^{T=95} (3.28358 * 10^{10} * e^{(\frac{-14593.4}{273.15 + T})}\ s^{-1}) dT \approx 3.2 * 10^{-8}\ s^{-1}$$

And then calculate a fraction cytosine deamination due to the anneal:

$$frac[cytosine\ deamination;\ 1\ cytosine\ base \mid (pH \approx 7.4) \mid$$
$$\tau_{(95^{\circ}C\ to\ 40^{\circ}C,\ 0.1^{\circ}C/s)} \approx 550\ s] = 1 - e^{(-k_{(deamination,\ mean::T \in [40^{\circ}C,\ 95^{\circ}C])} * t)}$$
$$1 - e^{(-(3.2 * 10^{-8}\ s^{-1}) * 550\ s)} \approx 1.8 * 10^{-5}$$

Putting this together, after photo-DEAN [193–196] step **(s.1)**, the fraction of deaminated cytosine bases will only be $\approx 1 - (1 - 9.9 * 10^{-5}) * (1 - 1.8 * 10^{-5}) \approx 1.2 * 10^{-4}$. So, even if we had a 10^2 nucleotide fully poly(C) DCN sequence [74, 128, 129, 193–196] on a photo-DEAN [193–196] probe, only $\approx 1\%$ would have any $C \longrightarrow U$ transition mutations (which is again, on order the same value we found for the generation of abasic sites in **6.4**). Thus, because uracil is a "coding" base that should not block polymerase procession, and minor thermodynamic penalties to gene chip / etc. hybridization of DCN [74,128,129,193–196] sequences during the eventual photo-DEAN [193–196] "decoding" steps aside, cytosine deamination should not be of significant concern for the current photo-DEAN [193–196] protocol.

However, spontaneous deamination should be of enormous concern for alternative strategies where target deamination represents a critical component of chemically modifying a probe to indicate discovery / hybridization to a target and no affinity purification methods are employed. Consider that in the context of the GEP-DEAN [74], a $(10\ fmol)/(18\ zmol) \approx 5.6 * 10^5$ fold excess of "searcher" molecules is used to find what we estimate to be $> 50\%$ of the $N \approx 10840$ targets, and thus, that after photo-DEAN [193–196] step **(s.1)** we'll have a $\approx (1.2 * 10^{-4}) * (5.6 * 10^5) \approx 67$ fold excess of false-positives relative to target signals will be generated. This clearly places us in the "danger limit" (discussed in section **6.2**) where the number of correct probe $\langle\rangle$ target hybridization reporting signals is on-order the square root of the fold excess of false-positive signals: $\sigma = \lambda^{1/2} = (k_{(f.p.rxn)} * t)^{1/2} \approx 67^{1/2} \approx 8.2$.

Now, the only methods we are aware of for "targeted" deamination are those of Fujimoto et. al. [61,63,121], where either ^{CV}U [63,121] or ^{CNV}K [61] is used to form a (reversible) photoadduct via $[2 + 2]$ cycloaddition to a cytosine nucleobase. This

cycloaddition process directly saturates the pyrimidine base's C5=C6 double bond and presumably enhances the nucleophilicity of its C4 carbon [76,106] hosting the primary amine group (NH_2) group that differentiates cytosine from uracil (which has a double-bonded oxygen on its C4 carbon). Here, whether or not one uses $^{\text{CV}}$U [63, 121] or $^{\text{CNV}}$K [61] to catalyze cytosine deamination, the half-life of the cytosine C4 primary amine group is stated to be around $\approx 40\ min \approx 2.4 * 10^3\ s$ to $\approx 50\ min \approx 3.0 * 10^3\ s$ at $\approx 90^{\circ}C$ in $pH \approx 7$ buffer [61,63] (though a more specific half-life estimate of $\approx 42\ min \approx 2.52 * 10^3\ s$ is provided for the $^{\text{CV}}$U case, implying a rate constant of $k_{(^{CV}U-cat.cyt.deamination,\ 90^{\circ}C)} \approx 2.8 * 10^{-4}\ s^{-1}$ [63].

However, in contrast to the $^{\text{CV}}$U or $^{\text{CNV}}$K facilitated process, the rate of spontaneous cytosine deamination after $\approx 40\ min \approx 2.4 * 10^3\ s$ at $\approx 90^{\circ}C$ can be calculated as:

$$k_{(deamination,90^{\circ}C)} \approx (3.28358 * 10^{10} \times e^{(\frac{-14593.4}{273.15+T})}\ s^{-1}) \approx (1.2 * 10^{-7}\ s^{-1})$$

$$frac\left[cytosine\ deamination;\ 1\ cytosine\ base \mid (pH \approx 7) \mid \tau_{90^{\circ}C} \approx 2.4 * 10^3\ s\right] =$$

$$(1 - e^{(-k_{(deamination,90^{\circ}C))} * t)}) \approx (1 - e^{(-(1.2 * 10^{-7}\ s^{-1}) * (2.4 * 10^3)\ s)}) \approx 2.9 * 10^{-4}$$

In the context of the GEP-DEAN [74] approach, where up to a $(\frac{10\ fmol}{18\ zmol}) \approx 5.6 * 10^5$ fold excess of probes is used relative to a population of cDNA targets, this is bad news as it implies that, after the above "cooking procedure" for targeted deamination, and concerning ourselves with a particular cytosine base on each probe, even without photocrosslinking to $^{\text{CV}}$U or $^{\text{CNV}}$K the said cytosine will be deaminated on $\approx (5.6 * 10^5) * (2.9 * 10^{-4}) \approx 160$ fold the number of probes as target molecules. Once again we find that we are in the "danger limit" (again, discussed in section **6.2**) where in this case we have: $\sigma = \sqrt{\lambda} = \sqrt{k_{(f.p.rxn)} * t} \approx \sqrt{160} \approx 12.6$.

This may also be bad news specifically regarding the use of $^{\text{CV}}$U [63, 121] or $^{\text{CNV}}$K [61] as a genomic editing tool to (with reasonable kinetics) induce site-directed $C \longrightarrow U$ transition mutations (where "U" codes as a "T" in the context of deoxyribonucleic acid). Specifically, if the procedure of heating for $\approx 1\ h \approx 3600\ sat \approx 90^{\circ}C$ is employed (see ref. [63] for ^{CV}U, ref. [61] for $^{\text{CNV}}$K), this would imply that we would have a fraction of off-target or spontaneous cytosine deamination events of:

$$frac[uncatalyzed\ cyt.\ deamination\ (ssDNA);\ 1\ cytosine\ base \mid (pH \approx 7) \mid \tau_{90^{\circ}C} \approx 3.6 * 10^3\ s]$$

$$\approx (1 - e^{(-k_{(deamination,90^{\circ}C)} * t)}) \approx (1 - e^{(-(1.2 * 10^{-7}\ s^{-1}) * (3.6 * 10^3)\ s)}) \approx 4.3 * 10^{-4}$$

This means that for a $\approx 10kb$ genomic fragment, and after $\approx 3600\ s \approx 1\ h$ of heating at $\approx 90^{\circ}C$ in $pH \approx 7$ buffer, on average $\approx 4.3\ C \longrightarrow U$ transition mutations will spontaneously occur while the $^{\text{CV}}$U or $^{\text{CNV}}$K targeted cytosine is being deaminated.

However, some hope comes from Luyen et. al.'s [121] result where it was apparently demonstrated that a $\approx 5\%$ yield of targeted cytosine deamination could be achieved using ^{CV}U at $\approx 37^{\circ}C$ in $pH \approx 7$ buffer after $\approx 48\ h \approx 1.728 * 10^5\ s$, implying a rate constant: $k_{(^{CV}U-cat.cyc.deamination,37^{\circ}C)} \approx 3.0 * 10^{-7}\ s^{-1}$ and a half-life for the

process of $\approx 2.3{*}10^6\ s \approx 27\ days$. Taking the value for spontaneous deamination of dsDNA at $\approx 37^{\circ}C$ in "Lindahl Buffer B" [115] at $pH \approx 7.4$ measured by (Frederico et. al., 1990) [57], $k_{(deamination,37^{\circ}C,dsDNA)} \approx (7 * 10^{-13}\ s^{-1})$, the corresponding rate of cytosine deamination over this time interval of $\approx 2.3 * 10^6\ s \approx 27\ days$ can be calculated as:

$$frac[uncatalyzed\ cyt.\ deamination\ (ssDNA);\ 1\ cytosine\ base \mid (pH \approx 7) \mid \tau_{37^{\circ}C} \approx 2.3 * 10^6\ s] \approx (1 - e^{(-k_{(deamination,37^{\circ}C)} * t)}) \approx (1 - e^{(-(7*10^{-13}\ s^{-1}) * (2.3*10^6)\ s)}) \approx 1.6 * 10^{-6}$$

In any case, this means that, after $\approx 2.3 * 10^6\ s \approx 27\ days$ of heating at $\approx 37^{\circ}C$ in $pH \approx 7$ buffer to match the calculated half-life for the ^{CV}U catalyzed targeted cytosine deamination process, we'll only have on average $\approx$ 1.6 spontaneous cytosine deamination events per $\approx 1Mb$ (i.e. ≈ 1 *megabase*) of genomic DNA. Importantly, the rest of the genomic DNA should remain double-stranded (i.e. in native form) as (Frederico et. al., 1990) [57] also found a rate for spontaneous deamination in ssDNA of $k_{(deamination,37^{\circ}C,ssDNA)} \approx 1{*}10^{-10}\ s^{-1}$, which is ≈ 143 fold faster than the rate for dsDNA.

While this kind of a rate for the spontaneous deamination process seems reasonable for an (arbitrarily parallelizable) means of targeting genomic $C \longrightarrow U$ transition mutations in genomic DNA, the ≈ 1 *month* long reaction half-life is less reasonable, and in particular, seems to mostly rule out the suggestion by Luyen et. al. [121] that ^{CV}U catalyzed cytosine deamination can serve as a reasonable mRNA editing platform. However, we do note that targeted deamination appears to proceed at a ≈ 2 fold higher rate for RNA relative to DNA [121].

6.6 RNA transesterification, other (pseudo)random cleavage processes, and their consequences

At the temperatures and buffer pH ranges employed for the DigiTag [128], DigiTag2 [129], GEP-DEAN [74], and photo-DEAN [193–196] assays, transesterification should serve as the primary mechanism for RNA decay (deoxyribonucleic acid exhibits $\approx 10^5$ fold enhanced stability to hydrolysis relative to ribonucleic acid under physiological conditions) [149].

Of enormous convenience, Li & Breaker [110] provide a general formula for calculating the rate of ssRNA transesterification (assuming well-mixed sequence composition) under a variety of ionic strength and temperature conditions, which tends to agree with experimental measured kinetic data within a factor of ≈ 3.5 or less (see "Table 1" in Li & Breaker [110]) for monovalent and divalent molar ion concentrations in the range $0.03\ M \leq \left[K^+\right] \leq 3.16\ M$ and $0.005\ M \leq \left[Mg^{2+}\right] \leq 0.05\ M$, respectively. With some minor modifications (e.g. to account for ssRNA polynucleotide length, to yield a rate in s^{-1} as opposed to min^{-1}, and to account for the case where $\left[Mg^{2+}\right] \approx 0$) the relevant equation, i.e. Li & Breaker's [110] "equation e", is as follows:

If $\left[Mg^{2+}\right] \approx 0$ (**Exp. 5.1**):

$$k_{(RNA\ transesterification)} \approx (L-1) \times k_{ref} \times 10^{(0.983*(pH-6))} \times 10^{(-0.24*(3.16-[K^+]))} \times 10^{(0.07*(T_1-23))}$$

If $\left[Mg^{2+}\right] > 0$ (**Exp. 5.2**):

$$k_{(RNA\ transesterification)} \approx (L-1) \times k_{ref} \times 10^{(0.983*(pH-6))} \times 10^{(-0.24*(3.16-[K^+]))} \times 69.3 * \left[Mg^{2+}\right]^{0.80} \times 3.57 * \left[K^{2+}\right] \times 10^{(0.07*(T_1-23))}$$

Where $k_{ref} \approx 2.17 * 10^{-11}\ s^{-1}$ :: *measurement conditions* $\approx \{T_0 \approx 23^\circ C;\ pH \approx 6;\ \left[K^+\right] \approx 3.16\ M;\ \left[Mg^{2+}\right] \approx 0\ M\}$ [110]; L is the length in nucleotides of the ssRNA species of interest (we subtract one from this value in the above expression to avoid the fence-post error); and values for $\{T_1(^\circ C);\ pH;\ \left[K^+\right];\ \left[Mg^{2+}\right]\}$ can be specified as desired (being mindful of the aforementioned suggested ionic strength ranges) to predict $k_{(RNA\ transesterification)}$ for buffer and temperature conditions of interest. Note that, while it should be safe to assume that $\left[K^+\right]$ and $\left[Na^+\right]$ are roughly interchangeable as the monovalent ion species, we suggest that more care should be taken if divalent ions other than $\left[Mg^{2+}\right]$ are present in a given reaction buffer.

Concerning ourselves with step **(s.1)** (described in **(II.3)**) we analyze the expect fraction of ssRNA transesterification after the photo-DEAN [193–196] reaction volume is held at $\approx 95^\circ C$ for $\approx 180\ s$, and then after the probe $\langle\rangle$ RNA target annealing step where the temperature is ramped from $\approx 95^\circ C$ to $\approx 40^\circ C$ at a rate of $\approx 0.1^\circ C/s$ (implying a total annealing time of $\approx 180\ s + (95-39) * 10\ s \approx (95-40) * 10\ s \approx 180\ s + 550\ s \approx 730\ s$).

First, for the $\approx$ 180 s at $\approx$ $95^\circ C$ in $\approx$ $1x$ TE buffer: [0.01 M TE, 0.02 M DTT, 0.150 M $NaCl$, $pH \approx 8$], which implies a $\left[Na^+\right] \approx 0.150\ M$ monovalent ion concentration and the absence of any divalent ions, from **(Exp. 6.5.1)** we find an RNA dinucleotide transesterification rate of: $k_{(RNA\ transesterification, (95^\circ C, pH \approx 8), no\ Mg^{2+})} \approx 4.17 * 10^{-5}\ s^{-1}$.

This tells us that after $\approx$ 180 s at $\approx 95^{\circ}C$ we should have a fraction ssRNA dinucleotide cleavage of:

$$frac[transesterification;\ 2nt\ RNA \mid (pH \approx 8) \mid \tau_{95^{\circ}C} \approx 180\ s]$$
$$\approx (1 - e^{(-k_{(RNA\ transesterification, (95^{\circ}C, pH \approx 8),\ no\ Mg^{2+}))} * t)})$$
$$\approx (1 - e^{(-(6.2 * 10^{-9}\ s^{-1}) * 180\ s)} \approx 7.48 * 10^{-3}$$

This further implies that a $\approx 1\ kb$ tract of ssRNA (and any secondary structure will be mostly denatured at $\approx 95^{\circ}C$) will be cleaved $\approx$ 7.48 times. To understanding the contribution of divalent metal to the RNA transesterification rate, note that if only $\approx$ 0.005 M of Mg^{2+} ions are added to the reaction buffer the rate of cleavage increases by approximately an order of magnitude to: $k_{(RNA\ transesterification, (95^{\circ}C, pH \approx 8),\ 0.005\ M\ Mg^{2+}))} \approx 3.30 * 10^{-4}\ s^{-1}$

This yields a fraction cleavage of ssRNA dinucleotides of $\approx 7.31 * 10^{-2}$ and implying $\approx$ 57.7 cleavage events along the length of the same $\approx 1kb$ ssRNA transcript.

Next, for the anneal in $\approx 1x\ TE$ buffer from $\approx 95^{\circ}C$ to $\approx 40^{\circ}C$ at a flat rate of $\approx 0.1^{\circ}C/s$, we can calculate an average rate of transesterification of:

$$k_{(RNA\ transesterification, mean::T \in [40^{\circ}C, 95^{\circ}C])}$$
$$\approx \left(\frac{1}{95 - 40}\right) \int_{T=40}^{T=95} [(Exp.6.5.1)]\ dT \approx 4.70 * 10^{-6}\ s^{-1}$$

This implies that the anneal leads to a fraction cleavage of ssRNA dinucleotides of:

$$frac[transesterification;\ 2nt\ RNA \mid (pH \approx 8) \mid \tau_{(40^{\circ}C\ to\ 95^{\circ}C,\ 0.1^{\circ}C/s)} \approx 550\ s]$$
$$\approx (1 - e^{(k_{(RNA\ transesterification,\ mean::T \in [40^{\circ}C\ to\ 95^{\circ}C])} * t)})$$
$$\approx (1 - e^{(-4.70 * (10^{-6}\ s^{-1}) * 550\ s)}) \approx 2.58 * 10^{-3}$$

Thus, the aforementioned anneal should statistically lead to $\approx$ 2.58 transesterification cleavage events along a $\approx 1\ kb$ tract of ssRNA, which is within the same order of magnitude as the $\approx$ 7.48 cuts we expect for holding at $\approx 95^{\circ}C$ for $\approx 180\ s$.

We quickly note that the rate of transesterification will increase or decrease by $\approx$ 1 order of magnitude for an increase or decrease of one buffer pH unit, respectively (regardless of the divalent metal concentration). Thus, we suggest that one can use buffer pH as an effective means of tuning the rate of RNA transesterification.

So what do the previous derivations for expected RNA transesterification rates tell us?

To begin to answer this question, consider that after photo-DEAN [193–196] step **(s.1)**, which employs a $\approx 95^{\circ}C$ incubation (for $\approx 3\ min \approx 180\ s$) and anneal from $\approx 95^{\circ}C$ to $\approx 40^{\circ}C$ (for $\approx 550\ s$), we're going to cut a $\approx 1\ kb$ tract of ssRNA around $\approx 10^3 * (1 - (1 - 7.48 * 10^{-3}) * (1 - 2.58 * 10^{-3})) \approx 10.0$ times, which we argue is non-negligible. If we estimate the "target region" on an mRNA transcript to be a contiguous stretch of $\approx 30\ nts$, we can correspondingly estimate that a single target will survive step **(s.1)** with a probability of only around $(1 - 7.48 * 10^{-3})^{29} * (1 - 2.58 * 10^{-3})^2 9 \approx 74.6\%$. In order words, we're cutting up and destroying $\approx 25.4\%$ of our targets via the process meant to assist molecular probe hybridization to their targets.

Regarding the ability to compensate for this false-negative rate with proper control experiments, we need to note that the number of cleaved target sequences is going to be approximately Poisson distributed (as discussed in section **6.2**), and as such that the inherent variance associated with a Poisson distribution should serve as a fundamental source of noise for the assay measurement. Here, let N be the number of "target sites" (i.e. photo-DEAN [193–196] molecular probe targets), where $\mu \approx 0.254 * N$ gives the number of cleaved or destroyed targets in the previous example. The Poisson distribution for the number of destroyed targets via transesterification implies, for example, that there is a $\approx 31.73\%$ chance for a divergence from the expected mean by more than $\sigma = \sqrt{\mu} \approx \sqrt{0.254 * N}$.

First, let;s call $\sigma = \sqrt{\mu} \approx \sqrt{0.254 * N}$ small relative to N, which is just another way of saying that N is "sufficiently large". To pick a good example, we can set $N \approx 10,840$, such that N corresponds to the number of target molecules at the lowerbound sensitivity range of the GEP-DEAN [74] assay. This yields $\sigma = \sqrt{\mu} \approx \sqrt{0.254 * 10840} \approx 52.5$ target molecules, which is certainly a reasonably small standard deviation to accept as a negligible contribution to the false-negative rate. Even if we set $N = 10$, we would only expect a Poisson distribution standard deviation of $\sigma = \sqrt{\mu} \approx \sqrt{0.254 * 10} \approx 1.59$ target molecules.

Thus, with the use of proper control experiments, the unavoidable variance in the number of randomly cleaved RNA target sequences should minimally contributed to the photo-DEAN [193–196] / etc. assay noise floor. Be aware though that the target copy number depression may bring the final number of uncleaved targets within the same order-of-magnitude as the standard deviation for other false-negative (e.g. nucleic acids sticking to the walls of a polypropylene test tube [18]) or potentially false-positive generating reactions. Target copy number depression is always a "bad thing" and should be appropriately attenuated.

A further point of concern is that we cannot necessarily assume that RNA transesterification is "sufficiently" (pseudo)random. In particular, consider that RNA secondary structure can significantly perturb transesterification rates [110], and as mentioned previously, even in the context of ssRNA some dinucleotide pairs will deviate up to ≈ 3.5 fold from the average rate of transesterification predicted by Li & Breaker's [110] "equation e" (i.e. our **(Exp. 6.5.1)** and **(Exp. 6.5.2)**). Care should therefore be taken to account for the (pseudo)randomness of spontaneous transesterification.

Finally, it's important to note that the fragment lengths produced by $Q * p_{(RNA\ transesterification)}$ cleavage events along long tracts of RNA (e.g. $\approx 1\ kb$ tracts of mRNA) with mixed sequence composition should not be assumed to have a simple exponential distribution. Here, due to the correlation between

secondary structure and RNA length, one needs to carefully analyze how this effects assay-to-assay variability at the limit of low target copy numbers. For an intuitive probabilistic model for the expected counts of each RNA fragment length (following some amount of backbone cleavage events via transesterification) by transforming the problem from its current form into one of calculating the number of expected "streaks" after some number of sequential flips of a biased coin.

Specifically, following earlier discussions on spontaneous RNA cleavage via transesterification, notice that the "survival probability" of an RNA internucleotide linkage can be written as:

$$\begin{aligned} p &= Prob[Intact\ RNA\ internucleotide\ linkage] \\ &\approx 1 - Prob[RNA\ internucleotide\ linkage\ transesterification] \\ &\approx e^{(-k_{(RNA\ transesterification, T^{\circ}C)} * \tau_{(RNA\ transesterification)})} \end{aligned}$$

We can therefore model the fragmentation of an RNA polynucleotide after some time $\tau_{(RNA\ transesterification)}$, under conditions to yield some rate of transesterification $k_{(RNA\ transesterification, T^{\circ}C)}$, by flipping a biased coin for each RNA internucleotide linkage with a probability $p = Prob[Intact\ RNA\ linkage]$ of yielding "heads", where we allow the internucleotide linkage to remain intact, and a $1-p$ probability of yielding "tails", where we cut the internucleotide linkage.

-

Here's our proposed probabilistic model for the fragment length counts after an RNA polymer is subject to conditions conducive to transesterification:

Take a biased coin, with probability p of yielding "heads" (i.e. H) and probability $1-p$ of yielding "tails" (i.e. T), and toss it $n+2$ times. Write down each result (H or T) as sequential elements in a one-dimensional array (or a string), and regardless of the actual result for the first and last coin toss, write down the outcome in either case as tails (i.e. T). For example, if we toss the biased coin $(n+2) = (5+2) = 7$ times, we might have the result: $\{T, T, H, T, H, H, H\}$, and after transforming the first and last coin toss to give the outcome of tails (i.e. T), we would write this result down as: $\{T, T, H, T, H, H, T\}$.

We now define a "streak" of length k where $k \in \{1, 2, 3, ..., n\}$ as an instance where we have a subarray (or substring) of length $k+2$ of the form: $\{T, ..., T\}$, where "..." represents k sequential H characters in a row (e.g. a $k = 3$ streak implies the existence of the subarray or substring $\{T, H, H, H, T\}$). However, we need to define a length $k = 0$ streak in a special way as the total count of (potentially overlapping) instances where the coin lands "tails up" twice in a row (i.e. where we see the subarray or substring $\{T, T\}$). For example, if we toss our biased coin once again $(n+2) = (5+2) = 7$ times, we might see a result that looks like $\{T, H, T, T, T, H, T\}$, where the first and last toss must be recorded as "tails" (i.e. T). Here, there are two (overlapping) instances where subarray or substring $\{T, T\}$ appears, and so we would have two runs of length $k = 0$. Provided this set up, we can now ask: what are the expected counts $\{c_0, c_1, c_2, ..., c_n\} \in C$ for non-overlapping streaks of "heads" (i.e. sequential H characters) of length k for $k \in \{0, 1, 2, ..., n\}$?

So how do we map the answer to the above question back to the original problem of looking for counts of the various RNA fragment lengths after some time $\tau_{(RNA\ transesterification)}$, during which transesterification occurs at some rate $k_{(RNA\ transesterification, T^{\circ}C)}$ along the linkages of a larger RNA polymer?

First notice that a streak of k heads (i.e. sequential H characters in the array or string where we record coin tosses) corresponds to an RNA fragment of length $L = (k + 1)$ nucleotides held together by k intact internucleotide linkages. Notice that the matter of recording the first and last coin toss as "tails" (i.e. T) is just to allow for proper treatment of the ends of the RNA polymer, where the first and last nucleotides can be (conceptually) thought of as being immediately upstream and downstream of a cleaved internucleotide linkage. Finally, notice that our somewhat contrived definition for a streak of length $k = 0$ corresponds to a situation where RNA transesterification cuts both immediately upstream and downstream of a single nucleotide base, liberating it from the polymer. Thus, we have that the counts $\{c_0, c_1, c_2, ..., c_n\} \in C$ for streaks of length $k \in \{0, 1, 2, ..., n\}$ as counts for RNA fragments of length $L \in \{0, 1, 2, ..., n + 1\}$ nucleotides when one assigns a transesterification survival probability of p to each internucleotide linkage.

We provide a Mathematica (v10.4.1.0) script (below), which employs the described probabilistic model, and will approximate expected counts for RNA fragments of length $\{0, 1, 2, ..., Q\}$ nucleotides, generated via assigning a uniform probability of transesterification based cleavage to each internucleotide linkage in an original length Q nucleotide RNA polymer. For this script, the probability of transesterification-based cleavage per internucleotide linkage is set via specification of the variable "pTransesterification" and the value Q for the length in nucleotides of the original RNA polynucleotide can be set by specifying the value of the parameter "mRNALength". One must also specify the number of simulation iterations with parameter "numTestIterations". Here, for convenience, simulation progress (in terms of the number of iterations thus far) can also be reported at desired intervals by specifying the parameter "progressUpdateInterval".

This script will output: (1) count weighted mean (μ) and median ($\tilde{x}$) fragment lengths, as well as the standard deviation (σ) for fragment lengths (all parameters are computed from simulation data); (2) an output list of expectation values for fragments of length $[1, Q]$ (i.e. all possible fragment lengths) where each element $\{L, count\}$ indicates the number of counts for a particular length L fragment; and (3) plots of the fragment length distribution with linearly scaling and logarithmically scaling y-axis tick-mark values.

```
(* SCRIPT START *)

mRNALength=10^3;
pTransesterification=(9.61*10^(-3));
numTestIterations=10^5;
progressUpdateInterval=10^2;
numberOfCoinTosses=(mRNALength-1);
pTails=pTransesterification;
pHeads=1-pTails;
streakCountArray=Array[{#-1,0}&,numberOfCoinTosses+1];

For[testIterations=1,testIterations<=numTestIterations,
    testIterations++,

    testString=StringJoin[Join[{"0"},
    RandomChoice[{pTails,pHeads}->{"0","1"},
```

```
    numberOfCoinTosses],{"0"}]];

    streakCountArray[[1,2]]+=StringCount[testString,
    "00",Overlaps->All];

    For[i=1,i<=StringLength[testString]-2,i++,
        streakCountArray[[i+1,2]]+=
        StringCount[testString,StringJoin[{"0"},
        Table[{"1"},{b,1,i}],{"0"}]];
    ];

    If[Mod[testIterations,progressUpdateInterval]==0,
        Print["Iterations thus far: ",testIterations]
    ];

];

streakCountArray={#[[1]],#[[2]]/(testIterations-1)}
&/@streakCountArray;

rnaFragmentLengthArray=#+{1,0}&/@streakCountArray;

weightedFragmentLengthData=
WeightedData[rnaFragmentLengthArray[[All,1]],
rnaFragmentLengthArray[[All,2]]];

Print[]
Print["Mean fragment length (\[Mu]): ",
N[Mean[weightedFragmentLengthData],12]]
Print["Median fragment length (",
Overscript["x","~"],"): ",
N[Median[weightedFragmentLengthData],12]]
Print["Standard deviation of fragment lengths
(\[Sigma]): ",
N[StandardDeviation[weightedFragmentLengthData],12]]
Print[]
Print["Predicted counts for all mRNA fragment
lengths."]
Print["Format :: {fragment length (nts),
expected count}"]

rnaFragmentLengthArray
ListPlot[rnaFragmentLengthArray,PlotStyle->
PointSize[0.02],PlotRange->All]
ListLogPlot[rnaFragmentLengthArray,PlotStyle->
PointSize[0.02],PlotRange->All]

(* SCRIPT END *)
```

For the above script, please note that we have pre-specified a value for "mRNALength" of $Q = 1\ kb$ (i.e. $Q = 10^3$ nucleotides) for the length of the RNA polynucleotide undergoing transesterification, implying that this polymer will, by default, have $10^3 - 1 = 999$ internucleotide linkages. We have also assigned a default value to the variable "pTransesterification", i.e. the probability of RNA internucleotide cleavage, of $p \approx (1-(1-7.16*10^{-3})*(1-2.47*10^{-3})) \approx 9.61*10^{-3}$,

which is meant to correspond to expected fraction of RNA transesterification after photo-DEAN [193–196] step **(s.1)** (see section **6.3** and our earlier discussion in this current section). Please also note that the number of simulation iterations "numTestIterations" has a default value of 10^5.

Now, putting the (above) script to use - specifying $R = 10^5$ trials, a 1 kb RNA transcript with $10^3 - 1 = 999$ internucleotide linkages, and setting the expected probability of internucleotide survival or cleavage of $p = \left(\frac{1}{2}\right)$ (i.e. where we are tossing a fair coin in our probabilistic model) - we find the following expected counts for fragment lengths:

$$Data format - - - c_L(k = simulation\ streak\ length,\ L)$$
$$c_0(k = 0, L = 1) : 250.486650000$$
$$c_1(k = 1, L = 2) : 100.121320000$$
$$c_2(k = 2, L = 3) : 55.6006900000$$
$$c_3(k = 3, L = 4) : 29.3849800000$$
$$c_4(k = 4, L = 5) : 15.1302900000$$
$$c_5(k = 5, L = 6) : 7.65834000000$$
$$c_6(k = 6, L = 7) : 3.85638000000$$
$$c_7(k = 7, L = 8) : 1.93903000000$$
$$c_8(k = 8, L = 9) : 0.967610000000$$
$$c_9(k = 9, L = 10) : 0.487570000000$$
$$c_{10}(k = 10, L = 11) : 0.242780000000$$
$$c_{11}(k = 11, L = 12) : 0.121020000000$$
$$c_{12}(k = 12, L = 13) : 0.0602100000000$$
$$c_{13}(k = 13, L = 14) : 0.0303400000000$$
$$c_{14}(k = 14, L = 15) : 0.0153200000000$$
$$c_{15}(k = 15, L = 16) : 0.00756000000000$$
$$c_{16}(k = 16, L = 17) : 0.00411000000000$$
$$c_{17}(k = 17, L = 18) : 0.00197000000000$$
$$c_{18}(k = 18, L = 19) : 0.000950000000000$$
$$c_{19}(k = 19, L = 20) : 0.000510000000000$$
$$c_{20}(k = 20, L = 21) : 0.000220000000000$$
$$c_{21}(k = 21, L = 22) : 0.000100000000000$$
$$c_{22}(k = 22, L = 23) : 0.0000700000000000$$
$$c_{23}(k = 23, L = 24) : 0.0000500000000000$$
$$c_{24}(k = 24, L = 25) : 0$$
$$c_{25}(k = 25, L = 26) : 0$$
$$c_{26}(k = 26, L = 27) : 0$$
$$c_{27}(k = 27, L = 28) : 0$$
$$c_{28}(k = 28, L = 29) : 0.0000100000000000$$

As might be obvious from the non-zero data point at $c_{28}(k = 28, L = 29)$, and perhaps fitting with intuition, for $(n+2) = 1001$ coin tosses (albeit where the first

and last coin tosses are rigged to yield tails), this simulation has a long convergence time. If not for lack of computational resources, the number of random simulation trials, R, should have been a few orders of magnitude larger than the current value of $R = 10^5$.

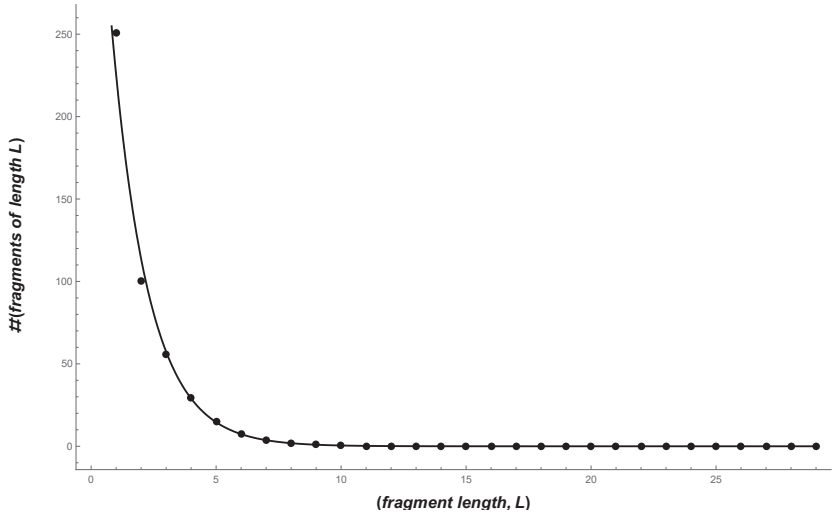
Without consideration of the outlier point at $c_{28}(k = 28, L = 29)$, we find a reasonable fit to the simulation data with the expression:

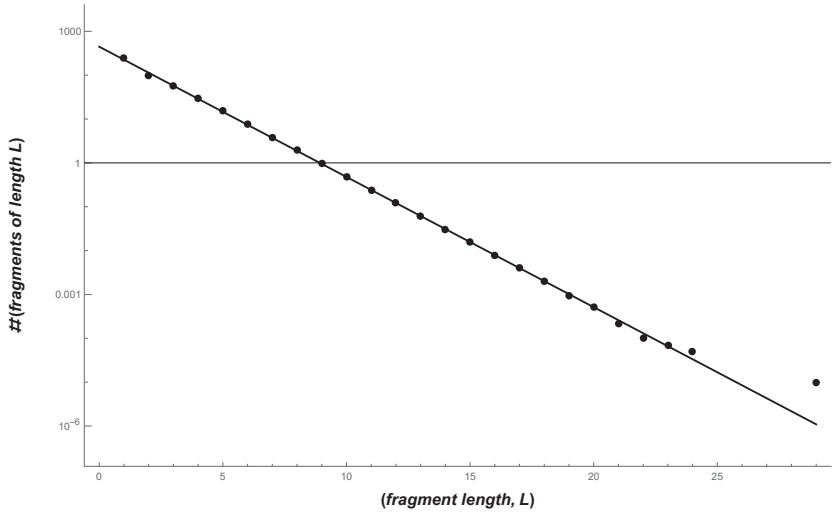
$$\#(fragments\ of\ length\ L) \approx e^{(6.101249 - 0.683838 * L)}$$

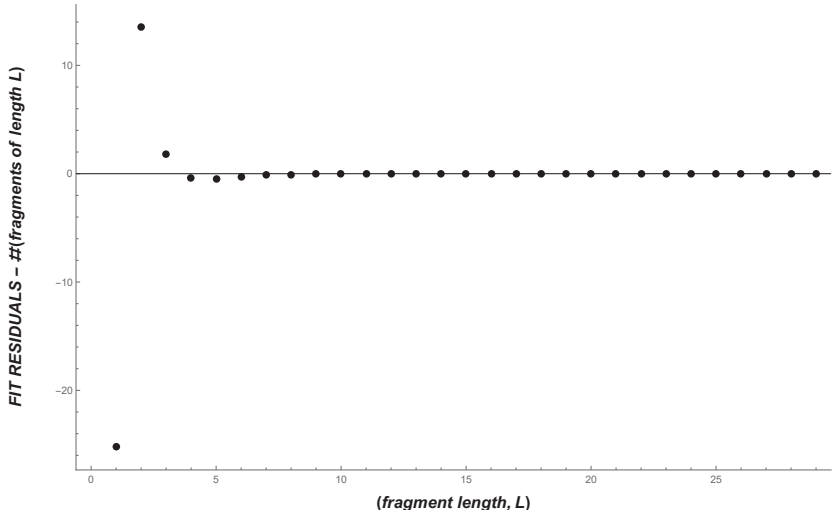
Furthermore, from the data (including the outlier point at $c_{28}(k = 28, L = 29)$) we find:

$\mu_{(fragment\ length)}(mean\ fragment\ length) \approx 1.9707204$
$\tilde{x}_{(fragment\ length)}(median\ fragment\ length) \approx 1.000000$
$\sigma_{(fragment\ length)}(standard\ deviation\ of\ fragment\ lengths) \approx 1.8014077$

We can plot our simulation data for the "fair coin" results and allow the reader to notice the clearly Gamma or exponential distribution of the expected RNA fragment counts as a function of fragment length L (measured in nucleotides):







For convenience, in Mathematica notation (where each element $\{L, count\}$ indicates the number of counts for a particular length L fragment), we can write these simulation results as:

```
{{1,5009733/20000},
{2,2503033/25000},
{3,5560069/100000},
{4,1469249/50000},
{5,1513029/100000},
{6,382917/50000},
{7,192819/50000},
{8,193903/100000},
{9,96761/100000},
{10,48757/100000},
```

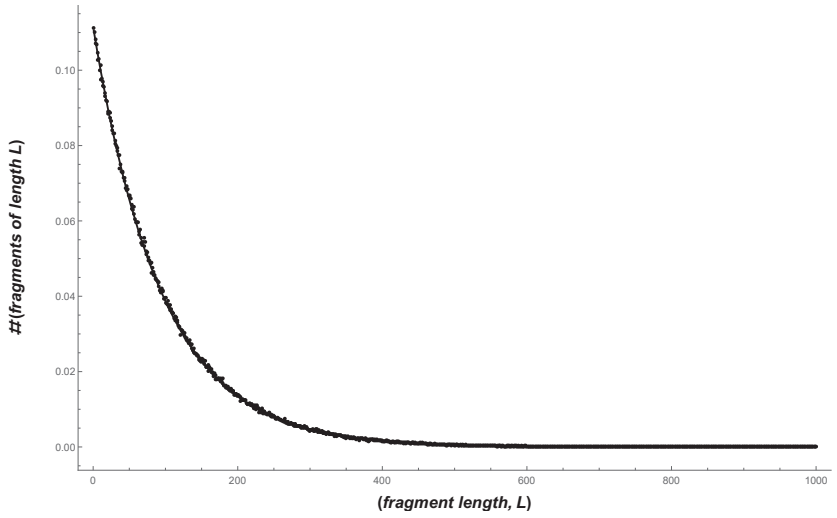
```
{11,12139/50000},
{12,6051/50000},
{13,6021/100000},
{14,1517/50000},
{15,383/25000},
{16,189/25000},
{17,411/100000},
{18,197/100000},
{19,19/20000},
{20,51/100000},
{21,11/50000},
{22,1/10000},
{23,7/100000},
{24,1/20000},
{25,0},
{26,0},
{27,0},
{28,0},
{29,1/100000}};
```

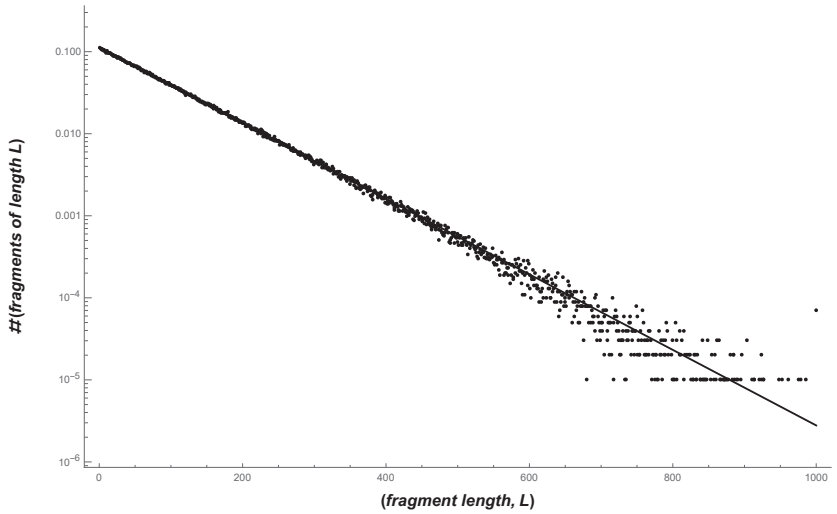
Now, setting an expected probability of internucleotide survival or cleavage of $p \approx (1-(1-7.16*10^{-3}*(1-2.47*10^{-3})) \approx (9.61*10^{-3})$, corresponding to expected fraction of RNA transesterification after photo-DEAN [193–196] step **(s.1)** (see section **6.3**), here we have a far greater distribution of product fragment sizes as well as almost perfect convergence (after $R = 10^5$ trials) to a fitted exponential decay function of the same form as the one previously used to fit the "fair coin" simulation results.

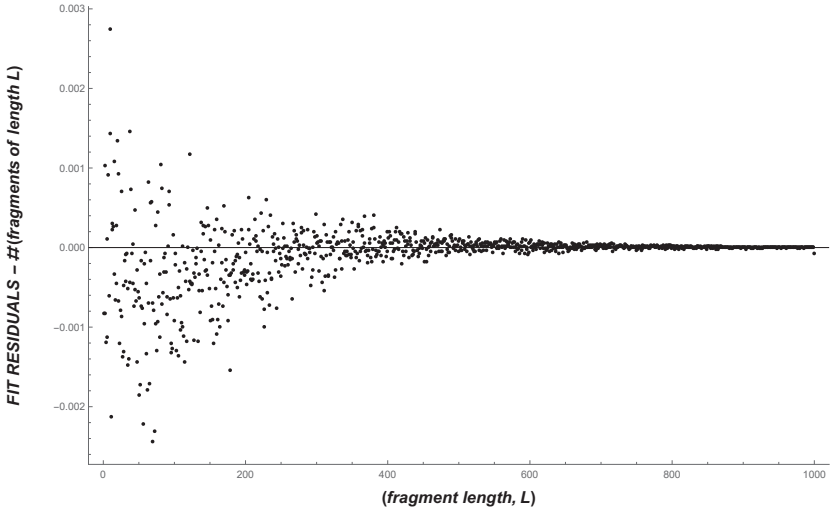
Without needing to discard any points as outliers, we find a reasonable fit to the simulation data with the expression:

$\mu_{(fragment\ length)}(mean\ fragment\ length) \approx 94.598602$
$\tilde{x}_{(fragment\ length)}(median\ fragment\ length) \approx 66.000000$
$\sigma_{(fragment\ length)}(standard\ deviation\ of\ fragment\ lengths) \approx 93.222862$

We can now plot our simulation data for these "biased coin" results and allow the reader to notice the clearly Gamma or exponential distribution of the fragment counts as a function of length L in nucleotides:


0.10
0.08
0.06
0.04
0.02
0.00
0
200
400
600
800
1000
#(fragments of length L)
(fragment length, L)


0.100
0.010
0.001
10^-4
10^-5
10^-6
0
200
400
600
800
1000
#(fragments of length L)
(fragment length, L)



There is, however, an important caveat regarding these simulation results. Consider a situation where we have a linear RNA polymer of some length L nucleotides, and we randomly cleave this polymer at some number of points by (as previously discussed) assigning each of the $(Q-1)$ internucleotide linkages in the polymer some approximately constant probability of transesterification. Consider a contiguous sequence of S nucleotides composing a photo-DEAN [193–196] "target". In this section (i.e. section **6.6**) we have already discussed the survival probability for this contiguous target sequence. However, conditioned on the survival of the target sequence, what probability will a particular length $L \leq Q$ RNA fragment (generated via the aforementioned transesterification process) have of encoding the target sequence, and how does this scale as a function of the fragment's length L? The naïve answer is that that probability $p_{(L\ nts, target\ encoding)}$ will be non-zero only for fragments of length $L \geq S$, and will then scale linearly will L as:

$$p_{(L\ nts, target\ encoding)} = \left(\frac{L-S+1}{\Sigma_{i \geq S}^{Q}(i-S+1)} \right) = \left(\frac{2*(L-S+1)}{(Q-S+1)(Q-S+2)} \right)$$

The argument is simply that, walking from $5'$ to $3'$ along a particular fragment, each of the fragment's $\{1, 2, ..., (L-S+1)\}$ nucleotides should have the same probability of being the first nucleotide in the target sequence.

Thus, it's perhaps worth keeping in mind the obvious point that target sequences will have a bias to appear on longer RNA fragments. We suggest that this will, in turn, have consequences for how many times one needs to cut the original Q nucleotide RNA polymer in order to sufficiently drive down the secondary structure around target sequence(s) to allow for kinetically and thermodynamically favorable photo-DEAN [193–196] probe hybridization. The reader should remember here that increasing the specified mean number of cuts will mean a corresponding increase in the probability of undesired target cleavage events (i.e. false-negative events).

6.7 *I just tried being a little blue*: Cyclobutane Pyrimidine Photodimers (CPDs) and other photochemical terrors at $\approx 254\ nm$

If the reader would be so kind as to take a bright flashlight (perhaps one of those new LED ones), press it into the palm of their hand, and turn it on, he or she will notice a faint red glow emanating from the backside of their hand. This simple experiment yield a rather strong clue that "red" light penetrates more deeply into biological tissue than "blue" light, and importantly for our purposes in this chapter, that the higher energy photons further along towards the "blue" or "UV" end of the spectrum (which made the initial flashlight beam "white") are being absorbed by some set of proteins, lipids, nucleic acids, or other biomolecules. We can further notice that even a single 500 nm photon of "green" light corresponds to $\approx 2.48\ eV$ of energy, which is the same order of magnitude as the energy of hydrogen or covalent bonds.

To now arrive at the point, the irradiation photo-crosslinking step **(s.2)** in Yokomori et. al.'s photo-DEAN [193–196] scheme (see section **6.3**) is not necessarily benign. Specifically, the cyclobutane products formed in the ^{RV}U (where "R" = "CN", "COOH", etc.; "R" = "COOH" in the case of photo-DEAN [193–196]) [62–64, 121, 126, 131–135, 199, 200],[s1], or roughly similar p-^{CV}p, ^{VZ}A, or ^{CNV}K catalyzed [2 + 2] photocycloaddition reactions also readily occur in unmodified nucleic acids after absorption of UVB and UVC band [h8] radiation, where $\approx$ 254 nm seems to be one most repeatedly cited "culprit" wavelength. Cyclobutane Pyrimidine Photodimers (CPD)s, particularly the cis-syn diastereoisomer variants of these dimers, are in fact some of the most common (^{UV}B, ^{UV}C)-induced lesions in nucleic acids [20, 36, 142, 145, 150, 192, 204],[h4], and purine-pyrimidine cyclobutane photoadducts, e.g. the d(TpA) photodimer [27–29, 47, 104, 205] where [2 + 2] cycloaddition takes place over the C5=C6 bonds of thymine and adenine, have also been noted to occur albeit at much reduced yield [70],[h5]. Like the reactions between ^{VZ}A or ^{CNV}K and their pyrimidine targets, many [2 + 2] photocycloaddition dimers involving the C5=C6 double bond in thymine are reversible via irradiation at $\approx$ 254 nm, though pyrimidine-purine adducts like d(TpA) appear to occur irreversibly [28]. It has been argued that the irreversibility of these latter adducts arises due to ring opening or other secondary reactions [47, 205], perhaps catalyzed by the high $\approx 26.3\ kcal/mol$ cyclobutane ring strain (see "Figure 2.15" of ref. [8]).

From this natural precedent for internucleotide photoadduct formation, we can understand the importance that photocrosslinking chemistries suggested for use in the context of nucleic acids (e.g. p-^{CV}p, ^{VZ}A, or ^{CNV}K, etc.) have high quantum yields in the UVA regime [h6], and for the corresponding photosplitting reactions to occur as far to the "red tail" of the UVB spectrum as possible. To further emphasize the importance of high quantum efficiencies at "redder" wavelengths, we refer the reader to Kladwang et. al.'s surprising 2012 report [100] that 254 nm UV shadowing of $\approx$ 200-mer RNA for only $\approx 20\ s$ caused damage to $\approx 16\% - 27\%$ of the molecules (unfortunately we were unable to determine a power density to attach to these numbers). Consider also that irradiation at UVA $\approx 365\ nm$ wavelengths can still catalyze DNA photodamage, where CPD formation followed by pyrimidine oxidation are the primary mutagenesis routes [50, 70, 124]. However, we can also note a $\approx 10^5$ fold decrease in the efficiency of photoadduct formation for UVA irradiation relative to UVC irradiation at $\approx 254\ nm$ [144, 179].

Putting everything together, while the 10 $min \approx 600\ s$ irradiation at $\lambda_{max} = 365\ nm$ (at a power density of $\approx 2.3\ W/cm^2$; see section **6.3**) step **(s.2)** in Yokomori

et. al.'s photo-DEAN [193–196] scheme is certainly something that should be optimized, it seems unlikely that irradiation will have a meaningful influence on false-positive or false-negative rates.

Finally, we remark there have been rather remarkable advances in the synthesis of nucleic acid photocrosslinkers with high UVA quantum efficiencies. Perhaps most notably, in circa 2008, Yoshimura & Fujimoto [66] coupled 3-Iodo-9H-carbazole with acrylonitrile via the palladium-catalyzed Mizoroki-Heck reaction, then reacted the carbazole secondary nitrogen with chlorosugar to yield a 3-cyanovinylcarbazole nucleoside analogue. They denoted the synthesized product $^{\mathrm{CNV}}$K [59, 60, 65, 66, 158, 201, 202],[h7] and soon took notice of the compound's extraordinary quantum efficiency at $\approx 366\ nm$ ($\Phi_{366} \approx 0.251$, $\epsilon_{366} \approx 6 * 10^3 M^{-1} cm^{-1}$). Upon absorption of a UVA $\approx 366\ nm$ photon [h8], and in a matter of seconds at a power density of $\approx 1.6\ W/cm^2$, $^{\mathrm{CNV}}$K was shown to be capable of mediating [2+2] photocycloaddition to a pyrimidine base at the +1 position of a Watson-Crick hybridized strand [59, 60, 65, 66, 158, 201, 202],[h7]. Moreover, the reaction was shown to be reversible with $\approx 312\ nm$ UVB irradiation [59, 60, 65, 66, 158, 201, 202],[h7],[h8]. Unfortunately, Fujimoto et. al. does not appear to report explicit power densities for photosplitting reaction [59, 60, 65, 66, 158, 201, 202],[h7].

The synthesis and characterization of $^{\mathrm{CNV}}$K appears to have been a watershed moment in Fujimoto's effort to significantly increase the photocrosslinking quantum yield of previous 5-R-vinyluracil (^{RV}U) (e.g. "R" = "CN", "COOH", etc.) [62–64,121,126,131–135,199,200],[h9] and p-carbamoylvinyl phenol nucleoside (p-$^{\mathrm{CV}}$p) [7, 197, 198] chemistries (we note the two distinct variants of p-$^{\mathrm{CV}}$p) [197, 198],[h10]. To the best of our knowledge, all tested $^{\mathrm{RV}}$U and p-$^{\mathrm{CV}}$p derivatives required tens of minutes to hours of irradiation time at $\approx 366\ nm$ for efficient photocrosslinking, though unfortunately, due to the lack of reported power densities, we are unable to make a comparison between the photocrosslinking or UVB photosplitting quantum efficiencies of $^{\mathrm{CNV}}$K and 7-carboxyvinyl-7-deaza-2′-deoxyadenosine ($^{\mathrm{VZ}}$A) [151] (a purine variant of $^{\mathrm{RV}}$U). Also, we note that the reversibility of the p-$^{\mathrm{CV}}$p crosslink appears to be either uncharacterized or unpublished [7, 197, 198].

In conclusion, can understand Yoshimura & Fujimoto's $^{\mathrm{CNV}}$K not just as an "ultrafast" photocrosslinker [66], but also as the gentlest non-enzymatic nucleic acid photocrosslinking chemistry known, we therefore recommend that this chemistry be adapted for use with the photo-DEAN scheme [66, 198–200] scheme in the place of $^{\mathrm{RV}}$U (where again we have that "R" = "COOH").

6.8 Special remarks (s*,h*,w* references)

Supplementary Note s1 :: In ref. [126], regarding 5-vinyldeoxyuridine (^{V}U) and 5-carboxyvinyldeoxyuridine ($^{\mathrm{CV}}\mathrm{U}$) photosplitting reaction conditions, there appears to be an error in the reported power output of the referenced $\approx$ 312 nm transilluminator. Quoting from "Section 3.1" of ref. [126]: "Irradiation was performed by UV-LED (OMRON, ZUV, 366 nm, 1.6 W/cm^2) or 2 W transilluminator (FUNAKOSHI, TR-312R/J, 312 nm)." However, the FUNAKOSHI TR-312R/J transilluminator has a power density of 15 W rather than the stated 2 W [w5]. Provided that the correct 15 W power output for this transilluminator is explicitly stated in later publications by Fujimoto et. al., e.g. in the materials section of ref. [202], we argue that it is more likely that there was a typographical error for the power output than for the name of the transilluminator. Finally, while no power density is explicitly reported for this transilluminator, provided the filter size of (15.2 $cm \times 35.6$ cm) we can estimate a power density of: 15 $W/(15.2$ $cm \times 35.6$ $cm) \approx 27.7$ mW/cm^2.

*

Supplementary Note s2 :: We suggest that it may be possible to competitively inhibit or attenuate non-specific hydrophobic interaction mediated carryover of DCN [74, 128, 129, 193–196] encoding photo-DEAN [193–196] probes during "magtration" by pre-incubating (or co-incubating during magtration) the employed streptavidin coated magnetic beads in a very high concentration solution (≈ 1 mM or so) of poly(T), poly(A), etc. DNA oligonucleotides (of some currently undetermined optimal length). Consider that if the rate of non-specific binding can be dropped by an order of magnitude, this would increase the sensitivity of GEP-DEAN [74] and very likely photo-DEAN [193–196] to a sufficient extent to allow for < 1 $zmol$ sensitivities ($\approx$ 602 molecules, or a $\approx$ 33.9 $attomolar$ (aM) concentration of molecules in GEP-DEAN's [74] ≈ 29.5 μL reaction volume). We readily admit that, even if the method works, this is a "substance over style" approach to the problem of attenuating false-positives.

[h1] :: According to the mass spectrometry results of Kim et. al. [96], the mismatch stringency of Taq DNA ligase arises during the nick closure reaction as opposed to the preliminary 5′-phosphate adenylation step. Thus, as an aside, this seems like an interesting way to inexpensively prepare 5′-adenylated substrates for use with adenylation-incompetent ligases (e.g. truncated variants of T4 RNA Ligase 2), ribozymes, or deoxyribozymes. However, we're certain a proper literature review will show that this hypothetical 5′-adenylation strategy has already been exploited.

*

[h2] :: In short, one will generally achieve superior thermodynamic discrimination of SNPs and/or mismatches with short oligonucleotides. The simple explanation is that, just as $\lim_{a\to\infty}(\frac{a}{a+n}) = 1$, the following ratios converge to ≈ 1 as the length, L, of an oligonucleotide increases:

$$r_{\Delta H} = \left(\frac{\Delta H(hybridization \mid mismatch)}{\Delta H(hybridization \mid no\ mismatch)} \right) \approx 1_{(as\ L \to \infty)}$$

$$r_{\Delta S} = \left(\frac{\Delta S(hybridization \mid mismatch)}{\Delta S(hybridization \mid no\ mismatch)} \right) \approx 1_{(as\ L \to \infty)}$$

And, therefore, the ratio of the free energies of the mismatched vs. perfectly paired duplex should converge to ≈ 1 in the same manner:

$$\begin{aligned} r_{\Delta G} &= \left(\frac{\Delta G(hybridization \mid mismatch)}{\Delta G(hybridization \mid no\ mismatch)} \right) \\ &= \left(\frac{\Delta H(hybridization \mid mismatch) - T * \Delta S(hybridization \mid mismatch)}{\Delta H(hybridization \mid no\ mismatch) - T * \Delta S(hybridization \mid no\ mismatch)} \right) \\ &\approx 1_{(as\ L \to \infty)} \end{aligned}$$

With regard to looking at duplex melting temperature T_m ($^\circ C$) depression to detect a mismatch, consider that the melting temperature of a perfectly Watson-Crick duplex is approximately [26, 58]:

...assuming non-self-complementarity and letting C_T be the sum of the concentrations of the two hybridizing oligonucleotides:

$$T_M(^\circ C) \approx \left(\frac{\Delta H}{\Delta S * R * ln\left(\frac{C_T}{4}\right)} \right) - 273.15^\circ C$$

...assuming self-complementary where C_T is the oligonucleotide concentration:

$$T_M(^\circ C, self - complementary) \approx \left(\frac{\Delta H}{\Delta S * R * ln(C_T)} \right) - 273.15^\circ C$$

Where:

$$R = Universal\ Gas\ Constant \approx k_B * N_A \approx 1.9872041 * 10^{-3} \left(\frac{kcal}{mol * K} \right)$$

Here ΔS needs to be adjusted according to the ionic strength of the buffer e.g. via SantaLucia's empirical monovalent salt correction [152]:

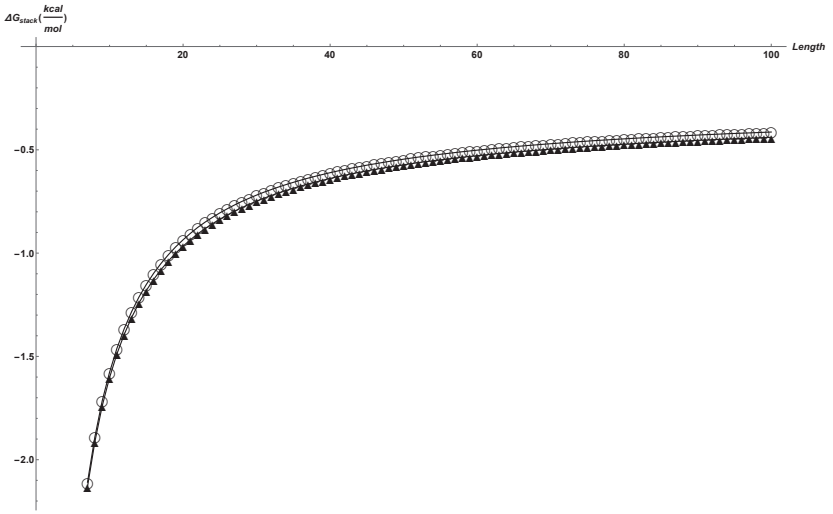
$$\begin{aligned} \Delta S \left(salt\ correction, units : \frac{kcal}{K * mol} \right) \approx \Delta S \left(1M\ NaCl; units : \frac{kcal}{K * mol} \right) + \\ (0.368 * 10^{-3}) * \left(\frac{1}{2} \right) (\#\ duplex\ phosphates) * ln(Na^+) \end{aligned}$$

(Note that unmodified oligonucleotides synthesized via phosphoramidite solid-phase methods will have 5′-hydroxyl termini.)

Considering that $T_m(^\circ C) \propto \frac{DeltaH}{DeltaS}$ (assuming fixed concentrations of oligonucleotide hybridization partners) and understanding a duplex mismatch as a constant penalty to the overall $\{\Delta H, \Delta S\}$ - on order two stacking interactions if the mismatch is internal (i.e. $\approx 2 * \Delta G_{stack}$) - it should therefore be the case that the melting temperatures of a perfectly paired duplex, and a duplex with a mismatch, should converge as L increases. This will, in turn, make it increasingly difficult to detect the mismatch looking at the fraction of hybridized versus dissociated oligonucleotides at any temperature.

Let's make this argument a bit more rigorous by statistically sampling (with replacement) from the set of individual melting temperatures for oligonucleotides of length $L = 7$ to 100 to observe the convergence in T_m for oligonucleotides of length L and $L+1$ (or L and $L+2$ to be truer to the penalty for an internal duplex mismatch). Regarding the set up for our calculation, each base is chosen uniformly over the alphabet $\Sigma = \{A, T, G, C\}$, we make the simplifying assumptions of non-self-complementary and lack of secondary structure, we set the concentration of each oligonucleotide and its Watson-Crick complement to 0.1 μM, and we assume a solution with an $[Na^+] = 0.1\ M$ concentration of monovalent ions (note that under these conditions $L \geq 7 \implies T_m > 0$). Regarding the number of sampling with replacement events for each length L, we set this value to $N = 10^6$ in order to insure reasonable convergence to true mean and median T_m values, balancing this desire against the fact that this calculation already takes on order $\approx 1\ day$ to perform. Finally, for each calculated (median) melting temperature, T_m, we compute the value for the mean and median base stack free energy, ΔG_{stack}, based on SantaLucia's nearest-neighbor stacking parameters [152].

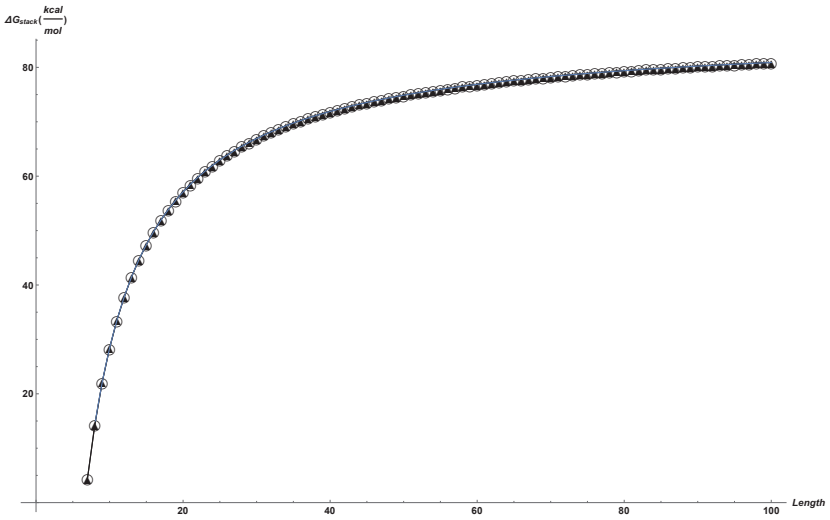
The result of the calculations for mean (solid triangle) and median (hollow circle) oligonucleotide T_m values as a function of oligonucleotide length (measured in nucleotides) is presented below:



We find a very reasonable fit to the mean and median T_m values with the expression:

$$T_m(0.1\ \mu M\ oligo, 0.1 \mu M\ complement, 0.1\ M\ NaCl) \approx \left(\frac{\Delta H * L + c_1}{\Delta S * L + c_2}\right)(^\circ C)$$
$$\approx \left(\frac{185.481 * L - 1227.88}{2.13297 * L + 0.791078}\right)(^\circ C)$$

We can also use the median T_m result to calculate mean (solid triangle) and median (hollow circle) average ΔG_{stack} (units: $kcal/mol$) energies at an oligonucleotides melting temperature as a function of oligo length (taking the mean or median over SantaLucia's ΔH and ΔS terms for each base stack [152]):



While a discussion of the full implications of the above graph is beyond the scope of this note, consider that as the length and thus the T_m of an oligonucleotide rises, the mean and median free energy of each individual stacking interaction will correspondingly decrease. Thus, holding constant the amount of time one is willing to spend to conduct an assay, methods of mismatch discrimination that look specifically at dissociation kinetics, i.e. an oligonucleotide's k_{off} parameter, will necessarily become less sensitive as a function of oligonucleotide length.

*

[h3] :: Considering that the 5-carboxyvinyl or 5-vinyl groups on $^{\mathrm{V}}$U and $^{\mathrm{CV}}$U [62–64, 121, 126, 131–135, 199, 200],[h9] are not much bulkier than a thymine 5-methyl group, we speculate that it may also be possible to have photo-DEAN probes expressed endogenously in live cells. Perhaps this could be done using vectors encoding Benner et. al.'s [168] enzymatically incorporable 6-amino-2-ketopurine (isoG) :: 2-amino-4-ketopyrimidine (isoC) bases, or Kimoto et. al.'s [97] more recent modified nucleotides, where in either case these modified nucleotides are used to direct the insertion of a $^{\mathrm{CV}}$U-like free NTP or dNTP? We are also intrigued by the possibility of biosynthesis of $^{\mathrm{CV}}$U-like derivatives. Consider, for example, that the 7-cyano-7-deazaguanine (preQ$_0$ base) precursor to the hypermodified RNA queuosine nucleoside [102] is structurally similar to a guanosine variant

of 7-carboxyvinyl-7-deaza-2′-deoxyadenosine ($^{\mathrm{VZ}}$A) [151], with a alkyne group replacing the usual alkene vinyl derivative on Fujimoto et. al.'s [2 + 2] photocycloaddition nucleotide derivatives [7, 59, 60, 62–66, 121, 126, 132–135, 151, 158, 197–202],[h7],[h9].

*

[h4] :: A mimic of the cis-syn thymine dimer is commercially available as a phosphoramidite from Glen Research [w6].

*

[h5] :: In general, pyrimidine nucleobases tend to be more reactive than their purine counterparts, likely due to the presence or greater accessibility of Conical Intersections (CI)'s allowing for high InterSystem Crossing (ISC) triplet yields [70]. One can perhaps argue that slow electronic excited state decay processes, e.g. phosphorescence, can allow additional configuration search time for favorable transition state entry for e.g. $[2+2]$ photocycloaddition.

*

[h6] :: An important question to consider is whether there exists an optimal wavelength for $^{\mathrm{CNV}}$k crosslinking "red" of $\approx 366\ nm$ that minimizes cyclobutane pyrimidine dimer (CPD) formation, the formation of oxidized bases, and other undesired photoproduct formation at the expense of requiring longer irradiation times. We suspect this will all depend on the existence of strong non-linearities in unmodified pyrimidine and/or purine absorbance at these longer wavelengths, and that it will likely not prove fruitful to consider wavelengths significantly red-shifted beyond $\approx 405\ nm$.

*

[h7] :: This is the SMILES string for 3-cyanovinylcarbazole-1′-β-deoxyriboside, i.e. Yoshimura and Fujimoto product "4" (see "Figure 1" in their paper) [66]:

N#C\C=C\c2cc3c1ccccc1n(c3cc2)C4CC(O)C(CO)O4

Note that product "4" is the precursor the final phosphoramidite product "5" of the reaction reported by Yoshimura and Fujimoto [66], which is generated via the coupling of (4, 4′-Dimethoxytrityl) and (2-cyanoethyl N,N,N',N'-tetraisopropylphosphorodiamidite) protection groups to the 5′ and 3′ hydroxyls of compound "4".

*

[h8] :: We remark here that "UVA" is meant to refer to ultraviolet wavelengths in the range of $\approx 400 - 320\ nm$, "UVB" is meant to refer to the wavelength range of $\approx 320 - 280\ nm$, and "UVC" is meant to refer to the wavelength range of $\approx 280 - 100\ nm$. Though there appear to be discrepancies in the literature regarding

the partitioning of the UV spectrum, we note that these differing partitionings tend to agree within $\approx 5 - 10\ nm$.

*

[h9] :: In ref. [201], regarding 5-vinyldeoxyuridine (^{V}U) and 5-carboxyvinyldeoxyuridine (^{CV}U) photosplitting reaction conditions, there appears to be an error in the reported power output of the referenced $\approx 312\ nm$ transilluminator. Quoting from "Section 3.1" of ref. [201]: "Irradiation was performed by UV-LED (OMRON, ZUV, 366 nm, 1.6 W/cm^2) or 2 W transilluminator (FUNAKOSHI, TR-312R/J, 312 nm)." However, the FUNAKOSHI TR-312R/J transilluminator has a power density of $\approx 15\ W$ rather than the stated $\approx 2\ W$ [w5]. Provided that the correct $\approx 15\ W$ power output for this transilluminator is explicitly stated in later publications by Fujimoto et. al., e.g. in the materials section of ref. [59], we argue that it is more likely there was a typographical error for the power output than for the name of the transilluminator. Finally, while no power density is explicitly reported for this transilluminator, provided the filter size of $\approx (15.2\ cm \times 35.6\ cm)$ we can estimate a power density of: $15\ W/((15.2\ cm \times 35.6\ cm)) \approx 27.7\ mW/cm^2$.

*

[h10] :: In ref. [197], p-^{CV}P has the SMILES descriptor:

NC(=O)/C=C/c1cc(ccc1)OC2CC(O)C(CO)O2

Whereas in ref. [7, 198], and p-^{CV}P has the SMILES descriptor:

NC(=O)/C=C/c2ccc(OC1CC(O)C(CO)O1)cc2

Generally, speaking the vinyl group is attached to the meta position of the benzene group in ref. [197] and the para position of the benzene group in ref. [7, 198]. Interestingly, the latter para-conjugated system appears to have a much higher photocrosslinking quantum efficiency. Though we do not have power densities to work with, forcing us to assume equivalent conditions to make any statements, we can guess that roughly equivalent power densities were employed in both studies, and then note that a $\approx 5\ h$ irradiation period of p-^{CV}P (meta) at $\approx 366\ nm$ resulted in a $\approx$ 71% photocrosslinking yield [197], whereas only a $\approx 30\ min$ irradiation period of p-^{CV}P (para) resulted in a much higher $\approx$ 96% photocrosslinking yield [198]. While meta and para-substituted benzene rings will certainly exhibit differences in their electronic structure, we do not have a theoretical explanation for the higher quantum efficiency of p-^{CV}P (para) vs. p-^{CV}P (meta). The difference in crosslinking efficiency for these two p-^{CV}P variants could also certainly be based on differing stacking kinetics with the target C5=C6 pyrimidine double bond.

Website Links

[w1] :: https://www.neb.com/products/m0262-lambda-exonuclease

[w2] :: https://www.neb.com/products/m0363-t5-exonuclease

[w3] :: https://www.neb.com/products/m0263-t7-exonuclease

[w4] :: http://www.toyobo-global.com/seihin/xr/lifescience/products/pcr_001.html

[w5] :: http://www.funakoshi.co.jp/download/catalog/FUN4916.pdf

[w6] :: http://www.glenresearch.com/ProductFiles/11-1330.html

Chapter 7

Supplementary Information for Computational Methods

7.1 Brief overview of Mathematica, Combinatorica, SAGE, and the 'igraph' R package

Mathematica [190]:

Mathematica is a (proprietary / closed source) commercial Computer Algebra System (CAS) and one of the primary software products of Wolfram Research, a private company founded by Stephen Wolfram in 1987 and located in Champaign, Illinois. It is written in C++ as well as its own language, which was recently denoted the Wolfram Language. The original version of Mathematica (v1) was originally released in June 1988, the (v9) version was initially released in November 2012, and the newest version (v10) of the software was released in July 2014. Between version releases, Wolfram Research typically rolls out various updates (e.g. v5.2 or v10.3) for the purpose of adding features or fixing bugs.

For additional details, we refer the reader to ---

Wolfram Research's page for Mathematica:
<http://www.wolfram.com/mathematica/>

Wolfram Research's page on Mathematica's revision history:
<http://www.wolfram.com/mathematica/quick-revision-history.html>

Mathematica's Wikipedia page:
<https://en.wikipedia.org/wiki/Mathematica>

Combinatorica [143,190]:

Combinatorica is an open source (however, written in the proprietary Mathematica language) add-on for Mathematica that introduced a wide range of functions (≈ 450) relevant to combinatorics and graph theory. The package was originally developed by Steven S. Skiena, currently a professor of computer science at SUNY Stony Brook: <http://www3.cs.stonybrook.edu/~skiena/>, and after revisions by Sriram V. Pemmaraju, currently a professor of computer science at the University of Iowa: <http://homepage.cs.uiowa.edu/~sriram/>, was eventually bundled with Mathematica 4.2 (released in June 2002).

For additional details, we refer the reader to ---

Steven S. Skiena's page for Combinatorica:
<http://www3.cs.stonybrook.edu/~skiena/combinatorica/>

The source code for the latest version of Combinatorica (v2.0.0):
<http://homepage.cs.uiowa.edu/~sriram/Combinatorica/NewCombinatorica.m>

The textbook guide to Combinatorica (written by Pemmaraju and Skiena):
[Pemmaraju, S. V., Skiena, S. S. Computational discrete mathematics: combinatorics and graph theory with Mathematica. Cambridge University Press: New York, NY (2003).] [143]

System for Algebra and Geometry Experimentation (SAGE) [164]:

System for Algebra and Geometry Experimentation (SAGE), also known as Sage or SageMath, is an open source software package licensed under the GPL (GNU Public License), and developed by collaboration with a stated mission of (quoting from: <http://www.sagemath.org/>): "Creating a viable free open source alternative to Magma, Maple, Mathematica and Matlab." SAGE was initially conceived of by William A. Stein, currently a professor of mathematics at the University of Washington, who also served as its lead developer, and was first released on February 24th, 2005. We note that there is now a platform for executing SAGE scripts online (in the "cloud" so to speak): <https://cloud.sagemath.com/>.

For additional details, we refer the reader to ---

The homepage for SAGE:
<http://www.sagemath.org/>

A tutorial for programming in SAGE's "Python-like" language:
<http://doc.sagemath.org/html/en/thematic_tutorials/tutorial-programming-python.html>

The Wikipedia page for SAGE:
<https://en.wikipedia.org/wiki/SageMath>

Finally, if the reader is previously familiar with Mathematica and/or Combinatorica, he or she may find the following links helpful:

<https://wiki.sagemath.org/sage_mathematica>
<https://wiki.sagemath.org/CombinatoricaCompare>

The 'igraph' package (for C, R, and Python) [45]:

The 'igraph' package is an open source software package written in C and licensed under the GPL (GNU Public License) with a primary focus on the implementation of algorithms for network analysis on large graph objects. We note that 'igraph' packages exist for the R and Python languages, and for this work, we make use of the 'igraph' R package. The 'igraph' package was developed by Gábor Csárdi and Támas Nepusz, was initially released in 2006, and its current version (which we make use of) is v1.0.0 (released on June 29th, 2015).

For additional details, we refer the reader to ---

Gábor Csárdi and Tamas Nepusz's published manuscript introducing and detailing the 'igraph' package:
[Csárdi, G., Nepusz, T.: The igraph software package for complex network research. *InterJournal Complex Systems* (manuscript no. 1695), pp. 1 - 9 (2006)] [45]

The homepage for the 'igraph' R package:
<http://igraph.org/r/>

The GitHub page for the 'igraph' package:
<https://github.com/igraph/igraph>

Documentation for the latest release (v1.0.0) of the 'igraph' package for the R language:
<http://igraph.org/r/doc/igraph.pdf>

The Wikipedia page for the 'igraph' package:
<https://en.wikipedia.org/wiki/Igraph>

7.2 Graph plotting in Mathematica 10.4.1 and SAGE 7.2

How to plot graphs in Mathematica 10.4.1
(see:
<https://reference.wolfram.com/language/ref/Graph.html> for additional documentation and options.) ::

Procedure:

```
graph6EncodingString="SPECIFY_GRAPH6_STRING";

mmaCoordinateList=SPECIFY_MATHEMATICAv9_FORMAT_VERTEX_COORDINATES;

mmaTargetGraph=ImportString[graph6EncodingString,"graph6"];

mmaTargetGraphPlot=SetProperty[mmaTargetGraph,{VertexCoordinates-
>mmaCoordinateList,VertexSize->SPECIFY_VERTEX_SIZE,VertexStyle-
>SPECIFY_VERTEX_COLOR,ImageSize->{SPECIFY_PLOT_DIM_X,SPECIFY_PLOT_DIM_Y}}];

mmaTargetGraphPlot
```

--

Note: One may specify a graph directly via an edge list in the following manner: 'mmaTargetGraph=Graph[SPECIFY_GRAPH_EDGE_LIST]'.

How to plot graphs in SAGE 7.2
(see: <http://doc.sagemath.org/html/en/reference/plotting/sage/graphs/graph_plot.html> for additional documentation and options.) ::

Procedure:

```
graph6EncodingString='SPECIFY_GRAPH6_STRING'

sageCoordinateList=SPECIFY_SAGE_FORMAT_VERTEX_COORDINATES

sageTargetGraph=Graph(graph6EncodingString)

sageTargetGraphPlot=sageTargetGraph.graphplot(pos=sageCoordinateList,vertex_shape="SPE
CIFY_VERTEX_SHAPE",vertex_size=SPECIFY_VERTEX_SIZE,vertex_colors='SPECIFY_VERTEX_COLOR
')

sageTargetGraphPlot.show(figsize=[SPECIFY_PLOT_DIM_X,SPECIFY_PLOT_DIM_Y])
```

--

Note 1: One may specify a graph directly via an edge list in the following manner: 'sageTargetGraph=Graph(SPECIFY_GRAPH_EDGE_LIST)'.

Note 2: Possible values for 'vertex_shape' appear to correspond to Matplotlib markers (for a list of these markers, see the Matplotlib API documentation: <http://matplotlib.org/api/markers_api.html>).

7.3 Export and import of graph6 (.g6) strings in Mathematica 10.4.1 and SAGE 7.2

(Graph6.MMA) To obtain graph6 (.g6) string encoding a Mathematica 10.4.1 graph object, we can make use of Mathematica's 'ExportString' and/or 'Export'. To transform a graph6 (.g6) string into a Mathematica 10.4.1 graph object, we can use Mathematica's 'ImportString[]' function.

Usage (exporting a graph6 (.g6) string): Letting 'mmaTargetGraph' be a Mathematica 10.4.1 object encoding a simple undirected and connected graph, we can generate a graph6 (.g6) string encoding 'mmaTargetGraph' with the command: 'ExportString[mmaTargetGraph,"graph6"]'. We can also export the graph6 (.g6) string to a file as follows: 'Export["PATH\\targetFile.g6", mmaTargetGraph]'.

Usage (importing a graph6 (.g6) string): Letting 's' be a graph6 (.g6) string, we can create a Mathematica 10.4.1 graph object from this string via the command: 'mmaTargetGraph=ImportString[s,"graph6"]'.

(Graph6.SAGE.Out) To obtain graph6 (.g6) string encoding a SAGE 7.2 graph object we can use SAGE's 'graph6_string()' function.

Usage: Letting 'sageTargetGraph' be a SAGE 7.2 object encoding a simple undirected and connected graph, we can generate a graph6 (.g6) string encoding this graph with the command: 'sageTargetGraph.graph6_string()'. We can also export the graph6 (.g6) string to a file with the following three lines: f = file(' ~/targetFile.txt','w') // f.write(str(sageTargetGraph.graph6_string()+'\n'))// f.close(). To transform a graph6 (.g6) string into a SAGE 7.2 graph object, we can write the SAGE 7.2 command: 'sageTargetGraph=Graph(s)'.

7.4 Mathematica 10.4.1 scripts to convert Mathematica 10.4.1 graph object edge lists to Combinatorica, SAGE 7.2, and 'igraph' R package graph object edge lists

(Mathematica 10.4.1 → Combinatorica) The below Mathematica 10.4.1 script will transform a Mathematica 10.4.1 graph object edge list into a Combinatorica graph object edge list:

Undirected Graphs ::

Input: Some MMA 10.4.1 parsible edge list (with integer labeled vertices) for an undirected graph, denoted here as (e.g.) 'mmaEdgeList = {1<->2,1<->3, … }'.

(Note :: See the posting by 'Szabolcs Horvat' at this link --- <http://community.wolfram.com/groups/-/m/t/880870?p_p_auth=S7lctraj> --- regarding a solution (which we make use of) for employing 'Block[]' to allow usage of Combinatorica functions without adding Combinatorica to '$ContextPath'.)

Script (Method 1):

```
Quiet[Block[{$ContextPath},Needs["Combinatorica`"]]]
combinatoricaTargetGraph=Combinatorica`FromUnorderedPairs[mmaEdgeList]
Combinatorica`Edges[combinatoricaTargetGraph]
```

Script (Method 2):

```
Quiet[Block[{$ContextPath},Needs["Combinatorica`"]]]
targetGraphAM=Normal[AdjacencyMatrix[Graph[mmaEdgeList]]];
combinatoricaTargetGraph=Combinatorica`FromAdjacencyMatrix[targetGraphAM];
combinatoricaEdgeList=Combinatorica`Edges[combinatoricaTargetGraph]
```

Output (from either method): (1) a Mathematica 10.4.1 object 'combinatoricaTargetGraph' encoding a Combinatorica undirected graph object; (2) a Mathematica 10.4.1 object 'combinatoricaEdgeList' encoding the edge list for the aforementioned Combinatorica undirected graph object.

-

Digraphs ::

Input: Some MMA 10.4.1 parsible edge list (with integer labeled vertices) for a digraph, denoted here as (e.g.) 'mmaEdgeList = {1\[DirectedEdge]2,1\[DirectedEdge]3, … }'.

(Note :: See the posting by 'Szabolcs Horvat' at this link --- <http://community.wolfram.com/groups/-/m/t/880870?p_p_auth=S7lctraj> --- regarding a solution (which we make use of) for employing 'Block[]' to allow usage of Combinatorica functions without adding Combinatorica to '$ContextPath'.)

Script:

```
Quiet[Block[{$ContextPath},Needs["Combinatorica`"]]]
combinatoricaTargetGraph=Combinatorica`FromOrderedPairs[mmaEdgeList]
Combinatorica`Edges[combinatoricaTargetGraph]
```

Output: (1) a Mathematica 10.4.1 object 'combinatoricaTargetGraph' encoding a Combinatorica digraph object; (2) a Mathematica 10.4.1 object 'combinatoricaEdgeList' encoding the edge list for the aforementioned Combinatorica digraph object.

(Mathematica 10.4.1 → SAGE 7.2) The below Mathematica 10.4.1 script will transform a Mathematica 10.4.1 graph object edge list into a SAGE 7.2 graph object edge list:

Undirected Graphs ::

Input: Some MMA 10.4.1 parsible edge list (with integer labeled vertices) for an undirected graph, denoted here as (e.g.) 'mmaEdgeList = {1<->2,1<->3, … }'.

Script:

```
sageEdgeListString=StringReplace[ToString[Table[StringJoin["(",ToString[mmaEdgeList[[i
,1]]-1],",",ToString[mmaEdgeList[[i,2]]-1],")"],{i,1,Length[mmaEdgeList]}]],{" "-
>"","{"->"[","}"->"]"}]
```

Output: A string 'sageEdgeListString' encoding a SAGE 7.2 format edge list for some undirected graph object (e.g. "{(1,2),(1,3), … }"). Note that we needed to account for the 1 → 0 vertex index origin change moving from MMA 10.4.1 to SAGE 7.2.

-

Digraphs ::

Input: Some MMA 10.4.1 parsible edge list (with integer labeled vertices) for a digraph, denoted here as (e.g.) 'mmaEdgeList = {1\[DirectedEdge]2,1\[DirectedEdge]3, … }'.

Script:

```
mmaEdgeList={1\[DirectedEdge]2,2\[DirectedEdge]3,3\[DirectedEdge]4,4\[DirectedEdge]1};
mmaTargetGraph=Graph[mmaEdgeList];
sageEdgeListString=StringReplace[StringJoin["sageDirectedTargetGraph=DiGraph({",Table[
StringJoin[ToString[VertexList[mmaTargetGraph][[i]]-
1],":[",StringReplace[ToString[Map[If[MemberQ[EdgeList[mmaTargetGraph],VertexList[mmaT
argetGraph][[i]]\[DirectedEdge]#]==True,#-
1,Nothing]&,AdjacencyList[mmaTargetGraph,VertexList[mmaTargetGraph][[i]]]]],{" "-
>"","{"->"","}"->""}],"],"],{i,1,Length[VertexList[mmaTargetGraph]]}],"})"],{",}"-
>"}"}]
```

Output: A string 'sageEdgeListString' encoding a SAGE 7.2 format edge list for some digraph object (e.g. "{(1,2),(1,3), … }"). Note that we needed to account for the 1 → 0 vertex index origin change moving from MMA 10.4.1 to SAGE 7.2. Please note also that we created a Mathematica 10.4.1 graph object 'mmaTargetGraph' in the above script.

(SAGE 7.2 → Mathematica 10.4.1) The below Mathematica 10.4.1 script will transform a SAGE 7.2 graph object edge list into a Mathematica 10.4.1 graph object edge list:

Undirected Graphs ::

Input: A string representing some SAGE 7.2 parsible edge list (with integer labeled vertices) for an undirected graph (e.g. we might have 'sageEdgeListString = "[(0,1),(0,2), …]"').

Script:

```
mmaEdgeList=Flatten[Map[{#[[1]]+1\[UndirectedEdge]#[[2]]+1}&,ToExpression[StringReplac
e[sageEdgeListString,{"("->"{",")"->"}"}]]],1]
```

Output: A MMA 10.4.1 parsible edge list for some undirected graph object (e.g. '{1<->2,1<->23, … }'). Note that we needed to account for the 0 → 1 vertex index origin change moving from SAGE 7.2 to MMA 10.4.1.

-

Digraphs ::

Input: A string representing some SAGE 7.2 parsible edge list (with integer labeled vertices) for a digraph (e.g. we might have 'sageEdgeListString = "{0:[1],1:[2],2:[3],3:[0]}"').

Script:

```
mmaEdgeList=Flatten[Map[Table[#[[1]]\[DirectedEdge]#[[2,i]],{i,1,Length[#[[2]]]}]&,Par
tition[ToExpression[StringReplace[sageEdgeListString,{":"->",","["->"{","]"-
>"}"}]],2]]]
```

Output: A MMA 10.4.1 parsible edge list for some digraph object (e.g. '{1\[DirectedEdge]2,1\[DirectedEdge]3, … }'). Note that we needed to account for the 0 → 1 vertex index origin change moving from SAGE 7.2 to MMA 10.4.1.

(Mathematica 10.4.1 <or> SAGE 7.2 → 'igraph' R package) The below Mathematica 10.4.1 script will transform a string representation of either a SAGE 7.2 or Mathematica 10.4.1 graph object edge list into an 'igraph' R-package graph object:

Undirected Graphs ::

Input: There are two inputs: (Input 1) A string representing some SAGE 7.2 or Mathematica 10.4.1 parsible edge list (with integer labeled vertices) for an undirected graph (e.g. we might have a string in the SAGE 7.2 format like '[(0,1),(0,2), …]', or a string in the Mathematica 10.4.1 format like '{1\[UndirectedEdge]2,1\[UndirectedEdge]3, … }'); the script will automatically detect and correct 0/1-indexing as necessary. (Input 2) An integer for the variable 'maximumNumberOfEdgesPerPartition', which specifies the maximum number of edges per line in the R script that will be output to specify an 'igraph' graph object. This is important for users of R editors like RStudio (see: <https://en.wikipedia.org/wiki/RStudio>), where we observed that there appears to be a maximum number of characters per line that can be recognized for read-in. In our hands, for graphs with at most 3-character names, a value of 'maximumNumberOfEdgesPerPartition = 100' creates an RStudio parsible output script (we specify '100' as an example value for this input).

Script:

```
(* SPECIFY INPUT VALUES HERE for 'mmaOrSageEdgeListString' and
'maximumNumberOfEdgesPerPartition'. *)
mmaOrSageEdgeListString="";
maximumNumberOfEdgesPerPartition=100;
(* SCRIPT START *)
mmaOrSageFlattenedEdgeList=ToExpression[StringReplace[mmaOrSageEdgeListString,{"["-
>"{","]"->"}","("->"","")"->"","<->"->",","\[UndirectedEdge]"->","}]];
If[Min[mmaOrSageFlattenedEdgeList]==0,
mmaOrSageFlattenedEdgeList =Map[#+1&, mmaOrSageFlattenedEdgeList];
];

Print["igraphTargetGraph <- make_empty_graph(directed=FALSE) %>%"]
Print[StringJoin["add_vertices(",ToString[Length[DeleteDuplicates[mmaOrSageFlattenedEd
geList]]],") %>%"]]

igraphPreProcessedEdgeList=Flatten[Sort[Partition[mmaOrSageFlattenedEdgeList,2]]];
edgeListPartitions=Map[Flatten[#]&,Partition[Partition[igraphPreProcessedEdgeList,2],m
aximumNumberOfEdgesPerPartition,maximumNumberOfEdgesPerPartition,{1,1},{}]];
igraphProcessedEdgeListStrings={};

For[k=1,k<=Length[edgeListPartitions],k++,
```

```
igraphProcessedEdgeListStrings=Append[igraphProcessedEdgeListStrings,{"add_edges(c("}]
;
igraphProcessedEdgeListStrings[[k]]=StringJoin[igraphProcessedEdgeListStrings[[k]],Tab
le[{ToString[edgeListPartitions[[k,i]]],",",ToString[edgeListPartitions[[k,i+1]]],",
"},{i,1,Length[edgeListPartitions[[k]]],2}]];

igraphProcessedEdgeListStrings[[k]]=StringDrop[igraphProcessedEdgeListStrings[[k]],-
2];
If[k!=Length[edgeListPartitions],
igraphProcessedEdgeListStrings[[k]]=StringJoin[igraphProcessedEdgeListStrings[[k]],"))
 %>%"];
,
igraphProcessedEdgeListStrings[[k]]=StringJoin[igraphProcessedEdgeListStrings[[k]],"))
"];
];
Print[igraphProcessedEdgeListStrings[[k]]];
];
(* SCRIPT END; Copy-paste and run output as an R script. *)
```

Output: The output here is a script that, when run in R with the 'igraph' package installed (via the command: 'install.packages("igraph")') and loaded (via the command: 'library(igraph)') will generate an 'igraph' graph object 'igraphTargetGraph' corresponding to the input edge list. As mentioned in the 'Input' section, the maximum number of edges specified per line in the output script will be bounded by the integer input value 'maximumNumberOfEdgesPerPartition' ('100' is a reasonable value for this parameter).

As an example, we can output an 'igraph' R-package script to generate a 3-cube, allowing for only four edges per line to be specified in the R script, as follows:

```
mmaOrSageEdgeListString="{1\[UndirectedEdge]2,1\[UndirectedEdge]3,1\[UndirectedEdge]4,
2\[UndirectedEdge]5,2\[UndirectedEdge]6,3\[UndirectedEdge]5,3\[UndirectedEdge]8,4\[Und
irectedEdge]6,4\[UndirectedEdge]8,5\[UndirectedEdge]7,6\[UndirectedEdge]7,7\[Undirecte
dEdge]8}";
maximumNumberOfEdgesPerPartition=4;
```

Here, the output for the script will be:

```
igraphTargetGraph <- make_empty_graph(directed=FALSE) %>%
add_vertices(8) %>%
add_edges(c(1,2, 1,3, 1,4, 2,5)) %>%
add_edges(c(2,6, 3,5, 3,8, 4,6)) %>%
add_edges(c(4,8, 5,7, 6,7, 7,8))
```

We can now execute the command 'plot(igraphTargetGraph)' in R to generate an image of an undirected graph corresponding to the 3-cube.

(Warning: The 'igraph' package may fail to parse graph edge lists if the specified vertex labels, which must be integers, only constitute a subset of some domain $[1,\ldots,n]$ where n is the maximum integer value for a vertex label. To ensure that there are not errors, we recommend "resetting" the vertex labels for the graph by e.g. exporting and re-importing the graph as a 'graph6' string like so: 'EdgeList[ImportString[ExportString[mmaTargetGraph,"graph6"],"graph6"]]'. We note that the Mathematica 10.4.1 command 'FindGraphIsomorphism[]' can allow one to determine the mapping between the original vertex labels and the "reset" vertex labels.)

-

Digraphs ::

Input: There are two inputs: (Input 1) A string representing some SAGE 7.2 or Mathematica 10.4.1 parsible edge list (with integer labeled vertices) for a digraph (e.g. we might have a string in the SAGE 7.2 format like '{0:[1],1:[2],2:[3],3:[0]}', or a string in the Mathematica 10.4.1 format like '{1\[UndirectedEdge]2,1\[UndirectedEdge]3, … }'); the script will automatically detect and correct 0/1-indexing as necessary. (Input 2) An integer for the variable 'maximumNumberOfEdgesPerPartition', which specifies the maximum number of edges per line in the R script that will be output to specify an 'igraph' graph object. This is important for users of R editors like RStudio (see: <https://en.wikipedia.org/wiki/RStudio>), where we observed that there appears to be a maximum number of characters per line that can be recognized for read-in. In our hands, for graphs with at most 3-character names, a value of 'maximumNumberOfEdgesPerPartition = 100' creates an RStudio parsible output script (we specify '100' as an example value for this input).

Script:

```
(* SPECIFY INPUT VALUES HERE for 'mmaOrSageEdgeListString' and
'maximumNumberOfEdgesPerPartition'. *)
mmaOrSageEdgeListString="";
maximumNumberOfEdgesPerPartition=100;
(* SCRIPT START *)
If[StringContainsQ[mmaOrSageEdgeListString,"\[DirectedEdge]"]==False&&StringContainsQ[
mmaOrSageEdgeListString,"->"]==False,
mmaOrSageEdgeListString=ToString[Flatten[Map[Table[#[[1]]\[DirectedEdge]#[[2,i]],{i,1,
Length[#[[2]]]}]&,Partition[ToExpression[StringReplace[mmaOrSageEdgeListString,{":"-
>","","["->"{","]"->"}"}]],2]]];
];
mmaOrSageFlattenedEdgeList=ToExpression[StringReplace[mmaOrSageEdgeListString,{"["-
>"{","]"->"}","("->"",")"->"","<->"->",","\[DirectedEdge]"->","}]];
If[Min[mmaOrSageFlattenedEdgeList]==0,mmaOrSageFlattenedEdgeList=Map[#+1&,mmaOrSageFla
ttenedEdgeList];];

Print["igraphDirectedTargetGraph <- make_empty_graph(directed=TRUE) %>%"];
Print[StringJoin["add_vertices(",ToString[Length[DeleteDuplicates[mmaOrSageFlattenedEd
geList]]],") %>%"]];

igraphPreProcessedEdgeList=Flatten[Sort[Partition[mmaOrSageFlattenedEdgeList,2]]];
edgeListPartitions=Map[Flatten[#]&,Partition[Partition[igraphPreProcessedEdgeList,2],m
aximumNumberOfEdgesPerPartition,maximumNumberOfEdgesPerPartition,{1,1},{}]];
igraphProcessedEdgeListStrings={};

For[k=1,k<=Length[edgeListPartitions],k++,igraphProcessedEdgeListStrings=Append[igraph
ProcessedEdgeListStrings,{"add_edges(c("}];
igraphProcessedEdgeListStrings[[k]]=StringJoin[igraphProcessedEdgeListStrings[[k]],Tab
le[{ToString[edgeListPartitions[[k,i]]],",",ToString[edgeListPartitions[[k,i+1]]],",
"},{i,1,Length[edgeListPartitions[[k]]],2}]];
igraphProcessedEdgeListStrings[[k]]=StringDrop[igraphProcessedEdgeListStrings[[k]],-
2];
If[k!=Length[edgeListPartitions],igraphProcessedEdgeListStrings[[k]]=StringJoin[igraph
ProcessedEdgeListStrings[[k]],")) %>%"];,igraphProcessedEdgeListStrings[[k]]=StringJoi
n[igraphProcessedEdgeListStrings[[k]],"))"];
];
Print[igraphProcessedEdgeListStrings[[k]]
];
];
(* SCRIPT END; Copy-paste and run output as an R script. *)
```

Output: The output here is a script that, when run in R with the 'igraph' package installed (via the command: 'install.packages("igraph")') and loaded (via the command: 'library(igraph)') will generate an 'igraph' digraph object 'igraphTargetGraph' corresponding to the input edge list. As mentioned in the 'Input' section, the maximum number of edges specified per line in the output script will be bounded by the integer input value 'maximumNumberOfEdgesPerPartition' ('100' is a reasonable value for this parameter).

As an example, we can output an 'igraph' R-package script to generate a digraph corresponding to the 3-cube with oriented edges (specifically where edges are oriented to point from vertex $v_a \to v_b$ whenever the integer label corresponding to vertex v_b is greater than the integer label corresponding to vertex v_a), allowing for only four edges per line to be specified in the R script, as follows:

```
mmaOrSageEdgeListString="{1\[DirectedEdge]2,1\[DirectedEdge]3,1\[DirectedEdge]4,2\[Dir
ectedEdge]5,2\[DirectedEdge]6,3\[DirectedEdge]5,3\[DirectedEdge]8,4\[DirectedEdge]6,4\
[DirectedEdge]8,5\[DirectedEdge]7,6\[DirectedEdge]7,7\[DirectedEdge]8}";
maximumNumberOfEdgesPerPartition=4;
```

Here, the output for the script will be:

```
igraphDirectedTargetGraph <- make_empty_graph(directed=TRUE) %>%
add_vertices(8) %>%
add_edges(c(1,2, 1,3, 1,4, 2,5)) %>%
add_edges(c(2,6, 3,5, 3,8, 4,6)) %>%
add_edges(c(4,8, 5,7, 6,7, 7,8))
```

We can now execute the command 'plot(igraphDirectedTargetGraph)' in R to generate an image of an digraph corresponding to the oriented 3-cube.

(Warning: The 'igraph' package may fail to parse graph edge lists if the specified vertex labels, which must be integers, only constitute a subset of some domain $[1,\ldots,n]$ where n is the maximum integer value for a vertex label. To ensure that there are not errors, we recommend "resetting" the vertex labels for the graph by e.g. exporting and re-importing the graph as a 'graph6' string like so: 'EdgeList[ImportString[ExportString[mmaTargetGraph,"graph6"],"graph6"]]'. We note that the Mathematica 10.4.1 command 'FindGraphIsomorphism[]' can allow one to determine the mapping between the original vertex labels and the "reset" vertex labels.)

7.5 Specifying and interconverting Mathematica 10.4.1 and SAGE 7.2 graph object vertex coordinate lists

The below procedure will transform a Mathematica 10.4.1 vertex coordinate list into a SAGE 7.2 vertex coordinate list:

Input: Some MMA 10.4.1 parsible coordinate list for a graph (e.g. we might have ‘mmaCoordinateList = {1 -> {x1, y1}, 2 -> {x2, y2}, … }’).

Procedure:

```
sageCoordinateListString=StringReplace[ToString[Table[StringJoin[ToString[mmaCoordinat
eList[[i,1]]-
1],":[",ToString[N[mmaCoordinateList[[i,2,1]]]],",",ToString[N[mmaCoordinateList[[i,2,
2]]]],"]"],{i,1,Length[mmaCoordinateList]}]],"  "->""]
```

Output: A string ‘sageCoordinateListString’ encoding a SAGE 7.2 format coordinate list for some graph object (e.g. ‘{0:[x1, y1], 1:[x2, y2], … }’). Note that we needed to account for the 1 → 0 index origin change moving from MMA 10.4.1 to SAGE 7.2.

The below procedure will transform a SAGE 7.2 vertex coordinate list into a Mathematica 10.4.1 vertex coordinate list:

Input: A string representing some SAGE 7.2 parsible coordinate list for a graph (e.g. we might have ‘sageCoordinateListString = {0:[x1, y1], 1:[x2, y2], … }’).

Procedure:

```
mmaCoordinateList=Flatten[Map[{#[[1]]+1-
>#[[2]]}&,ToExpression[StringReplace[sageCoordinateListString,{":"->"->","["->"{","]"-
>"}"}]]],1]
```

Output: A MMA 10.4.1 parsible coordinate list for some graph object (e.g. ‘{1 -> {x1, y1}, 2 -> {x2, y2}, … }’). Note that we needed to account for the 0 → 1 index origin change moving from SAGE 7.2 to MMA 10.4.1.

7.6 Computational methods for finding and enumerating Hamiltonian cycles and paths with Mathematica 10.4.1, Combinatorica, SAGE 7.2, and the 'igraph' R package

Using the software packages Mathematica (version 10.4.1), SAGE 7.2, and the 'igraph' (version 1.0.0) R language (version 3.2.3) package, we have employed four (three) redundant methods to count Hamiltonian cycle as well as Hamiltonian path flows through the gadgets corresponding to undirected (directed) graphs described in this document. Wherever it is possible for us to count Hamiltonian path flows through a gadget, it is always possible for us to check this calculation by counting Hamiltonian cycle flows through a slightly modified version of the gadget.

We note that one may also enumerate Hamiltonian cycles and paths by e.g. generating all simple paths (or an appropriate subset of all simple graph) in a simple undirected graph via a Breadth-First-Search (BFS) or Depth-First-Search (DFS), and then simply scan and test path to determine whether it is a Hamiltonian cycle or path. However, in our hands, naïve versions of such approaches do not approach scale well for graph objects (specifically of interest to this work) composed of more than about ≈ 10 to ≈ 20 vertices.

For Hamiltonian cycles, we make use of the following ::

(HC.1) // (HC_digraph.1) Mathematica's (version 10.4.1) 'FindHamiltonianCycle[]' function:

Usage (Undirected Graphs): Letting 'mmaTargetGraph' be a Mathematica 10.4.1 object encoding a simple connected undirected graph, the function call 'FindHamiltonianCycle[mmaTargetGraph, All]' will return all Hamiltonian cycles in 'mmaTargetGraph' and 'Length[FindHamiltonianCycle[mmaTargetGraph, All]]' will return the count for all Hamiltonian cycles.

-

Usage (Digraphs): Equivalent to the above method for undirected graphs (under the same assumptions that the graph is simple as well as connected).

--

(Undirected Graphs) Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
mmaTargetGraph=ImportString["GsXOXc","graph6"];
Length[FindHamiltonianCycle[mmaTargetGraph,All]]
```

Output: '6' (Output is correct, see e.g. <http://oeis.org/A066037>)

-

(Digraphs) Example with a directed cycle on $(k = 6)$ vertices:

```
mmaDirectedTargetGraph=Graph[{1->2,2->3,3->4,4->5,5->6,6->1}];
Length[FindHamiltonianCycle[mmaDirectedTargetGraph,All]]
```

Output: '1'

(HC.2) Combinatorica's 'HamiltonianCycle[]' function:

Usage (Undirected Graphs): Letting 'combinatoricaTargetGraph' be a Combinatorica object encoding a simple connected undirected graph, the function call '(1/2)*Length[HamiltonianCycle[combinatoricaTargetGraph,All]]' will return the number

of Hamiltonian cycles in ‘combinatoricaTargetGraph’ (we divide by two to account for the reversals of each Hamiltonian cycle).

(Note :: See the posting by ‘Szabolcs Horvat’ at this link --- <http://community.wolfram.com/groups/-/m/t/880870?p_p_auth=S7lctraj> --- regarding a solution (which we make use of) for employing ‘Block[]’ to allow usage of Combinatorica functions without adding Combinatorica to ‘$ContextPath’.)

(Warning :: Loading the Graph Utilities package along with Combinatorica may interfere with the ‘HamiltonianCycle[]’ function.)

-

Usage (Digraphs): The Combinatorica package does not appear to include a method for counting Hamiltonian cycles in digraphs.

--

(Undirected Graphs) Example with an (n=3)-cube graph (graph6 (.g6) string: “GsXOXc”):

```
Quiet[Block[{$ContextPath},Needs["Combinatorica`"]]]
targetGraphAdjacencyMatrix = Normal[AdjacencyMatrix[ImportString["GsXOXc","graph6"]]];
combinatoricaTargetGraph=
Combinatorica`FromAdjacencyMatrix[targetGraphAdjacencyMatrix];
(1/2)*Length[Combinatorica`HamiltonianCycle[combinatoricaTargetGraph,All]]
```

Output: ‘6’ (Output is correct, see e.g. <http://oeis.org/A066037>)

(HC.3) // (HC_digraph.3) SAGE 7.2's 'SubgraphSearch()' function:

Usage (Undirected Graphs): Let 'sageTargetGraph' be a SAGE 7.2 object encoding a simple connected undirected graph. Here, we first need to generate a simple cycle graph isomorphic to any Hamiltonian cycle in 'sageTargetGraph' (should one or more exist), which can be done via the command: 'cycleGraph=graphs.CycleGraph(len(sageTargetGraph))'. We can then import 'SubgraphSearch()' via the command: 'from sage.graphs.generic_graph_pyx import SubgraphSearch', and immediately following this, call 'SubgraphSearch(sageTargetGraph, cycleGraph)', to return the set of all labeled subgraphs in 'sageTargetGraph' isomorphic to 'cycleGraph' corresponding to Hamiltonian cycles and cycles and their '(2*len(cycleGraph))' automorphisms.

Putting everything together, we can ask for the total Hamiltonian cycle count in 'sageTargetGraph' via the line:
'SubgraphSearch(sageTargetGraph, cycleGraph).cardinality() / (2*len(cycleGraph))'.

We note that Nathann Cohen (<http://www.steinertriples.fr/ncohen/tut/Graphs/>) is the primary author for 'SubgraphSearch', and that the 'generic_graph_pyx' Cython package of Cython functions for graphs it was incorporated into was originally authored by Robert L. Miller.

For additional documentation please see:
<http://www.sagemath.org/documentation/html/en/reference/graphs/sage/graphs/generic_graph_pyx.html>.

For the source code please see: <http://www.sagenb.org/src/graphs/generic_graph.py>.

-

Usage (Digraphs): The method of using SAGE 7.2 to count Hamiltonian cycles in digraphs is nearly the same as the above method for counting Hamiltonian cycles in undirected graphs, save for following three minor changes:

(1) The target graph, 'sageTargetGraph', referred to hereafter as 'sageDirectedTargetGraph', must be specified properly as a digraph (see the example provided below as well as the SAGE documentation for digraphs: <http://doc.sagemath.org/html/en/reference/graphs/sage/graphs/digraph.html>);

(2) We must specify a directed cycle as the subgraph corresponding to Hamiltonian cycles in the digraph. This can easily be done via the line 'directedCycleGraph=DiGraph([[1..len(sageDirectedTargetGraph)], lambda a,b: b==mod(a+1,len(sageDirectedTargetGraph))])';

(3) We remove a factor of '2' from the automorphism count for cycles in undirected graphs (i.e. '(2*len(cycleGraph))' becomes 'len(directedCycleGraph)').

--

(Undirected Graphs) Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
from sage.graphs.generic_graph_pyx import SubgraphSearch
sageTargetGraph=Graph('GsXOXc')
cycleGraph=graphs.CycleGraph(len(sageTargetGraph))
SubgraphSearch(sageTargetGraph, cycleGraph).cardinality() / (2*len(cycleGraph))
```

Output: '6' (Output is correct, see e.g. <http://oeis.org/A066037>)

-

(Digraphs) Example with a directed cycle on $(k = 6)$ vertices:

```
from sage.graphs.generic_graph_pyx import SubgraphSearch
sageDirectedTargetGraph=DiGraph({1:[2],2:[3],3:[4],4:[5],5:[6],6:[1]})

directedCycleGraph=DiGraph([[1..len(sageDirectedTargetGraph)], lambda a,b:
b==mod(a+1,len(sageDirectedTargetGraph))])

SubgraphSearch(sageDirectedTargetGraph, directedCycleGraph).cardinality() /
len(directedCycleGraph)

Output: '1'
```

(HC.4.1) // (HC_digraph.4.1) & (HC.4.2) // (HC_digraph.4.2) The 'igraph' (version 1.0.0) R language (version 3.2.3) package 'count_subgraph_isomorphisms()' function:

Usage (Undirected Graphs): Let 'igraphTargetGraph' be an 'igraph' R package object encoding a simple connected undirected graph. Here, we first need to generate a simple cycle graph isomorphic to any Hamiltonian cycle in 'igraphTargetGraph' (should one or more exist), which can be done via the R language command: 'cycleGraph <- make_ring(vcount(igraphTargetGraph), directed = FALSE, mutual = FALSE, circular = TRUE)'. One can then call 'count_subgraph_isomorphisms(cycleGraph, igraphTargetGraph, method = [SELECT "lad" OR "vf2"])', to return the set of all labeled subgraphs in 'igraphTargetGraph' isomorphic to 'cycleGraph' corresponding to Hamiltonian cycles and cycles and their '(2*vcount(igraphTargetGraph)' automorphisms.

Putting everything together, we can ask for the total Hamiltonian cycle count in 'igraphTargetGraph' via the line:
'count_subgraph_isomorphisms(cycleGraph, igraphTargetGraph, method = [SELECT "lad" OR "vf2"]) / (2*vcount(igraphTargetGraph))'

We note that there are two algorithms one can select to enumerate subgraphs via the 'count_subgraph_isomorphisms()' function: "vf2" or "lad". The choice "lad" corresponds to the algorithm detailed in: [Solnon, C.: AllDifferent-based filtering for subgraph isomorphism. *Artificial Intelligence* **174**(12-13), pp. 850 - 864 (2010)] [162]; the choice "vf2" corresponds to the algorithm detailed in: [Cordella, L. P., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(10), pp. 1367 – 1372 (2004)] [42] (see also the earlier conference paper [41]). We denote the choice of the "lad" method as **(HC.4.1)** and the choice of the "vf2" method **(HC.4.2)**.

Regarding the documentation for 'count_subgraph_isomorphisms()' and 'make_ring()' functions, see the 'igraph' R package documentation:
<http://igraph.org/r/doc/igraph.pdf>.

SUPPLEMENTARY NOTE: While this feature is poorly documented, the command 'graph.subisomorphic.lad(cycleGraph,igraphTargetGraph)' will return a SINGLE explicit mapping of 'cycleGraph' to 'igraphTargetGraph' (i.e. a single Hamiltonian cycle). This function is of use because, in our hands, we were unable to find a way to have 'subgraph_isomorphisms(cycleGraph, igraphTargetGraph, method = [SELECT "lad" OR "vf2"])' return some subset of all such mappings (eq. Hamiltonian cycles), and this is of course problematic if 'igraphTargetGraph' has a very large number of possible Hamiltonian cycles.

-

Usage (Digraphs): The method of using the 'igraph' R package to count Hamiltonian cycles in digraphs is nearly the same as the above method for counting Hamiltonian cycles in undirected graphs, save for following three minor changes:

(1) The target graph, 'igraphTargetGraph', referred to hereafter as 'igraphDirectedTargetGraph', must be specified properly as a digraph (see the example provided below as well as the 'igraph' R package documentation:
<http://igraph.org/r/doc/igraph.pdf> --- all that's required here is to toggle all of the 'directed = FALSE' flags in the undirected example to 'directed = TRUE' and when adding edges 'add_edges(c(1,2, 2,3, 3,4, …' to note that the first item in each pair, e.g. '1,2,' corresponds to a vertex with a directed edge pointing at the second item in the pair);

(2) We must specify a directed cycle as the subgraph corresponding to Hamiltonian cycles in the digraph. This can easily be done via the line 'directedCycleGraph <- make_ring(vcount(igraphDirectedTargetGraph), directed = TRUE, mutual = FALSE, circular = TRUE)';

(3) We remove a factor of ‘2’ from the automorphism count for cycles in undirected graphs (i.e. ‘(2*vcount(igraphTargetGraph))’ becomes ‘vcount(igraphTargetGraph)’).

--

(Undirected Graphs) Example with an (n=3)-cube graph (graph6 (.g6) string: “GsXOXc”) via method (HC.4.1):

```
library(igraph)
igraphTargetGraph <- make_empty_graph(directed=FALSE) %>%
add_vertices(8) %>%
add_edges(c(1,2, 1,3, 1,4, 2,5)) %>%
add_edges(c(2,6, 3,5, 3,8, 4,6)) %>%
add_edges(c(4,8, 5,7, 6,7, 7,8))
cycleGraph <- make_ring(vcount(igraphTargetGraph), directed = FALSE, mutual = FALSE,
circular = TRUE)

count_subgraph_isomorphisms(cycleGraph, igraphTargetGraph, method = "lad") /
(2*vcount(igraphTargetGraph))
```

Output: ‘6’ (Output is correct, see e.g. <http://oeis.org/A066037>)

-

(Digraphs) Example with a directed cycle on $(k = 6)$ vertices via method (HC.4.1):

```
library(igraph)
igraphDirectedTargetGraph <- make_empty_graph(directed=TRUE) %>%
add_vertices(6) %>%
add_edges(c(1,2, 2,3, 3,4, 4,5,  5,6,  6,1))
directedCycleGraph <- make_ring(vcount(igraphDirectedTargetGraph), directed = TRUE,
mutual = FALSE, circular = TRUE)

count_subgraph_isomorphisms(directedCycleGraph, igraphDirectedTargetGraph, method =
"lad") / vcount(igraphDirectedTargetGraph)
```

Output: ‘1’

For Hamiltonian paths, we make use of the following ::

(HP.1) // (HP_digraph.1) Mathematica's (version 10.4.1) 'FindHamiltonianCycle[]' function with the simple Hamiltonian cycle to Hamiltonian path reduction method of connecting all vertices in the MMA target graph to a single vertex (denoted 'globalVertex'):

Usage (Undirected Graphs): Letting 'mmaTargetGraph' be a Mathematica 10.4.1 object encoding a simple connected undirected graph, the function call 'FindHamiltonianCycle[EdgeAdd[mmaTargetGraph,Table["globalVertex"<->VertexList[mmaTargetGraph][[i]],{i,1,Length[VertexList[mmaTargetGraph]]}]],All]' will return all Hamiltonian paths in 'mmaTargetGraph' and 'Length[FindHamiltonianCycle[EdgeAdd[mmaTargetGraph,Table["globalVertex"<->VertexList[mmaTargetGraph][[i]],{i,1,Length[VertexList[mmaTargetGraph]]}]],All]]' will return the count for all Hamiltonian cycles.

-

Usage (Digraphs): For digraphs the method is nearly equivalent to the above method for undirected graphs (under the same assumptions that the graph is simple as well as connected), however, we need to add two edges (one pointing in either direction) between each vertex in the original graph and the 'globalVertex'.

--

(Undirected Graphs) Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
mmaTargetGraph=ImportString["GsXOXc","graph6"];
Length[FindHamiltonianCycle[EdgeAdd[mmaTargetGraph,Table["globalVertex"<->VertexList[mmaTargetGraph][[i]],{i,1,Length[VertexList[mmaTargetGraph]]}]],All]]
```

Output: '72' (Output is correct; this value appropriately corresponds to 1/2 the number of (directed) Hamiltonian paths in the (n=3)-cube graph: <https://oeis.org/A091299>).

-

(Digraphs) Example with a directed cycle on $(k = 6)$ vertices:

```
mmaDirectedTargetGraph=Graph[{1->2,2->3,3->4,4->5,5->6,6->1}];
Length[FindHamiltonianCycle[EdgeAdd[mmaDirectedTargetGraph,Flatten[Table[{"globalVertex"->VertexList[mmaDirectedTargetGraph][[i]], VertexList[mmaDirectedTargetGraph][[i]]->"globalVertex"},{i,1,Length[VertexList[mmaDirectedTargetGraph]]}]]],All]]
```

Output: '6'

-

Regarding the enumeration of Hamiltonian paths in either the case of (Undirected Graphs) or (Digraphs):

To return a list of all Hamiltonian paths rather than a count (via the above method), replace the 'Length[…]' wrapper for 'FindHamiltonianCycle[…]' with the following ---

```
Map[Take[RotateLeft[#,SortBy[Position[#,"globalVertex"],Last][[1, 1]]],{1,-3}]&,
FindHamiltonianCycle[…]]
```

(HP.2) // (HP_digraph.2) Combinatorica's 'HamiltonianPath[]' function:

Usage (Undirected Graphs): Letting 'combinatoricaTargetGraph' be a Combinatorica object encoding a simple connected undirected graph, the function call '(1/2)*Length[HamiltonianPath[combinatoricaTargetGraph,All]]' will return all Hamiltonian paths in 'combinatoricaTargetGraph' (we divide by two to account for the reversals of each Hamiltonian path).

(Note :: See the posting by 'Szabolcs Horvat' at this link --- <http://community.wolfram.com/groups/-/m/t/880870?p_p_auth=S7lctraj> --- regarding a solution (which we make use of) for employing 'Block[]' to allow usage of Combinatorica functions without adding Combinatorica to '$ContextPath'.)

(Warning :: Loading the Graph Utilities package along with Combinatorica can interfere with the 'HamiltonianPath' function. E.g. for the (n=3)-cube graph 'Length[HamiltonianPath[combinatoricaTargetGraph,All]]' returns '126' instead of the appropriate value of '144' for the total number of (directed) Hamiltonian paths in the graph: <https://oeis.org/A091299>.)

-

Usage (Digraphs): The Combinatorica package does not appear to include a method for counting Hamiltonian cycles in digraphs.

--

Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
Quiet[Block[{$ContextPath},Needs["Combinatorica`"]]]
targetGraphAdjacencyMatrix=Normal[AdjacencyMatrix[ImportString["GsXOXc","graph6"]]];
combinatoricaTargetGraph=
Combinatorica`FromAdjacencyMatrix[targetGraphAdjacencyMatrix];
(1/2)*Length[Combinatorica`HamiltonianPath[combinatoricaTargetGraph,All]]
```

Output: '72' (Output is correct; this value appropriately corresponds to 1/2 the number of (directed) Hamiltonian paths in the (n=3)-cube graph: <https://oeis.org/A091299>).

(HP.3) // (HP_digraph.3) SAGE 7.2's 'SubgraphSearch()' function:

Usage (Undirected Graphs): Let 'sageTargetGraph' by a SAGE 7.2 object encoding a simple connected undirected graph. Here, we first need to generate a simple path graph (i.e. linear chain of vertices) isomorphic to any Hamiltonian path in 'sageTargetGraph' (should one or more exist), which can be done via the command: 'pathGraph=graphs.PathGraph(len(sageTargetGraph))'. We can then import 'SubgraphSearch()' via the command: the command 'from sage.graphs.generic_graph_pyx import SubgraphSearch', and call 'SubgraphSearch(sageTargetGraph, pathGraph)', to return the set of all labeled subgraphs in 'sageTargetGraph' isomorphic to 'pathGraph' corresponding to Hamiltonian paths and their reversals / automorphisms.

Putting all of this together, we can ask for the total Hamiltonian path count in 'sageTargetGraph' via the line:
'SubgraphSearch(sageTargetGraph, pathGraph).cardinality() / 2'.

We note that Nathann Cohen (<http://www.steinertriples.fr/ncohen/tut/Graphs/>) is the primary author for 'SubgraphSearch', and that the 'generic_graph_pyx' Cython package of Cython functions for graphs it was incorporated into was originally authored by Robert L. Miller.

For additional documentation please see:
<http://www.sagemath.org/documentation/html/en/reference/graphs/sage/graphs/generic_graph_pyx.html>.

For the source code please see: <http://www.sagenb.org/src/graphs/generic_graph.py>.

-

Usage (Digraphs): The method of using SAGE 7.2 to count Hamiltonian paths in digraphs is nearly the same as the above method for counting Hamiltonian cycles in undirected graphs, save for following three minor changes:

(1) The target graph, 'sageTargetGraph', referred to hereafter as 'sageDirectedTargetGraph', must be specified properly as a digraph (see the example provided below as well as the SAGE documentation for digraphs: <http://doc.sagemath.org/html/en/reference/graphs/sage/graphs/digraph.html>);

(2) We must specify a directed path as the subgraph corresponding to Hamiltonian path in the digraph. This can easily be done via the line 'directedPathGraph=DiGraph([[1..len(sageDirectedTargetGraph)], lambda a,b: b==a+1])';

(3) We remove a factor of '2' from the automorphism count for paths in undirected graphs (i.e. we no longer divide the output of 'SubgraphSearch()' by '2').

--

(Undirected Graphs) Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
from sage.graphs.generic_graph_pyx import SubgraphSearch
sageTargetGraph=Graph('GsXOXc')
pathGraph=graphs.PathGraph(len(sageTargetGraph))
SubgraphSearch(sageTargetGraph, pathGraph).cardinality() / 2
```

Output: '72' (Output is correct; this value appropriately corresponds to 1/2 the number of (directed) Hamiltonian paths in the (n=3)-cube graph: <https://oeis.org/A091299>).

-

(Digraphs) Example with a directed cycle on $(k = 6)$ vertices:

```
from sage.graphs.generic_graph_pyx import SubgraphSearch
sageDirectedTargetGraph=DiGraph({1:[2],2:[3],3:[4],4:[5],5:[6],6:[1]})
directedPathGraph=DiGraph([[1..len(sageDirectedTargetGraph)], lambda a,b: b==a+1])
SubgraphSearch(sageDirectedTargetGraph, directedCycleGraph).cardinality()

Output: '6'
```

(HP.4.1) // (HP_digraph.4.1) & (HP.4.2) // (HP_digraph.4.2) The 'igraph' (version 1.0.0) R language (version 3.2.3) package 'count_subgraph_isomorphisms()' function:

Usage (Undirected Graphs): Let 'igraphTargetGraph' be an 'igraph' R package object encoding a simple connected undirected graph. Here, we first need to generate a simple path graph (i.e. a linear chain of vertices) isomorphic to any Hamiltonian path in 'igraphTargetGraph' (should one or more exist), which can be done via the R language command: 'pathGraph <- make_ring(vcount(igraphTargetGraph), directed = FALSE, mutual = FALSE, circular = FALSE)'. One can then call 'count_subgraph_isomorphisms(pathGraph, igraphTargetGraph, method = [SELECT "lad" OR "vf2"])', to return the set of all labeled subgraphs in 'igraphTargetGraph' isomorphic to 'pathGraph' corresponding to Hamiltonian paths and their reversals / automorphisms.

Putting everything together, we can ask for the total Hamiltonian path count in 'igraphTargetGraph' via the line:
'count_subgraph_isomorphisms(pathGraph, igraphTargetGraph, method = [SELECT "lad" OR "vf2"]) / 2'

We note as in the prior section that there are two algorithms one can select to enumerate subgraphs via the 'count_subgraph_isomorphisms()' function: "vf2" or "lad". The choice "lad" corresponds to the algorithm detailed in: [Solnon, C.: AllDifferent-based filtering for subgraph isomorphism. *Artificial Intelligence* **174**(12-13), pp. 850 - 864 (2010)] [162]; the choice "vf2" corresponds to the algorithm detailed in: [Cordella, L. P., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(10), pp. 1367 – 1372 (2004)] [42] (see also the earlier conference paper [41]). We denote the choice of the "lad" method as **(HP.4.1)** and the choice of the "vf2" method **(HP.4.2)**.

Regarding the documentation for 'count_subgraph_isomorphisms()' and 'make_ring()' functions, see the 'igraph' R package documentation: <http://igraph.org/r/doc/igraph.pdf>.

SUPPLEMENTARY NOTE: While this feature is poorly documented, the command 'graph.subisomorphic.lad(pathGraph,igraphTargetGraph)' will return a SINGLE explicit mapping of 'pathGraph' to 'igraphTargetGraph' (i.e. a single Hamiltonian path). This function is of use because, in our hands, we were unable to find a way to have 'subgraph_isomorphisms(pathGraph, igraphTargetGraph, method = [SELECT "lad" OR "vf2"])' return some subset of all such mappings (eq. Hamiltonian path), and this is of course problematic for obvious reason if 'igraphTargetGraph' has a very large number of possible Hamiltonian paths.

-

Usage (Digraphs): The method of using the 'igraph' R package to count Hamiltonian paths in digraphs is nearly the same as the above method for counting Hamiltonian paths in undirected graphs, save for following three minor changes:

(1) The target graph, 'igraphTargetGraph', referred to hereafter as 'igraphDirectedTargetGraph', must be specified properly as a digraph (see the example provided below as well as the 'igraph' R package documentation: <http://igraph.org/r/doc/igraph.pdf> --- all that's required here is to toggle all of the 'directed = FALSE' flags in the undirected example to 'directed = TRUE' and when adding edges 'add_edges(c(1,2, 2,3, 3,4, …' to note that the first item in each pair, e.g. '1,2,' corresponds to a vertex with a directed edge pointing at the second item in the pair);

(2) We must specify a directed path as the subgraph corresponding to Hamiltonian paths in the digraph. This can easily be done via the line 'directedPathGraph <- make_ring(vcount(igraphDirectedTargetGraph), directed = TRUE, mutual = FALSE, circular = FALSE)';

(3) We remove a factor of ‘2’ from the automorphism count for paths in undirected graphs (i.e. we no longer divide the output of ‘count_subgraph_isomorphisms()’ by ‘2’).

--

(Undirected Graphs) Example with an (n=3)-cube graph (graph6 (.g6) string: “GsXOXc”) via method (HP.4.1):

```
igraphTargetGraph <- make_empty_graph(directed=FALSE) %>%
add_vertices(8) %>%
add_edges(c(1,2, 1,3, 1,4, 2,5)) %>%
add_edges(c(2,6, 3,5, 3,8, 4,6)) %>%
add_edges(c(4,8, 5,7, 6,7, 7,8))
pathGraph <- make_ring(vcount(igraphTargetGraph), directed = FALSE, mutual = FALSE,
circular = FALSE)

count_subgraph_isomorphisms(pathGraph, igraphTargetGraph, method = "lad") / 2
```

Output: ‘72’ (Output is correct; this value appropriately corresponds to 1/2 the number of (directed) Hamiltonian paths in the (n=3)-cube graph: <https://oeis.org/A091299>).

-

(Digraphs) Example with a directed cycle on $(k = 6)$ vertices via method (HC.4.1):

```
library(igraph)
igraphDirectedTargetGraph <- make_empty_graph(directed=TRUE) %>%
add_vertices(6) %>%
add_edges(c(1,2, 2,3, 3,4, 4,5, 5,6, 6,1))
directedPathGraph <- make_ring(vcount(igraphDirectedTargetGraph), directed = TRUE,
mutual = FALSE, circular = FALSE)

count_subgraph_isomorphisms(directedPathGraph, igraphDirectedTargetGraph, method =
"lad")
```

Output: ‘6’

7.7 A method of contracting $(k \leq 3)$-edge-connected subgraphs to show non-Hamiltonicity of large graph structures; the Mathematica 10.4.1 'SubgraphContract[]' function

If one wishes to show that some undirected graph G is non-Hamiltonian, and the size of the vertex and/or edge set for G frustrates direct application of e.g. the backtracking algorithms discusse in section **(7.6)**, one manner of proceeding is to attempt to establish the non-Hamiltonicity of a special minor H of G and to somehow show that this establishes the non-Hamiltonicity of G.

Specifically, we observe the following lemma:

Lemma :: If H is a minor of an undirected graph G, generated via the contraction of one or more disjoint $(k \leq 3)$-edge-connected subgraphs (or equivalently, one or more disjoint subgraphs that can only be entered once by any Hamiltonian cycle in G), then if H is non-Hamiltonian we have that G is non-Hamiltonian.

Proof: Let S be a $(k \leq 3)$-edge-connected subgraph of G, i.e. a subgraph that may be disconnected from $G \setminus S$ via removal of k (where $k \leq 3$) edges from the set Q of edges with strictly one end in S. Let V_S and E_S represent the vertex and edge sets of S, respectively. Consider that a Hamiltonian cycle in G, should one exist, may only ever flow in and out of the subgraph S once by traversing two of the edges in the set Q. Said differently, if we have a cycle graph (i.e. a cyclic chain of connected vertices) corresponding to any Hamiltonian cycle in G, the traversal of this Hamiltonian cycle through S will correspond to a linear chain of vertices with exactly $|V_S| - 1$ edges. If we now generate a minor H of G be contracting S to a single vertex v_S, for any Hamiltonian cycle in G we can likewise contract the linear chain of vertices corresponding to a traversal of the subgraph S to generate a Hamiltonian cycle for H. Therefore, if H is a minor of an undirected graph G generated via the contraction of a $(k \leq 3)$-edge-connected subgraph, or by the same argument, a set of disjoint $(k \leq 3)$-edge-connected subgraphs, if H is non-Hamiltonian then G is non-Hamiltonian. We ask the reader to please note, however, that the converse of this statement need not hold.

The below Mathematica 10.4.1 function --- 'SubgraphContract[]' --- can be used to contract a specified subgraph in a Mathematica 10.4.1 graph object ::

```
(*
New function 'SubgraphContract[]' ::
Input 1: Original MMA 10.4.1 (undirected / directed / mixed) graph object;
Input 2: List of vertices specifying the target subgraph for contraction;
Input 3: Label for vertex representing contracted subgraph;
*)
SubgraphContract:=
    With[{originalVertexCoordinateList=
    Table[VertexList[#1][[i]]->GraphEmbedding[#1][[i]],
    {i,1,Length[VertexList[#1]]}],
        subgraphAdjacencyList=Complement[Union[Flatten[
        Table[AdjacencyList[#1,#2[[i]]],{i,1,Length[#2]}]]],#2]},
        If[Length[#2]>1&&ConnectedGraphQ[Subgraph[#1,#2]],
            SetProperty[VertexDelete[EdgeAdd[VertexAdd[#1,{#3}],
                Table[
                    Which[
                        Select[EdgeList[#1],
                        ((subgraphAdjacencyList[[i]]==#[[1]]&&
                        MemberQ[subgraphAdjacencyList,#[[2]]]==False)||
                        (subgraphAdjacencyList[[i]]==#[[2]]&&
                        MemberQ[subgraphAdjacencyList,#[[1]]]==False))&&
                        (StringContainsQ[ToString[#],"\[UndirectedEdge]"]==True||
                        StringContainsQ[ToString[#],"<->"]==True)&]!={},
                            #3<->subgraphAdjacencyList[[i]],
                        Select[EdgeList[#1],subgraphAdjacencyList[[i]]==#[[1]]&&
                        MemberQ[subgraphAdjacencyList,#[[2]]]==False&&
                        (StringContainsQ[ToString[#],"\[DirectedEdge]"]==True||
                        StringContainsQ[ToString[#],"->"]==True)&]!={},
                            #3\[DirectedEdge]subgraphAdjacencyList[[i]],
                        Select[EdgeList[#1],subgraphAdjacencyList[[i]]==#[[2]]&&
                        MemberQ[subgraphAdjacencyList,#[[1]]]==False&&
                        (StringContainsQ[ToString[#],"\[DirectedEdge]"]==True||
                        StringContainsQ[ToString[#],"->"]==True)&]!={},
                            subgraphAdjacencyList[[i]]\[DirectedEdge]#3],
                {i,1,Length[subgraphAdjacencyList]}]],#2],
            VertexCoordinates->Append[originalVertexCoordinateList,
            #3->Mean[Table[originalVertexCoordinateList[[All,2]][[Position[
            originalVertexCoordinateList[[All,1]],
            #2[[i]]][[1,1]]]],{i,1,Length[#2]}]]]
            ,
            "Subgraph contraction failure; check inputs."
        ]
    ]&;
```

-

Example usage of the 'SubgraphContract[]' function; we contract (to a vertex labeled 'contractedVertex') four vertices defining a face of the 3-cube, yielding a graph isomorphic to the wheel graph W_5:

```
mmaTargetGraph=Graph[{1<->2,1<->3,1<->4,2<->5,2<->6,3<->5,3<->8,4<->6,4<->8,5<->7,6<-
>7,7<->8},VertexLabels->"Name"];
contractedMMATargetGraph=SubgraphContract[mmaTargetGraph,{1,2,4,6},"contractedVertex"]
w5WheelGraph=Graph[{1<->2,1<->3,1<->4,1<->5,2<->3,2<->5,3<->4,4<->5}]
IsomorphicGraphQ[contractedMMATargetGraph,w5WheelGraph]
```

Output: (1) a drawing of 'contractedMMATargetGraph'; (2) 'True' as the output of 'IsomorphicGraphQ[contractedMMATargetGraph,w5WheelGraph]'.

7.8 Planarity testing with Mathematica 10.4.1, Combinatorica, and SAGE 7.2

Using Mathematica's 'PlanarGraphQ[]' function:

Usage: Letting 'mmaTargetGraph' be a Mathematica 10.4.1 graph object encoding a simple undirected graph, the function call 'PlanarGraphQ[mmaTargetGraph]' will return 'True' if 'mmaTargetGraph' is a planar graph and 'False' otherwise.

--

Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
mmaTargetGraph = ImportString["GsXOXc","graph6"];
PlanarGraphQ[mmaTargetGraph]
```

Output: 'True'

Using Combinatorica's 'PlanarQ[]' function:

Usage: Letting 'combinatoricaTargetGraph' be a Combinatorica graph object encoding a simple undirected graph, the function call 'PlanarQ[combinatoricaTargetGraph]' will return 'True' if 'combinatoricaTargetGraph' is a planar graph and 'False' otherwise.

Regarding the procedure underlying the 'PlanarQ[]' function, we refer the reader to (pg. 370) of the textbook guide to Combinatorica (written by Pemmaraju and Skiena): [Pemmaraju, S. V., Skiena, S. S.: Computational discrete mathematics: combinatorics and graph theory with Mathematica. Cambridge University Press: New York, NY (2003).] [143]. Providing our interpretation of the author's words, while it is true that multiple O(n) planarity testing algorithms exist (one of the first was reported in: [Hopcroft, J., Tarjan, R.: Efficient planarity testing. *J. ACM* **21**(4), pp. 549 - 568 (1974)] [81]), these implementations are "…difficult to implement…". The authors then state, however, that "…Most of these algorithms are based on ideas from an old $O(n^3)$ algorithm by Auslander and Parter [AP61]…" (here [AP61] refers to: [Auslander, L., Parter, S.: On imbedding graphs in the sphere. *J. Mathematics and Mechanics* **10**(3), pp. 517 - 523 (1961)] [11]; regarding the title, recall that there exists a topological embedding of a graph into S^2 iff there exists a topological embedding of the graph into $\mathbb{R}^2$). Therefore, we assume that the 'PlanarQ[]' implementation was likewise based on the Auslander-Parter $O(n^3)$ algorithm. We also refer the reader to the source code for the latest version of Combinatorica (v2.0.0): <http://homepage.cs.uiowa.edu/~sriram/Combinatorica/NewCombinatorica.m>.

(Note :: See the posting by 'Szabolcs Horvat' at this link --- <http://community.wolfram.com/groups/-/m/t/880870?p_p_auth=S7lctraj> --- regarding a solution (which we make use of) for employing 'Block[]' to allow usage of Combinatorica functions without adding Combinatorica to '$ContextPath'.)

--

Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
Quiet[Block[{$ContextPath},Needs["Combinatorica`"]]]
targetGraphAdjacencyMatrix = Normal[AdjacencyMatrix[ImportString["GsXOXc","graph6"]]];
combinatoricaTargetGraph=
Combinatorica`FromAdjacencyMatrix[targetGraphAdjacencyMatrix];
Combinatorica`PlanarQ[combinatoricaTargetGraph]
```

Output: 'True'

Using SAGE 7.2's 'is_planar()' function:

Usage: After first loading the 'is_planar()' function via the command 'from sage.graphs.planarity import is_planar', and letting 'sageTargetGraph' be a SAGE 7.2

graph object encoding a simple undirected graph, the function call
'is_planar(sageTargetGraph)' will return 'True' if 'sageTargetGraph' is a planar graph
and 'False' otherwise.

We note the procedure underlying the 'is_planar()' function is a Cython implementation
of the O(n) Boyer-Myrvold planarity testing algorithm: [Boyer, J. M., Myrvold, W. J.:
On the cutting edge: simplified O(n) planarity by edge addition. *J. Graph Algorithms
Appl.* **8**(3), pp. 241 - 273 (2004)] [30].

Regarding the SAGE documentation for this function, we refer the reader to the
following link:
<http://doc.sagemath.org/html/en/reference/graphs/sage/graphs/planarity.html>

--

Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
from sage.graphs.planarity import is_planar
sageTargetGraph=Graph('GsXOXc')
is_planar(sageTargetGraph)
```

Output: 'True'

7.9 Determining the minimum vertex cut for a graph with Combinatorica, SAGE 7.2, and the 'igraph' R package

Using Mathematica 10.4.1's 'VertexConnectivity[]' function:

Usage: Letting 'mmaTargetGraph' be a Mathematica 10.4.1 graph object encoding a simple undirected and connected graph, the function call 'VertexConnectivity[mmaTargetGraph]' will return the minimum vertex cut of the graph.

(Warning :: Combinatorica's function for the minimum vertex cut is also denoted 'VertexConnectivity[]'; adding the Combinatorica package (e.g. via 'Needs["Combinatorica`"]') to the general MMA context path (i.e. '$ContextPath') will disable Mathematica 10.4.1's native 'VertexConnectivity[]' function. See the posting by 'Szabolcs Horvat' at this link --- <http://community.wolfram.com/groups/-/m/t/880870?p_p_auth=S7lctraj> --- regarding a solution (which we make use of) for employing 'Block[]' to allow usage of Combinatorica functions without adding Combinatorica to '$ContextPath'.)

--

Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
mmaTargetGraph = ImportString["GsXOXc","graph6"];
VertexConnectivity[mmaTargetGraph]
```

Output: '3'

Using Combinatorica's 'VertexConnectivity[]' function:

Usage: Letting 'combinatoricaTargetGraph' be a Combinatorica graph object encoding a simple undirected and connected graph, the function call 'VertexConnectivity[combinatoricaTargetGraph]' will return the minimum vertex cut of the graph.

(Note :: Mathamtica 10.4.1's function for the minimum vertex cut is also denoted 'VertexConnectivity[]'; adding the Combinatorica package (e.g. via 'Needs["Combinatorica`"]') to the general MMA context path (i.e. '$ContextPath') will disable Mathematica 10.4.1's native 'VertexConnectivity[]' function. See the posting by 'Szabolcs Horvat' at this link --- <http://community.wolfram.com/groups/-/m/t/880870?p_p_auth=S7lctraj> --- regarding a solution (which we make use of) for employing 'Block[]' to allow usage of Combinatorica functions without adding Combinatorica to '$ContextPath'.)

(Warning :: Loading the Graph Utilities package along with Combinatorica may interfere with the 'VertexConnectivity[]' function.)

--

Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
Quiet[Block[{$ContextPath},Needs["Combinatorica`"]]]
targetGraphAdjacencyMatrix = Normal[AdjacencyMatrix[ImportString["GsXOXc","graph6"]]];
combinatoricaTargetGraph=
Combinatorica`FromAdjacencyMatrix[targetGraphAdjacencyMatrix];
Combinatorica`VertexConnectivity[combinatoricaTargetGraph]
```

Output: '3'

Using SAGE 7.2's 'vertex_connectivity()' function:

Usage: Letting 'sageTargetGraph' be a SAGE 7.2 graph object encoding a simple undirected and connected graph, the command: 'sageTargetGraph.vertex_connectivity()' will return the minimum vertex cut of the graph.

--

Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
sageTargetGraph=Graph('GsXOXc')
sageTargetGraph.vertex_connectivity()
```

Output: '3'

Using the 'igraph' R package's 'vertex_connectivity()' function:

Usage: Letting 'igraphTargetGraph' be an 'igraph' R package graph object encoding a simple undirected and connected graph, the command:
'vertex_connectivity(igraphTargetGraph, source = NULL, target = NULL, checks = TRUE)' will return the minimum vertex cut of the graph.

--

Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
library(igraph)
igraphTargetGraph <- make_empty_graph(directed=FALSE) %>%
add_vertices(8) %>%
add_edges(c(1,2, 1,3, 1,4, 2,5)) %>%
add_edges(c(2,6, 3,5, 3,8, 4,6)) %>%
add_edges(c(4,8, 5,7, 6,7, 7,8))
vertex_connectivity(igraphTargetGraph, source = NULL, target = NULL, checks = TRUE)
```

Output: '3'

7.10 Determining the chromatic number $\chi(G)$ for a graph with Mathematica 10.4.1, Combinatorica, and SAGE 7.2

Computing the chromatic number $\chi(G)$ of a graph with Mathematica 10.4.1:

While Mathematica 10.4.1 has a 'BipartiteGraphQ[]' function, which efficiently tests if a Mathematica 10.4.1 graph object is bipartite, we are unaware of any other built-in functionality to compute chromatic numbers or chromatic polynomials for graphs.

Computing the chromatic number $\chi(G)$ of a graph with Combinatorica:

Combinatorica has a 'ChromaticNumber[]' function, which will return to the chromatic number of a Combinatorica graph object. This function, after checking e.g. that that the input graph is not bipartite, makes repeated calls to Combinatorica's 'Backtrack[]' algorithm.

Computing the chromatic number $\chi(G)$ of a graph with SAGE 7.2:

See:
<http://doc.sagemath.org/html/en/reference/graphs/sage/graphs/graph_coloring.html>

There are (at least) two functions that may be called in SAGE 7.2 to compute the chromatic number $\chi(G)$ of a graph: 'chromatic_number(sageTargetGraph)' and 'vertex_coloring()'. Here, we also make use of SAGE 7.2's 'is_bipartite()' function.

Note that prior to using 'chromatic_number()' or 'vertex_coloring()', the two functions must be imported. For the 'chromatic_number()' function this can be done via the command 'from sage.graphs.graph_coloring import chromatic_number'; for the 'vertex_coloring' function this can be done via the command 'from sage.graphs.graph_coloring import vertex_coloring'. Once imported the relevant function(s) are imported, either the call 'chromatic_number(sageTargetGraph)' or 'vertex_coloring(sageTargetGraph, value_only=True)' will return the chromatic number of a specified SAGE 7.2 graph object ('sageTargetGraph').

There are multiple additional options for 'vertex_coloring', which are detailed at the aforementioned link to the SAGE documentation on graph coloring. For example, with regards to the 'vertex_coloring()' function, while we specify 'value_only=True' above, if one specifies 'value_only=False' a minimal k-coloring of the graph vertices will be returned. One can directly test for a existence of a k-coloring for a specified value of k. While this may lead to speedups if one expects a small range of possible minimum colorations for a given 'sageTargetGraph', as a word of caution, one is NOT testing here if a given k-coloration is a minimal coloring, only that such a k-coloration exists. Here, for example, to test if 'sageTargetGraph' is $(k = 3)$-colorable, which will return 'True' if 'sageTargetGraph' is either 3-colorable or bipartite, one can call 'vertex_coloring()' as follows:
'vertex_coloring(sageTargetGraph, k=3, value_only=True)'.

Finally, if one is working with large and/or pathological graph objects, it's important to note that Mixed Integer Linear Programming (MILP) solvers, such as CPLEX and GUROBI, can be specified for use by the 'chromatic_number(sageTargetGraph)' and 'vertex_coloring()' functions. See the following SAGE documentation page for installation instructions:
<http://doc.sagemath.org/html/en/thematic_tutorials/linear_programming.html>. We do NOT, however, install or otherwise make use of optional MILP solvers for this work.

7.11 Determining the girth for a graph with Combinatorica, SAGE 7.2, and the 'igraph' R package

Output of Combinatorica's 'Girth[]' function:

Output of SAGE 7.2's 'girth()' function:

Output for the 'igraph' R package 'girth()' function:

Using Combinatorica's 'Girth[]' function:

Usage: Letting 'combinatoricaTargetGraph' be a Combinatorica graph object encoding a simple undirected and connected graph, the function call 'Girth[combinatoricaTargetGraph]' will return the girth (i.e. minimum cycle length) of the graph.

(Note :: See the posting by 'Szabolcs Horvat' at this link --- <http://community.wolfram.com/groups/-/m/t/880870?p_p_auth=S7lctraj> --- regarding a solution (which we make use of) for employing 'Block[]' to allow usage of Combinatorica functions without adding Combinatorica to '$ContextPath'.)

(Warning :: Loading the Graph Utilities package along with Combinatorica may interfere with the 'Girth[]' function.)

--

Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
Quiet[Block[{$ContextPath},Needs["Combinatorica`"]]]
mmaEdgeList=EdgeList[ImportString["GsXOXc","graph6"]];
targetGraphAM=Normal[AdjacencyMatrix[Graph[mmaEdgeList]]];
combinatoricaTargetGraph=Combinatorica`FromAdjacencyMatrix[targetGraphAM];
Combinatorica`Girth[combinatoricaTargetGraph]
```

Output: '4'

Using SAGE 7.2's 'girth()' function:

Usage: Letting 'sageTargetGraph' be a SAGE 7.2 graph object encoding a simple undirected and connected graph, the command: 'sageTargetGraph.girth()' will return the girth (i.e. minimum cycle length) of the graph.

--

Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
sageTargetGraph=Graph('GsXOXc')
sageTargetGraph.girth()
```

Output: '4'

Using the 'igraph' R package's 'girth()' function:

Usage: Letting 'igraphTargetGraph' be an 'igraph' R package graph object encoding a simple undirected and connected graph, the command: 'vertex_connectivity(igraphTargetGraph, source = NULL, target = NULL, checks = TRUE)' will return the girth (i.e. minimum cycle length) of the graph.

--

Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
library(igraph)
igraphTargetGraph <- make_empty_graph(directed=FALSE) %>%
add_vertices(8) %>%
add_edges(c(1,2, 1,3, 1,4, 2,5)) %>%
add_edges(c(2,6, 3,5, 3,8, 4,6)) %>%
add_edges(c(4,8, 5,7, 6,7, 7,8))
girth(igraphTargetGraph)
```

Output:

```
$girth
[1] 4

$circle
+ 4/8 vertices:
[1] 7 8 4 6
```

7.12 Determining the minimum genus $\gamma_{min}(G)$ and minimum genus embedding of a graph with SAGE 7.2

The genus of a graph is a natural number $k \in \mathbb{N}$ specifying the minimum number of handles that must be added to a planar surface S to allow for the embedding of a given graph G, or said differently, a "drawing" on S without edge crossings.

Determining if a graph has genus $\leq k \in \mathbb{N}$ is an NP-complete problem (see: [Thomassen, C.: The graph genus problem is NP-complete. *J. Algorithms* **10**(4), pp. 568 - 576 (1989)] [169] where the classic NP-complete problem of determining if a graph G has an independent vertex set of size $\alpha(G) \geq k \in \mathbb{N}$ is reduced to the problem of determining if the graph has genus $\gamma(G) \leq k \in \mathbb{N}$), however the genus of special types of graphs, such as complete graphs or complete bipartite graphs, may be computing in polynomial-time. For examples of these special families of graphs and corresponding literature references, we refer the reader to the Wolfram Mathworld "Graph Genus" page: <http://mathworld.wolfram.com/GraphGenus.html>.

For make use of SAGE 7.2's method of calculating a graph's genus by determining the genus of all possible combinatorial embeddings of the given graph and determining the minimum value:
<http://combinat.sagemath.org/doc/reference/graphs/sage/graphs/genus.html>.

Using SAGE 7.2's 'genus()' function:

Usage: Letting 'sageTargetGraph' be a SAGE 7.2 graph object encoding a simple undirected and connected graph, the routine shown in the examples (below), which requires one to import 'sage.graphs.genus', will return the genus of 'sageTargetGraph'. Additionally, if one calls 'genus(record_embedding = True)', as shown in the example, the function 'get_embedding()' can then be used to return a clockwise-ordered list of nearest-neighbors for each vertex in a minimum genus embedding, which is sufficient to specify an embedding if the surface one is embedding into is orientable.

--

Example with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
import sage.graphs.genus

sageTargetGraph=Graph('GsXOXc')

G = Graph(sageTargetGraph, implementation='c_graph', sparse=False)
gb = sage.graphs.genus.simple_connected_genus_backtracker(G._backend.c_graph()[0])
gb.genus(record_embedding = True)
```

Output ::

'0'

```
-

gb.get_embedding()

Output ::

{0: [1, 2, 3],
 1: [0, 5, 4],
 2: [0, 4, 7],
 3: [0, 7, 5],
 4: [1, 6, 2],
 5: [1, 3, 6],
 6: [4, 5, 7],
 7: [2, 6, 3]}

--
```

Example with K_6, i.e. the complete graph on $k = 6$ vertices (graph6 (.g6) string: "E~~w"):

```
import sage.graphs.genus

sageTargetGraph=Graph('E~~w')

G = Graph(sageTargetGraph, implementation='c_graph', sparse=False)
gb = sage.graphs.genus.simple_connected_genus_backtracker(G._backend.c_graph()[0])
gb.genus(record_embedding = True)

Output ::

'1'

-

gb.get_embedding()

Output ::

{0: [1, 2, 3, 4, 5],
 1: [0, 2, 3, 5, 4],
 2: [0, 4, 3, 1, 5],
 3: [0, 5, 1, 2, 4],
 4: [0, 3, 2, 1, 5],
 5: [0, 4, 1, 3, 2]}
```

--

Example with the (Figure 4.9.b) cubic 3-connected bipartite graph (on 42 vertices):

```
import sage.graphs.genus

sageTargetGraph=Graph([(0,2),(1,5),(2,6),(2,17),(3,6),(3,7),(3,8),(4,9),(4,10),(4,11),
(5,11),(5,20),(6,21),(7,12),(7,31),(8,13),(8,32),(9,15),(9,33),(10,16),(10,34),(11,24)
,(12,17),(12,18),(13,18),(13,27),(14,15),(14,26),(14,27),(15,19),(16,19),(16,20),(17,3
0),(18,22),(19,23),(20,35),(21,25),(21,36),(22,25),(22,26),(23,28),(23,29),(24,29),(24
,39),(25,31),(26,32),(27,28),(28,33),(29,34),(30,36),(30,37),(31,37),(32,37),(33,38),(
34,38),(35,38),(35,39),(36,40),(39,41)])

G = Graph(sageTargetGraph, implementation='c_graph', sparse=False)
gb = sage.graphs.genus.simple_connected_genus_backtracker(G._backend.c_graph()[0])
gb.genus(record_embedding = True)
```

Output ::

```
'2'
```

-

```
gb.get_embedding()
```

Output (embedding in the S_2 orientable surface --- FORMAT: "vertex v_i: [clockwise ordering of vertices about v_i in $S_g = S_2$ embedding]) ::

```
{0: [2],
 1: [5],
 2: [0, 6, 17],
 3: [6, 7, 8],
 4: [9, 10, 11],
 5: [1, 11, 20],
 6: [2, 3, 21],
 7: [3, 12, 31],
 8: [3, 32, 13],
 9: [4, 33, 15],
 10: [4, 16, 34],
 11: [4, 5, 24],
 12: [7, 17, 18],
 13: [8, 27, 18],
 14: [15, 27, 26],
 15: [9, 14, 19],
 16: [10, 19, 20],
 17: [2, 30, 12],
 18: [12, 13, 22],
 19: [15, 23, 16],
 20: [5, 35, 16],
 21: [6, 36, 25],
 22: [18, 26, 25],
 23: [19, 28, 29],
 24: [11, 29, 39],
 25: [21, 31, 22],
 26: [14, 32, 22],
 27: [13, 14, 28],
 28: [23, 27, 33],
 29: [23, 34, 24],
 30: [17, 37, 36],
 31: [7, 25, 37],
 32: [8, 37, 26],
 33: [9, 38, 28],
 34: [10, 29, 38],
```

```
35: [20, 38, 39],
36: [21, 40, 30],
37: [30, 32, 31],
38: [33, 35, 34],
39: [24, 35, 41],
40: [36],
41: [39]}
```

7.13 Introduction to Brendan McKay's NAUTY and Plantri packages

References ::

NAUTY package:
[McKay, B. D., Piperno, A.: Practical graph isomorphism, II. *J. Symbolic. Comput.* **60**, pp. 94 - 112 (2014)] [122]

Plantri package:
[Brinkmann, G., McKay, B. D.: Fast generation of planar graphs. *Match Commun. Math. Co.* **58**(2), pp. 323 - 357 (2007)] [33]

NOTE: For usage in the context of SAGE 7.2, either or both packages must be installed. See <http://doc.sagemath.org/html/en/reference/misc/sage/misc/package.html> for a list of optional packages for SAGE ('nauty' corresponds to NAUTY and 'plantri' corresponds to Plantri) as well as package installation instructions.

NAUTY (No AUTomorphisms, Yes?) ::

Brendan McKay's NAUTY (No AUTomorphisms, Yes?) software, consisting of a set of routines written in C, includes the world's fastest practical algorithms for determination of the generators, cardinalities, and orbits for the automorphism groups of connected and unconnected graphs (and hence, graph isomorphism testing), and was the first software used to enumerate all 1,182,004 order (n=11) non-isomorphic graphs. NAUTY also includes an absolutely fantastic algorithm denoted 'geng()' for enumerating all elements in sets of connected and/or unconnected graphs of specified order and with specified restrictions for e.g. upper- and lower-bounds on vertex degrees, edge counts, and edge 2-connectivity, as well as properties like bipartiteness.

The NAUTY package, as well as instructions for its use (including instructions for the use of the 'geng()' algorithm and its variants), are available at the following website: <http://users.cecs.anu.edu.au/~bdm/nauty/>

-

Example usage of NAUTY's 'geng()' algorithm to generate all 2-connected bipartite cubic (planar and non-planar) graphs of order $(n = 16)$:

Let's call 'geng()' with the following command: ./geng -C -b -d2 -D3 16 24:24 -g 'outputTarget.txt'

This will result in the following example command prompt output:

">A ./geng -Cbd2D3 n=16 e=24
>Z 38 graphs generated in 0.14 sec"

This will also output, to the specified file 'outputTarget.txt', all non-isomorphic graphs that: (1) are 2-edge-connected (toggle '-C'); (2) are bipartite (toggle -b); (3) have minimium vertex degree 2 (toggle 'd2'); (4) have maximum degree 3 (toggle 'D3'); (5) are of order $(n = 16)$; (6) have between k1:k2=24:24 edges (where we set k1=k2=24 to force the graph to be cubic despite the allowance of minimum vertex degree of 2). Finally, the toggle '-g' specifies a graph6 (.g6) encoding scheme for the graphs outputted to the specified file 'outputTarget.txt'. Note that the order in which the toggles are specified is of no consequence.

Plantri ::

Brinkmann and McKay's 'plantri()' algorithm is an extension of NAUTY's 'geng()' that allows for very fast and memory efficient generation of planar graphs (i.e. graphs embeddable in $\mathbb{R}^2$ or S^2 without edge crossings) and their planar duals (e.g. the dual of a 3-connected bipartite cubic planar graph is a eulerian planar triangulation).

The Plantri package, as well as instructions for its use, are available at the following website: <https://users.cecs.anu.edu.au/~bdm/plantri/>

-

Example usage of the 'plantri()' algorithm to generate all 3-connected bipartite cubic planar graphs of order $(n = 32)$:

Let's call 'plantri()' with the following command: ./plantri -bp –c3 16 –e24:24 –g 'outputTarget.txt'

This will result in the following example command prompt output:

"2 bipartite graphs written to plantriOutputDUMP.txt; cpu=0.00 sec"

This will also output, to the specified file 'outputTarget.txt', all non-isomorphic planar graphs that: (1) are at least 3-edge-connected (toggle '–c3'); (2) are bipartite (toggle –bp; note that just specifying 'b' will cause 'plantri()' to output a Eulerian planar triangulation); (3) are of order (n=16); (4) and have between –ek1:k2=–e24:24 edges (where we set k1=k2=24 to force the graph to be cubic despite the allowance of minimum vertex degree of 2). Finally, and just like with 'geng()', note that the toggle '–g' specifies a graph6 (.g6) encoding scheme for the graphs outputted to the specified file 'outputTarget.txt' (unlike 'geng()' though this is no longer "default" behavior), and note that the order in which the toggles are specified is of no consequence.

Briefly elaborating on this example, the output for a call to 'plantri()' of the form --- ./plantri -bp –c3 n –e(3n/2):(3n/2) –g 'outputTarget.txt' --- where 'n' corresponds to the order of the target family of graphs, will exactly match the following On-Line Encyclopedia of Integer Sequences (OEIS) entry for 3-connected bipartite cubic planar graphs (Barnette) graphs: "Number of unlabeled trivalent 3-connected bipartite planar graphs with 2n nodes." <https://oeis.org/A007083>

(n)	a(n)
4	0
6	0
8	1
10	0
12	1
14	1
16	2
18	2
20	8
22	8
24	32
26	57
28	185
30	466
32	1543

(NOTE / WARNING :: The "A007083 as a simple table" link (i.e. <https://oeis.org/A007083/list>) incorrectly states that their list corresponds to 'n vs. a(n)'. This is a mistake as their list should be labeled '2n vs. a(n)'. For the reader's convenience, we provide the corrected 'n vs. a(n)' list immediately above.)

7.14 A Mathematica 10.4.1 script to trace the faces and return the dual of an arbitrary 3-connected planar graph

The following two scripts take as input an arbitrary 3-connected planar graph and return, in the case of 'TraceFaces[]', a set of edge lists corresponding to the faces of the graph, and in the case of 'GenerateDualGraph[]', the dual of the graph. From [Whitney, H.: Congruent graphs and the connectivity of graphs. *Amer. J. Math.* **54**(1), pp. 150 - 168 (1932)] [189] we have that any 3-connected planar graph is "uniquely embeddable" in $\mathbb{R}^2$ or S^2 and that any two planar embeddings of the same 3-connected planar graph will necessarily have isomorphic combinatorial (eq. geometrical) duals.

```
(*Provided an arbitrary 3-connected planar graph 'TraceFaces[]' will return a list of
edge sets defining the faces of the graph.*)
TraceFaces[mmaInputGraph_]:=Module[{mmaTargetGraph,mmaVertexList,mmaEdgeList,mmaVCList
,allPossibleDirectedEdgesList,counterClockwiseVertexOrderings,currentFace,faceList,cur
rentEdge,initialVertex,initialVertexReturnFlag},vertexLabelResetMap=Table[VertexList[m
maInputGraph][[i]]->i,{i,1,Length[VertexList[mmaInputGraph]]}];
mmaVertexList=Sort[Flatten[ReplaceAll[VertexList[mmaInputGraph],vertexLabelResetMap]]]
;
mmaEdgeList=Flatten[ReplaceAll[EdgeList[mmaInputGraph],vertexLabelResetMap]];
mmaTargetGraph=Graph[mmaVertexList,mmaEdgeList,GraphLayout->"PlanarEmbedding"];
mmaVCList=SortBy[Table[VertexList[mmaTargetGraph][[i]]-
>GraphEmbedding[mmaTargetGraph][[i]],{i,1,Length[mmaVertexList]}],First];
allPossibleDirectedEdgesList=EdgeList[DirectedGraph[mmaEdgeList,VertexCoordinates-
>mmaVCList]];
counterClockwiseVertexOrderings=Table[mmaVCList[[All,1]][[Flatten[Map[Position[mmaVCLi
st[[All,2]],#]&,Sort[Take[GraphEmbedding[SetProperty[NeighborhoodGraph[mmaTargetGraph,
mmaVertexList[[i]],1],VertexCoordinates->mmaVCList]],{2,-
1}],ToPolarCoordinates[#1+({0,0}-mmaVCList[[i,2]])][[2]]<ToPolarCoordinates[#2+({0,0}-
mmaVCList[[i,2]])][[2]]&]]]],{i,1,VertexCount[mmaTargetGraph]}];
faceList={};
While[Length[allPossibleDirectedEdgesList]>0,
currentEdge=allPossibleDirectedEdgesList[[1]];
currentFace={};
initialVertexReturnFlag=False;
initialVertex=currentEdge[[1]];
While[initialVertexReturnFlag==False,
currentFace=Append[currentFace,currentEdge[[1]]];
currentEdge=currentEdge[[2]]-
>counterClockwiseVertexOrderings[[currentEdge[[2]]]][[Mod[Position[counterClockwiseVer
texOrderings[[currentEdge[[2]]]],currentEdge[[1]]][[1,1]]+1,Length[counterClockwiseVer
texOrderings[[currentEdge[[2]]]],1]]];
If[currentEdge[[1]]==initialVertex,
initialVertexReturnFlag=True;
];
];
allPossibleDirectedEdgesList=Delete[allPossibleDirectedEdgesList,Partition[Map[Positio
n[allPossibleDirectedEdgesList,#][[1,1]]&,Append[Table[currentFace[[r]]\[DirectedEdge]
currentFace[[r+1]],{r,1,Length[currentFace]-1}],currentFace[[-
1]]\[DirectedEdge]currentFace[[1]]]],1]];
currentFace=Append[Table[Sort[{currentFace[[r]],currentFace[[r+1]]}][[1]]<-
>Sort[{currentFace[[r]],currentFace[[r+1]]}][[2]],{r,1,Length[currentFace]-
1}],Sort[{currentFace[[-1]],currentFace[[1]]}][[1]]<->Sort[{currentFace[[-
1]],currentFace[[1]]}][[2]]];
faceList=Append[faceList,currentFace];
];
Table[Map[Replace[#[[1]],Map[#[[2]]->#[[1]]&,vertexLabelResetMap]]<-
>Replace[#[[2]],Map[#[[2]]-
>#[[1]]&,vertexLabelResetMap]]&,faceList[[i]]],{i,1,Length[faceList]}]
];
```

```
(*Provided an arbitrary 3-connected planar graph 'GenerateDualGraph[]' will return its
dual; note that this procedure requires 'TraceFaces[]' to be defined prior to its
use.*)
GenerateDualGraph[mmaInputGraph_]:=Module[{faces=TraceFaces[mmaInputGraph],pairsOfElem
ents},pairsOfElements=Subsets[Range[Length[faces]],{2}];
Graph[Pick[Table[pairsOfElements[[i,1]]<-
>pairsOfElements[[i,2]],{i,1,Length[pairsOfElements]}],Map[#1>0&,Map[Length[Intersecti
on[#[[1]],#[[2]]]]&,Subsets[faces,{2}]],True],GraphLayout->"PlanarEmbedding"]
];
```

-

Example usage of 'TraceFaces[]' with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc"):

```
mmaTargetGraph=Graph[{1<->2,1<->3,1<->4,2<->5,2<->6,3<->5,3<->8,4<->6,4<->8,5<->7,6<-
>7,7<->8}];

TraceFaces[mmaTargetGraph]

Output ::

'{{1<->2,2<->6,4<->6,1<->4},{1<->3,3<->5,2<->5,1<->2},{1<->4,4<->8,3<->8,1<->3},{2<-
>5,5<->7,6<->7,2<->6},{3<->8,8<->7,5<->7,3<->5},{4<->6,6<->7,8<->7,4<->8}}'
```

-

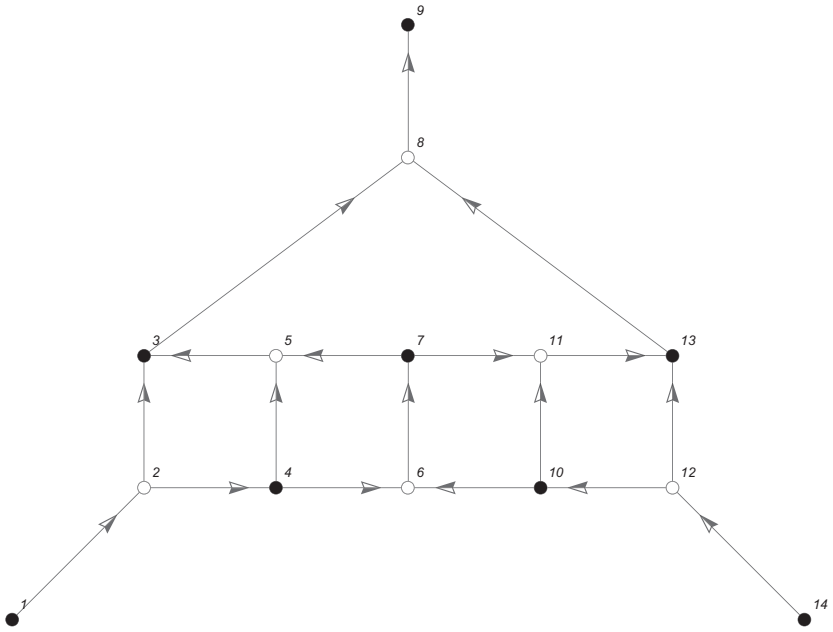
Example usage of 'GenerateDualGraph[]' with an (n=3)-cube graph (graph6 (.g6) string: "GsXOXc") which will generate the skeleton graph of the octahedron:

```
mmaTargetGraph=Graph[{1<->2,1<->3,1<->4,2<->5,2<->6,3<->5,3<->8,4<->6,4<->8,5<->7,6<-
>7,7<->8}];

octahedronSkeletonGraph=Graph[{1<->2,1<->3,1<->4,1<->5,2<->3,2<->5,2<->6,3<->4,3<-
>6,4<->5,4<->6,5<->6}];

IsomorphicGraphQ[GenerateDualGraph[mmaTargetGraph],octahedronSkeletonGraph]

Output ::

'True'
```

Chapter 8

Appendix: Gadget Properties, Edge Lists, and Embedding Coordinates

8.1 (Figure 2.2) gadget (height (q = 1) instance)



Graph Properties ::

(NOTE: Properties in this section are computed assuming that the digraph is undirected.)

--

Number of Vertices: '14'

Number of Edges: '18'

Minimum vertex degree $\delta(G)$: '1'

Maximum vertex degree $\Delta(G)$: '3'

Output of SAGE 7.2's 'is_regular(k=3)' function: 'False'

--

Planarity:

Output of Mathematica 10.4.1's 'PlanarGraphQ[]' function: 'True'

Output of Combinatorica's 'PlanarQ[]' function: 'True'

Output of SAGE 7.2's 'is_planar(sageTargetGraph)' function: 'True'

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1's 'VertexConnectivity[]' function: '1'

Output of Combinatorica's 'VertexConnectivity[]' function: '1'

Output of SAGE 7.2's 'vertex_connectivity()' function: '1'

Output for the 'igraph' R package command 'vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)': '1'

--

Chromatic Number $\chi(G)$:

Output of Combinatorica's 'ChromaticNumber[]' function: '2'

Output of SAGE 7.2's 'chromatic_number(sageTargetGraph)' function: '2'

Output of Mathematica 10.4.1's 'BipartiteGraphQ[]' function: 'True'

Output of Combinatorica's 'BipartiteQ[]' function: 'True'

Output of SAGE 7.2's 'is_bipartite()' function: 'True'

--

Girth:

Output of Combinatorica's 'Girth[]' function: '4'

Output of SAGE 7.2's 'girth()' function: '4'

Output for the 'igraph' R package 'girth()' function: '4'

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1\[DirectedEdge]2,2\[DirectedEdge]3,2\[DirectedEdge]4,3\[DirectedEdge]8,4\[DirectedEd
ge]5,4\[DirectedEdge]6,5\[DirectedEdge]3,6\[DirectedEdge]7,7\[DirectedEdge]5,7\[Direct
edEdge]11,8\[DirectedEdge]9,10\[DirectedEdge]6,10\[DirectedEdge]11,11\[DirectedEdge]13
,12\[DirectedEdge]10,12\[DirectedEdge]13,13\[DirectedEdge]8,14\[DirectedEdge]12}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-3.,-1.6429},2->{-2.,-0.6429},3->{-2.,0.3571},4->{-1.,-0.6429},5->{-1.,0.3571},6-
>{0.,-0.6429},7->{0.,0.3571},8->{0.,1.8571},9->{0.,2.8571},10->{1.,-0.6429},11-
>{1.,0.3571},12->{2.,-0.6429},13->{2.,0.3571},14->{3.,-1.6429}}

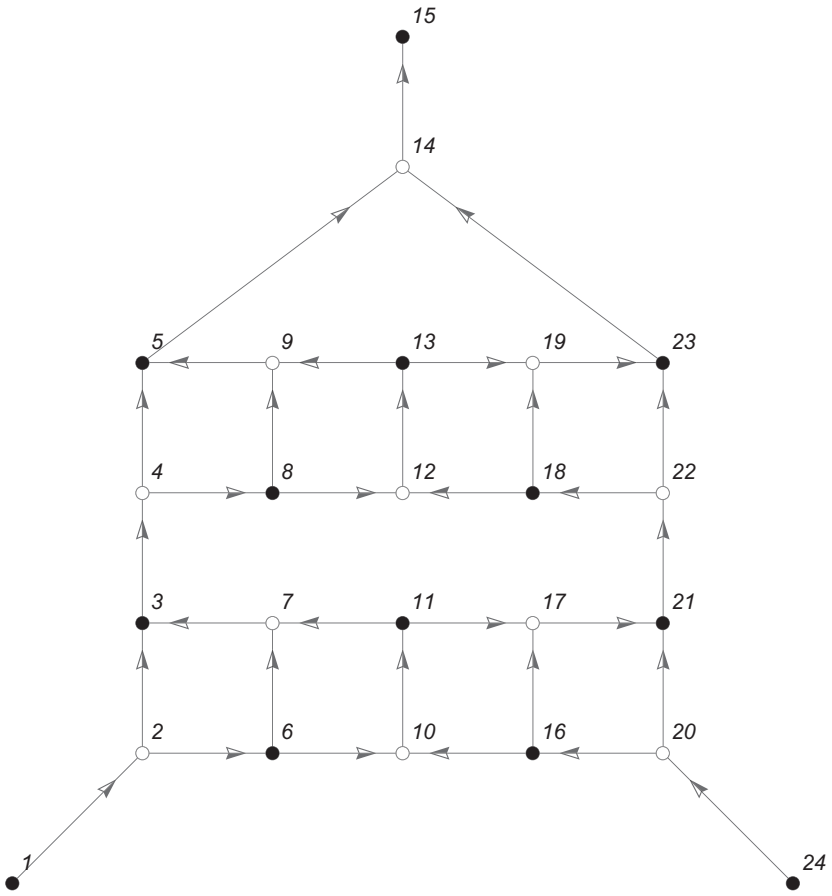
Edge list (SAGE 7.2):
[(0,1),(1,2),(1,3),(2,7),(3,4),(3,5),(4,2),(5,6),(6,4),(6,10),(7,8),(9,5),(9,10),(10,1
2),(11,9),(11,12),(12,7),(13,11)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-3.,-1.6429],1:[-2.,-0.6429],2:[-2.,0.3571],3:[-1.,-0.6429],4:[-1.,0.3571],5:[0.,-
0.6429],6:[0.,0.3571],7:[0.,1.8571],8:[0.,2.8571],9:[1.,-
0.6429],10:[1.,0.3571],11:[2.,-0.6429],12:[2.,0.3571],13:[3.,-1.6429]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorTwo,3->colorOne,4->colorOne,5->colorTwo,6->colorTwo,7-
>colorOne,8->colorTwo,9->colorOne,10->colorOne,11->colorTwo,12->colorTwo,13-
>colorOne,14->colorOne}

8.2 (Figure 2.2) gadget (height (q = 2) instance)



Graph Properties ::

(NOTE: Properties in this section are computed assuming that the digraph is undirected.)

--

Number of Vertices: '24

Number of Edges: '33

Minimum vertex degree $\delta(G)$: '1'

Maximum vertex degree $\Delta(G)$: '3'

Output of SAGE 7.2's 'is_regular(k=3)' function: 'False'

--

Planarity:

Output of Mathematica 10.4.1's 'PlanarGraphQ[]' function: 'True'

Output of Combinatorica's 'PlanarQ[]' function: 'True'

Output of SAGE 7.2's 'is_planar(sageTargetGraph)' function: 'True'

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1's 'VertexConnectivity[]' function: '1'

Output of Combinatorica's 'VertexConnectivity[]' function: '1'

Output of SAGE 7.2's 'vertex_connectivity()' function: '1'

Output for the 'igraph' R package command 'vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)': '1'

--

Chromatic Number $\chi(G)$:

Output of Combinatorica's 'ChromaticNumber[]' function: '2'

Output of SAGE 7.2's 'chromatic_number(sageTargetGraph)' function: '2'

Output of Mathematica 10.4.1's 'BipartiteGraphQ[]' function: 'True'

Output of Combinatorica's 'BipartiteQ[]' function: 'True'

Output of SAGE 7.2's 'is_bipartite()' function: 'True'

--

Girth:

Output of Combinatorica's 'Girth[]' function: '4'

Output of SAGE 7.2's 'girth()' function: '4'

Output for the 'igraph' R package 'girth()' function: '4'

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1\[DirectedEdge]2,2\[DirectedEdge]3,2\[DirectedEdge]6,3\[DirectedEdge]4,4\[DirectedEdge]5,4\[DirectedEdge]8,5\[DirectedEdge]14,6\[DirectedEdge]7,6\[DirectedEdge]10,7\[DirectedEdge]3,8\[DirectedEdge]9,8\[DirectedEdge]12,9\[DirectedEdge]5,10\[DirectedEdge]11,11\[DirectedEdge]7,11\[DirectedEdge]17,12\[DirectedEdge]13,13\[DirectedEdge]9,13\[DirectedEdge]19,14\[DirectedEdge]15,16\[DirectedEdge]10,16\[DirectedEdge]17,17\[DirectedEdge]21,18\[DirectedEdge]12,18\[DirectedEdge]19,19\[DirectedEdge]23,20\[DirectedEdge]16,20\[DirectedEdge]21,21\[DirectedEdge]22,22\[DirectedEdge]18,22\[DirectedEdge]23,23\[DirectedEdge]14,24\[DirectedEdge]20}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-3.,-2.5833},2->{-2.,-1.5833},3->{-2.,-0.5833},4->{-2.,0.4167},5->{-2.,1.4167},6->{-1.,-1.5833},7->{-1.,-0.5833},8->{-1.,0.4167},9->{-1.,1.4167},10->{0.,-1.5833},11->{0.,-0.5833},12->{0.,0.4167},13->{0.,1.4167},14->{0.,2.9167},15->{0.,3.9167},16->{1.,-1.5833},17->{1.,-0.5833},18->{1.,0.4167},19->{1.,1.4167},20->{2.,-1.5833},21->{2.,-0.5833},22->{2.,0.4167},23->{2.,1.4167},24->{3.,-2.5833}}

Edge list (SAGE 7.2):

[(0,1),(1,2),(1,5),(2,3),(3,4),(3,7),(4,13),(5,6),(5,9),(6,2),(7,8),(7,11),(8,4),(9,10),(10,6),(10,16),(11,12),(12,8),(12,18),(13,14),(15,9),(15,16),(16,20),(17,11),(17,18),(18,22),(19,15),(19,20),(20,21),(21,17),(21,22),(22,13),(23,19)]

-

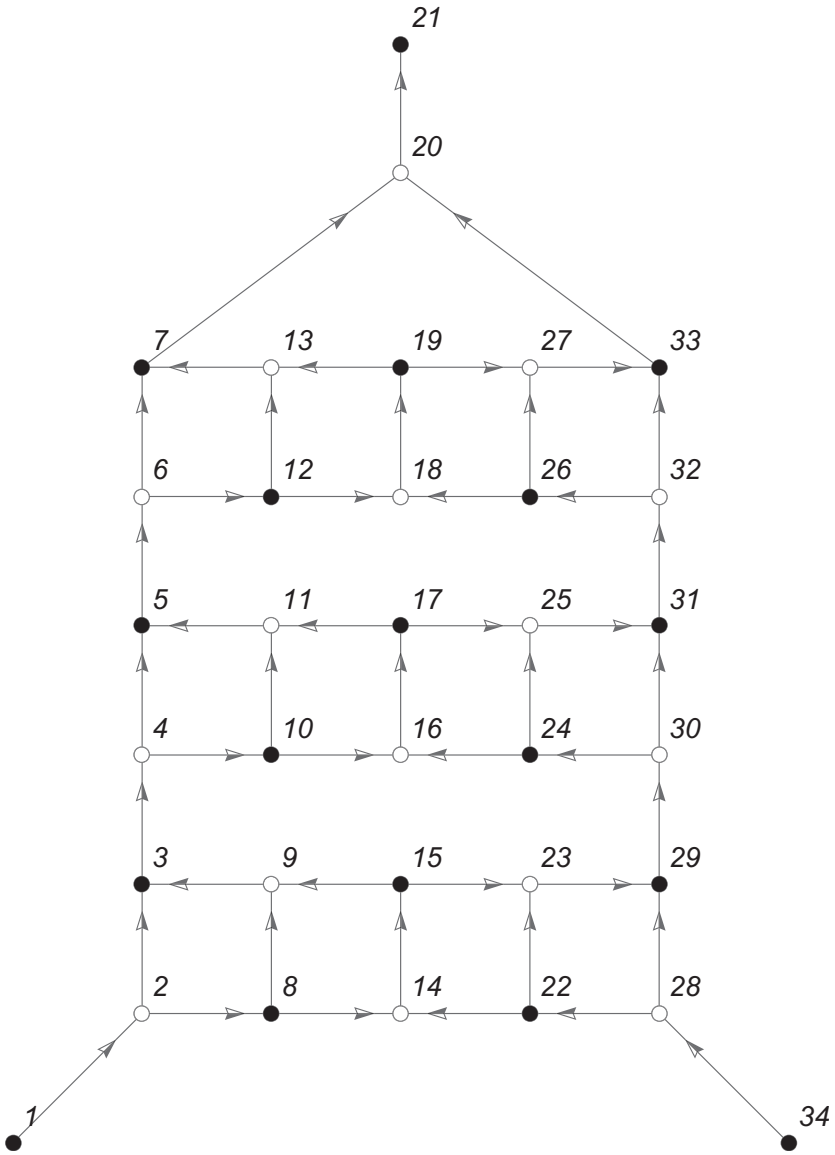
Example embedding coordinates (SAGE 7.2):

{0:[-3.,-2.5833],1:[-2.,-1.5833],2:[-2.,-0.5833],3:[-2.,0.4167],4:[-2.,1.4167],5:[-1.,-1.5833],6:[-1.,-0.5833],7:[-1.,0.4167],8:[-1.,1.4167],9:[0.,-1.5833],10:[0.,-0.5833],11:[0.,0.4167],12:[0.,1.4167],13:[0.,2.9167],14:[0.,3.9167],15:[1.,-1.5833],16:[1.,-0.5833],17:[1.,0.4167],18:[1.,1.4167],19:[2.,-1.5833],20:[2.,-0.5833],21:[2.,0.4167],22:[2.,1.4167],23:[3.,-2.5833]}

Canonical vertex $(k = 2)$-coloring ::

{1->colorOne,2->colorTwo,3->colorOne,4->colorTwo,5->colorOne,6->colorOne,7->colorTwo,8->colorOne,9->colorTwo,10->colorTwo,11->colorOne,12->colorTwo,13->colorOne,14->colorTwo,15->colorOne,16->colorOne,17->colorTwo,18->colorOne,19->colorTwo,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorOne}

8.3 (Figure 2.2) gadget (height (q = 3) instance)



Graph Properties ::

(NOTE: Properties in this section are computed assuming that the digraph is undirected.)

--

Number of Vertices: '34'

Number of Edges: '48'

Minimum vertex degree $\delta(G)$: '1'

Maximum vertex degree $\Delta(G)$: '3'

Output of SAGE 7.2's 'is_regular(k=3)' function: 'False'

--

Planarity:

Output of Mathematica 10.4.1's 'PlanarGraphQ[]' function: 'True'

Output of Combinatorica's 'PlanarQ[]' function: 'True'

Output of SAGE 7.2's 'is_planar(sageTargetGraph)' function: 'True'

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1's 'VertexConnectivity[]' function: '1'

Output of Combinatorica's 'VertexConnectivity[]' function: '1'

Output of SAGE 7.2's 'vertex_connectivity()' function: '1'

Output for the 'igraph' R package command 'vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)': '1'

--

Chromatic Number $\chi(G)$:

Output of Combinatorica's 'ChromaticNumber[]' function: '2'

Output of SAGE 7.2's 'chromatic_number(sageTargetGraph)' function: '2'

Output of Mathematica 10.4.1's 'BipartiteGraphQ[]' function: 'True'

Output of Combinatorica's 'BipartiteQ[]' function: 'True'

Output of SAGE 7.2's 'is_bipartite()' function: 'True'

--

Girth:

Output of Combinatorica's 'Girth[]' function: '4'

Output of SAGE 7.2's 'girth()' function: '4'

Output for the 'igraph' R package 'girth()' function: '4'

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1\[DirectedEdge]2,2\[DirectedEdge]3,2\[DirectedEdge]8,3\[DirectedEdge]4,4\[DirectedEdge]5,4\[DirectedEdge]10,5\[DirectedEdge]6,6\[DirectedEdge]7,6\[DirectedEdge]12,7\[DirectedEdge]20,8\[DirectedEdge]9,8\[DirectedEdge]14,9\[DirectedEdge]3,10\[DirectedEdge]11,10\[DirectedEdge]16,11\[DirectedEdge]5,12\[DirectedEdge]13,12\[DirectedEdge]18,13\[DirectedEdge]7,14\[DirectedEdge]15,15\[DirectedEdge]9,15\[DirectedEdge]23,16\[DirectedEdge]17,17\[DirectedEdge]11,17\[DirectedEdge]25,18\[DirectedEdge]19,19\[DirectedEdge]13,19\[DirectedEdge]27,20\[DirectedEdge]21,22\[DirectedEdge]14,22\[DirectedEdge]23,23\[DirectedEdge]29,24\[DirectedEdge]16,24\[DirectedEdge]25,25\[DirectedEdge]31,26\[DirectedEdge]18,26\[DirectedEdge]27,27\[DirectedEdge]33,28\[DirectedEdge]22,28\[DirectedEdge]29,29\[DirectedEdge]30,30\[DirectedEdge]24,30\[DirectedEdge]31,31\[DirectedEdge]32,32\[DirectedEdge]26,32\[DirectedEdge]33,33\[DirectedEdge]20,34\[DirectedEdge]28}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-3.,-3.5588},2->{-2.,-2.5588},3->{-2.,-1.5588},4->{-2.,-0.5588},5->{-2.,0.4412},6->{-2.,1.4412},7->{-2.,2.4412},8->{-1.,-2.5588},9->{-1.,-1.5588},10->{-1.,-0.5588},11->{-1.,0.4412},12->{-1.,1.4412},13->{-1.,2.4412},14->{0.,-2.5588},15->{0.,-1.5588},16->{0.,-0.5588},17->{0.,0.4412},18->{0.,1.4412},19->{0.,2.4412},20->{0.,3.9412},21->{0.,4.9412},22->{1.,-2.5588},23->{1.,-1.5588},24->{1.,-0.5588},25->{1.,0.4412},26->{1.,1.4412},27->{1.,2.4412},28->{2.,-2.5588},29->{2.,-1.5588},30->{2.,-0.5588},31->{2.,0.4412},32->{2.,1.4412},33->{2.,2.4412},34->{3.,-3.5588}}

Edge list (SAGE 7.2):

[(0,1),(1,2),(1,7),(2,3),(3,4),(3,9),(4,5),(5,6),(5,11),(6,19),(7,8),(7,13),(8,2),(9,10),(9,15),(10,4),(11,12),(11,17),(12,6),(13,14),(14,8),(14,22),(15,16),(16,10),(16,24),(17,18),(18,12),(18,26),(19,20),(21,13),(21,22),(22,28),(23,15),(23,24),(24,30),(25,17),(25,26),(26,32),(27,21),(27,28),(28,29),(29,23),(29,30),(30,31),(31,25),(31,32),(32,19),(33,27)]

-

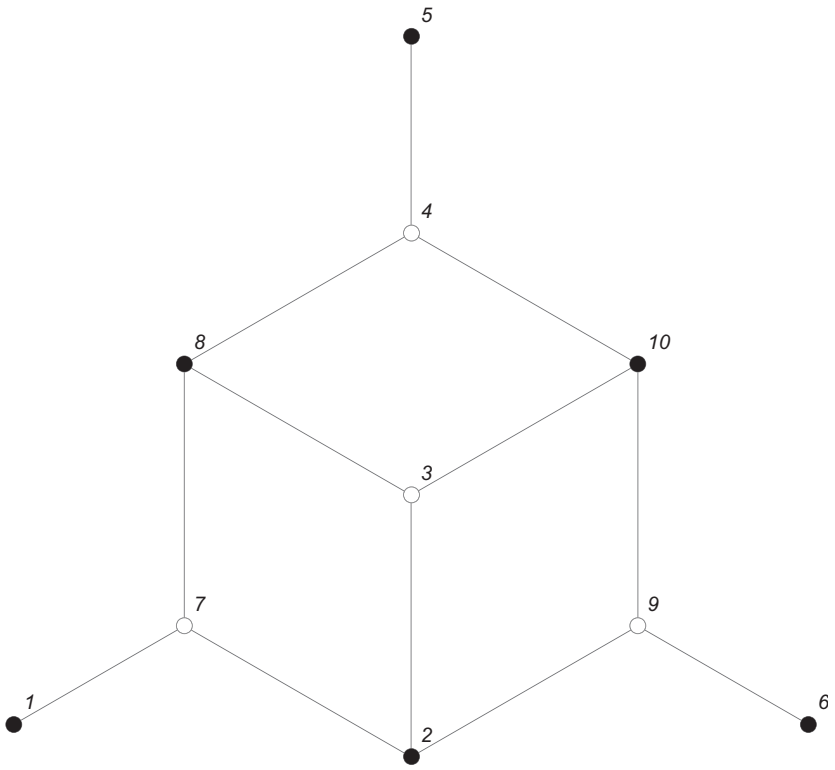
Example embedding coordinates (SAGE 7.2):

{0:[-3.,-3.5588],1:[-2.,-2.5588],2:[-2.,-1.5588],3:[-2.,-0.5588],4:[-2.,0.4412],5:[-2.,1.4412],6:[-2.,2.4412],7:[-1.,-2.5588],8:[-1.,-1.5588],9:[-1.,-0.5588],10:[-1.,0.4412],11:[-1.,1.4412],12:[-1.,2.4412],13:[0.,-2.5588],14:[0.,-1.5588],15:[0.,-0.5588],16:[0.,0.4412],17:[0.,1.4412],18:[0.,2.4412],19:[0.,3.9412],20:[0.,4.9412],21:[1.,-2.5588],22:[1.,-1.5588],23:[1.,-0.5588],24:[1.,0.4412],25:[1.,1.4412],26:[1.,2.4412],27:[2.,-2.5588],28:[2.,-1.5588],29:[2.,-0.5588],30:[2.,0.4412],31:[2.,1.4412],32:[2.,2.4412],33:[3.,-3.5588]}

Canonical vertex $(k = 2)$-coloring ::

```
{1->colorOne,2->colorTwo,3->colorOne,4->colorTwo,5->colorOne,6->colorTwo,7-
>colorOne,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13-
>colorTwo,14->colorTwo,15->colorOne,16->colorTwo,17->colorOne,18->colorTwo,19-
>colorOne,20->colorTwo,21->colorOne,22->colorOne,23->colorTwo,24->colorOne,25-
>colorTwo,26->colorOne,27->colorTwo,28->colorTwo,29->colorOne,30->colorTwo,31-
>colorOne,32->colorTwo,33->colorOne,34->colorOne}
```

8.4 (Figure 2.3) gadget (depth (q = 1) instance)



Graph Properties ::

--

Number of Vertices: ‘10’

Number of Edges: ‘12’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->7,2<->3,2<->7,2<->9,3<->8,3<->10,4<->5,4<->8,4<->10,6<->9,7<->8,9<->10}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-1.5155,-0.875},2->{0.,-1.},3->{0.,0.},4->{0.,1.},5->{0.,1.75},6->{1.5155,-0.875},7->{-0.866,-0.5},8->{-0.866,0.5},9->{0.866,-0.5},10->{0.866,0.5}}

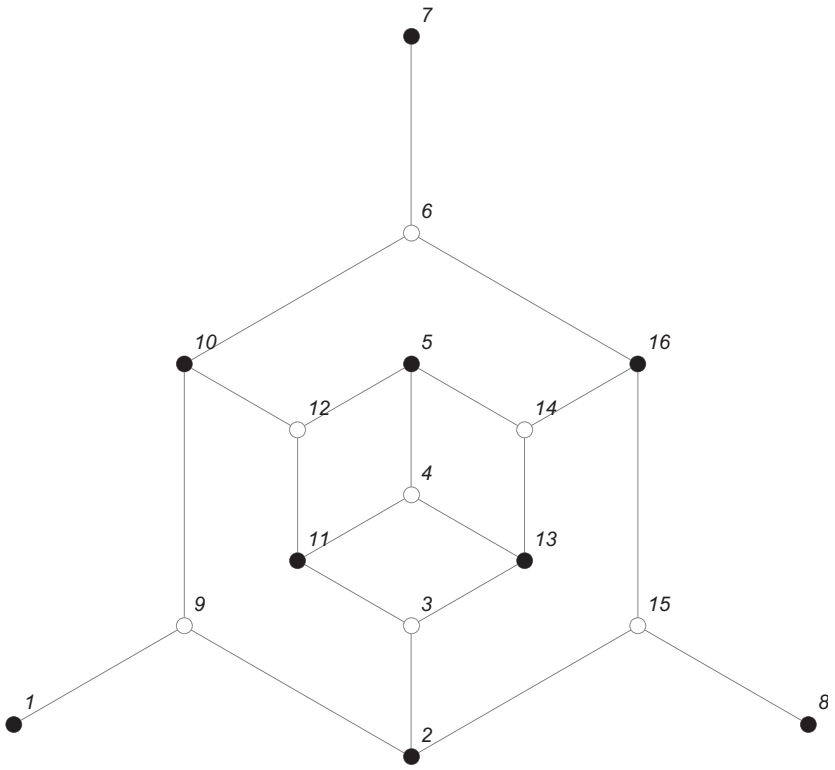
Edge list (SAGE 7.2):
[(0,6),(1,2),(1,6),(1,8),(2,7),(2,9),(3,4),(3,7),(3,9),(5,8),(6,7),(8,9)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-1.5155,-0.875],1:[0.,-1.],2:[0.,0.],3:[0.,1.],4:[0.,1.75],5:[1.5155,-0.875],6:[-0.866,-0.5],7:[-0.866,0.5],8:[0.866,-0.5],9:[0.866,0.5]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorOne,3->colorTwo,4->colorTwo,5->colorOne,6->colorOne,7->colorTwo,8->colorOne,9->colorTwo,10->colorOne}

8.5 (Figure 2.3) gadget (depth (q = 2) instance)



Graph Properties ::

--

Number of Vertices: ‘16’

Number of Edges: ‘21’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->9,2<->3,2<->9,2<->15,3<->11,3<->13,4<->5,4<->11,4<->13,5<->12,5<->14,6<->7,6<->10,6<->16,8<->15,9<->10,10<->12,11<->12,13<->14,14<->16,15<->16}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-1.5155,-0.875},2->{0.,-1.},3->{0.,-0.5},4->{0.,0.},5->{0.,0.5},6->{0.,1.},7->{0.,1.75},8->{1.5155,-0.875},9->{-0.866,-0.5},10->{-0.866,0.5},11->{-0.433,-0.25},12->{-0.433,0.25},13->{0.433,-0.25},14->{0.433,0.25},15->{0.866,-0.5},16->{0.866,0.5}}

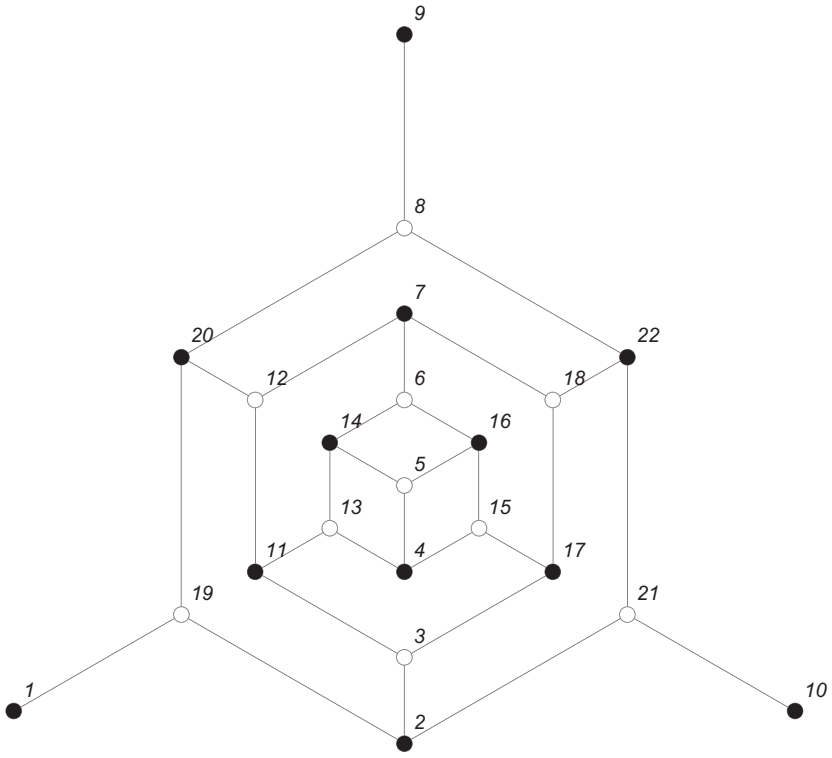
Edge list (SAGE 7.2):
[(0,8),(1,2),(1,8),(1,14),(2,10),(2,12),(3,4),(3,10),(3,12),(4,11),(4,13),(5,6),(5,9),(5,15),(7,14),(8,9),(9,11),(10,11),(12,13),(13,15),(14,15)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-1.5155,-0.875],1:[0.,-1.],2:[0.,-0.5],3:[0.,0.],4:[0.,0.5],5:[0.,1.],6:[0.,1.75],7:[1.5155,-0.875],8:[-0.866,-0.5],9:[-0.866,0.5],10:[-0.433,-0.25],11:[-0.433,0.25],12:[0.433,-0.25],13:[0.433,0.25],14:[0.866,-0.5],15:[0.866,0.5]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorOne,3->colorTwo,4->colorTwo,5->colorOne,6->colorTwo,7->colorOne,8->colorOne,9->colorTwo,10->colorOne,11->colorOne,12->colorTwo,13->colorOne,14->colorTwo,15->colorTwo,16->colorOne}

8.6 (Figure 2.3) gadget (depth (q = 3) instance)



Graph Properties ::

--

Number of Vertices: ‘22’

Number of Edges: ‘30’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->19,2<->3,2<->19,2<->21,3<->11,3<->17,4<->5,4<->13,4<->15,5<->14,5<->16,6<->7,6<->14,6<->16,7<->12,7<->18,8<->9,8<->20,8<->22,10<->21,11<->12,11<->13,12<->20,13<->14,15<->16,15<->17,17<->18,18<->22,19<->20,21<->22}

-

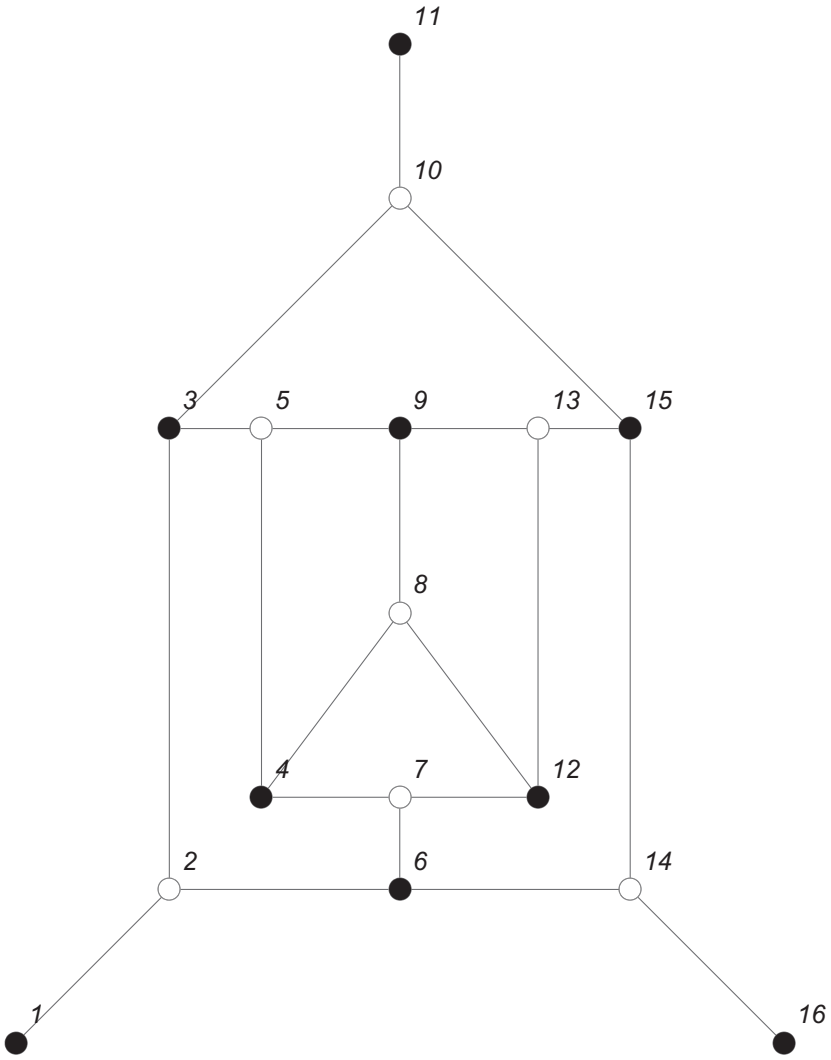
Example embedding coordinates (Mathematica 10.4.1):
{1->{-1.5155,-0.875},2->{0.,-1.},3->{0.,-0.6667},4->{0.,-0.3333},5->{0.,0.},6->{0.,0.3333},7->{0.,0.6667},8->{0.,1.},9->{0.,1.75},10->{1.5155,-0.875},11->{-0.5774,-0.3333},12->{-0.5774,0.3333},13->{-0.2887,-0.1667},14->{-0.2887,0.1667},15->{0.2887,-0.1667},16->{0.2887,0.1667},17->{0.5774,-0.3333},18->{0.5774,0.3333},19->{-0.866,-0.5},20->{-0.866,0.5},21->{0.866,-0.5},22->{0.866,0.5}}

Edge list (SAGE 7.2):
[(0,18),(1,2),(1,18),(1,20),(2,10),(2,16),(3,4),(3,12),(3,14),(4,13),(4,15),(5,6),(5,13),(5,15),(6,11),(6,17),(7,8),(7,19),(7,21),(9,20),(10,11),(10,12),(11,19),(12,13),(14,15),(14,16),(16,17),(17,21),(18,19),(20,21)]

Example embedding coordinates (SAGE 7.2):
{0:[-1.5155,-0.875],1:[0.,-1.],2:[0.,-0.6667],3:[0.,-0.3333],4:[0.,0.],5:[0.,0.3333],6:[0.,0.6667],7:[0.,1.],8:[0.,1.75],9:[1.5155,-0.875],10:[-0.5774,-0.3333],11:[-0.5774,0.3333],12:[-0.2887,-0.1667],13:[-0.2887,0.1667],14:[0.2887,-0.1667],15:[0.2887,0.1667],16:[0.5774,-0.3333],17:[0.5774,0.3333],18:[-0.866,-0.5],19:[-0.866,0.5],20:[0.866,-0.5],21:[0.866,0.5]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorOne,3->colorTwo,4->colorOne,5->colorTwo,6->colorTwo,7->colorOne,8->colorTwo,9->colorOne,10->colorOne,11->colorOne,12->colorTwo,13->colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorOne,18->colorTwo,19->colorTwo,20->colorOne,21->colorTwo,22->colorOne}

8.7 (Figure 3.1) gadget (depth (q = 1) instance)



Graph Properties ::

--

Number of Vertices: ‘16’

Number of Edges: ‘21’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->2,2<->3,2<->6,3<->5,3<->10,4<->5,4<->7,4<->8,5<->9,6<->7,6<->14,7<->12,8<->9,8<->12,9<->13,10<->11,10<->15,12<->13,13<->15,14<->15,14<->16}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-0.8333,-0.8875},2->{-0.5,-0.5542},3->{-0.5,0.4458},4->{-0.3,-0.3542},5->{-0.3,0.4458},6->{0.,-0.5542},7->{0.,-0.3542},8->{0.,0.0458},9->{0.,0.4458},10->{0.,0.9458},11->{0.,1.2792},12->{0.3,-0.3542},13->{0.3,0.4458},14->{0.5,-0.5542},15->{0.5,0.4458},16->{0.8333,-0.8875}}

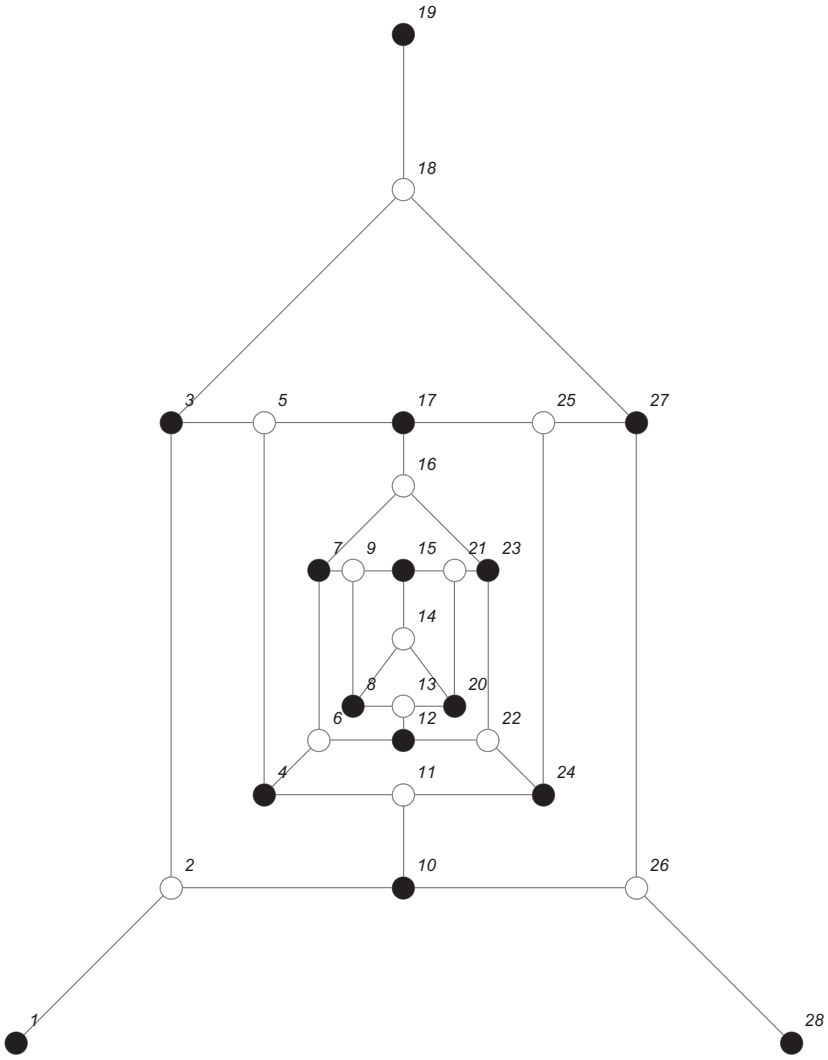
Edge list (SAGE 7.2):
[(0,1),(1,2),(1,5),(2,4),(2,9),(3,4),(3,6),(3,7),(4,8),(5,6),(5,13),(6,11),(7,8),(7,11),(8,12),(9,10),(9,14),(11,12),(12,14),(13,14),(13,15)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-0.8333,-0.8875],1:[-0.5,-0.5542],2:[-0.5,0.4458],3:[-0.3,-0.3542],4:[-0.3,0.4458],5:[0.,-0.5542],6:[0.,-0.3542],7:[0.,0.0458],8:[0.,0.4458],9:[0.,0.9458],10:[0.,1.2792],11:[0.3,-0.3542],12:[0.3,0.4458],13:[0.5,-0.5542],14:[0.5,0.4458],15:[0.8333,-0.8875]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorTwo,3->colorOne,4->colorOne,5->colorTwo,6->colorOne,7->colorTwo,8->colorTwo,9->colorOne,10->colorTwo,11->colorOne,12->colorOne,13->colorTwo,14->colorTwo,15->colorOne,16->colorOne}

8.8 (Figure 3.1) gadget (depth (q = 2) instance)



Graph Properties ::

--

Number of Vertices: ‘28’

Number of Edges: ‘39’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->2,2<->3,2<->10,3<->5,3<->18,4<->5,4<->6,4<->11,5<->17,6<->7,6<->12,7<->9,7<->16,8<->9,8<->13,8<->14,9<->15,10<->11,10<->26,11<->24,12<->13,12<->22,13<->20,14<->15,14<->20,15<->21,16<->17,16<->23,17<->25,18<->19,18<->27,20<->21,21<->23,22<->23,22<->24,24<->25,25<->27,26<->27,26<->28}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-0.8333,-0.876},2->{-0.5,-0.5426},3->{-0.5,0.4574},4->{-0.3,-0.3426},5->{-0.3,0.4574},6->{-0.1818,-0.2251},7->{-0.1818,0.1385},8->{-0.1091,-0.1524},9->{-0.1091,0.1385},10->{0.,-0.5426},11->{0.,-0.3426},12->{0.,-0.2251},13->{0.,-0.1524},14->{0.,-0.007},15->{0.,0.1385},16->{0.,0.3203},17->{0.,0.4574},18->{0.,0.9574},19->{0.,1.2907},20->{0.1091,-0.1524},21->{0.1091,0.1385},22->{0.1818,-0.2251},23->{0.1818,0.1385},24->{0.3,-0.3426},25->{0.3,0.4574},26->{0.5,-0.5426},27->{0.5,0.4574},28->{0.8333,-0.876}}

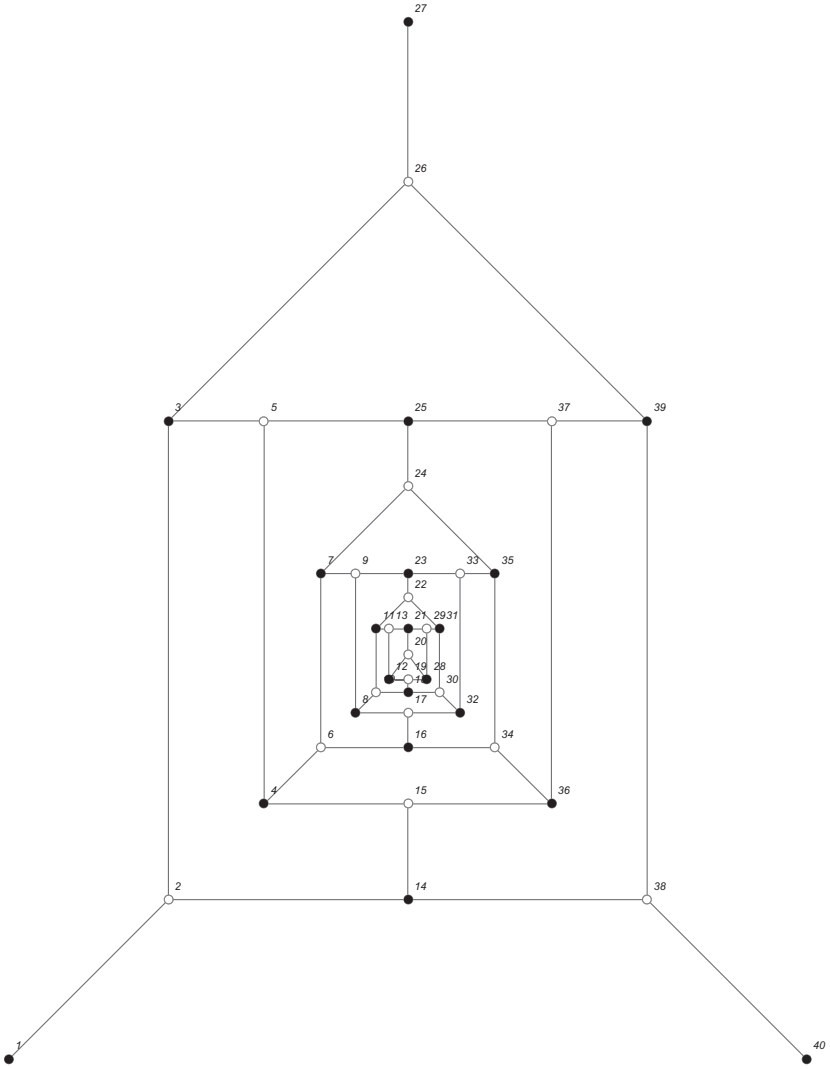
Edge list (SAGE 7.2):
[(0,1),(1,2),(1,9),(2,4),(2,17),(3,4),(3,5),(3,10),(4,16),(5,6),(5,11),(6,8),(6,15),(7,8),(7,12),(7,13),(8,14),(9,10),(9,25),(10,23),(11,12),(11,21),(12,19),(13,14),(13,19),(14,20),(15,16),(15,22),(16,24),(17,18),(17,26),(19,20),(20,22),(21,22),(21,23),(23,24),(24,26),(25,26),(25,27)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-0.8333,-0.876],1:[-0.5,-0.5426],2:[-0.5,0.4574],3:[-0.3,-0.3426],4:[-0.3,0.4574],5:[-0.1818,-0.2251],6:[-0.1818,0.1385],7:[-0.1091,-0.1524],8:[-0.1091,0.1385],9:[0.,-0.5426],10:[0.,-0.3426],11:[0.,-0.2251],12:[0.,-0.1524],13:[0.,-0.007],14:[0.,0.1385],15:[0.,0.3203],16:[0.,0.4574],17:[0.,0.9574],18:[0.,1.2907],19:[0.1091,-0.1524],20:[0.1091,0.1385],21:[0.1818,-0.2251],22:[0.1818,0.1385],23:[0.3,-0.3426],24:[0.3,0.4574],25:[0.5,-0.5426],26:[0.5,0.4574],27:[0.8333,-0.876]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorTwo,3->colorOne,4->colorOne,5->colorTwo,6->colorTwo,7->colorOne,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13->colorTwo,14->colorTwo,15->colorOne,16->colorTwo,17->colorOne,18->colorTwo,19->colorOne,20->colorOne,21->colorTwo,22->colorTwo,23->colorOne,24->colorOne,25->colorTwo,26->colorTwo,27->colorOne,28->colorOne}

8.9 (Figure 3.1) gadget (depth (q = 3) instance)



Graph Properties ::

--

Number of Vertices: ‘40’

Number of Edges: ‘57’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->2,2<->3,2<->14,3<->5,3<->26,4<->5,4<->6,4<->15,5<->25,6<->7,6<->16,7<->9,7<->24,8<->9,8<->10,8<->17,9<->23,10<->11,10<->18,11<->13,11<->22,12<->13,12<->19,12<->20,13<->21,14<->15,14<->38,15<->36,16<->17,16<->34,17<->32,18<->19,18<->30,19<->28,20<->21,20<->28,21<->29,22<->23,22<->31,23<->33,24<->25,24<->35,25<->37,26<->27,26<->39,28<->29,29<->31,30<->31,30<->32,32<->33,33<->35,34<->35,34<->36,36<->37,37<->39,38<->39,38<->40}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-0.8333,-0.866},2->{-0.5,-0.5326},3->{-0.5,0.4674},4->{-0.3,-0.3326},5->{-0.3,0.4674},6->{-0.1818,-0.2151},7->{-0.1818,0.1485},8->{-0.1091,-0.1424},9->{-0.1091,0.1485},10->{-0.0661,-0.0997},11->{-0.0661,0.0326},12->{-0.0397,-0.0732},13->{-0.0397,0.0326},14->{0.,-0.5326},15->{0.,-0.3326},16->{0.,-0.2151},17->{0.,-0.1424},18->{0.,-0.0997},19->{0.,-0.0732},20->{0.,-0.0203},21->{0.,0.0326},22->{0.,0.0987},23->{0.,0.1485},24->{0.,0.3303},25->{0.,0.4674},26->{0.,0.9674},27->{0.,1.3007},28->{0.0397,-0.0732},29->{0.0397,0.0326},30->{0.0661,-0.0997},31->{0.0661,0.0326},32->{0.1091,-0.1424},33->{0.1091,0.1485},34->{0.1818,-0.2151},35->{0.1818,0.1485},36->{0.3,-0.3326},37->{0.3,0.4674},38->{0.5,-0.5326},39->{0.5,0.4674},40->{0.8333,-0.866}}

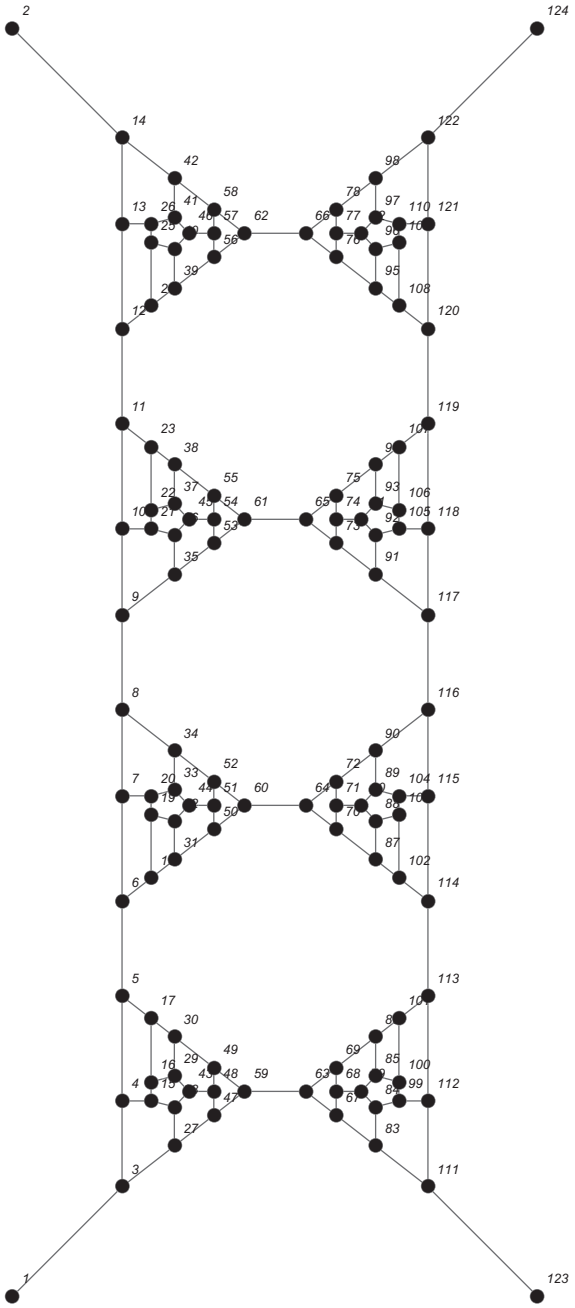
Edge list (SAGE 7.2):
[(0,1),(1,2),(1,13),(2,4),(2,25),(3,4),(3,5),(3,14),(4,24),(5,6),(5,15),(6,8),(6,23),(7,8),(7,9),(7,16),(8,22),(9,10),(9,17),(10,12),(10,21),(11,12),(11,18),(11,19),(12,20),(13,14),(13,37),(14,35),(15,16),(15,33),(16,31),(17,18),(17,29),(18,27),(19,20),(19,27),(20,28),(21,22),(21,30),(22,32),(23,24),(23,34),(24,36),(25,26),(25,38),(27,28),(28,30),(29,30),(29,31),(31,32),(32,34),(33,34),(33,35),(35,36),(36,38),(37,38),(37,39)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-0.8333,-0.866],1:[-0.5,-0.5326],2:[-0.5,0.4674],3:[-0.3,-0.3326],4:[-0.3,0.4674],5:[-0.1818,-0.2151],6:[-0.1818,0.1485],7:[-0.1091,-0.1424],8:[-0.1091,0.1485],9:[-0.0661,-0.0997],10:[-0.0661,0.0326],11:[-0.0397,-0.0732],12:[-0.0397,0.0326],13:[0.,-0.5326],14:[0.,-0.3326],15:[0.,-0.2151],16:[0.,-0.1424],17:[0.,-0.0997],18:[0.,-0.0732],19:[0.,-0.0203],20:[0.,0.0326],21:[0.,0.0987],22:[0.,0.1485],23:[0.,0.3303],24:[0.,0.4674],25:[0.,0.9674],26:[0.,1.3007],27:[0.0397,-0.0732],28:[0.0397,0.0326],29:[0.0661,-0.0997],30:[0.0661,0.0326],31:[0.1091,-0.1424],32:[0.1091,0.1485],33:[0.1818,-0.2151],34:[0.1818,0.1485],35:[0.3,-0.3326],36:[0.3,0.4674],37:[0.5,-0.5326],38:[0.5,0.4674],39:[0.8333,-0.866]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorTwo,3->colorOne,4->colorOne,5->colorTwo,6->colorTwo,7->colorOne,8->colorOne,9->colorTwo,10->colorTwo,11->colorOne,12->colorOne,13->colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorTwo,18->colorOne,19->colorTwo,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25->colorOne,26->colorTwo,27->colorOne,28->colorOne,29->colorTwo,30->colorTwo,31->colorOne,32->colorOne,33->colorTwo,34->colorTwo,35->colorOne,36->colorOne,37->colorTwo,38->colorTwo,39->colorOne,40->colorOne}

8.10 (Figure 4.4.a gadget) (equivalent to the (Figure 4.9.a) gadget and to the "Fig. 2.a" gadget from ref. [118])



Graph Properties ::

--

Number of Vertices: ‘124’

Number of Edges: ‘182’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘3’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘3’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘False’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘False’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘False’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):


```
{1<->3,2<->14,3<->4,3<->27,4<->5,4<->15,5<->6,5<->17,6<->7,6<->18,7<->8,7<->20,8<-
>9,8<->34,9<->10,9<->35,10<->11,10<->21,11<->12,11<->23,12<->13,12<->24,13<->14,13<-
>26,14<->42,15<->16,15<->28,16<->17,16<->29,17<->30,18<->19,18<->31,19<->20,19<-
>32,20<->33,21<->22,21<->36,22<->23,22<->37,23<->38,24<->25,24<->39,25<->26,25<-
>40,26<->41,27<->28,27<->47,28<->43,29<->30,29<->43,30<->49,31<->32,31<->50,32<-
>44,33<->34,33<->44,34<->52,35<->36,35<->53,36<->45,37<->38,37<->45,38<->55,39<-
>40,39<->56,40<->46,41<->42,41<->46,42<->58,43<->48,44<->51,45<->54,46<->57,47<-
>48,47<->59,48<->49,49<->59,50<->51,50<->60,51<->52,52<->60,53<->54,53<->61,54<-
>55,55<->61,56<->57,56<->62,57<->58,58<->62,59<->63,60<->64,61<->65,62<->66,63<-
>67,63<->69,64<->70,64<->72,65<->73,65<->75,66<->76,66<->78,67<->68,67<->83,68<-
>69,68<->79,69<->86,70<->71,70<->87,71<->72,71<->80,72<->90,73<->74,73<->91,74<-
>75,74<->81,75<->94,76<->77,76<->95,77<->78,77<->82,78<->98,79<->84,79<->85,80<-
>88,80<->89,81<->92,81<->93,82<->96,82<->97,83<->84,83<->111,84<->99,85<->86,85<-
>100,86<->101,87<->88,87<->102,88<->103,89<->90,89<->104,90<->116,91<->92,91<-
>117,92<->105,93<->94,93<->106,94<->107,95<->96,95<->108,96<->109,97<->98,97<-
>110,98<->122,99<->100,99<->112,100<->101,101<->113,102<->103,102<->114,103<-
>104,104<->115,105<->106,105<->118,106<->107,107<->119,108<->109,108<->120,109<-
>110,110<->121,111<->112,111<->123,112<->113,113<->114,114<->115,115<->116,116<-
>117,117<->118,118<->119,119<->120,120<->121,121<->122,122<->124}
```


-

Example embedding coordinates (Mathematica 10.4.1):


```
{1->{-6.,-14.5},2->{-6.,14.5},3->{-3.5,-12.},4->{-3.5,-10.0364},5->{-3.5,-7.6364},6-
>{-3.5,-5.4545},7->{-3.5,-3.0545},8->{-3.5,-1.0909},9->{-3.5,1.0909},10->{-
3.5,3.0545},11->{-3.5,5.4545},12->{-3.5,7.6364},13->{-3.5,10.0364},14->{-3.5,12.},15-
>{-2.8354,-10.0364},16->{-2.8354,-9.6},17->{-2.8354,-8.1542},18->{-2.8354,-4.9367},19-
>{-2.8354,-3.4909},20->{-2.8354,-3.0545},21->{-2.8354,3.0545},22->{-2.8354,3.4909},23-
>{-2.8354,4.9367},24->{-2.8354,8.1542},25->{-2.8354,9.6},26->{-2.8354,10.0364},27->{-
2.3028,-11.0671},28->{-2.3028,-10.1712},29->{-2.3028,-9.4652},30->{-2.3028,-
8.5693},31->{-2.3028,-4.5217},32->{-2.3028,-3.6258},33->{-2.3028,-2.9197},34->{-
2.3028,-2.0238},35->{-2.3028,2.0238},36->{-2.3028,2.9197},37->{-2.3028,3.6258},38->{-
2.3028,4.5217},39->{-2.3028,8.5693},40->{-2.3028,9.4652},41->{-2.3028,10.1712},42->{-
2.3028,11.0671},43->{-1.9736,-9.8182},44->{-1.9736,-3.2727},45->{-1.9736,3.2727},46-
>{-1.9736,9.8182},47->{-1.4,-10.3636},48->{-1.4,-9.8182},49->{-1.4,-9.2727},50->{-
1.4,-3.8182},51->{-1.4,-3.2727},52->{-1.4,-2.7273},53->{-1.4,2.7273},54->{-
1.4,3.2727},55->{-1.4,3.8182},56->{-1.4,9.2727},57->{-1.4,9.8182},58->{-
1.4,10.3636},59->{-0.7,-9.8182},60->{-0.7,-3.2727},61->{-0.7,3.2727},62->{-
0.7,9.8182},63->{0.7,-9.8182},64->{0.7,-3.2727},65->{0.7,3.2727},66->{0.7,9.8182},67-
>{1.4,-10.3636},68->{1.4,-9.8182},69->{1.4,-9.2727},70->{1.4,-3.8182},71->{1.4,-
3.2727},72->{1.4,-2.7273},73->{1.4,2.7273},74->{1.4,3.2727},75->{1.4,3.8182},76-
>{1.4,9.2727},77->{1.4,9.8182},78->{1.4,10.3636},79->{1.9736,-9.8182},80->{1.9736,-
3.2727},81->{1.9736,3.2727},82->{1.9736,9.8182},83->{2.3028,-11.0671},84->{2.3028,-
10.1712},85->{2.3028,-9.4652},86->{2.3028,-8.5693},87->{2.3028,-4.5217},88->{2.3028,-
3.6258},89->{2.3028,-2.9197},90->{2.3028,-2.0238},91->{2.3028,2.0238},92-
>{2.3028,2.9197},93->{2.3028,3.6258},94->{2.3028,4.5217},95->{2.3028,8.5693},96-
>{2.3028,9.4652},97->{2.3028,10.1712},98->{2.3028,11.0671},99->{2.8354,-10.0364},100-
>{2.8354,-9.6},101->{2.8354,-8.1542},102->{2.8354,-4.9367},103->{2.8354,-3.4909},104-
>{2.8354,-3.0545},105->{2.8354,3.0545},106->{2.8354,3.4909},107->{2.8354,4.9367},108-
>{2.8354,8.1542},109->{2.8354,9.6},110->{2.8354,10.0364},111->{3.5,-12.},112->{3.5,-
10.0364},113->{3.5,-7.6364},114->{3.5,-5.4545},115->{3.5,-3.0545},116->{3.5,-
1.0909},117->{3.5,1.0909},118->{3.5,3.0545},119->{3.5,5.4545},120->{3.5,7.6364},121-
>{3.5,10.0364},122->{3.5,12.},123->{6.,-14.5},124->{6.,14.5}}
```

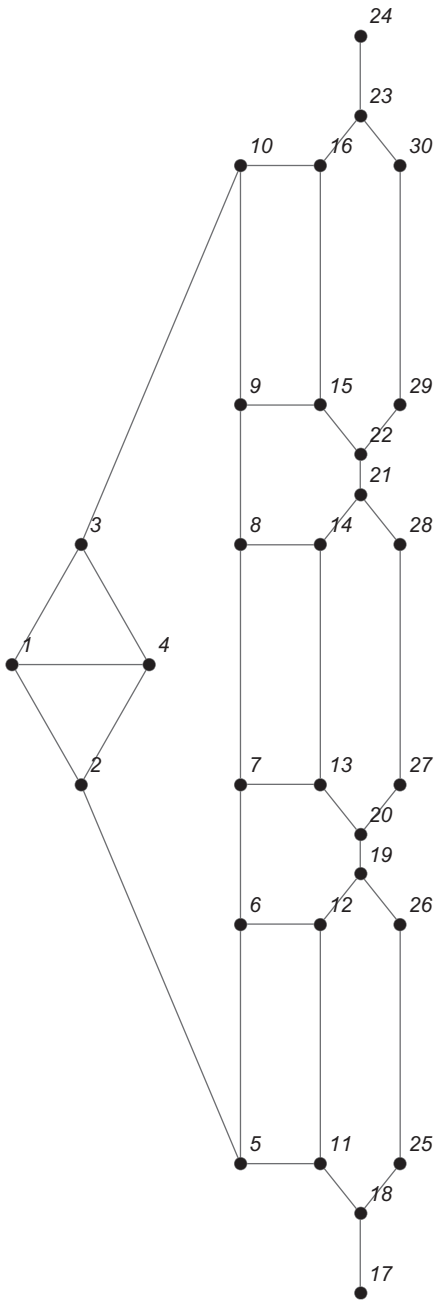
Edge list (SAGE 7.2):


```
[(0,2),(1,13),(2,3),(2,26),(3,4),(3,14),(4,5),(4,16),(5,6),(5,17),(6,7),(6,19),(7,8),(
7,33),(8,9),(8,34),(9,10),(9,20),(10,11),(10,22),(11,12),(11,23),(12,13),(12,25),(13,4
1),(14,15),(14,27),(15,16),(15,28),(16,29),(17,18),(17,30),(18,19),(18,31),(19,32),(20
,21),(20,35),(21,22),(21,36),(22,37),(23,24),(23,38),(24,25),(24,39),(25,40),(26,27),(
26,46),(27,42),(28,29),(28,42),(29,48),(30,31),(30,49),(31,43),(32,33),(32,43),(33,51)
,(34,35),(34,52),(35,44),(36,37),(36,44),(37,54),(38,39),(38,55),(39,45),(40,41),(40,4
5),(41,57),(42,47),(43,50),(44,53),(45,56),(46,47),(46,58),(47,48),(48,58),(49,50),(49
,59),(50,51),(51,59),(52,53),(52,60),(53,54),(54,60),(55,56),(55,61),(56,57),(57,61),(
58,62),(59,63),(60,64),(61,65),(62,66),(62,68),(63,69),(63,71),(64,72),(64,74),(65,75)
,(65,77),(66,67),(66,82),(67,68),(67,78),(68,85),(69,70),(69,86),(70,71),(70,79),(71,8
9),(72,73),(72,90),(73,74),(73,80),(74,93),(75,76),(75,94),(76,77),(76,81),(77,97),(78
,83),(78,84),(79,87),(79,88),(80,91),(80,92),(81,95),(81,96),(82,83),(82,110),(83,98),
(84,85),(84,99),(85,100),(86,87),(86,101),(87,102),(88,89),(88,103),(89,115),(90,91),(
90,116),(91,104),(92,93),(92,105),(93,106),(94,95),(94,107),(95,108),(96,97),(96,109),
(97,121),(98,99),(98,111),(99,100),(100,112),(101,102),(101,113),(102,103),(103,114),(
104,105),(104,117),(105,106),(106,118),(107,108),(107,119),(108,109),(109,120),(110,11
1),(110,122),(111,112),(112,113),(113,114),(114,115),(115,116),(116,117),(117,118),(11
8,119),(119,120),(120,121),(121,123)]
```


Example embedding coordinates (SAGE 7.2):


```
{0:[-6.,-14.5],1:[-6.,14.5],2:[-3.5,-12.],3:[-3.5,-10.0364],4:[-3.5,-7.6364],5:[-3.5,-
5.4545],6:[-3.5,-3.0545],7:[-3.5,-1.0909],8:[-3.5,1.0909],9:[-3.5,3.0545],10:[-
3.5,5.4545],11:[-3.5,7.6364],12:[-3.5,10.0364],13:[-3.5,12.],14:[-2.8354,-
10.0364],15:[-2.8354,-9.6],16:[-2.8354,-8.1542],17:[-2.8354,-4.9367],18:[-2.8354,-
3.4909],19:[-2.8354,-3.0545],20:[-2.8354,3.0545],21:[-2.8354,3.4909],22:[-
2.8354,4.9367],23:[-2.8354,8.1542],24:[-2.8354,9.6],25:[-2.8354,10.0364],26:[-2.3028,-
11.0671],27:[-2.3028,-10.1712],28:[-2.3028,-9.4652],29:[-2.3028,-8.5693],30:[-2.3028,-
4.5217],31:[-2.3028,-3.6258],32:[-2.3028,-2.9197],33:[-2.3028,-2.0238],34:[-
2.3028,2.0238],35:[-2.3028,2.9197],36:[-2.3028,3.6258],37:[-2.3028,4.5217],38:[-
2.3028,8.5693],39:[-2.3028,9.4652],40:[-2.3028,10.1712],41:[-2.3028,11.0671],42:[-
1.9736,-9.8182],43:[-1.9736,-3.2727],44:[-1.9736,3.2727],45:[-1.9736,9.8182],46:[-
1.4,-10.3636],47:[-1.4,-9.8182],48:[-1.4,-9.2727],49:[-1.4,-3.8182],50:[-1.4,-
3.2727],51:[-1.4,-2.7273],52:[-1.4,2.7273],53:[-1.4,3.2727],54:[-1.4,3.8182],55:[-
1.4,9.2727],56:[-1.4,9.8182],57:[-1.4,10.3636],58:[-0.7,-9.8182],59:[-0.7,-
3.2727],60:[-0.7,3.2727],61:[-0.7,9.8182],62:[0.7,-9.8182],63:[0.7,-
3.2727],64:[0.7,3.2727],65:[0.7,9.8182],66:[1.4,-10.3636],67:[1.4,-9.8182],68:[1.4,-
9.2727],69:[1.4,-3.8182],70:[1.4,-3.2727],71:[1.4,-
2.7273],72:[1.4,2.7273],73:[1.4,3.2727],74:[1.4,3.8182],75:[1.4,9.2727],76:[1.4,9.8182
],77:[1.4,10.3636],78:[1.9736,-9.8182],79:[1.9736,-
3.2727],80:[1.9736,3.2727],81:[1.9736,9.8182],82:[2.3028,-11.0671],83:[2.3028,-
10.1712],84:[2.3028,-9.4652],85:[2.3028,-8.5693],86:[2.3028,-4.5217],87:[2.3028,-
3.6258],88:[2.3028,-2.9197],89:[2.3028,-
2.0238],90:[2.3028,2.0238],91:[2.3028,2.9197],92:[2.3028,3.6258],93:[2.3028,4.5217],94
:[2.3028,8.5693],95:[2.3028,9.4652],96:[2.3028,10.1712],97:[2.3028,11.0671],98:[2.8354
,-10.0364],99:[2.8354,-9.6],100:[2.8354,-8.1542],101:[2.8354,-4.9367],102:[2.8354,-
3.4909],103:[2.8354,-
3.0545],104:[2.8354,3.0545],105:[2.8354,3.4909],106:[2.8354,4.9367],107:[2.8354,8.1542
],108:[2.8354,9.6],109:[2.8354,10.0364],110:[3.5,-12.],111:[3.5,-10.0364],112:[3.5,-
7.6364],113:[3.5,-5.4545],114:[3.5,-3.0545],115:[3.5,-
1.0909],116:[3.5,1.0909],117:[3.5,3.0545],118:[3.5,5.4545],119:[3.5,7.6364],120:[3.5,1
0.0364],121:[3.5,12.],122:[6.,-14.5],123:[6.,14.5]}
```

8.11 (Figure 4.4.b) gadget (equivalent to the (Figure 4.10.a) gadget and to the "Fig. 3.e" gadget from ref. [118])



Graph Properties ::

--

Number of Vertices: ‘30’

Number of Edges: ‘40’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘3’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘3’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘False’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘False’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘False’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘3’

Output of SAGE 7.2’s ‘girth()’ function: ‘3’

Output for the ‘igraph’ R package ‘girth()’ function: ‘3’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: '0'

Edge list (Mathematica 10.4.1):

{1<->2,1<->3,1<->4,2<->4,2<->5,3<->4,3<->10,5<->6,5<->11,6<->7,6<->12,7<->8,7<->13,8<->9,8<->14,9<->10,9<->15,10<->16,11<->12,11<->18,12<->19,13<->14,13<->20,14<->21,15<->16,15<->22,16<->23,17<->18,18<->25,19<->20,19<->26,20<->27,21<->22,21<->28,22<->29,23<->24,23<->30,25<->26,27<->28,29<->30}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-14.3619,0.},2->{-10.9333,-6.},3->{-10.9333,6.},4->{-7.5047,0.},5->{-2.9333,-25.},6->{-2.9333,-13.},7->{-2.9333,-6.},8->{-2.9333,6.},9->{-2.9333,13.},10->{-2.9333,25.},11->{1.0667,-25.},12->{1.0667,-13.},13->{1.0667,-6.},14->{1.0667,6.},15->{1.0667,13.},16->{1.0667,25.},17->{3.0667,-31.5},18->{3.0667,-27.5},19->{3.0667,-10.5},20->{3.0667,-8.5},21->{3.0667,8.5},22->{3.0667,10.5},23->{3.0667,27.5},24->{3.0667,31.5},25->{5.0667,-25.},26->{5.0667,-13.},27->{5.0667,-6.},28->{5.0667,6.},29->{5.0667,13.},30->{5.0667,25.}}

Edge list (SAGE 7.2):

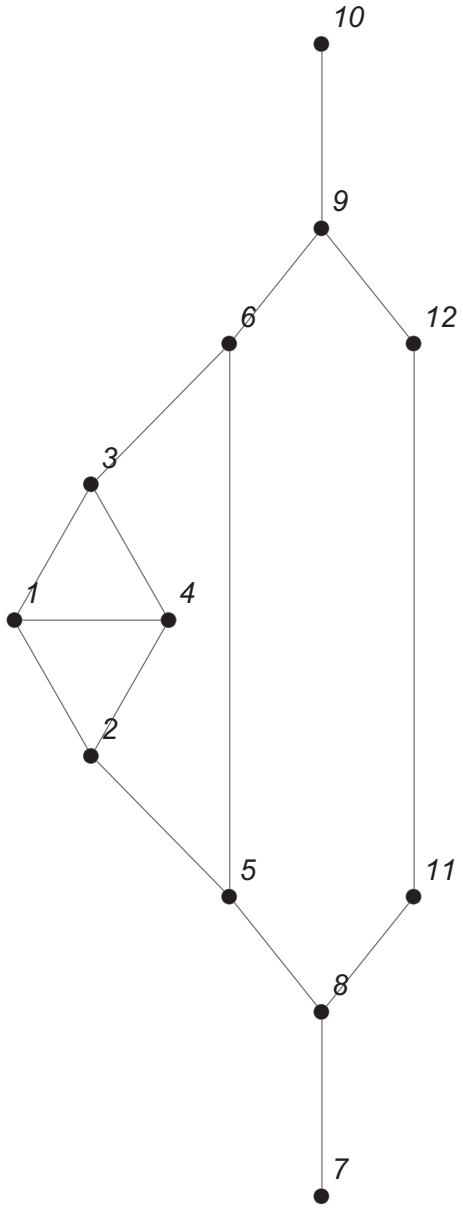
[(0,1),(0,2),(0,3),(1,3),(1,4),(2,3),(2,9),(4,5),(4,10),(5,6),(5,11),(6,7),(6,12),(7,8),(7,13),(8,9),(8,14),(9,15),(10,11),(10,17),(11,18),(12,13),(12,19),(13,20),(14,15),(14,21),(15,22),(16,17),(17,24),(18,19),(18,25),(19,26),(20,21),(20,27),(21,28),(22,23),(22,29),(24,25),(26,27),(28,29)]

-

Example embedding coordinates (SAGE 7.2):

{0:[-14.3619,0.],1:[-10.9333,-6.],2:[-10.9333,6.],3:[-7.5047,0.],4:[-2.9333,-25.],5:[-2.9333,-13.],6:[-2.9333,-6.],7:[-2.9333,6.],8:[-2.9333,13.],9:[-2.9333,25.],10:[1.0667,-25.],11:[1.0667,-13.],12:[1.0667,-6.],13:[1.0667,6.],14:[1.0667,13.],15:[1.0667,25.],16:[3.0667,-31.5],17:[3.0667,-27.5],18:[3.0667,-10.5],19:[3.0667,-8.5],20:[3.0667,8.5],21:[3.0667,10.5],22:[3.0667,27.5],23:[3.0667,31.5],24:[5.0667,-25.],25:[5.0667,-13.],26:[5.0667,-6.],27:[5.0667,6.],28:[5.0667,13.],29:[5.0667,25.]}

8.12 (Figure 4.4.c) gadget (equivalent to the (Figure 4.11.a) gadget and to the "single-literal clause" gadget shown in "Fig. 5" of ref. [118])



Graph Properties ::

--

Number of Vertices: ‘12’

Number of Edges: ‘15’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘3’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘3’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘False’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘False’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘False’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘3’

Output of SAGE 7.2’s ‘girth()’ function: ‘3’

Output for the ‘igraph’ R package ‘girth()’ function: ‘3’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: '0'

Edge list (Mathematica 10.4.1):
{1<->2,1<->3,1<->4,2<->4,2<->5,3<->4,3<->6,5<->6,5<->8,6<->9,7<->8,8<->11,9<->10,9<->12,11<->12}

-

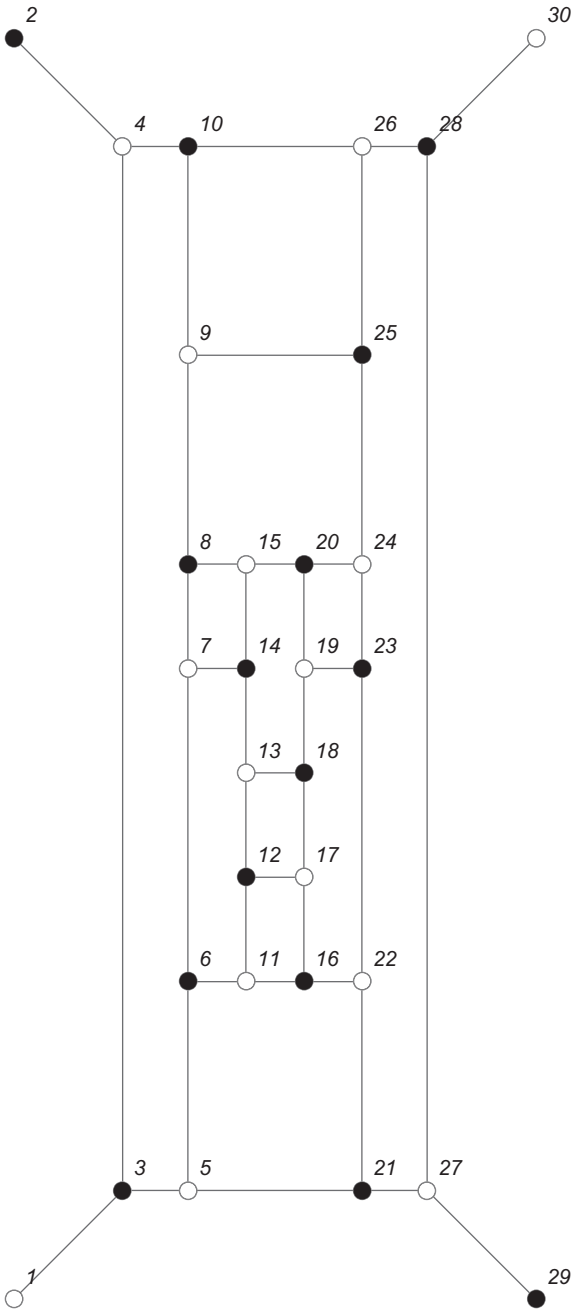
Example embedding coordinates (Mathematica 10.4.1):
{1->{-5.0133,0.},2->{-3.3333,-2.94},3->{-3.3333,2.94},4->{-1.6533,0.},5->{-0.3333,-6.},6->{-0.3333,6.},7->{1.6667,-12.5},8->{1.6667,-8.5},9->{1.6667,8.5},10->{1.6667,12.5},11->{3.6667,-6.},12->{3.6667,6.}}

Edge list (SAGE 7.2):
[(0,1),(0,2),(0,3),(1,3),(1,4),(2,3),(2,5),(4,5),(4,7),(5,8),(6,7),(7,10),(8,9),(8,11),(10,11)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-5.0133,0.],1:[-3.3333,-2.94],2:[-3.3333,2.94],3:[-1.6533,0.],4:[-0.3333,-6.],5:[-0.3333,6.],6:[1.6667,-12.5],7:[1.6667,-8.5],8:[1.6667,8.5],9:[1.6667,12.5],10:[3.6667,-6.],11:[3.6667,6.]}

8.13 (Figure 4.5.a) gadget (specifying (z = 1) blocks)



Graph Properties ::

--

Number of Vertices: ‘30’

Number of Edges: ‘41’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->3,2<->4,3<->4,3<->5,4<->10,5<->6,5<->21,6<->7,6<->11,7<->8,7<->14,8<->9,8<->15,9<->10,9<->25,10<->26,11<->12,11<->16,12<->13,12<->17,13<->14,13<->18,14<->15,15<->20,16<->17,16<->22,17<->18,18<->19,19<->20,19<->23,20<->24,21<->22,21<->27,22<->23,23<->24,24<->25,25<->26,26<->28,27<->28,27<->29,28<->30}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-6.,-13.86},2->{-6.,15.1399},3->{-3.5,-11.36},4->{-3.5,12.6399},5->{-2.,-11.36},6->{-2.,-6.5601},7->{-2.,0.64},8->{-2.,3.0403},9->{-2.,7.8402},10->{-2.,12.6399},11->{-0.6667,-6.5601},12->{-0.6667,-4.1602},13->{-0.6667,-1.7599},14->{-0.6667,0.64},15->{-0.6667,3.0403},16->{0.6667,-6.5601},17->{0.6667,-4.1602},18->{0.6667,-1.7599},19->{0.6667,0.64},20->{0.6667,3.0403},21->{2.,-11.36},22->{2.,-6.5601},23->{2.,0.64},24->{2.,3.0403},25->{2.,7.8402},26->{2.,12.6399},27->{3.5,-11.36},28->{3.5,12.6399},29->{6.,-13.86},30->{6.,15.1399}}

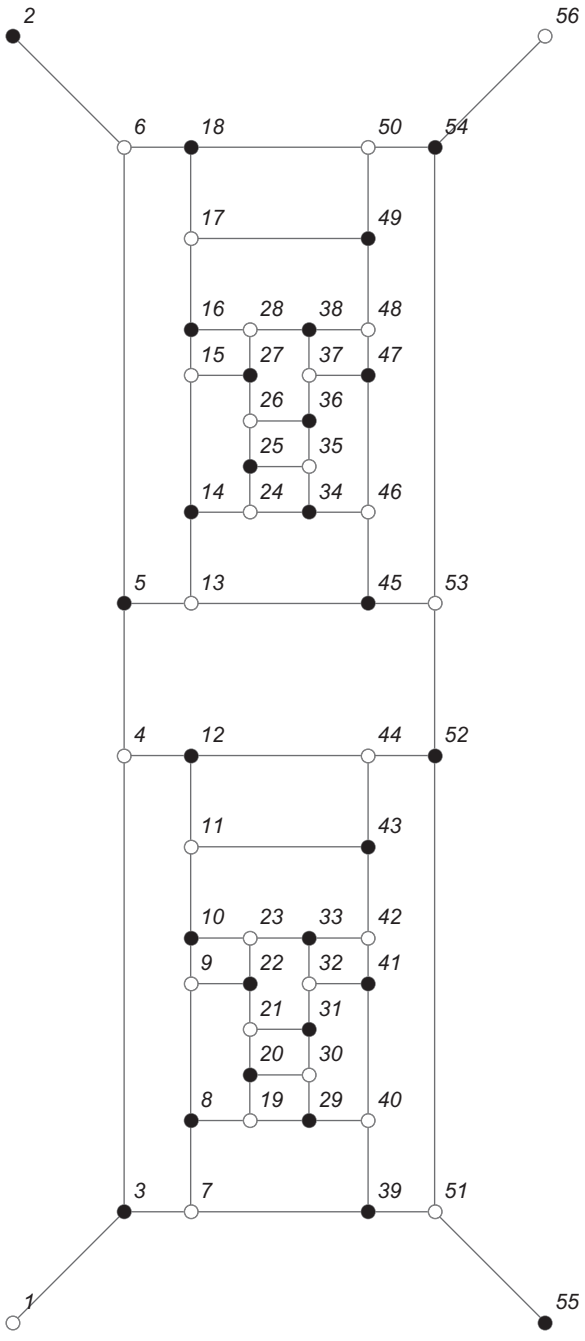
Edge list (SAGE 7.2):
[(0,2),(1,3),(2,3),(2,4),(3,9),(4,5),(4,20),(5,6),(5,10),(6,7),(6,13),(7,8),(7,14),(8,9),(8,24),(9,25),(10,11),(10,15),(11,12),(11,16),(12,13),(12,17),(13,14),(14,19),(15,16),(15,21),(16,17),(17,18),(18,19),(18,22),(19,23),(20,21),(20,26),(21,22),(22,23),(23,24),(24,25),(25,27),(26,27),(26,28),(27,29)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-6.,-13.86],1:[-6.,15.1399],2:[-3.5,-11.36],3:[-3.5,12.6399],4:[-2.,-11.36],5:[-2.,-6.5601],6:[-2.,0.64],7:[-2.,3.0403],8:[-2.,7.8402],9:[-2.,12.6399],10:[-0.6667,-6.5601],11:[-0.6667,-4.1602],12:[-0.6667,-1.7599],13:[-0.6667,0.64],14:[-0.6667,3.0403],15:[0.6667,-6.5601],16:[0.6667,-4.1602],17:[0.6667,-1.7599],18:[0.6667,0.64],19:[0.6667,3.0403],20:[2.,-11.36],21:[2.,-6.5601],22:[2.,0.64],23:[2.,3.0403],24:[2.,7.8402],25:[2.,12.6399],26:[3.5,-11.36],27:[3.5,12.6399],28:[6.,-13.86],29:[6.,15.1399]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorTwo,2->colorOne,3->colorOne,4->colorTwo,5->colorTwo,6->colorOne,7->colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13->colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorTwo,18->colorOne,19->colorTwo,20->colorOne,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25->colorOne,26->colorTwo,27->colorTwo,28->colorOne,29->colorOne,30->colorTwo}

8.14 (Figure 4.5.a) gadget (specifying (z = 2) blocks)



Graph Properties ::

--

Number of Vertices: ‘56’

Number of Edges: ‘80’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->3,2<->6,3<->4,3<->7,4<->5,4<->12,5<->6,5<->13,6<->18,7<->8,7<->39,8<->9,8<->19,9<->10,9<->22,10<->11,10<->23,11<->12,11<->43,12<->44,13<->14,13<->45,14<->15,14<->24,15<->16,15<->27,16<->17,16<->28,17<->18,17<->49,18<->50,19<->20,19<->29,20<->21,20<->30,21<->22,21<->31,22<->23,23<->33,24<->25,24<->34,25<->26,25<->35,26<->27,26<->36,27<->28,28<->38,29<->30,29<->40,30<->31,31<->32,32<->33,32<->41,33<->42,34<->35,34<->46,35<->36,36<->37,37<->38,37<->47,38<->48,39<->40,39<->51,40<->41,41<->42,42<->43,43<->44,44<->52,45<->46,45<->53,46<->47,47<->48,48<->49,49<->50,50<->54,51<->52,51<->55,52<->53,53<->54,54<->56}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-6.,-14.2061},2->{-6.,14.7939},3->{-3.5,-11.7061},4->{-3.5,-1.4205},5->{-3.5,2.0082},6->{-3.5,12.2939},7->{-2.,-11.7061},8->{-2.,-9.649},9->{-2.,-6.5633},10->{-2.,-5.5346},11->{-2.,-3.4775},12->{-2.,-1.4205},13->{-2.,2.0082},14->{-2.,4.0654},15->{-2.,7.1511},16->{-2.,8.1794},17->{-2.,10.2367},18->{-2.,12.2939},19->{-0.6667,-9.649},20->{-0.6667,-8.6205},21->{-0.6667,-7.5918},22->{-0.6667,-6.5633},23->{-0.6667,-5.5346},24->{-0.6667,4.0654},25->{-0.6667,5.0939},26->{-0.6667,6.1224},27->{-0.6667,7.1511},28->{-0.6667,8.1794},29->{0.6667,-9.649},30->{0.6667,-8.6205},31->{0.6667,-7.5918},32->{0.6667,-6.5633},33->{0.6667,-5.5346},34->{0.6667,4.0654},35->{0.6667,5.0939},36->{0.6667,6.1224},37->{0.6667,7.1511},38->{0.6667,8.1794},39->{2.,-11.7061},40->{2.,-9.649},41->{2.,-6.5633},42->{2.,-5.5346},43->{2.,-3.4775},44->{2.,-1.4205},45->{2.,2.0082},46->{2.,4.0654},47->{2.,7.1511},48->{2.,8.1794},49->{2.,10.2367},50->{2.,12.2939},51->{3.5,-11.7061},52->{3.5,-1.4205},53->{3.5,2.0082},54->{3.5,12.2939},55->{6.,-14.2061},56->{6.,14.7939}}

Edge list (SAGE 7.2):

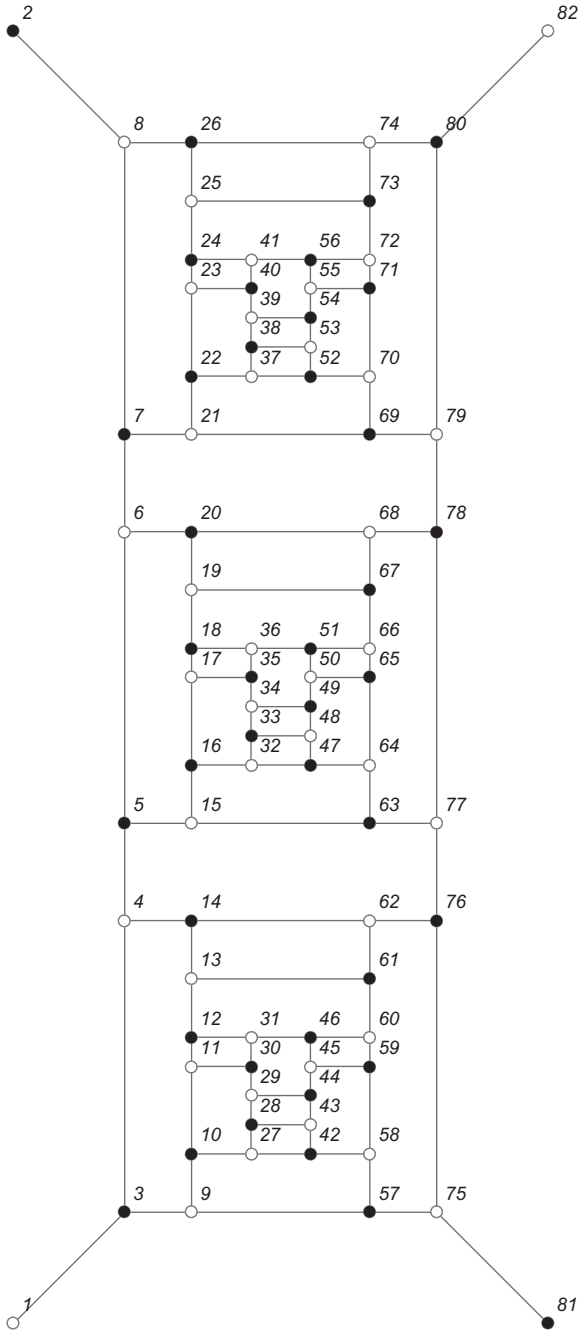
[(0,2),(1,5),(2,3),(2,6),(3,4),(3,11),(4,5),(4,12),(5,17),(6,7),(6,38),(7,8),(7,18),(8,9),(8,21),(9,10),(9,22),(10,11),(10,42),(11,43),(12,13),(12,44),(13,14),(13,23),(14,15),(14,26),(15,16),(15,27),(16,17),(16,48),(17,49),(18,19),(18,28),(19,20),(19,29),(20,21),(20,30),(21,22),(22,32),(23,24),(23,33),(24,25),(24,34),(25,26),(25,35),(26,27),(27,37),(28,29),(28,39),(29,30),(30,31),(31,32),(31,40),(32,41),(33,34),(33,45),(34,35),(35,36),(36,37),(36,46),(37,47),(38,39),(38,50),(39,40),(40,41),(41,42),(42,43),(43,51),(44,45),(44,52),(45,46),(46,47),(47,48),(48,49),(49,53),(50,51),(50,54),(51,52),(52,53),(53,55)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-6.,-14.2061],1:[-6.,14.7939],2:[-3.5,-11.7061],3:[-3.5,-1.4205],4:[-
3.5,2.0082],5:[-3.5,12.2939],6:[-2.,-11.7061],7:[-2.,-9.649],8:[-2.,-6.5633],9:[-2.,-
5.5346],10:[-2.,-3.4775],11:[-2.,-1.4205],12:[-2.,2.0082],13:[-2.,4.0654],14:[-
2.,7.1511],15:[-2.,8.1794],16:[-2.,10.2367],17:[-2.,12.2939],18:[-0.6667,-9.649],19:[-
0.6667,-8.6205],20:[-0.6667,-7.5918],21:[-0.6667,-6.5633],22:[-0.6667,-5.5346],23:[-
0.6667,4.0654],24:[-0.6667,5.0939],25:[-0.6667,6.1224],26:[-0.6667,7.1511],27:[-
0.6667,8.1794],28:[0.6667,-9.649],29:[0.6667,-8.6205],30:[0.6667,-7.5918],31:[0.6667,-
6.5633],32:[0.6667,-
5.5346],33:[0.6667,4.0654],34:[0.6667,5.0939],35:[0.6667,6.1224],36:[0.6667,7.1511],37
:[0.6667,8.1794],38:[2.,-11.7061],39:[2.,-9.649],40:[2.,-6.5633],41:[2.,-
5.5346],42:[2.,-3.4775],43:[2.,-
1.4205],44:[2.,2.0082],45:[2.,4.0654],46:[2.,7.1511],47:[2.,8.1794],48:[2.,10.2367],49
:[2.,12.2939],50:[3.5,-11.7061],51:[3.5,-
1.4205],52:[3.5,2.0082],53:[3.5,12.2939],54:[6.,-14.2061],55:[6.,14.7939]}

Canonical vertex ($k = 2$)-coloring ::
{1->colorTwo,2->colorOne,3->colorOne,4->colorTwo,5->colorOne,6->colorTwo,7-
>colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13-
>colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorTwo,18->colorOne,19-
>colorTwo,20->colorOne,21->colorTwo,22->colorOne,23->colorTwo,24->colorTwo,25-
>colorOne,26->colorTwo,27->colorOne,28->colorTwo,29->colorOne,30->colorTwo,31-
>colorOne,32->colorTwo,33->colorOne,34->colorOne,35->colorTwo,36->colorOne,37-
>colorTwo,38->colorOne,39->colorOne,40->colorTwo,41->colorOne,42->colorTwo,43-
>colorOne,44->colorTwo,45->colorOne,46->colorTwo,47->colorOne,48->colorTwo,49-
>colorOne,50->colorTwo,51->colorTwo,52->colorOne,53->colorTwo,54->colorOne,55-
>colorOne,56->colorTwo}

8.15 (Figure 4.5.a) gadget (specifying (z = 3) blocks)



Graph Properties ::

--

Number of Vertices: ‘82’

Number of Edges: ‘119’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->3,2<->8,3<->4,3<->9,4<->5,4<->14,5<->6,5<->15,6<->7,6<->20,7<->8,7<->21,8<->26,9<->10,9<->57,10<->11,10<->27,11<->12,11<->30,12<->13,12<->31,13<->14,13<->61,14<->62,15<->16,15<->63,16<->17,16<->32,17<->18,17<->35,18<->19,18<->36,19<->20,19<->67,20<->68,21<->22,21<->69,22<->23,22<->37,23<->24,23<->40,24<->25,24<->41,25<->26,25<->73,26<->74,27<->28,27<->42,28<->29,28<->43,29<->30,29<->44,30<->31,31<->46,32<->33,32<->47,33<->34,33<->48,34<->35,34<->49,35<->36,36<->51,37<->38,37<->52,38<->39,38<->53,39<->40,39<->54,40<->41,41<->56,42<->43,42<->58,43<->44,44<->45,45<->46,45<->59,46<->60,47<->48,47<->64,48<->49,49<->50,50<->51,50<->65,51<->66,52<->53,52<->70,53<->54,54<->55,55<->56,55<->71,56<->72,57<->58,57<->75,58<->59,59<->60,60<->61,61<->62,62<->76,63<->64,63<->77,64<->65,65<->66,66<->67,67<->68,68<->78,69<->70,69<->79,70<->71,71<->72,72<->73,73<->74,74<->80,75<->76,75<->81,76<->77,77<->78,78<->79,79<->80,80<->82}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-6.,-14.3084},2->{-6.,14.6916},3->{-3.5,-11.8084},4->{-3.5,-5.263},5->{-3.5,-3.0811},6->{-3.5,3.4643},7->{-3.5,5.6461},8->{-3.5,12.1916},9->{-2.,-11.8084},10->{-2.,-10.4993},11->{-2.,-8.5357},12->{-2.,-7.8811},13->{-2.,-6.572},14->{-2.,-5.263},15->{-2.,-3.0811},16->{-2.,-1.772},17->{-2.,0.1916},18->{-2.,0.846},19->{-2.,2.1552},20->{-2.,3.4643},21->{-2.,5.6461},22->{-2.,6.9551},23->{-2.,8.9189},24->{-2.,9.5734},25->{-2.,10.8825},26->{-2.,12.1916},27->{-0.6667,-10.4993},28->{-0.6667,-9.8448},29->{-0.6667,-9.1902},30->{-0.6667,-8.5357},31->{-0.6667,-7.8811},32->{-0.6667,-1.772},33->{-0.6667,-1.1175},34->{-0.6667,-0.463},35->{-0.6667,0.1916},36->{-0.6667,0.846},37->{-0.6667,6.9551},38->{-0.6667,7.6098},39->{-0.6667,8.2642},40->{-0.6667,8.9189},41->{-0.6667,9.5734},42->{0.6667,-10.4993},43->{0.6667,-9.8448},44->{0.6667,-9.1902},45->{0.6667,-8.5357},46->{0.6667,-7.8811},47->{0.6667,-1.772},48->{0.6667,-1.1175},49->{0.6667,-0.463},50->{0.6667,0.1916},51->{0.6667,0.846},52->{0.6667,6.9551},53->{0.6667,7.6098},54->{0.6667,8.2642},55->{0.6667,8.9189},56->{0.6667,9.5734},57->{2.,-11.8084},58->{2.,-10.4993},59->{2.,-8.5357},60->{2.,-7.8811},61->{2.,-6.572},62->{2.,-5.263},63->{2.,-3.0811},64->{2.,-1.772},65->{2.,0.1916},66->{2.,0.846},67->{2.,2.1552},68->{2.,3.4643},69->{2.,5.6461},70->{2.,6.9551},71->{2.,8.9189},72->{2.,9.5734},73->{2.,10.8825},74->{2.,12.1916},75->{3.5,-11.8084},76->{3.5,-5.263},77->{3.5,-3.0811},78->{3.5,3.4643},79->{3.5,5.6461},80->{3.5,12.1916},81->{6.,-14.3084},82->{6.,14.6916}}

Edge list (SAGE 7.2):

[(0,2),(1,7),(2,3),(2,8),(3,4),(3,13),(4,5),(4,14),(5,6),(5,19),(6,7),(6,20),(7,25),(8,9),(8,56),(9,10),(9,26),(10,11),(10,29),(11,12),(11,30),(12,13),(12,60),(13,61),(14,15),(14,62),(15,16),(15,31),(16,17),(16,34),(17,18),(17,35),(18,19),(18,66),(19,67),(20,21),(20,68),(21,22),(21,36),(22,23),(22,39),(23,24),(23,40),(24,25),(24,72),(25,73),(26,27),(26,41),(27,28),(27,42),(28,29),(28,43),(29,30),(30,45),(31,32),(31,46),(32,33),(32,47),(33,34),(33,48),(34,35),(35,50),(36,37),(36,51),(37,38),(37,52),(38,39),(38,53),(39,40),(40,55),(41,42),(41,57),(42,43),(43,44),(44,45),(44,58),(45,59),(46,47),(46,63),(47,48),(48,49),(49,50),(49,64),(50,65),(51,52),(51,69),(52,53),(53,54),(54,55),(54,70),(55,71),(56,57),(56,74),(57,58),(58,59),(59,60),(60,61),(61,75),(62,63),(62,76),(63,64),(64,65),(65,66),(66,67),(67,77),(68,69),(68,78),(69,70),(70,71),(71,72),(72,73),(73,79),(74,75),(74,80),(75,76),(76,77),(77,78),(78,79),(79,81)]

-

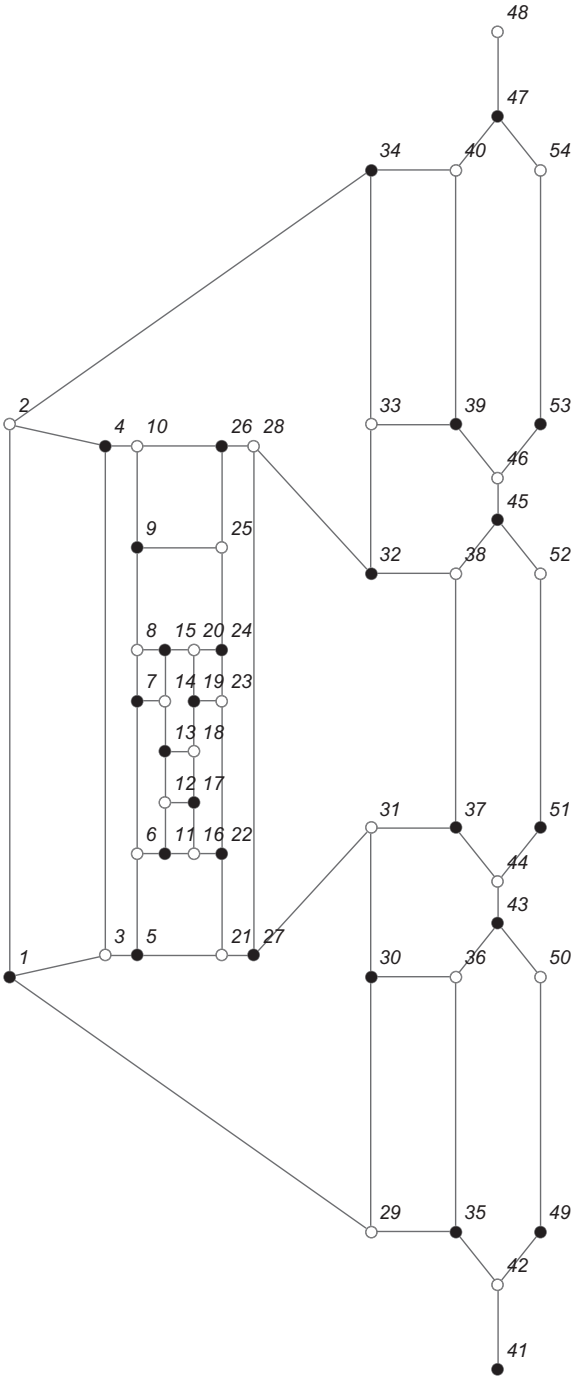
Example embedding coordinates (SAGE 7.2):

```
{0:[-6.,-14.3084],1:[-6.,14.6916],2:[-3.5,-11.8084],3:[-3.5,-5.263],4:[-3.5,-
3.0811],5:[-3.5,3.4643],6:[-3.5,5.6461],7:[-3.5,12.1916],8:[-2.,-11.8084],9:[-2.,-
10.4993],10:[-2.,-8.5357],11:[-2.,-7.8811],12:[-2.,-6.572],13:[-2.,-5.263],14:[-2.,-
3.0811],15:[-2.,-1.772],16:[-2.,0.1916],17:[-2.,0.846],18:[-2.,2.1552],19:[-
2.,3.4643],20:[-2.,5.6461],21:[-2.,6.9551],22:[-2.,8.9189],23:[-2.,9.5734],24:[-
2.,10.8825],25:[-2.,12.1916],26:[-0.6667,-10.4993],27:[-0.6667,-9.8448],28:[-0.6667,-
9.1902],29:[-0.6667,-8.5357],30:[-0.6667,-7.8811],31:[-0.6667,-1.772],32:[-0.6667,-
1.1175],33:[-0.6667,-0.463],34:[-0.6667,0.1916],35:[-0.6667,0.846],36:[-
0.6667,6.9551],37:[-0.6667,7.6098],38:[-0.6667,8.2642],39:[-0.6667,8.9189],40:[-
0.6667,9.5734],41:[0.6667,-10.4993],42:[0.6667,-9.8448],43:[0.6667,-
9.1902],44:[0.6667,-8.5357],45:[0.6667,-7.8811],46:[0.6667,-1.772],47:[0.6667,-
1.1175],48:[0.6667,-
0.463],49:[0.6667,0.1916],50:[0.6667,0.846],51:[0.6667,6.9551],52:[0.6667,7.6098],53:[
0.6667,8.2642],54:[0.6667,8.9189],55:[0.6667,9.5734],56:[2.,-11.8084],57:[2.,-
10.4993],58:[2.,-8.5357],59:[2.,-7.8811],60:[2.,-6.572],61:[2.,-5.263],62:[2.,-
3.0811],63:[2.,-
1.772],64:[2.,0.1916],65:[2.,0.846],66:[2.,2.1552],67:[2.,3.4643],68:[2.,5.6461],69:[2
.,6.9551],70:[2.,8.9189],71:[2.,9.5734],72:[2.,10.8825],73:[2.,12.1916],74:[3.5,-
11.8084],75:[3.5,-5.263],76:[3.5,-
3.0811],77:[3.5,3.4643],78:[3.5,5.6461],79:[3.5,12.1916],80:[6.,-
14.3084],81:[6.,14.6916]}
```

Canonical vertex $(k = 2)$-coloring ::

```
{1->colorTwo,3->colorOne,2->colorOne,8->colorTwo,4->colorTwo,9->colorTwo,5-
>colorOne,14->colorOne,6->colorTwo,15->colorTwo,7->colorOne,20->colorOne,21-
>colorTwo,26->colorOne,10->colorOne,57->colorOne,11->colorTwo,27->colorTwo,12-
>colorOne,30->colorOne,13->colorTwo,31->colorTwo,61->colorOne,62->colorTwo,16-
>colorOne,63->colorOne,17->colorTwo,32->colorTwo,18->colorOne,35->colorOne,19-
>colorTwo,36->colorTwo,67->colorOne,68->colorTwo,22->colorOne,69->colorOne,23-
>colorTwo,37->colorTwo,24->colorOne,40->colorOne,25->colorTwo,41->colorTwo,73-
>colorOne,74->colorTwo,28->colorOne,42->colorOne,29->colorTwo,43->colorTwo,44-
>colorOne,46->colorOne,33->colorOne,47->colorOne,34->colorTwo,48->colorTwo,49-
>colorOne,51->colorOne,38->colorOne,52->colorOne,39->colorTwo,53->colorTwo,54-
>colorOne,56->colorOne,58->colorTwo,45->colorTwo,59->colorOne,60->colorTwo,64-
>colorTwo,50->colorTwo,65->colorOne,66->colorTwo,70->colorTwo,55->colorTwo,71-
>colorOne,72->colorTwo,75->colorTwo,76->colorOne,77->colorTwo,78->colorOne,79-
>colorTwo,80->colorOne,81->colorOne,82->colorTwo}
```

8.16 (Figure 4.5.b) gadget (specifying (z = 1) blocks)



Graph Properties ::

--

Number of Vertices: ‘54’

Number of Edges: ‘76’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->2,1<->3,1<->29,2<->4,2<->34,3<->4,3<->5,4<->10,5<->6,5<->21,6<->7,6<->11,7<->8,7<->14,8<->9,8<->15,9<->10,9<->25,10<->26,11<->12,11<->16,12<->13,12<->17,13<->14,13<->18,14<->15,15<->20,16<->17,16<->22,17<->18,18<->19,19<->20,19<->23,20<->24,21<->22,21<->27,22<->23,23<->24,24<->25,25<->26,26<->28,27<->28,27<->31,28<->32,29<->30,29<->35,30<->31,30<->36,31<->37,32<->33,32<->38,33<->34,33<->39,34<->40,35<->36,35<->42,36<->43,37<->38,37<->44,38<->45,39<->40,39<->46,40<->47,41<->42,42<->49,43<->44,43<->50,44<->51,45<->46,45<->52,46<->53,47<->48,47<->54,49<->50,51<->52,53<->54}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-14.2592,-12.6445},2->{-14.2592,13.3555},3->{-9.7592,-11.6445},4->{-9.7592,12.3554},5->{-8.2592,-11.6445},6->{-8.2592,-6.8446},7->{-8.2592,0.3555},8->{-8.2592,2.7558},9->{-8.2592,7.5557},10->{-8.2592,12.3554},11->{-6.9259,-6.8446},12->{-6.9259,-4.4447},13->{-6.9259,-2.0444},14->{-6.9259,0.3555},15->{-6.9259,2.7558},16->{-5.5925,-6.8446},17->{-5.5925,-4.4447},18->{-5.5925,-2.0444},19->{-5.5925,0.3555},20->{-5.5925,2.7558},21->{-4.2592,-11.6445},22->{-4.2592,-6.8446},23->{-4.2592,0.3555},24->{-4.2592,2.7558},25->{-4.2592,7.5557},26->{-4.2592,12.3554},27->{-2.7592,-11.6445},28->{-2.7592,12.3554},29->{2.7407,-24.6445},30->{2.7407,-12.6445},31->{2.7407,-5.6445},32->{2.7407,6.3555},33->{2.7407,13.3555},34->{2.7407,25.3555},35->{6.7408,-24.6445},36->{6.7408,-12.6445},37->{6.7408,-5.6445},38->{6.7408,6.3555},39->{6.7408,13.3555},40->{6.7408,25.3555},41->{8.7408,-31.1445},42->{8.7408,-27.1445},43->{8.7408,-10.1445},44->{8.7408,-8.1445},45->{8.7408,8.8555},46->{8.7408,10.8555},47->{8.7408,27.8555},48->{8.7408,31.8555},49->{10.7408,-24.6445},50->{10.7408,-12.6445},51->{10.7408,-5.6445},52->{10.7408,6.3555},53->{10.7408,13.3555},54->{10.7408,25.3555}}

Edge list (SAGE 7.2):

[(0,1),(0,2),(0,28),(1,3),(1,33),(2,3),(2,4),(3,9),(4,5),(4,20),(5,6),(5,10),(6,7),(6,13),(7,8),(7,14),(8,9),(8,24),(9,25),(10,11),(10,15),(11,12),(11,16),(12,13),(12,17),(13,14),(14,19),(15,16),(15,21),(16,17),(17,18),(18,19),(18,22),(19,23),(20,21),(20,26),(21,22),(22,23),(23,24),(24,25),(25,27),(26,27),(26,30),(27,31),(28,29),(28,34),(29,30),(29,35),(30,36),(31,32),(31,37),(32,33),(32,38),(33,39),(34,35),(34,41),(35,42),(36,37),(36,43),(37,44),(38,39),(38,45),(39,46),(40,41),(41,48),(42,43),(42,49),(43,50),(44,45),(44,51),(45,52),(46,47),(46,53),(48,49),(50,51),(52,53)]

-

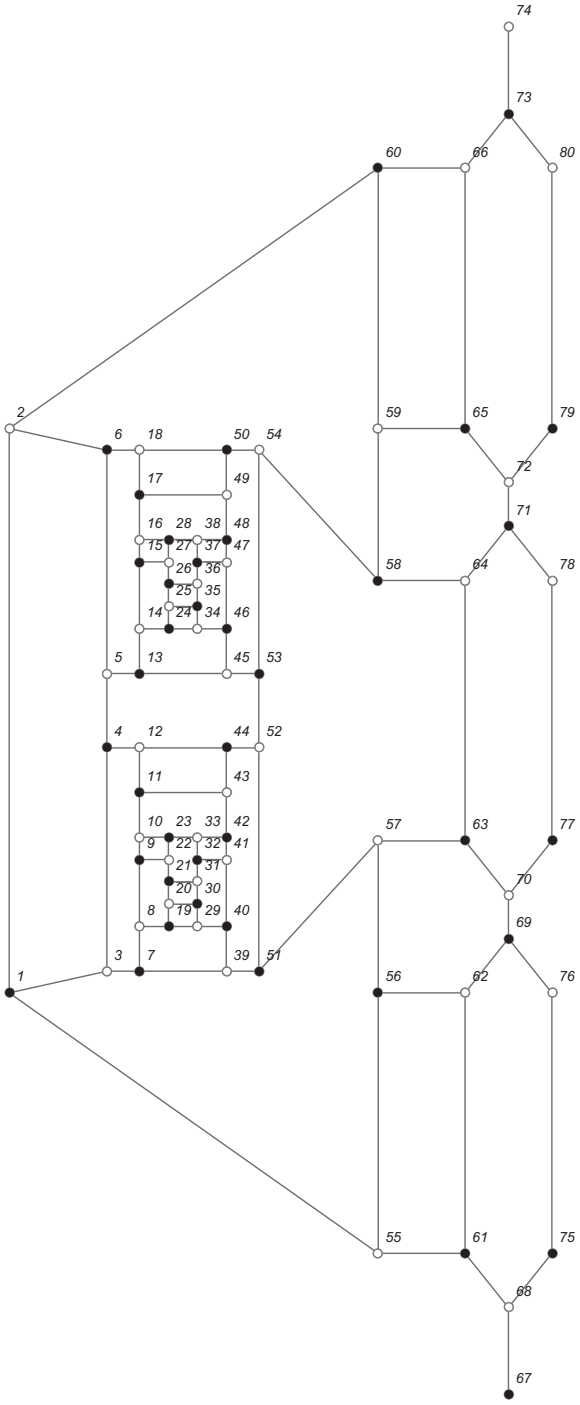
Example embedding coordinates (SAGE 7.2):

{0:[-14.2592,-12.6445],1:[-14.2592,13.3555],2:[-9.7592,-11.6445],3:[-
9.7592,12.3554],4:[-8.2592,-11.6445],5:[-8.2592,-6.8446],6:[-8.2592,0.3555],7:[-
8.2592,2.7558],8:[-8.2592,7.5557],9:[-8.2592,12.3554],10:[-6.9259,-6.8446],11:[-
6.9259,-4.4447],12:[-6.9259,-2.0444],13:[-6.9259,0.3555],14:[-6.9259,2.7558],15:[-
5.5925,-6.8446],16:[-5.5925,-4.4447],17:[-5.5925,-2.0444],18:[-5.5925,0.3555],19:[-
5.5925,2.7558],20:[-4.2592,-11.6445],21:[-4.2592,-6.8446],22:[-4.2592,0.3555],23:[-
4.2592,2.7558],24:[-4.2592,7.5557],25:[-4.2592,12.3554],26:[-2.7592,-11.6445],27:[-
2.7592,12.3554],28:[2.7407,-24.6445],29:[2.7407,-12.6445],30:[2.7407,-
5.6445],31:[2.7407,6.3555],32:[2.7407,13.3555],33:[2.7407,25.3555],34:[6.7408,-
24.6445],35:[6.7408,-12.6445],36:[6.7408,-
5.6445],37:[6.7408,6.3555],38:[6.7408,13.3555],39:[6.7408,25.3555],40:[8.7408,-
31.1445],41:[8.7408,-27.1445],42:[8.7408,-10.1445],43:[8.7408,-
8.1445],44:[8.7408,8.8555],45:[8.7408,10.8555],46:[8.7408,27.8555],47:[8.7408,31.8555]
,48:[10.7408,-24.6445],49:[10.7408,-12.6445],50:[10.7408,-
5.6445],51:[10.7408,6.3555],52:[10.7408,13.3555],53:[10.7408,25.3555]}

Canonical vertex ($k = 2$)-coloring ::

{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorOne,6->colorTwo,7-
>colorOne,8->colorTwo,9->colorOne,10->colorTwo,11->colorOne,12->colorTwo,13-
>colorOne,14->colorTwo,15->colorOne,16->colorTwo,17->colorOne,18->colorTwo,19-
>colorOne,20->colorTwo,21->colorTwo,22->colorOne,23->colorTwo,24->colorOne,25-
>colorTwo,26->colorOne,27->colorOne,28->colorTwo,29->colorTwo,30->colorOne,31-
>colorTwo,32->colorOne,33->colorTwo,34->colorOne,35->colorOne,36->colorTwo,37-
>colorOne,38->colorTwo,39->colorOne,40->colorTwo,41->colorOne,42->colorTwo,43-
>colorOne,44->colorTwo,45->colorOne,46->colorTwo,47->colorOne,48->colorTwo,49-
>colorOne,50->colorTwo,51->colorOne,52->colorTwo,53->colorOne,54->colorTwo}

8.17 (Figure 4.5.b) gadget (specifying (z = 2) blocks)



Graph Properties ::

--

Number of Vertices: ‘80’

Number of Edges: ‘115’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->2,1<->3,1<->55,2<->6,2<->60,3<->4,3<->7,4<->5,4<->12,5<->6,5<->13,6<->18,7<->8,7<->39,8<->9,8<->19,9<->10,9<->22,10<->11,10<->23,11<->12,11<->43,12<->44,13<->14,13<->45,14<->15,14<->24,15<->16,15<->27,16<->17,16<->28,17<->18,17<->49,18<->50,19<->20,19<->29,20<->21,20<->30,21<->22,21<->31,22<->23,23<->33,24<->25,24<->34,25<->26,25<->35,26<->27,26<->36,27<->28,28<->38,29<->30,29<->40,30<->31,31<->32,32<->33,32<->41,33<->42,34<->35,34<->46,35<->36,36<->37,37<->38,37<->47,38<->48,39<->40,39<->51,40<->41,41<->42,42<->43,43<->44,44<->52,45<->46,45<->53,46<->47,47<->48,48<->49,49<->50,50<->54,51<->52,51<->57,52<->53,53<->54,54<->58,55<->56,55<->61,56<->57,56<->62,57<->63,58<->59,58<->64,59<->60,59<->65,60<->66,61<->62,61<->68,62<->69,63<->64,63<->70,64<->71,65<->66,65<->72,66<->73,67<->68,68<->75,69<->70,69<->76,70<->77,71<->72,71<->78,72<->79,73<->74,73<->80,75<->76,77<->78,79<->80}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-12.225,-12.7943},2->{-12.225,13.2057},3->{-7.725,-11.7943},4->{-7.725,-1.5087},5->{-7.725,1.92},6->{-7.725,12.2057},7->{-6.225,-11.7943},8->{-6.225,-9.7372},9->{-6.225,-6.6515},10->{-6.225,-5.6228},11->{-6.225,-3.5657},12->{-6.225,-1.5087},13->{-6.225,1.92},14->{-6.225,3.9772},15->{-6.225,7.0629},16->{-6.225,8.0912},17->{-6.225,10.1485},18->{-6.225,12.2057},19->{-4.8917,-9.7372},20->{-4.8917,-8.7087},21->{-4.8917,-7.68},22->{-4.8917,-6.6515},23->{-4.8917,-5.6228},24->{-4.8917,3.9772},25->{-4.8917,5.0057},26->{-4.8917,6.0342},27->{-4.8917,7.0629},28->{-4.8917,8.0912},29->{-3.5583,-9.7372},30->{-3.5583,-8.7087},31->{-3.5583,-7.68},32->{-3.5583,-6.6515},33->{-3.5583,-5.6228},34->{-3.5583,3.9772},35->{-3.5583,5.0057},36->{-3.5583,6.0342},37->{-3.5583,7.0629},38->{-3.5583,8.0912},39->{-2.225,-11.7943},40->{-2.225,-9.7372},41->{-2.225,-6.6515},42->{-2.225,-5.6228},43->{-2.225,-3.5657},44->{-2.225,-1.5087},45->{-2.225,1.92},46->{-2.225,3.9772},47->{-2.225,7.0629},48->{-2.225,8.0912},49->{-2.225,10.1485},50->{-2.225,12.2057},51->{-0.725,-11.7943},52->{-0.725,-1.5087},53->{-0.725,1.92},54->{-0.725,12.2057},55->{4.7749,-24.7943},56->{4.7749,-12.7943},57->{4.7749,-5.7943},58->{4.7749,6.2057},59->{4.7749,13.2057},60->{4.7749,25.2057},61->{8.775,-24.7943},62->{8.775,-12.7943},63->{8.775,-5.7943},64->{8.775,6.2057},65->{8.775,13.2057},66->{8.775,25.2057},67->{10.775,-31.2943},68->{10.775,-27.2943},69->{10.775,-10.2943},70->{10.775,-8.2943},71->{10.775,8.7057},72->{10.775,10.7057},73->{10.775,27.7057},74->{10.775,31.7057},75->{12.775,-24.7943},76->{12.775,-12.7943},77->{12.775,-5.7943},78->{12.775,6.2057},79->{12.775,13.2057},80->{12.775,25.2057}}

Edge list (SAGE 7.2):

[(0,1),(0,2),(0,54),(1,5),(1,59),(2,3),(2,6),(3,4),(3,11),(4,5),(4,12),(5,17),(6,7),(6,38),(7,8),(7,18),(8,9),(8,21),(9,10),(9,22),(10,11),(10,42),(11,43),(12,13),(12,44),(13,14),(13,23),(14,15),(14,26),(15,16),(15,27),(16,17),(16,48),(17,49),(18,19),(18,28),(19,20),(19,29),(20,21),(20,30),(21,22),(22,32),(23,24),(23,33),(24,25),(24,34),(25,26),(25,35),(26,27),(27,37),(28,29),(28,39),(29,30),(30,31),(31,32),(31,40),(32,41),(33,34),(33,45),(34,35),(35,36),(36,37),(36,46),(37,47),(38,39),(38,50),(39,40),(40,41),(41,42),(42,43),(43,51),(44,45),(44,52),(45,46),(46,47),(47,48),(48,49),(49,53),(50,51),(50,56),(51,52),(52,53),(53,57),(54,55),(54,60),(55,56),(55,61),(56,62),(57,58),(57,63),(58,59),(58,64),(59,65),(60,61),(60,67),(61,68),(62,63),(62,69),(63,70),(64,65),(64,71),(65,72),(66,67),(67,74),(68,69),(68,75),(69,76),(70,71),(70,77),(71,78),(72,73),(72,79),(74,75),(76,77),(78,79)]

-

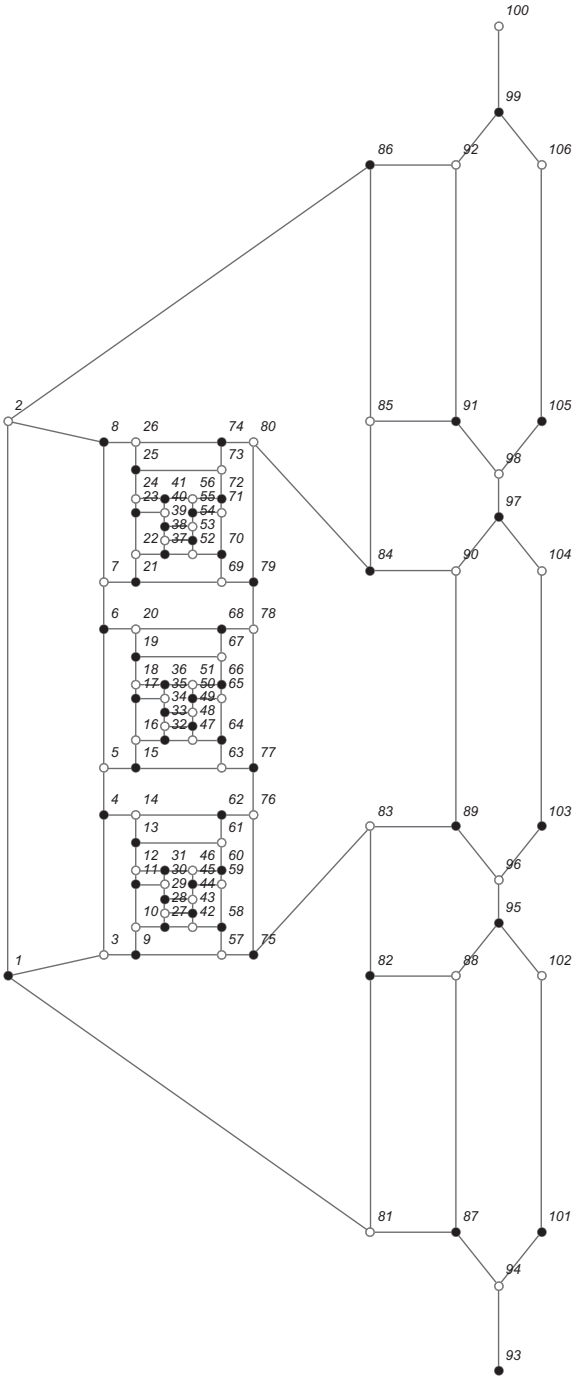
Example embedding coordinates (SAGE 7.2):

```
{0:[-12.225,-12.7943],1:[-12.225,13.2057],2:[-7.725,-11.7943],3:[-7.725,-1.5087],4:[-
7.725,1.92],5:[-7.725,12.2057],6:[-6.225,-11.7943],7:[-6.225,-9.7372],8:[-6.225,-
6.6515],9:[-6.225,-5.6228],10:[-6.225,-3.5657],11:[-6.225,-1.5087],12:[-
6.225,1.92],13:[-6.225,3.9772],14:[-6.225,7.0629],15:[-6.225,8.0912],16:[-
6.225,10.1485],17:[-6.225,12.2057],18:[-4.8917,-9.7372],19:[-4.8917,-8.7087],20:[-
4.8917,-7.68],21:[-4.8917,-6.6515],22:[-4.8917,-5.6228],23:[-4.8917,3.9772],24:[-
4.8917,5.0057],25:[-4.8917,6.0342],26:[-4.8917,7.0629],27:[-4.8917,8.0912],28:[-
3.5583,-9.7372],29:[-3.5583,-8.7087],30:[-3.5583,-7.68],31:[-3.5583,-6.6515],32:[-
3.5583,-5.6228],33:[-3.5583,3.9772],34:[-3.5583,5.0057],35:[-3.5583,6.0342],36:[-
3.5583,7.0629],37:[-3.5583,8.0912],38:[-2.225,-11.7943],39:[-2.225,-9.7372],40:[-
2.225,-6.6515],41:[-2.225,-5.6228],42:[-2.225,-3.5657],43:[-2.225,-1.5087],44:[-
2.225,1.92],45:[-2.225,3.9772],46:[-2.225,7.0629],47:[-2.225,8.0912],48:[-
2.225,10.1485],49:[-2.225,12.2057],50:[-0.725,-11.7943],51:[-0.725,-1.5087],52:[-
0.725,1.92],53:[-0.725,12.2057],54:[4.7749,-24.7943],55:[4.7749,-12.7943],56:[4.7749,-
5.7943],57:[4.7749,6.2057],58:[4.7749,13.2057],59:[4.7749,25.2057],60:[8.775,-
24.7943],61:[8.775,-12.7943],62:[8.775,-
5.7943],63:[8.775,6.2057],64:[8.775,13.2057],65:[8.775,25.2057],66:[10.775,-
31.2943],67:[10.775,-27.2943],68:[10.775,-10.2943],69:[10.775,-
8.2943],70:[10.775,8.7057],71:[10.775,10.7057],72:[10.775,27.7057],73:[10.775,31.7057]
,74:[12.775,-24.7943],75:[12.775,-12.7943],76:[12.775,-
5.7943],77:[12.775,6.2057],78:[12.775,13.2057],79:[12.775,25.2057]}
```

Canonical vertex $(k = 2)$-coloring ::

```
{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorTwo,6->colorOne,7-
>colorOne,8->colorTwo,9->colorOne,10->colorTwo,11->colorOne,12->colorTwo,13-
>colorOne,14->colorTwo,15->colorOne,16->colorTwo,17->colorOne,18->colorTwo,19-
>colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorOne,25-
>colorTwo,26->colorOne,27->colorTwo,28->colorOne,29->colorTwo,30->colorOne,31-
>colorTwo,32->colorOne,33->colorTwo,34->colorTwo,35->colorOne,36->colorTwo,37-
>colorOne,38->colorTwo,39->colorTwo,40->colorOne,41->colorTwo,42->colorOne,43-
>colorTwo,44->colorOne,45->colorTwo,46->colorOne,47->colorTwo,48->colorOne,49-
>colorTwo,50->colorOne,51->colorOne,52->colorTwo,53->colorOne,54->colorTwo,55-
>colorTwo,56->colorOne,57->colorTwo,58->colorOne,59->colorTwo,60->colorOne,61-
>colorOne,62->colorTwo,63->colorOne,64->colorTwo,65->colorOne,66->colorTwo,67-
>colorOne,68->colorTwo,69->colorOne,70->colorTwo,71->colorOne,72->colorTwo,73-
>colorOne,74->colorTwo,75->colorOne,76->colorTwo,77->colorOne,78->colorTwo,79-
>colorOne,80->colorTwo}
```

8.18 (Figure 4.5.b) gadget (specifying (z = 3) blocks)



Graph Properties ::

--

Number of Vertices: ‘106’

Number of Edges: ‘154’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->2,1<->3,1<->81,2<->8,2<->86,3<->4,3<->9,4<->5,4<->14,5<->6,5<->15,6<->7,6<->20,7<->8,7<->21,8<->26,9<->10,9<->57,10<->11,10<->27,11<->12,11<->30,12<->13,12<->31,13<->14,13<->61,14<->62,15<->16,15<->63,16<->17,16<->32,17<->18,17<->35,18<->19,18<->36,19<->20,19<->67,20<->68,21<->22,21<->69,22<->23,22<->37,23<->24,23<->40,24<->25,24<->41,25<->26,25<->73,26<->74,27<->28,27<->42,28<->29,28<->43,29<->30,29<->44,30<->31,31<->46,32<->33,32<->47,33<->34,33<->48,34<->35,34<->49,35<->36,36<->51,37<->38,37<->52,38<->39,38<->53,39<->40,39<->54,40<->41,41<->56,42<->43,42<->58,43<->44,44<->45,45<->46,45<->59,46<->60,47<->48,47<->64,48<->49,49<->50,50<->51,50<->65,51<->66,52<->53,52<->70,53<->54,54<->55,55<->56,55<->71,56<->72,57<->58,57<->75,58<->59,59<->60,60<->61,61<->62,62<->76,63<->64,63<->77,64<->65,65<->66,66<->67,67<->68,68<->78,69<->70,69<->79,70<->71,71<->72,72<->73,73<->74,74<->80,75<->76,75<->83,76<->77,77<->78,78<->79,79<->80,80<->84,81<->82,81<->87,82<->83,82<->88,83<->89,84<->85,84<->90,85<->86,85<->91,86<->92,87<->88,87<->94,88<->95,89<->90,89<->96,90<->97,91<->92,91<->98,92<->99,93<->94,94<->101,95<->96,95<->102,96<->103,97<->98,97<->104,98<->105,99<->100,99<->106,101<->102,103<->104,105<->106}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-11.1887,-12.8518},2->{-11.1887,13.1482},3->{-6.6887,-11.8518},4->{-6.6887,-5.3064},5->{-6.6887,-3.1245},6->{-6.6887,3.4209},7->{-6.6887,5.6027},8->{-6.6887,12.1482},9->{-5.1887,-11.8518},10->{-5.1887,-10.5427},11->{-5.1887,-8.5791},12->{-5.1887,-7.9245},13->{-5.1887,-6.6154},14->{-5.1887,-5.3064},15->{-5.1887,-3.1245},16->{-5.1887,-1.8154},17->{-5.1887,0.1482},18->{-5.1887,0.8026},19->{-5.1887,2.1118},20->{-5.1887,3.4209},21->{-5.1887,5.6027},22->{-5.1887,6.9117},23->{-5.1887,8.8755},24->{-5.1887,9.53},25->{-5.1887,10.8391},26->{-5.1887,12.1482},27->{-3.8554,-10.5427},28->{-3.8554,-9.8882},29->{-3.8554,-9.2336},30->{-3.8554,-8.5791},31->{-3.8554,-7.9245},32->{-3.8554,-1.8154},33->{-3.8554,-1.1609},34->{-3.8554,-0.5064},35->{-3.8554,0.1482},36->{-3.8554,0.8026},37->{-3.8554,6.9117},38->{-3.8554,7.5664},39->{-3.8554,8.2208},40->{-3.8554,8.8755},41->{-3.8554,9.53},42->{-2.522,-10.5427},43->{-2.522,-9.8882},44->{-2.522,-9.2336},45->{-2.522,-8.5791},46->{-2.522,-7.9245},47->{-2.522,-1.8154},48->{-2.522,-1.1609},49->{-2.522,-0.5064},50->{-2.522,0.1482},51->{-2.522,0.8026},52->{-2.522,6.9117},53->{-2.522,7.5664},54->{-2.522,8.2208},55->{-2.522,8.8755},56->{-2.522,9.53},57->{-1.1887,-11.8518},58->{-1.1887,-10.5427},59->{-1.1887,-8.5791},60->{-1.1887,-7.9245},61->{-1.1887,-6.6154},62->{-1.1887,-5.3064},63->{-1.1887,-3.1245},64->{-1.1887,-1.8154},65->{-1.1887,0.1482},66->{-1.1887,0.8026},67->{-1.1887,2.1118},68->{-1.1887,3.4209},69->{-1.1887,5.6027},70->{-1.1887,6.9117},71->{-1.1887,8.8755},72->{-1.1887,9.53},73->{-1.1887,10.8391},74->{-1.1887,12.1482},75->{0.3113,-11.8518},76->{0.3113,-5.3064},77->{0.3113,-3.1245},78->{0.3113,3.4209},79->{0.3113,5.6027},80->{0.3113,12.1482},81->{5.8112,-24.8518},82->{5.8112,-12.8518},83->{5.8112,-5.8518},84->{5.8112,6.1482},85->{5.8112,13.1482},86->{5.8112,25.1482},87->{9.8113,-24.8518},88->{9.8113,-12.8518},89->{9.8113,-5.8518},90->{9.8113,6.1482},91->{9.8113,13.1482},92->{9.8113,25.1482},93->{11.8113,-31.3518},94->{11.8113,-27.3518},95->{11.8113,-10.3518},96->{11.8113,-8.3518},97->{11.8113,8.6482},98->{11.8113,10.6482},99->{11.8113,27.6482},100->{11.8113,31.6482},101->{13.8113,-24.8518},102->{13.8113,-12.8518},103->{13.8113,-5.8518},104->{13.8113,6.1482},105->{13.8113,13.1482},106->{13.8113,25.1482}}

Edge list (SAGE 7.2):

```
[(0,1),(0,2),(0,80),(1,7),(1,85),(2,3),(2,8),(3,4),(3,13),(4,5),(4,14),(5,6),(5,19),(6
,7),(6,20),(7,25),(8,9),(8,56),(9,10),(9,26),(10,11),(10,29),(11,12),(11,30),(12,13),(
12,60),(13,61),(14,15),(14,62),(15,16),(15,31),(16,17),(16,34),(17,18),(17,35),(18,19)
,(18,66),(19,67),(20,21),(20,68),(21,22),(21,36),(22,23),(22,39),(23,24),(23,40),(24,2
5),(24,72),(25,73),(26,27),(26,41),(27,28),(27,42),(28,29),(28,43),(29,30),(30,45),(31
,32),(31,46),(32,33),(32,47),(33,34),(33,48),(34,35),(35,50),(36,37),(36,51),(37,38),(
37,52),(38,39),(38,53),(39,40),(40,55),(41,42),(41,57),(42,43),(43,44),(44,45),(44,58)
,(45,59),(46,47),(46,63),(47,48),(48,49),(49,50),(49,64),(50,65),(51,52),(51,69),(52,5
3),(53,54),(54,55),(54,70),(55,71),(56,57),(56,74),(57,58),(58,59),(59,60),(60,61),(61
,75),(62,63),(62,76),(63,64),(64,65),(65,66),(66,67),(67,77),(68,69),(68,78),(69,70),(
70,71),(71,72),(72,73),(73,79),(74,75),(74,82),(75,76),(76,77),(77,78),(78,79),(79,83)
,(80,81),(80,86),(81,82),(81,87),(82,88),(83,84),(83,89),(84,85),(84,90),(85,91),(86,8
7),(86,93),(87,94),(88,89),(88,95),(89,96),(90,91),(90,97),(91,98),(92,93),(93,100),(9
4,95),(94,101),(95,102),(96,97),(96,103),(97,104),(98,99),(98,105),(100,101),(102,103)
,(104,105)]
```

-

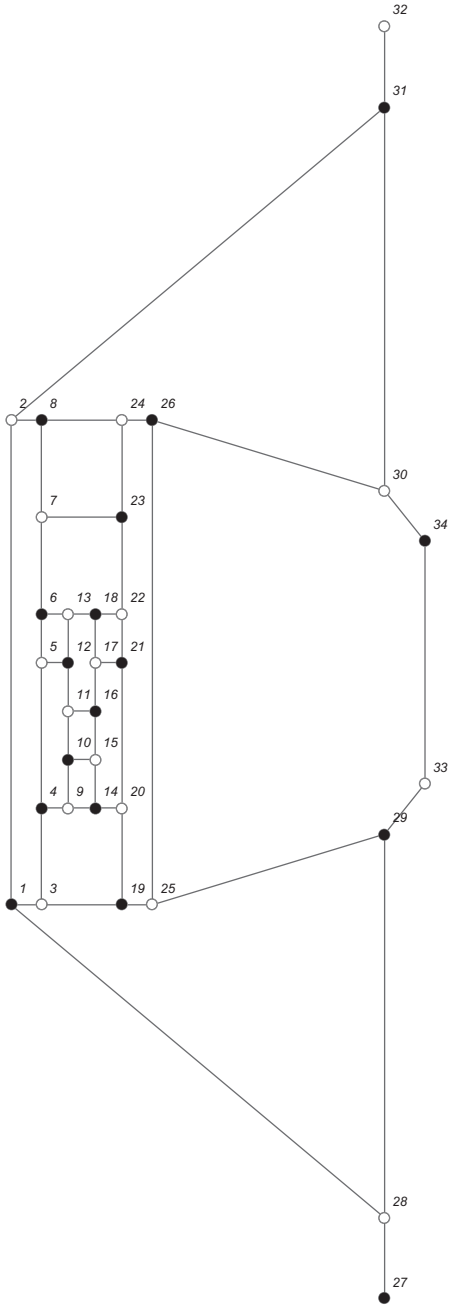
Example embedding coordinates (SAGE 7.2):

```
{0:[-11.1887,-12.8518],1:[-11.1887,13.1482],2:[-6.6887,-11.8518],3:[-6.6887,-
5.3064],4:[-6.6887,-3.1245],5:[-6.6887,3.4209],6:[-6.6887,5.6027],7:[-
6.6887,12.1482],8:[-5.1887,-11.8518],9:[-5.1887,-10.5427],10:[-5.1887,-8.5791],11:[-
5.1887,-7.9245],12:[-5.1887,-6.6154],13:[-5.1887,-5.3064],14:[-5.1887,-3.1245],15:[-
5.1887,-1.8154],16:[-5.1887,0.1482],17:[-5.1887,0.8026],18:[-5.1887,2.1118],19:[-
5.1887,3.4209],20:[-5.1887,5.6027],21:[-5.1887,6.9117],22:[-5.1887,8.8755],23:[-
5.1887,9.53],24:[-5.1887,10.8391],25:[-5.1887,12.1482],26:[-3.8554,-10.5427],27:[-
3.8554,-9.8882],28:[-3.8554,-9.2336],29:[-3.8554,-8.5791],30:[-3.8554,-7.9245],31:[-
3.8554,-1.8154],32:[-3.8554,-1.1609],33:[-3.8554,-0.5064],34:[-3.8554,0.1482],35:[-
3.8554,0.8026],36:[-3.8554,6.9117],37:[-3.8554,7.5664],38:[-3.8554,8.2208],39:[-
3.8554,8.8755],40:[-3.8554,9.53],41:[-2.522,-10.5427],42:[-2.522,-9.8882],43:[-2.522,-
9.2336],44:[-2.522,-8.5791],45:[-2.522,-7.9245],46:[-2.522,-1.8154],47:[-2.522,-
1.1609],48:[-2.522,-0.5064],49:[-2.522,0.1482],50:[-2.522,0.8026],51:[-
2.522,6.9117],52:[-2.522,7.5664],53:[-2.522,8.2208],54:[-2.522,8.8755],55:[-
2.522,9.53],56:[-1.1887,-11.8518],57:[-1.1887,-10.5427],58:[-1.1887,-8.5791],59:[-
1.1887,-7.9245],60:[-1.1887,-6.6154],61:[-1.1887,-5.3064],62:[-1.1887,-3.1245],63:[-
1.1887,-1.8154],64:[-1.1887,0.1482],65:[-1.1887,0.8026],66:[-1.1887,2.1118],67:[-
1.1887,3.4209],68:[-1.1887,5.6027],69:[-1.1887,6.9117],70:[-1.1887,8.8755],71:[-
1.1887,9.53],72:[-1.1887,10.8391],73:[-1.1887,12.1482],74:[0.3113,-
11.8518],75:[0.3113,-5.3064],76:[0.3113,-
3.1245],77:[0.3113,3.4209],78:[0.3113,5.6027],79:[0.3113,12.1482],80:[5.8112,-
24.8518],81:[5.8112,-12.8518],82:[5.8112,-
5.8518],83:[5.8112,6.1482],84:[5.8112,13.1482],85:[5.8112,25.1482],86:[9.8113,-
24.8518],87:[9.8113,-12.8518],88:[9.8113,-
5.8518],89:[9.8113,6.1482],90:[9.8113,13.1482],91:[9.8113,25.1482],92:[11.8113,-
31.3518],93:[11.8113,-27.3518],94:[11.8113,-10.3518],95:[11.8113,-
8.3518],96:[11.8113,8.6482],97:[11.8113,10.6482],98:[11.8113,27.6482],99:[11.8113,31.6
482],100:[13.8113,-24.8518],101:[13.8113,-12.8518],102:[13.8113,-
5.8518],103:[13.8113,6.1482],104:[13.8113,13.1482],105:[13.8113,25.1482]}
```

Canonical vertex $(k = 2)$-coloring ::

{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorTwo,6->colorOne,7-
>colorTwo,8->colorOne,9->colorOne,10->colorTwo,11->colorOne,12->colorTwo,13-
>colorOne,14->colorTwo,15->colorOne,16->colorTwo,17->colorOne,18->colorTwo,19-
>colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25-
>colorOne,26->colorTwo,27->colorOne,28->colorTwo,29->colorOne,30->colorTwo,31-
>colorOne,32->colorOne,33->colorTwo,34->colorOne,35->colorTwo,36->colorOne,37-
>colorOne,38->colorTwo,39->colorOne,40->colorTwo,41->colorOne,42->colorTwo,43-
>colorOne,44->colorTwo,45->colorOne,46->colorTwo,47->colorTwo,48->colorOne,49-
>colorTwo,50->colorOne,51->colorTwo,52->colorTwo,53->colorOne,54->colorTwo,55-
>colorOne,56->colorTwo,57->colorTwo,58->colorOne,59->colorTwo,60->colorOne,61-
>colorTwo,62->colorOne,63->colorTwo,64->colorOne,65->colorTwo,66->colorOne,67-
>colorTwo,68->colorOne,69->colorTwo,70->colorOne,71->colorTwo,72->colorOne,73-
>colorTwo,74->colorOne,75->colorOne,76->colorTwo,77->colorOne,78->colorTwo,79-
>colorOne,80->colorTwo,81->colorTwo,82->colorOne,83->colorTwo,84->colorOne,85-
>colorTwo,86->colorOne,87->colorOne,88->colorTwo,89->colorOne,90->colorTwo,91-
>colorOne,92->colorTwo,93->colorOne,94->colorTwo,95->colorOne,96->colorTwo,97-
>colorOne,98->colorTwo,99->colorOne,100->colorTwo,101->colorOne,102->colorTwo,103-
>colorOne,104->colorTwo,105->colorOne,106->colorTwo}

8.19 (Figure 4.5.c) gadget (specifying (z = 1) blocks)



Graph Properties ::

--

Number of Vertices: ‘34’

Number of Edges: ‘48’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->2,1<->3,1<->28,2<->8,2<->31,3<->4,3<->19,4<->5,4<->9,5<->6,5<->12,6<->7,6<->13,7<->8,7<->23,8<->24,9<->10,9<->14,10<->11,10<->15,11<->12,11<->16,12<->13,13<->18,14<->15,14<->20,15<->16,16<->17,17<->18,17<->21,18<->22,19<->20,19<->25,20<->21,21<->22,22<->23,23<->24,24<->26,25<->26,25<->29,26<->30,27<->28,28<->29,29<->33,30<->31,30<->34,31<->32,33<->34}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-7.1471,-11.4353},2->{-7.1471,12.5646},3->{-5.6471,-11.4353},4->{-5.6471,-6.6354},5->{-5.6471,0.5647},6->{-5.6471,2.965},7->{-5.6471,7.7649},8->{-5.6471,12.5646},9->{-4.3138,-6.6354},10->{-4.3138,-4.2355},11->{-4.3138,-1.8352},12->{-4.3138,0.5647},13->{-4.3138,2.965},14->{-2.9804,-6.6354},15->{-2.9804,-4.2355},16->{-2.9804,-1.8352},17->{-2.9804,0.5647},18->{-2.9804,2.965},19->{-1.6471,-11.4353},20->{-1.6471,-6.6354},21->{-1.6471,0.5647},22->{-1.6471,2.965},23->{-1.6471,7.7649},24->{-1.6471,12.5646},25->{-0.1471,-11.4353},26->{-0.1471,12.5646},27->{11.3529,-30.9353},28->{11.3529,-26.9353},29->{11.3529,-7.9353},30->{11.3529,9.0647},31->{11.3529,28.0647},32->{11.3529,32.0647},33->{13.3529,-5.4353},34->{13.3529,6.5647}}

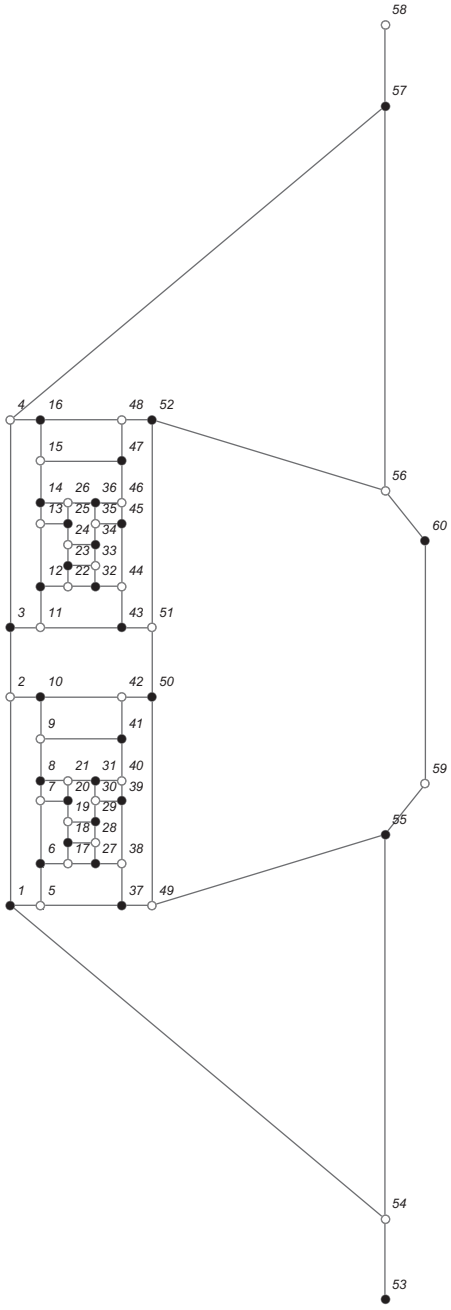
Edge list (SAGE 7.2):
[(0,1),(0,2),(0,27),(1,7),(1,30),(2,3),(2,18),(3,4),(3,8),(4,5),(4,11),(5,6),(5,12),(6,7),(6,22),(7,23),(8,9),(8,13),(9,10),(9,14),(10,11),(10,15),(11,12),(12,17),(13,14),(13,19),(14,15),(15,16),(16,17),(16,20),(17,21),(18,19),(18,24),(19,20),(20,21),(21,22),(22,23),(23,25),(24,25),(24,28),(25,29),(26,27),(27,28),(28,32),(29,30),(29,33),(30,31),(32,33)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-7.1471,-11.4353],1:[-7.1471,12.5646],2:[-5.6471,-11.4353],3:[-5.6471,-6.6354],4:[-5.6471,0.5647],5:[-5.6471,2.965],6:[-5.6471,7.7649],7:[-5.6471,12.5646],8:[-4.3138,-6.6354],9:[-4.3138,-4.2355],10:[-4.3138,-1.8352],11:[-4.3138,0.5647],12:[-4.3138,2.965],13:[-2.9804,-6.6354],14:[-2.9804,-4.2355],15:[-2.9804,-1.8352],16:[-2.9804,0.5647],17:[-2.9804,2.965],18:[-1.6471,-11.4353],19:[-1.6471,-6.6354],20:[-1.6471,0.5647],21:[-1.6471,2.965],22:[-1.6471,7.7649],23:[-1.6471,12.5646],24:[-0.1471,-11.4353],25:[-0.1471,12.5646],26:[11.3529,-30.9353],27:[11.3529,-26.9353],28:[11.3529,-7.9353],29:[11.3529,9.0647],30:[11.3529,28.0647],31:[11.3529,32.0647],32:[13.3529,-5.4353],33:[13.3529,6.5647]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorTwo,6->colorOne,7->colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13->colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorTwo,18->colorOne,19->colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25->colorTwo,26->colorOne,27->colorOne,28->colorTwo,29->colorOne,30->colorTwo,31->colorOne,32->colorTwo,33->colorTwo,34->colorOne}

8.20 (Figure 4.5.c) gadget (specifying (z = 2) blocks)



Graph Properties ::

--

Number of Vertices: ‘60’

Number of Edges: ‘87’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->2,1<->5,1<->54,2<->3,2<->10,3<->4,3<->11,4<->16,4<->57,5<->6,5<->37,6<->7,6<->17,7<->8,7<->20,8<->9,8<->21,9<->10,9<->41,10<->42,11<->12,11<->43,12<->13,12<->22,13<->14,13<->25,14<->15,14<->26,15<->16,15<->47,16<->48,17<->18,17<->27,18<->19,18<->28,19<->20,19<->29,20<->21,21<->31,22<->23,22<->32,23<->24,23<->33,24<->25,24<->34,25<->26,26<->36,27<->28,27<->38,28<->29,29<->30,30<->31,30<->39,31<->40,32<->33,32<->44,33<->34,34<->35,35<->36,35<->45,36<->46,37<->38,37<->49,38<->39,39<->40,40<->41,41<->42,42<->50,43<->44,43<->51,44<->45,45<->46,46<->47,47<->48,48<->52,49<->50,49<->55,50<->51,51<->52,52<->56,53<->54,54<->55,55<->59,56<->57,56<->60,57<->58,59<->60}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-5.5667,-11.7257},2->{-5.5667,-1.4401},3->{-5.5667,1.9886},4->{-5.5667,12.2743},5->{-4.0667,-11.7257},6->{-4.0667,-9.6686},7->{-4.0667,-6.5829},8->{-4.0667,-5.5542},9->{-4.0667,-3.4971},10->{-4.0667,-1.4401},11->{-4.0667,1.9886},12->{-4.0667,4.0458},13->{-4.0667,7.1315},14->{-4.0667,8.1598},15->{-4.0667,10.2171},16->{-4.0667,12.2743},17->{-2.7334,-9.6686},18->{-2.7334,-8.6401},19->{-2.7334,-7.6114},20->{-2.7334,-6.5829},21->{-2.7334,-5.5542},22->{-2.7334,4.0458},23->{-2.7334,5.0743},24->{-2.7334,6.1028},25->{-2.7334,7.1315},26->{-2.7334,8.1598},27->{-1.4,-9.6686},28->{-1.4,-8.6401},29->{-1.4,-7.6114},30->{-1.4,-6.5829},31->{-1.4,-5.5542},32->{-1.4,4.0458},33->{-1.4,5.0743},34->{-1.4,6.1028},35->{-1.4,7.1315},36->{-1.4,8.1598},37->{-0.0667,-11.7257},38->{-0.0667,-9.6686},39->{-0.0667,-6.5829},40->{-0.0667,-5.5542},41->{-0.0667,-3.4971},42->{-0.0667,-1.4401},43->{-0.0667,1.9886},44->{-0.0667,4.0458},45->{-0.0667,7.1315},46->{-0.0667,8.1598},47->{-0.0667,10.2171},48->{-0.0667,12.2743},49->{1.4333,-11.7257},50->{1.4333,-1.4401},51->{1.4333,1.9886},52->{1.4333,12.2743},53->{12.9333,-31.2257},54->{12.9333,-27.2257},55->{12.9333,-8.2257},56->{12.9333,8.7743},57->{12.9333,27.7743},58->{12.9333,31.7743},59->{14.9333,-5.7257},60->{14.9333,6.2743}}

Edge list (SAGE 7.2):

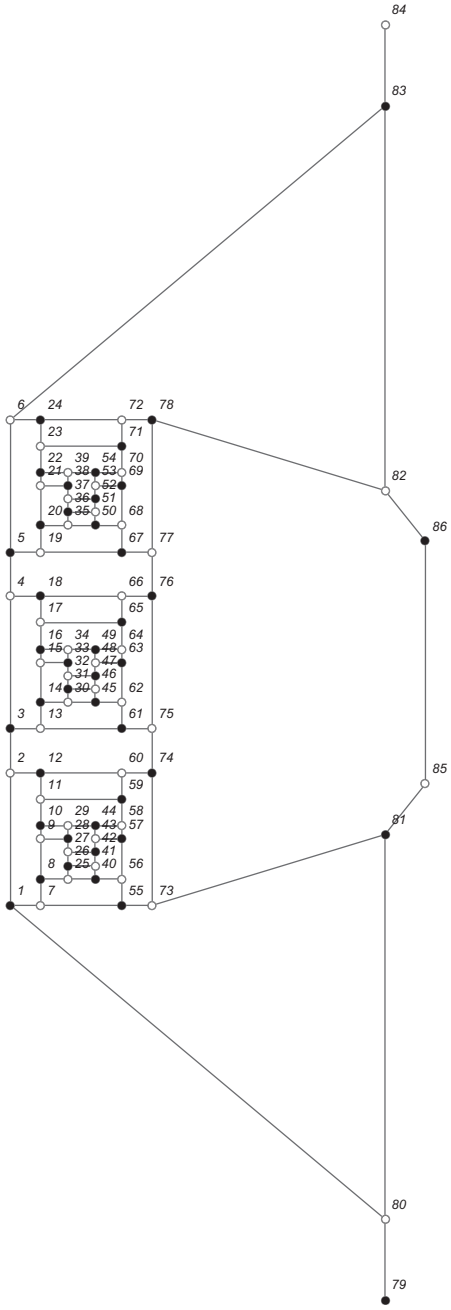
[(0,1),(0,4),(0,53),(1,2),(1,9),(2,3),(2,10),(3,15),(3,56),(4,5),(4,36),(5,6),(5,16),(6,7),(6,19),(7,8),(7,20),(8,9),(8,40),(9,41),(10,11),(10,42),(11,12),(11,21),(12,13),(12,24),(13,14),(13,25),(14,15),(14,46),(15,47),(16,17),(16,26),(17,18),(17,27),(18,19),(18,28),(19,20),(20,30),(21,22),(21,31),(22,23),(22,32),(23,24),(23,33),(24,25),(25,35),(26,27),(26,37),(27,28),(28,29),(29,30),(29,38),(30,39),(31,32),(31,43),(32,33),(33,34),(34,35),(34,44),(35,45),(36,37),(36,48),(37,38),(38,39),(39,40),(40,41),(41,49),(42,43),(42,50),(43,44),(44,45),(45,46),(46,47),(47,51),(48,49),(48,54),(49,50),(50,51),(51,55),(52,53),(53,54),(54,58),(55,56),(55,59),(56,57),(58,59)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-5.5667,-11.7257],1:[-5.5667,-1.4401],2:[-5.5667,1.9886],3:[-5.5667,12.2743],4:[-4.0667,-11.7257],5:[-4.0667,-9.6686],6:[-4.0667,-6.5829],7:[-4.0667,-5.5542],8:[-4.0667,-3.4971],9:[-4.0667,-1.4401],10:[-4.0667,1.9886],11:[-4.0667,4.0458],12:[-4.0667,7.1315],13:[-4.0667,8.1598],14:[-4.0667,10.2171],15:[-4.0667,12.2743],16:[-2.7334,-9.6686],17:[-2.7334,-8.6401],18:[-2.7334,-7.6114],19:[-2.7334,-6.5829],20:[-2.7334,-5.5542],21:[-2.7334,4.0458],22:[-2.7334,5.0743],23:[-2.7334,6.1028],24:[-2.7334,7.1315],25:[-2.7334,8.1598],26:[-1.4,-9.6686],27:[-1.4,-8.6401],28:[-1.4,-7.6114],29:[-1.4,-6.5829],30:[-1.4,-5.5542],31:[-1.4,4.0458],32:[-1.4,5.0743],33:[-1.4,6.1028],34:[-1.4,7.1315],35:[-1.4,8.1598],36:[-0.0667,-11.7257],37:[-0.0667,-9.6686],38:[-0.0667,-6.5829],39:[-0.0667,-5.5542],40:[-0.0667,-3.4971],41:[-0.0667,-1.4401],42:[-0.0667,1.9886],43:[-0.0667,4.0458],44:[-0.0667,7.1315],45:[-0.0667,8.1598],46:[-0.0667,10.2171],47:[-0.0667,12.2743],48:[1.4333,-11.7257],49:[1.4333,-1.4401],50:[1.4333,1.9886],51:[1.4333,12.2743],52:[12.9333,-31.2257],53:[12.9333,-27.2257],54:[12.9333,-8.2257],55:[12.9333,8.7743],56:[12.9333,27.7743],57:[12.9333,31.7743],58:[14.9333,-5.7257],59:[14.9333,6.2743]}

Canonical vertex ($k = 2$)-coloring :
{1->colorOne,2->colorTwo,3->colorOne,4->colorTwo,5->colorTwo,6->colorOne,7->colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13->colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorTwo,18->colorOne,19->colorTwo,20->colorOne,21->colorTwo,22->colorTwo,23->colorOne,24->colorTwo,25->colorOne,26->colorTwo,27->colorOne,28->colorTwo,29->colorOne,30->colorTwo,31->colorOne,32->colorOne,33->colorTwo,34->colorOne,35->colorTwo,36->colorOne,37->colorOne,38->colorTwo,39->colorOne,40->colorTwo,41->colorOne,42->colorTwo,43->colorOne,44->colorTwo,45->colorOne,46->colorTwo,47->colorOne,48->colorTwo,49->colorTwo,50->colorOne,51->colorTwo,52->colorOne,53->colorOne,54->colorTwo,55->colorOne,56->colorTwo,57->colorOne,58->colorTwo,59->colorTwo,60->colorOne}

8.21 (Figure 4.5.c) gadget (specifying (z = 3) blocks)



Graph Properties ::

--

Number of Vertices: ‘86’

Number of Edges: ‘126’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->2,1<->7,1<->80,2<->3,2<->12,3<->4,3<->13,4<->5,4<->18,5<->6,5<->19,6<->24,6<->83,7<->8,7<->55,8<->9,8<->25,9<->10,9<->28,10<->11,10<->29,11<->12,11<->59,12<->60,13<->14,13<->61,14<->15,14<->30,15<->16,15<->33,16<->17,16<->34,17<->18,17<->65,18<->66,19<->20,19<->67,20<->21,20<->35,21<->22,21<->38,22<->23,22<->39,23<->24,23<->71,24<->72,25<->26,25<->40,26<->27,26<->41,27<->28,27<->42,28<->29,29<->44,30<->31,30<->45,31<->32,31<->46,32<->33,32<->47,33<->34,34<->49,35<->36,35<->50,36<->37,36<->51,37<->38,37<->52,38<->39,39<->54,40<->41,40<->56,41<->42,42<->43,43<->44,43<->57,44<->58,45<->46,45<->62,46<->47,47<->48,48<->49,48<->63,49<->64,50<->51,50<->68,51<->52,52<->53,53<->54,53<->69,54<->70,55<->56,55<->73,56<->57,57<->58,58<->59,59<->60,60<->74,61<->62,61<->75,62<->63,63<->64,64<->65,65<->66,66<->76,67<->68,67<->77,68<->69,69<->70,70<->71,71<->72,72<->78,73<->74,73<->81,74<->75,75<->76,76<->77,77<->78,78<->82,79<->80,80<->81,81<->85,82<->83,82<->86,83<->84,85<->86}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-4.9419,-11.8173},2->{-4.9419,-5.2719},3->{-4.9419,-3.09},4->{-4.9419,3.4554},5->{-4.9419,5.6372},6->{-4.9419,12.1827},7->{-3.4419,-11.8173},8->{-3.4419,-10.5082},9->{-3.4419,-8.5446},10->{-3.4419,-7.89},11->{-3.4419,-6.5809},12->{-3.4419,-5.2719},13->{-3.4419,-3.09},14->{-3.4419,-1.7809},15->{-3.4419,0.1827},16->{-3.4419,0.8371},17->{-3.4419,2.1463},18->{-3.4419,3.4554},19->{-3.4419,5.6372},20->{-3.4419,6.9462},21->{-3.4419,8.91},22->{-3.4419,9.5645},23->{-3.4419,10.8736},24->{-3.4419,12.1827},25->{-2.1086,-10.5082},26->{-2.1086,-9.8537},27->{-2.1086,-9.1991},28->{-2.1086,-8.5446},29->{-2.1086,-7.89},30->{-2.1086,-1.7809},31->{-2.1086,-1.1264},32->{-2.1086,-0.4719},33->{-2.1086,0.1827},34->{-2.1086,0.8371},35->{-2.1086,6.9462},36->{-2.1086,7.6009},37->{-2.1086,8.2553},38->{-2.1086,8.91},39->{-2.1086,9.5645},40->{-0.7752,-10.5082},41->{-0.7752,-9.8537},42->{-0.7752,-9.1991},43->{-0.7752,-8.5446},44->{-0.7752,-7.89},45->{-0.7752,-1.7809},46->{-0.7752,-1.1264},47->{-0.7752,-0.4719},48->{-0.7752,0.1827},49->{-0.7752,0.8371},50->{-0.7752,6.9462},51->{-0.7752,7.6009},52->{-0.7752,8.2553},53->{-0.7752,8.91},54->{-0.7752,9.5645},55->{0.5581,-11.8173},56->{0.5581,-10.5082},57->{0.5581,-8.5446},58->{0.5581,-7.89},59->{0.5581,-6.5809},60->{0.5581,-5.2719},61->{0.5581,-3.09},62->{0.5581,-1.7809},63->{0.5581,0.1827},64->{0.5581,0.8371},65->{0.5581,2.1463},66->{0.5581,3.4554},67->{0.5581,5.6372},68->{0.5581,6.9462},69->{0.5581,8.91},70->{0.5581,9.5645},71->{0.5581,10.8736},72->{0.5581,12.1827},73->{2.0581,-11.8173},74->{2.0581,-5.2719},75->{2.0581,-3.09},76->{2.0581,3.4554},77->{2.0581,5.6372},78->{2.0581,12.1827},79->{13.5581,-31.3173},80->{13.5581,-27.3173},81->{13.5581,-8.3173},82->{13.5581,8.6827},83->{13.5581,27.6827},84->{13.5581,31.6827},85->{15.5581,-5.8173},86->{15.5581,6.1827}}

Edge list (SAGE 7.2):


```
[(0,1),(0,6),(0,79),(1,2),(1,11),(2,3),(2,12),(3,4),(3,17),(4,5),(4,18),(5,23),(5,82),
(6,7),(6,54),(7,8),(7,24),(8,9),(8,27),(9,10),(9,28),(10,11),(10,58),(11,59),(12,13),(
12,60),(13,14),(13,29),(14,15),(14,32),(15,16),(15,33),(16,17),(16,64),(17,65),(18,19)
,(18,66),(19,20),(19,34),(20,21),(20,37),(21,22),(21,38),(22,23),(22,70),(23,71),(24,2
5),(24,39),(25,26),(25,40),(26,27),(26,41),(27,28),(28,43),(29,30),(29,44),(30,31),(30
,45),(31,32),(31,46),(32,33),(33,48),(34,35),(34,49),(35,36),(35,50),(36,37),(36,51),(
37,38),(38,53),(39,40),(39,55),(40,41),(41,42),(42,43),(42,56),(43,57),(44,45),(44,61)
,(45,46),(46,47),(47,48),(47,62),(48,63),(49,50),(49,67),(50,51),(51,52),(52,53),(52,6
8),(53,69),(54,55),(54,72),(55,56),(56,57),(57,58),(58,59),(59,73),(60,61),(60,74),(61
,62),(62,63),(63,64),(64,65),(65,75),(66,67),(66,76),(67,68),(68,69),(69,70),(70,71),(
71,77),(72,73),(72,80),(73,74),(74,75),(75,76),(76,77),(77,81),(78,79),(79,80),(80,84)
,(81,82),(81,85),(82,83),(84,85)]
```


-

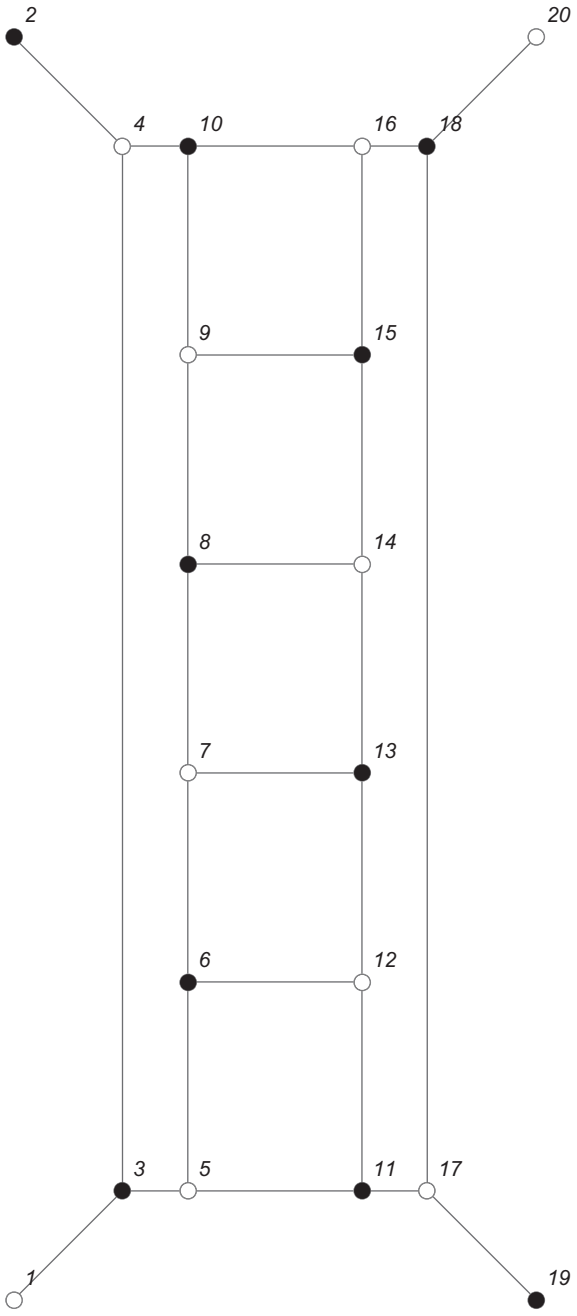
Example embedding coordinates (SAGE 7.2):


```
{0:[-4.9419,-11.8173],1:[-4.9419,-5.2719],2:[-4.9419,-3.09],3:[-4.9419,3.4554],4:[-
4.9419,5.6372],5:[-4.9419,12.1827],6:[-3.4419,-11.8173],7:[-3.4419,-10.5082],8:[-
3.4419,-8.5446],9:[-3.4419,-7.89],10:[-3.4419,-6.5809],11:[-3.4419,-5.2719],12:[-
3.4419,-3.09],13:[-3.4419,-1.7809],14:[-3.4419,0.1827],15:[-3.4419,0.8371],16:[-
3.4419,2.1463],17:[-3.4419,3.4554],18:[-3.4419,5.6372],19:[-3.4419,6.9462],20:[-
3.4419,8.91],21:[-3.4419,9.5645],22:[-3.4419,10.8736],23:[-3.4419,12.1827],24:[-
2.1086,-10.5082],25:[-2.1086,-9.8537],26:[-2.1086,-9.1991],27:[-2.1086,-8.5446],28:[-
2.1086,-7.89],29:[-2.1086,-1.7809],30:[-2.1086,-1.1264],31:[-2.1086,-0.4719],32:[-
2.1086,0.1827],33:[-2.1086,0.8371],34:[-2.1086,6.9462],35:[-2.1086,7.6009],36:[-
2.1086,8.2553],37:[-2.1086,8.91],38:[-2.1086,9.5645],39:[-0.7752,-10.5082],40:[-
0.7752,-9.8537],41:[-0.7752,-9.1991],42:[-0.7752,-8.5446],43:[-0.7752,-7.89],44:[-
0.7752,-1.7809],45:[-0.7752,-1.1264],46:[-0.7752,-0.4719],47:[-0.7752,0.1827],48:[-
0.7752,0.8371],49:[-0.7752,6.9462],50:[-0.7752,7.6009],51:[-0.7752,8.2553],52:[-
0.7752,8.91],53:[-0.7752,9.5645],54:[0.5581,-11.8173],55:[0.5581,-
10.5082],56:[0.5581,-8.5446],57:[0.5581,-7.89],58:[0.5581,-6.5809],59:[0.5581,-
5.2719],60:[0.5581,-3.09],61:[0.5581,-
1.7809],62:[0.5581,0.1827],63:[0.5581,0.8371],64:[0.5581,2.1463],65:[0.5581,3.4554],66
:[0.5581,5.6372],67:[0.5581,6.9462],68:[0.5581,8.91],69:[0.5581,9.5645],70:[0.5581,10.
8736],71:[0.5581,12.1827],72:[2.0581,-11.8173],73:[2.0581,-5.2719],74:[2.0581,-
3.09],75:[2.0581,3.4554],76:[2.0581,5.6372],77:[2.0581,12.1827],78:[13.5581,-
31.3173],79:[13.5581,-27.3173],80:[13.5581,-
8.3173],81:[13.5581,8.6827],82:[13.5581,27.6827],83:[13.5581,31.6827],84:[15.5581,-
5.8173],85:[15.5581,6.1827]}
```

Canonical vertex $(k = 2)$-coloring ::


```
{1->colorOne,2->colorTwo,3->colorOne,4->colorTwo,5->colorOne,6->colorTwo,7-
>colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13-
>colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorTwo,18->colorOne,19-
>colorTwo,20->colorOne,21->colorTwo,22->colorOne,23->colorTwo,24->colorOne,25-
>colorTwo,26->colorOne,27->colorTwo,28->colorOne,29->colorTwo,30->colorTwo,31-
>colorOne,32->colorTwo,33->colorOne,34->colorTwo,35->colorTwo,36->colorOne,37-
>colorTwo,38->colorOne,39->colorTwo,40->colorOne,41->colorTwo,42->colorOne,43-
>colorTwo,44->colorOne,45->colorOne,46->colorTwo,47->colorOne,48->colorTwo,49-
>colorOne,50->colorOne,51->colorTwo,52->colorOne,53->colorTwo,54->colorOne,55-
>colorOne,56->colorTwo,57->colorOne,58->colorTwo,59->colorOne,60->colorTwo,61-
>colorOne,62->colorTwo,63->colorOne,64->colorTwo,65->colorOne,66->colorTwo,67-
>colorOne,68->colorTwo,69->colorOne,70->colorTwo,71->colorOne,72->colorTwo,73-
>colorTwo,74->colorOne,75->colorTwo,76->colorOne,77->colorTwo,78->colorOne,79-
>colorOne,80->colorTwo,81->colorOne,82->colorTwo,83->colorOne,84->colorTwo,85-
>colorTwo,86->colorOne}
```

8.22 (Figure 4.8.a) gadget (specifying (z = 1) blocks)



Graph Properties ::

--

Number of Vertices: ‘20’

Number of Edges: ‘26’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->3,2<->4,3<->4,3<->5,4<->10,5<->6,5<->11,6<->7,6<->12,7<->8,7<->13,8<->9,8<->14,9<->10,9<->15,10<->16,11<->12,11<->17,12<->13,13<->14,14<->15,15<->16,16<->18,17<->18,17<->19,18<->20}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-6.,-14.5},2->{-6.,14.5},3->{-3.5,-12.},4->{-3.5,12.},5->{-2.,-12.},6->{-2.,-7.2},7->{-2.,-2.4},8->{-2.,2.4},9->{-2.,7.2},10->{-2.,12.},11->{2.,-12.},12->{2.,-7.2},13->{2.,-2.4},14->{2.,2.4},15->{2.,7.2},16->{2.,12.},17->{3.5,-12.},18->{3.5,12.},19->{6.,-14.5},20->{6.,14.5}}

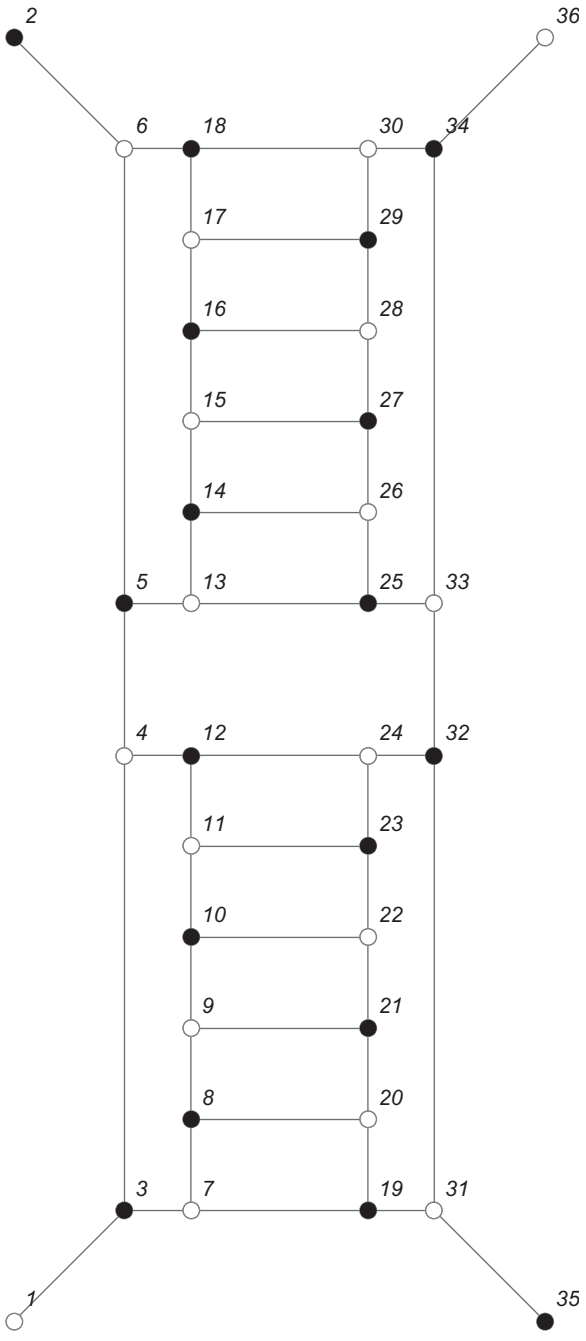
Edge list (SAGE 7.2):
[(0,2),(1,3),(2,3),(2,4),(3,9),(4,5),(4,10),(5,6),(5,11),(6,7),(6,12),(7,8),(7,13),(8,9),(8,14),(9,15),(10,11),(10,16),(11,12),(12,13),(13,14),(14,15),(15,17),(16,17),(16,18),(17,19)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-6.,-14.5],1:[-6.,14.5],2:[-3.5,-12.],3:[-3.5,12.],4:[-2.,-12.],5:[-2.,-7.2],6:[-2.,-2.4],7:[-2.,2.4],8:[-2.,7.2],9:[-2.,12.],10:[2.,-12.],11:[2.,-7.2],12:[2.,-2.4],13:[2.,2.4],14:[2.,7.2],15:[2.,12.],16:[3.5,-12.],17:[3.5,12.],18:[6.,-14.5],19:[6.,14.5]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorTwo,2->colorOne,3->colorOne,4->colorTwo,5->colorTwo,6->colorOne,7->colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorOne,12->colorTwo,13->colorOne,14->colorTwo,15->colorOne,16->colorTwo,17->colorTwo,18->colorOne,19->colorOne,20->colorTwo}

8.23 (Figure 4.8.a) gadget (specifying (z = 2) blocks)



Graph Properties ::

--

Number of Vertices: ‘’

Number of Edges: ‘’

Minimum vertex degree $\delta(G)$: ‘’

Maximum vertex degree $\Delta(G)$: ‘’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘’

Output of SAGE 7.2’s ‘girth()’ function: ‘’

Output for the ‘igraph’ R package ‘girth()’ function: ‘’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->3,2<->6,3<->4,3<->7,4<->5,4<->12,5<->6,5<->13,6<->18,7<->8,7<->19,8<->9,8<->20,9<->10,9<->21,10<->11,10<->22,11<->12,11<->23,12<->24,13<->14,13<->25,14<->15,14<->26,15<->16,15<->27,16<->17,16<->28,17<->18,17<->29,18<->30,19<->20,19<->31,20<->21,21<->22,22<->23,23<->24,24<->32,25<->26,25<->33,26<->27,27<->28,28<->29,29<->30,30<->34,31<->32,31<->35,32<->33,33<->34,34<->36}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-6.,-14.5},2->{-6.,14.5},3->{-3.5,-12.},4->{-3.5,-1.7142},5->{-3.5,1.7143},6->{-3.5,12.},7->{-2.,-12.},8->{-2.,-9.9429},9->{-2.,-7.8857},10->{-2.,-5.8285},11->{-2.,-3.7714},12->{-2.,-1.7142},13->{-2.,1.7143},14->{-2.,3.7715},15->{-2.,5.8285},16->{-2.,7.8855},17->{-2.,9.9428},18->{-2.,12.},19->{2.,-12.},20->{2.,-9.9429},21->{2.,-7.8857},22->{2.,-5.8285},23->{2.,-3.7714},24->{2.,-1.7142},25->{2.,1.7143},26->{2.,3.7715},27->{2.,5.8285},28->{2.,7.8855},29->{2.,9.9428},30->{2.,12.},31->{3.5,-12.},32->{3.5,-1.7142},33->{3.5,1.7143},34->{3.5,12.},35->{6.,-14.5},36->{6.,14.5}}

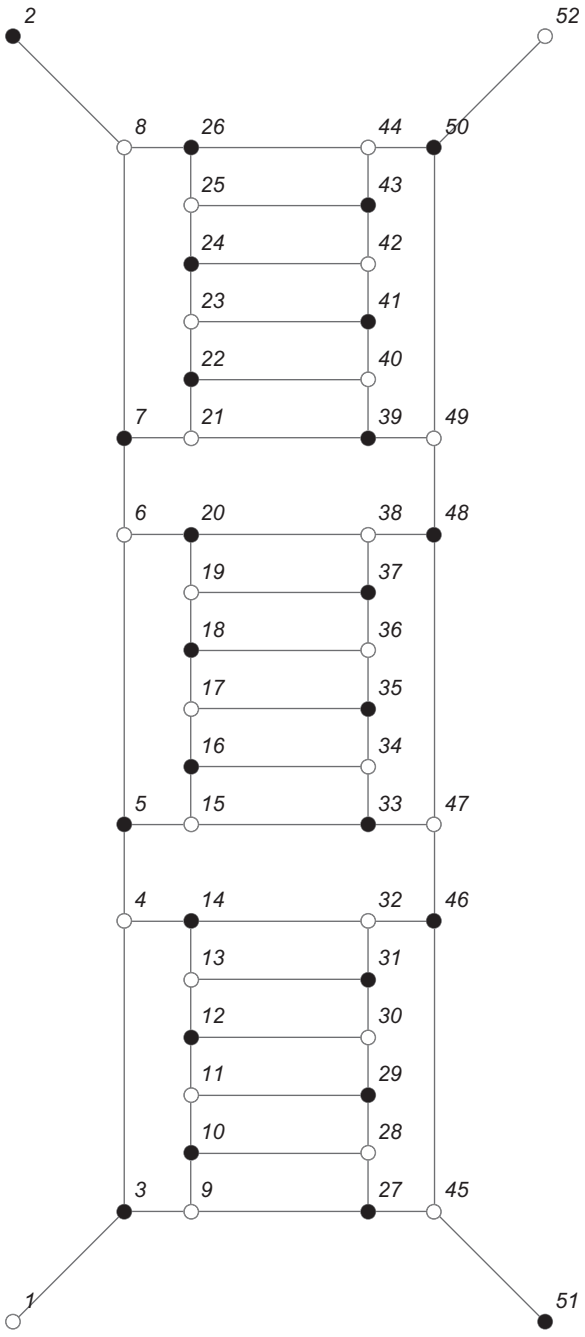
Edge list (SAGE 7.2):
[(0,2),(1,5),(2,3),(2,6),(3,4),(3,11),(4,5),(4,12),(5,17),(6,7),(6,18),(7,8),(7,19),(8,9),(8,20),(9,10),(9,21),(10,11),(10,22),(11,23),(12,13),(12,24),(13,14),(13,25),(14,15),(14,26),(15,16),(15,27),(16,17),(16,28),(17,29),(18,19),(18,30),(19,20),(20,21),(21,22),(22,23),(23,31),(24,25),(24,32),(25,26),(26,27),(27,28),(28,29),(29,33),(30,31),(30,34),(31,32),(32,33),(33,35)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-6.,-14.5],1:[-6.,14.5],2:[-3.5,-12.],3:[-3.5,-1.7142],4:[-3.5,1.7143],5:[-3.5,12.],6:[-2.,-12.],7:[-2.,-9.9429],8:[-2.,-7.8857],9:[-2.,-5.8285],10:[-2.,-3.7714],11:[-2.,-1.7142],12:[-2.,1.7143],13:[-2.,3.7715],14:[-2.,5.8285],15:[-2.,7.8855],16:[-2.,9.9428],17:[-2.,12.],18:[2.,-12.],19:[2.,-9.9429],20:[2.,-7.8857],21:[2.,-5.8285],22:[2.,-3.7714],23:[2.,-1.7142],24:[2.,1.7143],25:[2.,3.7715],26:[2.,5.8285],27:[2.,7.8855],28:[2.,9.9428],29:[2.,12.],30:[3.5,-12.],31:[3.5,-1.7142],32:[3.5,1.7143],33:[3.5,12.],34:[6.,-14.5],35:[6.,14.5]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorTwo,2->colorOne,3->colorOne,4->colorTwo,5->colorOne,6->colorTwo,7->colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13->colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorTwo,18->colorOne,19->colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25->colorOne,26->colorTwo,27->colorOne,28->colorTwo,29->colorOne,30->colorTwo,31->colorTwo,32->colorOne,33->colorTwo,34->colorOne,35->colorOne,36->colorTwo}

8.24 (Figure 4.8.a) gadget (specifying (z = 3) blocks)



Graph Properties ::

--

Number of Vertices: ‘52’

Number of Edges: ‘74’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->3,2<->8,3<->4,3<->9,4<->5,4<->14,5<->6,5<->15,6<->7,6<->20,7<->8,7<->21,8<->26,9<->10,9<->27,10<->11,10<->28,11<->12,11<->29,12<->13,12<->30,13<->14,13<->31,14<->32,15<->16,15<->33,16<->17,16<->34,17<->18,17<->35,18<->19,18<->36,19<->20,19<->37,20<->38,21<->22,21<->39,22<->23,22<->40,23<->24,23<->41,24<->25,24<->42,25<->26,25<->43,26<->44,27<->28,27<->45,28<->29,29<->30,30<->31,31<->32,32<->46,33<->34,33<->47,34<->35,35<->36,36<->37,37<->38,38<->48,39<->40,39<->49,40<->41,41<->42,42<->43,43<->44,44<->50,45<->46,45<->51,46<->47,47<->48,48<->49,49<->50,50<->52}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-6.,-14.5},2->{-6.,14.5},3->{-3.5,-12.},4->{-3.5,-5.4545},5->{-3.5,-3.2727},6->{-3.5,3.2727},7->{-3.5,5.4546},8->{-3.5,12.},9->{-2.,-12.},10->{-2.,-10.6909},11->{-2.,-9.3818},12->{-2.,-8.0727},13->{-2.,-6.7636},14->{-2.,-5.4545},15->{-2.,-3.2727},16->{-2.,-1.9636},17->{-2.,-0.6546},18->{-2.,0.6544},19->{-2.,1.9636},20->{-2.,3.2727},21->{-2.,5.4546},22->{-2.,6.7635},23->{-2.,8.0727},24->{-2.,9.3818},25->{-2.,10.6909},26->{-2.,12.},27->{2.,-12.},28->{2.,-10.6909},29->{2.,-9.3818},30->{2.,-8.0727},31->{2.,-6.7636},32->{2.,-5.4545},33->{2.,-3.2727},34->{2.,-1.9636},35->{2.,-0.6546},36->{2.,0.6544},37->{2.,1.9636},38->{2.,3.2727},39->{2.,5.4546},40->{2.,6.7635},41->{2.,8.0727},42->{2.,9.3818},43->{2.,10.6909},44->{2.,12.},45->{3.5,-12.},46->{3.5,-5.4545},47->{3.5,-3.2727},48->{3.5,3.2727},49->{3.5,5.4546},50->{3.5,12.},51->{6.,-14.5},52->{6.,14.5}}

Edge list (SAGE 7.2):

[(0,2),(1,7),(2,3),(2,8),(3,4),(3,13),(4,5),(4,14),(5,6),(5,19),(6,7),(6,20),(7,25),(8,9),(8,26),(9,10),(9,27),(10,11),(10,28),(11,12),(11,29),(12,13),(12,30),(13,31),(14,15),(14,32),(15,16),(15,33),(16,17),(16,34),(17,18),(17,35),(18,19),(18,36),(19,37),(20,21),(20,38),(21,22),(21,39),(22,23),(22,40),(23,24),(23,41),(24,25),(24,42),(25,43),(26,27),(26,44),(27,28),(28,29),(29,30),(30,31),(31,45),(32,33),(32,46),(33,34),(34,35),(35,36),(36,37),(37,47),(38,39),(38,48),(39,40),(40,41),(41,42),(42,43),(43,49),(44,45),(44,50),(45,46),(46,47),(47,48),(48,49),(49,51)]

-

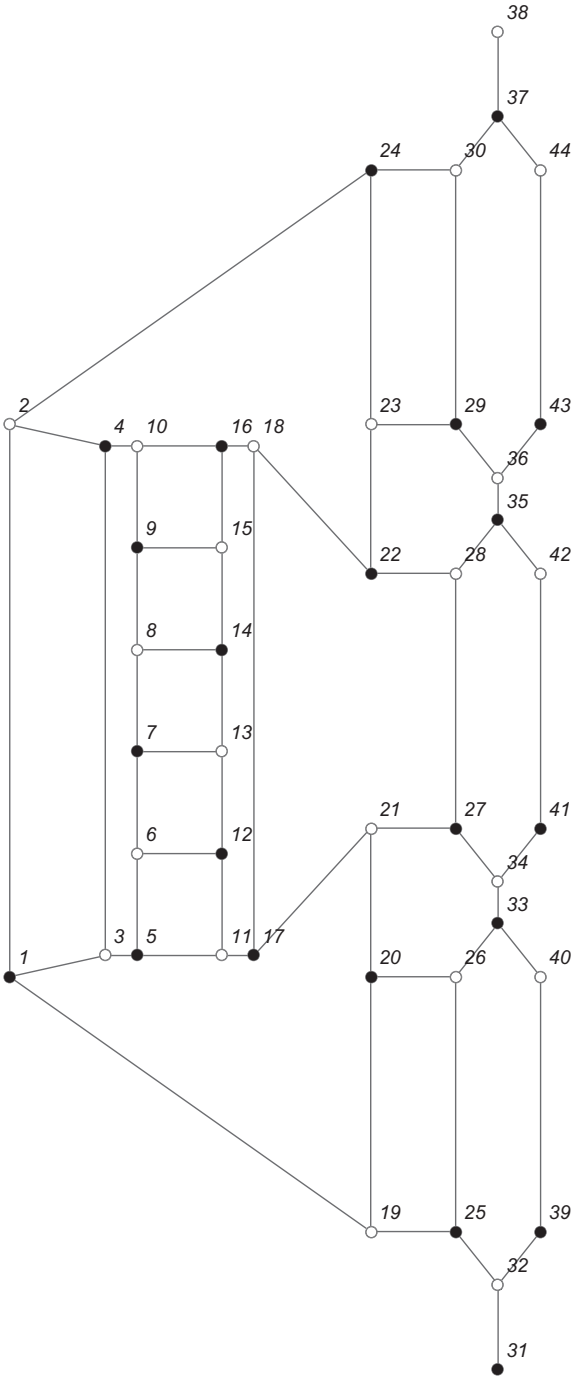
Example embedding coordinates (SAGE 7.2):

{0:[-6.,-14.5],1:[-6.,14.5],2:[-3.5,-12.],3:[-3.5,-5.4545],4:[-3.5,-3.2727],5:[-3.5,3.2727],6:[-3.5,5.4546],7:[-3.5,12.],8:[-2.,-12.],9:[-2.,-10.6909],10:[-2.,-9.3818],11:[-2.,-8.0727],12:[-2.,-6.7636],13:[-2.,-5.4545],14:[-2.,-3.2727],15:[-2.,-1.9636],16:[-2.,-0.6546],17:[-2.,0.6544],18:[-2.,1.9636],19:[-2.,3.2727],20:[-2.,5.4546],21:[-2.,6.7635],22:[-2.,8.0727],23:[-2.,9.3818],24:[-2.,10.6909],25:[-2.,12.],26:[2.,-12.],27:[2.,-10.6909],28:[2.,-9.3818],29:[2.,-8.0727],30:[2.,-6.7636],31:[2.,-5.4545],32:[2.,-3.2727],33:[2.,-1.9636],34:[2.,-0.6546],35:[2.,0.6544],36:[2.,1.9636],37:[2.,3.2727],38:[2.,5.4546],39:[2.,6.7635],40:[2.,8.0727],41:[2.,9.3818],42:[2.,10.6909],43:[2.,12.],44:[3.5,-12.],45:[3.5,-5.4545],46:[3.5,-3.2727],47:[3.5,3.2727],48:[3.5,5.4546],49:[3.5,12.],50:[6.,-14.5],51:[6.,14.5]}

Canonical vertex ($k = 2$)-coloring ::

```
{1->colorTwo,3->colorOne,2->colorOne,8->colorTwo,4->colorTwo,9->colorTwo,5-
>colorOne,14->colorOne,6->colorTwo,15->colorTwo,7->colorOne,20->colorOne,21-
>colorTwo,26->colorOne,10->colorOne,27->colorOne,11->colorTwo,28->colorTwo,12-
>colorOne,29->colorOne,13->colorTwo,30->colorTwo,31->colorOne,32->colorTwo,16-
>colorOne,33->colorOne,17->colorTwo,34->colorTwo,18->colorOne,35->colorOne,19-
>colorTwo,36->colorTwo,37->colorOne,38->colorTwo,22->colorOne,39->colorOne,23-
>colorTwo,40->colorTwo,24->colorOne,41->colorOne,25->colorTwo,42->colorTwo,43-
>colorOne,44->colorTwo,45->colorTwo,46->colorOne,47->colorTwo,48->colorOne,49-
>colorTwo,50->colorOne,51->colorOne,52->colorTwo}
```

8.25 (Figure 4.8.b) gadget (specifying (z = 1) blocks)



Graph Properties ::

--

Number of Vertices: ‘44’

Number of Edges: ‘61’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->2,1<->3,1<->19,2<->4,2<->24,3<->4,3<->5,4<->10,5<->6,5<->11,6<->7,6<->12,7<->8,7<->13,8<->9,8<->14,9<->10,9<->15,10<->16,11<->12,11<->17,12<->13,13<->14,14<->15,15<->16,16<->18,17<->18,17<->21,18<->22,19<->20,19<->25,20<->21,20<->26,21<->27,22<->23,22<->28,23<->24,23<->29,24<->30,25<->26,25<->32,26<->33,27<->28,27<->34,28<->35,29<->30,29<->36,30<->37,31<->32,32<->39,33<->34,33<->40,34<->41,35<->36,35<->42,36<->43,37<->38,37<->44,39<->40,41<->42,43<->44}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-15.6818,-13.},2->{-15.6818,13.},3->{-11.1818,-12.},4->{-11.1818,12.},5->{-9.6818,-12.},6->{-9.6818,-7.2},7->{-9.6818,-2.4},8->{-9.6818,2.4},9->{-9.6818,7.2},10->{-9.6818,12.},11->{-5.6818,-12.},12->{-5.6818,-7.2},13->{-5.6818,-2.4},14->{-5.6818,2.4},15->{-5.6818,7.2},16->{-5.6818,12.},17->{-4.1818,-12.},18->{-4.1818,12.},19->{1.3181,-25.},20->{1.3181,-13.},21->{1.3181,-6.},22->{1.3181,6.},23->{1.3181,13.},24->{1.3181,25.},25->{5.3182,-25.},26->{5.3182,-13.},27->{5.3182,-6.},28->{5.3182,6.},29->{5.3182,13.},30->{5.3182,25.},31->{7.3182,-31.5},32->{7.3182,-27.5},33->{7.3182,-10.5},34->{7.3182,-8.5},35->{7.3182,8.5},36->{7.3182,10.5},37->{7.3182,27.5},38->{7.3182,31.5},39->{9.3182,-25.},40->{9.3182,-13.},41->{9.3182,-6.},42->{9.3182,6.},43->{9.3182,13.},44->{9.3182,25.}}

Edge list (SAGE 7.2):

[(0,1),(0,2),(0,18),(1,3),(1,23),(2,3),(2,4),(3,9),(4,5),(4,10),(5,6),(5,11),(6,7),(6,12),(7,8),(7,13),(8,9),(8,14),(9,15),(10,11),(10,16),(11,12),(12,13),(13,14),(14,15),(15,17),(16,17),(16,20),(17,21),(18,19),(18,24),(19,20),(19,25),(20,26),(21,22),(21,27),(22,23),(22,28),(23,29),(24,25),(24,31),(25,32),(26,27),(26,33),(27,34),(28,29),(28,35),(29,36),(30,31),(31,38),(32,33),(32,39),(33,40),(34,35),(34,41),(35,42),(36,37),(36,43),(38,39),(40,41),(42,43)]

-

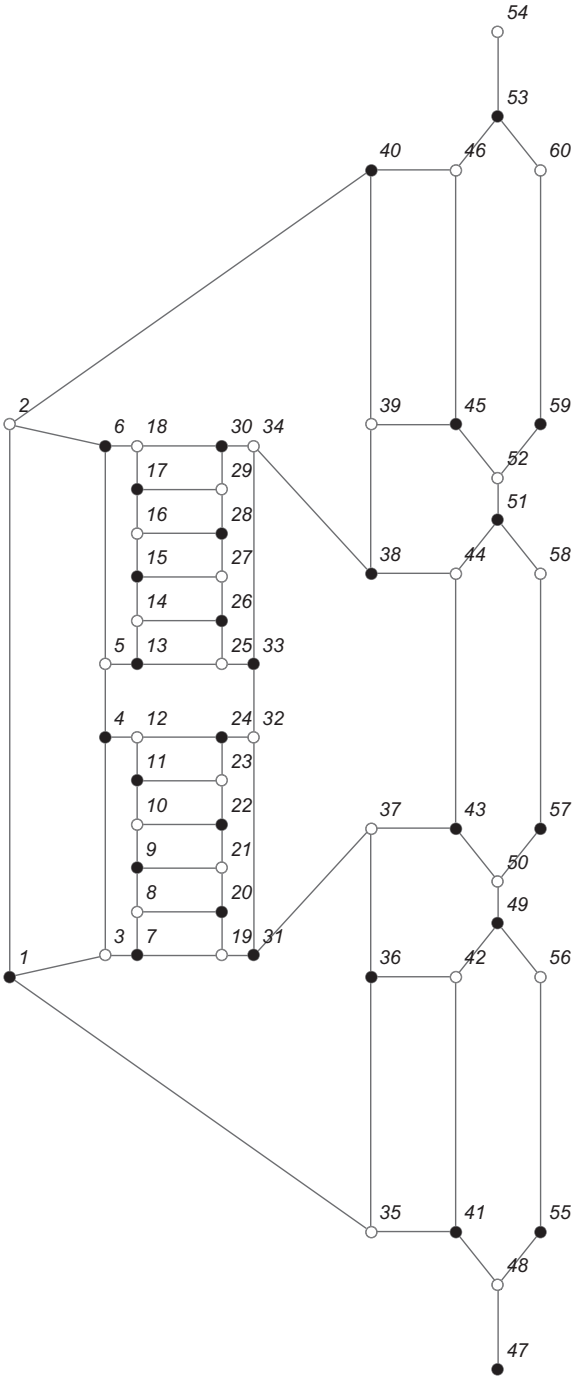
Example embedding coordinates (SAGE 7.2):

{0:[-15.6818,-13.],1:[-15.6818,13.],2:[-11.1818,-12.],3:[-11.1818,12.],4:[-9.6818,-12.],5:[-9.6818,-7.2],6:[-9.6818,-2.4],7:[-9.6818,2.4],8:[-9.6818,7.2],9:[-9.6818,12.],10:[-5.6818,-12.],11:[-5.6818,-7.2],12:[-5.6818,-2.4],13:[-5.6818,2.4],14:[-5.6818,7.2],15:[-5.6818,12.],16:[-4.1818,-12.],17:[-4.1818,12.],18:[1.3181,-25.],19:[1.3181,-13.],20:[1.3181,-6.],21:[1.3181,6.],22:[1.3181,13.],23:[1.3181,25.],24:[5.3182,-25.],25:[5.3182,-13.],26:[5.3182,-6.],27:[5.3182,6.],28:[5.3182,13.],29:[5.3182,25.],30:[7.3182,-31.5],31:[7.3182,-27.5],32:[7.3182,-10.5],33:[7.3182,-8.5],34:[7.3182,8.5],35:[7.3182,10.5],36:[7.3182,27.5],37:[7.3182,31.5],38:[9.3182,-25.],39:[9.3182,-13.],40:[9.3182,-6.],41:[9.3182,6.],42:[9.3182,13.],43:[9.3182,25.]}

Canonical vertex $(k = 2)$-coloring ::

{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorOne,6->colorTwo,7-
>colorOne,8->colorTwo,9->colorOne,10->colorTwo,11->colorTwo,12->colorOne,13-
>colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorOne,18->colorTwo,19-
>colorTwo,20->colorOne,21->colorTwo,22->colorOne,23->colorTwo,24->colorOne,25-
>colorOne,26->colorTwo,27->colorOne,28->colorTwo,29->colorOne,30->colorTwo,31-
>colorOne,32->colorTwo,33->colorOne,34->colorTwo,35->colorOne,36->colorTwo,37-
>colorOne,38->colorTwo,39->colorOne,40->colorTwo,41->colorOne,42->colorTwo,43-
>colorOne,44->colorTwo}

8.26 (Figure 4.8.b) gadget (specifying (z = 2) blocks)



Graph Properties ::

--

Number of Vertices: ‘60’

Number of Edges: ‘85’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->2,1<->3,1<->35,2<->6,2<->40,3<->4,3<->7,4<->5,4<->12,5<->6,5<->13,6<->18,7<->8,7<->19,8<->9,8<->20,9<->10,9<->21,10<->11,10<->22,11<->12,11<->23,12<->24,13<->14,13<->25,14<->15,14<->26,15<->16,15<->27,16<->17,16<->28,17<->18,17<->29,18<->30,19<->20,19<->31,20<->21,21<->22,22<->23,23<->24,24<->32,25<->26,25<->33,26<->27,27<->28,28<->29,29<->30,30<->34,31<->32,31<->37,32<->33,33<->34,34<->38,35<->36,35<->41,36<->37,36<->42,37<->43,38<->39,38<->44,39<->40,39<->45,40<->46,41<->42,41<->48,42<->49,43<->44,43<->50,44<->51,45<->46,45<->52,46<->53,47<->48,48<->55,49<->50,49<->56,50<->57,51<->52,51<->58,52<->59,53<->54,53<->60,55<->56,57<->58,59<->60}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-13.6333,-13.},2->{-13.6333,13.},3->{-9.1333,-12.},4->{-9.1333,-1.7142},5->{-9.1333,1.7143},6->{-9.1333,12.},7->{-7.6333,-12.},8->{-7.6333,-9.9429},9->{-7.6333,-7.8857},10->{-7.6333,-5.8285},11->{-7.6333,-3.7714},12->{-7.6333,-1.7142},13->{-7.6333,1.7143},14->{-7.6333,3.7715},15->{-7.6333,5.8285},16->{-7.6333,7.8855},17->{-7.6333,9.9428},18->{-7.6333,12.},19->{-3.6333,-12.},20->{-3.6333,-9.9429},21->{-3.6333,-7.8857},22->{-3.6333,-5.8285},23->{-3.6333,-3.7714},24->{-3.6333,-1.7142},25->{-3.6333,1.7143},26->{-3.6333,3.7715},27->{-3.6333,5.8285},28->{-3.6333,7.8855},29->{-3.6333,9.9428},30->{-3.6333,12.},31->{-2.1333,-12.},32->{-2.1333,-1.7142},33->{-2.1333,1.7143},34->{-2.1333,12.},35->{3.3666,-25.},36->{3.3666,-13.},37->{3.3666,-6.},38->{3.3666,6.},39->{3.3666,13.},40->{3.3666,25.},41->{7.3667,-25.},42->{7.3667,-13.},43->{7.3667,-6.},44->{7.3667,6.},45->{7.3667,13.},46->{7.3667,25.},47->{9.3667,-31.5},48->{9.3667,-27.5},49->{9.3667,-10.5},50->{9.3667,-8.5},51->{9.3667,8.5},52->{9.3667,10.5},53->{9.3667,27.5},54->{9.3667,31.5},55->{11.3667,-25.},56->{11.3667,-13.},57->{11.3667,-6.},58->{11.3667,6.},59->{11.3667,13.},60->{11.3667,25.}}

Edge list (SAGE 7.2):

[(0,1),(0,2),(0,34),(1,5),(1,39),(2,3),(2,6),(3,4),(3,11),(4,5),(4,12),(5,17),(6,7),(6,18),(7,8),(7,19),(8,9),(8,20),(9,10),(9,21),(10,11),(10,22),(11,23),(12,13),(12,24),(13,14),(13,25),(14,15),(14,26),(15,16),(15,27),(16,17),(16,28),(17,29),(18,19),(18,30),(19,20),(20,21),(21,22),(22,23),(23,31),(24,25),(24,32),(25,26),(26,27),(27,28),(28,29),(29,33),(30,31),(30,36),(31,32),(32,33),(33,37),(34,35),(34,40),(35,36),(35,41),(36,42),(37,38),(37,43),(38,39),(38,44),(39,45),(40,41),(40,47),(41,48),(42,43),(42,49),(43,50),(44,45),(44,51),(45,52),(46,47),(47,54),(48,49),(48,55),(49,56),(50,51),(50,57),(51,58),(52,53),(52,59),(54,55),(56,57),(58,59)]

-

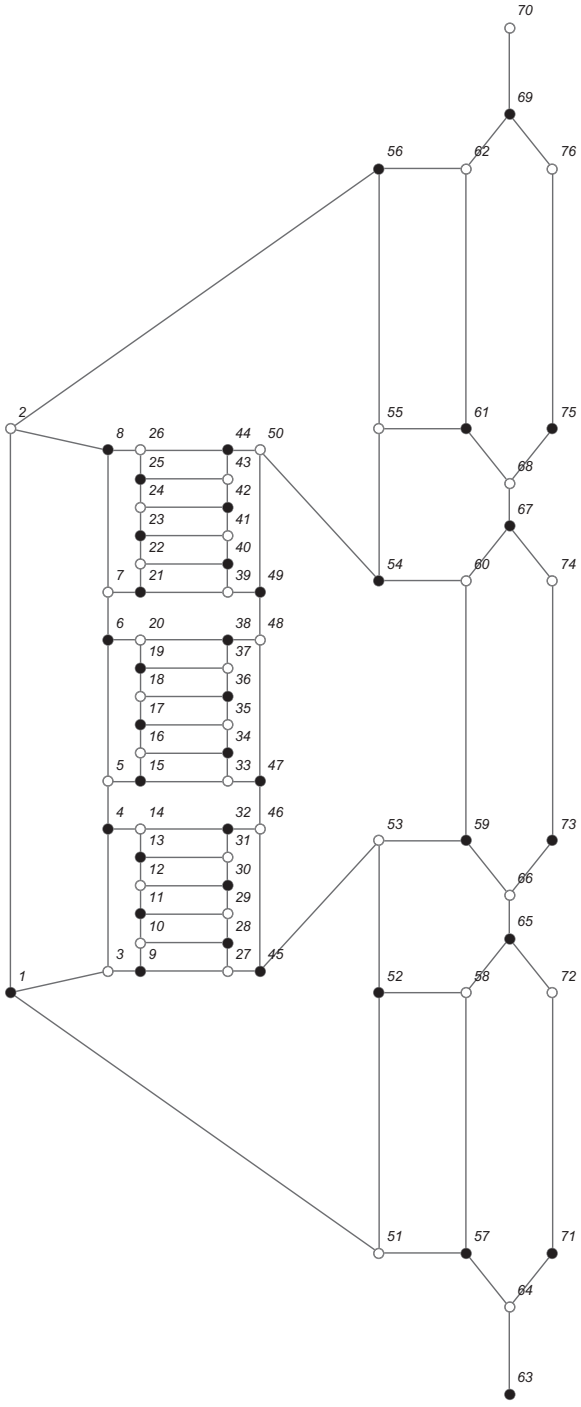
Example embedding coordinates (SAGE 7.2):

{0:[-13.6333,-13.],1:[-13.6333,13.],2:[-9.1333,-12.],3:[-9.1333,-1.7142],4:[-
9.1333,1.7143],5:[-9.1333,12.],6:[-7.6333,-12.],7:[-7.6333,-9.9429],8:[-7.6333,-
7.8857],9:[-7.6333,-5.8285],10:[-7.6333,-3.7714],11:[-7.6333,-1.7142],12:[-
7.6333,1.7143],13:[-7.6333,3.7715],14:[-7.6333,5.8285],15:[-7.6333,7.8855],16:[-
7.6333,9.9428],17:[-7.6333,12.],18:[-3.6333,-12.],19:[-3.6333,-9.9429],20:[-3.6333,-
7.8857],21:[-3.6333,-5.8285],22:[-3.6333,-3.7714],23:[-3.6333,-1.7142],24:[-
3.6333,1.7143],25:[-3.6333,3.7715],26:[-3.6333,5.8285],27:[-3.6333,7.8855],28:[-
3.6333,9.9428],29:[-3.6333,12.],30:[-2.1333,-12.],31:[-2.1333,-1.7142],32:[-
2.1333,1.7143],33:[-2.1333,12.],34:[3.3666,-25.],35:[3.3666,-13.],36:[3.3666,-
6.],37:[3.3666,6.],38:[3.3666,13.],39:[3.3666,25.],40:[7.3667,-25.],41:[7.3667,-
13.],42:[7.3667,-6.],43:[7.3667,6.],44:[7.3667,13.],45:[7.3667,25.],46:[9.3667,-
31.5],47:[9.3667,-27.5],48:[9.3667,-10.5],49:[9.3667,-
8.5],50:[9.3667,8.5],51:[9.3667,10.5],52:[9.3667,27.5],53:[9.3667,31.5],54:[11.3667,-
25.],55:[11.3667,-13.],56:[11.3667,-
6.],57:[11.3667,6.],58:[11.3667,13.],59:[11.3667,25.]}

Canonical vertex ($k = 2$)-coloring ::

{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorTwo,6->colorOne,7-
>colorOne,8->colorTwo,9->colorOne,10->colorTwo,11->colorOne,12->colorTwo,13-
>colorOne,14->colorTwo,15->colorOne,16->colorTwo,17->colorOne,18->colorTwo,19-
>colorTwo,20->colorOne,21->colorTwo,22->colorOne,23->colorTwo,24->colorOne,25-
>colorTwo,26->colorOne,27->colorTwo,28->colorOne,29->colorTwo,30->colorOne,31-
>colorOne,32->colorTwo,33->colorOne,34->colorTwo,35->colorTwo,36->colorOne,37-
>colorTwo,38->colorOne,39->colorTwo,40->colorOne,41->colorOne,42->colorTwo,43-
>colorOne,44->colorTwo,45->colorOne,46->colorTwo,47->colorOne,48->colorTwo,49-
>colorOne,50->colorTwo,51->colorOne,52->colorTwo,53->colorOne,54->colorTwo,55-
>colorOne,56->colorTwo,57->colorOne,58->colorTwo,59->colorOne,60->colorTwo}

8.27 (Figure 4.8.b) gadget (specifying (z = 3) blocks)



Graph Properties ::

--

Number of Vertices: ‘76’

Number of Edges: ‘109’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->2,1<->3,1<->51,2<->8,2<->56,3<->4,3<->9,4<->5,4<->14,5<->6,5<->15,6<->7,6<->20,7<->8,7<->21,8<->26,9<->10,9<->27,10<->11,10<->28,11<->12,11<->29,12<->13,12<->30,13<->14,13<->31,14<->32,15<->16,15<->33,16<->17,16<->34,17<->18,17<->35,18<->19,18<->36,19<->20,19<->37,20<->38,21<->22,21<->39,22<->23,22<->40,23<->24,23<->41,24<->25,24<->42,25<->26,25<->43,26<->44,27<->28,27<->45,28<->29,29<->30,30<->31,31<->32,32<->46,33<->34,33<->47,34<->35,35<->36,36<->37,37<->38,38<->48,39<->40,39<->49,40<->41,41<->42,42<->43,43<->44,44<->50,45<->46,45<->53,46<->47,47<->48,48<->49,49<->50,50<->54,51<->52,51<->57,52<->53,52<->58,53<->59,54<->55,54<->60,55<->56,55<->61,56<->62,57<->58,57<->64,58<->65,59<->60,59<->66,60<->67,61<->62,61<->68,62<->69,63<->64,64<->71,65<->66,65<->72,66<->73,67<->68,67<->74,68<->75,69<->70,69<->76,71<->72,73<->74,75<->76}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-12.4474,-13.},2->{-12.4474,13.},3->{-7.9474,-12.},4->{-7.9474,-5.4545},5->{-7.9474,-3.2727},6->{-7.9474,3.2727},7->{-7.9474,5.4546},8->{-7.9474,12.},9->{-6.4474,-12.},10->{-6.4474,-10.6909},11->{-6.4474,-9.3818},12->{-6.4474,-8.0727},13->{-6.4474,-6.7636},14->{-6.4474,-5.4545},15->{-6.4474,-3.2727},16->{-6.4474,-1.9636},17->{-6.4474,-0.6546},18->{-6.4474,0.6544},19->{-6.4474,1.9636},20->{-6.4474,3.2727},21->{-6.4474,5.4546},22->{-6.4474,6.7635},23->{-6.4474,8.0727},24->{-6.4474,9.3818},25->{-6.4474,10.6909},26->{-6.4474,12.},27->{-2.4474,-12.},28->{-2.4474,-10.6909},29->{-2.4474,-9.3818},30->{-2.4474,-8.0727},31->{-2.4474,-6.7636},32->{-2.4474,-5.4545},33->{-2.4474,-3.2727},34->{-2.4474,-1.9636},35->{-2.4474,-0.6546},36->{-2.4474,0.6544},37->{-2.4474,1.9636},38->{-2.4474,3.2727},39->{-2.4474,5.4546},40->{-2.4474,6.7635},41->{-2.4474,8.0727},42->{-2.4474,9.3818},43->{-2.4474,10.6909},44->{-2.4474,12.},45->{-0.9474,-12.},46->{-0.9474,-5.4545},47->{-0.9474,-3.2727},48->{-0.9474,3.2727},49->{-0.9474,5.4546},50->{-0.9474,12.},51->{4.5525,-25.},52->{4.5525,-13.},53->{4.5525,-6.},54->{4.5525,6.},55->{4.5525,13.},56->{4.5525,25.},57->{8.5526,-25.},58->{8.5526,-13.},59->{8.5526,-6.},60->{8.5526,6.},61->{8.5526,13.},62->{8.5526,25.},63->{10.5526,-31.5},64->{10.5526,-27.5},65->{10.5526,-10.5},66->{10.5526,-8.5},67->{10.5526,8.5},68->{10.5526,10.5},69->{10.5526,27.5},70->{10.5526,31.5},71->{12.5526,-25.},72->{12.5526,-13.},73->{12.5526,-6.},74->{12.5526,6.},75->{12.5526,13.},76->{12.5526,25.}}

Edge list (SAGE 7.2):

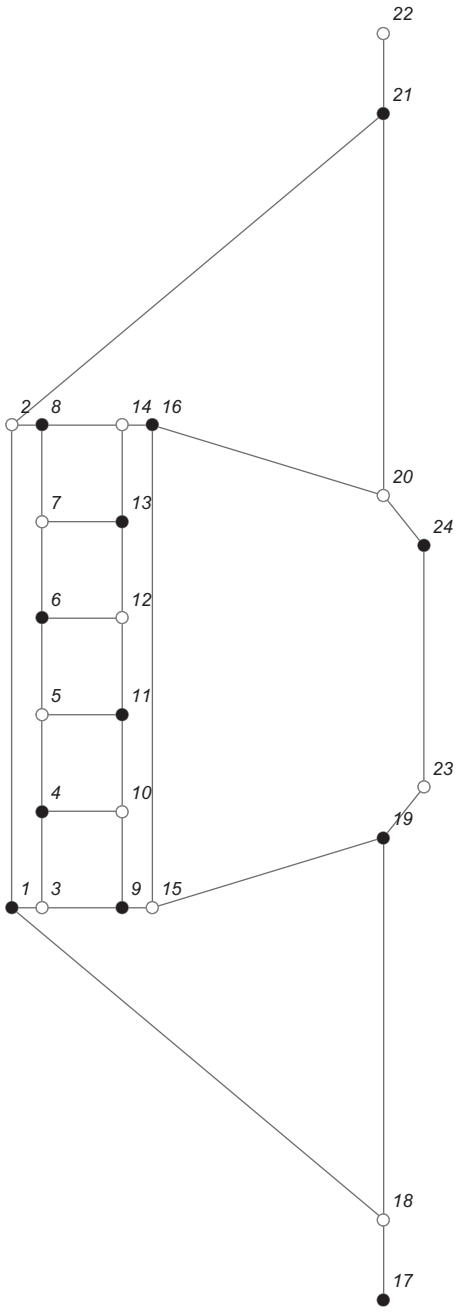
[(0,1),(0,2),(0,50),(1,7),(1,55),(2,3),(2,8),(3,4),(3,13),(4,5),(4,14),(5,6),(5,19),(6,7),(6,20),(7,25),(8,9),(8,26),(9,10),(9,27),(10,11),(10,28),(11,12),(11,29),(12,13),(12,30),(13,31),(14,15),(14,32),(15,16),(15,33),(16,17),(16,34),(17,18),(17,35),(18,19),(18,36),(19,37),(20,21),(20,38),(21,22),(21,39),(22,23),(22,40),(23,24),(23,41),(24,25),(24,42),(25,43),(26,27),(26,44),(27,28),(28,29),(29,30),(30,31),(31,45),(32,33),(32,46),(33,34),(34,35),(35,36),(36,37),(37,47),(38,39),(38,48),(39,40),(40,41),(41,42),(42,43),(43,49),(44,45),(44,52),(45,46),(46,47),(47,48),(48,49),(49,53),(50,51),(50,56),(51,52),(51,57),(52,58),(53,54),(53,59),(54,55),(54,60),(55,61),(56,57),(56,63),(57,64),(58,59),(58,65),(59,66),(60,61),(60,67),(61,68),(62,63),(63,70),(64,65),(64,71),(65,72),(66,67),(66,73),(67,74),(68,69),(68,75),(70,71),(72,73),(74,75)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-12.4474,-13.],1:[-12.4474,13.],2:[-7.9474,-12.],3:[-7.9474,-5.4545],4:[-7.9474,-
3.2727],5:[-7.9474,3.2727],6:[-7.9474,5.4546],7:[-7.9474,12.],8:[-6.4474,-12.],9:[-
6.4474,-10.6909],10:[-6.4474,-9.3818],11:[-6.4474,-8.0727],12:[-6.4474,-6.7636],13:[-
6.4474,-5.4545],14:[-6.4474,-3.2727],15:[-6.4474,-1.9636],16:[-6.4474,-0.6546],17:[-
6.4474,0.6544],18:[-6.4474,1.9636],19:[-6.4474,3.2727],20:[-6.4474,5.4546],21:[-
6.4474,6.7635],22:[-6.4474,8.0727],23:[-6.4474,9.3818],24:[-6.4474,10.6909],25:[-
6.4474,12.],26:[-2.4474,-12.],27:[-2.4474,-10.6909],28:[-2.4474,-9.3818],29:[-2.4474,-
8.0727],30:[-2.4474,-6.7636],31:[-2.4474,-5.4545],32:[-2.4474,-3.2727],33:[-2.4474,-
1.9636],34:[-2.4474,-0.6546],35:[-2.4474,0.6544],36:[-2.4474,1.9636],37:[-
2.4474,3.2727],38:[-2.4474,5.4546],39:[-2.4474,6.7635],40:[-2.4474,8.0727],41:[-
2.4474,9.3818],42:[-2.4474,10.6909],43:[-2.4474,12.],44:[-0.9474,-12.],45:[-0.9474,-
5.4545],46:[-0.9474,-3.2727],47:[-0.9474,3.2727],48:[-0.9474,5.4546],49:[-
0.9474,12.],50:[4.5525,-25.],51:[4.5525,-13.],52:[4.5525,-
6.],53:[4.5525,6.],54:[4.5525,13.],55:[4.5525,25.],56:[8.5526,-25.],57:[8.5526,-
13.],58:[8.5526,-6.],59:[8.5526,6.],60:[8.5526,13.],61:[8.5526,25.],62:[10.5526,-
31.5],63:[10.5526,-27.5],64:[10.5526,-10.5],65:[10.5526,-
8.5],66:[10.5526,8.5],67:[10.5526,10.5],68:[10.5526,27.5],69:[10.5526,31.5],70:[12.552
6,-25.],71:[12.5526,-13.],72:[12.5526,-
6.],73:[12.5526,6.],74:[12.5526,13.],75:[12.5526,25.]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorTwo,6->colorOne,7-
>colorTwo,8->colorOne,9->colorOne,10->colorTwo,11->colorOne,12->colorTwo,13-
>colorOne,14->colorTwo,15->colorOne,16->colorTwo,17->colorOne,18->colorTwo,19-
>colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25-
>colorOne,26->colorTwo,27->colorTwo,28->colorOne,29->colorTwo,30->colorOne,31-
>colorTwo,32->colorOne,33->colorTwo,34->colorOne,35->colorTwo,36->colorOne,37-
>colorTwo,38->colorOne,39->colorTwo,40->colorOne,41->colorTwo,42->colorOne,43-
>colorTwo,44->colorOne,45->colorOne,46->colorTwo,47->colorOne,48->colorTwo,49-
>colorOne,50->colorTwo,51->colorTwo,52->colorOne,53->colorTwo,54->colorOne,55-
>colorTwo,56->colorOne,57->colorOne,58->colorTwo,59->colorOne,60->colorTwo,61-
>colorOne,62->colorTwo,63->colorOne,64->colorTwo,65->colorOne,66->colorTwo,67-
>colorOne,68->colorTwo,69->colorOne,70->colorTwo,71->colorOne,72->colorTwo,73-
>colorOne,74->colorTwo,75->colorOne,76->colorTwo}

8.28 (Figure 4.8.c) gadget (specifying (z = 1) blocks)



Graph Properties ::

--

Number of Vertices: ‘24’

Number of Edges: ‘33’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->2,1<->3,1<->18,2<->8,2<->21,3<->4,3<->9,4<->5,4<->10,5<->6,5<->11,6<->7,6<->12,7<->8,7<->13,8<->14,9<->10,9<->15,10<->11,11<->12,12<->13,13<->14,14<->16,15<->16,15<->19,16<->20,17<->18,18<->19,19<->23,20<->21,20<->24,21<->22,23<->24}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-8.6667,-12.},2->{-8.6667,12.},3->{-7.1667,-12.},4->{-7.1667,-7.2},5->{-7.1667,-2.4},6->{-7.1667,2.4},7->{-7.1667,7.2},8->{-7.1667,12.},9->{-3.1667,-12.},10->{-3.1667,-7.2},11->{-3.1667,-2.4},12->{-3.1667,2.4},13->{-3.1667,7.2},14->{-3.1667,12.},15->{-1.6667,-12.},16->{-1.6667,12.},17->{9.8333,-31.5},18->{9.8333,-27.5},19->{9.8333,-8.5},20->{9.8333,8.5},21->{9.8333,27.5},22->{9.8333,31.5},23->{11.8333,-6.},24->{11.8333,6.}}

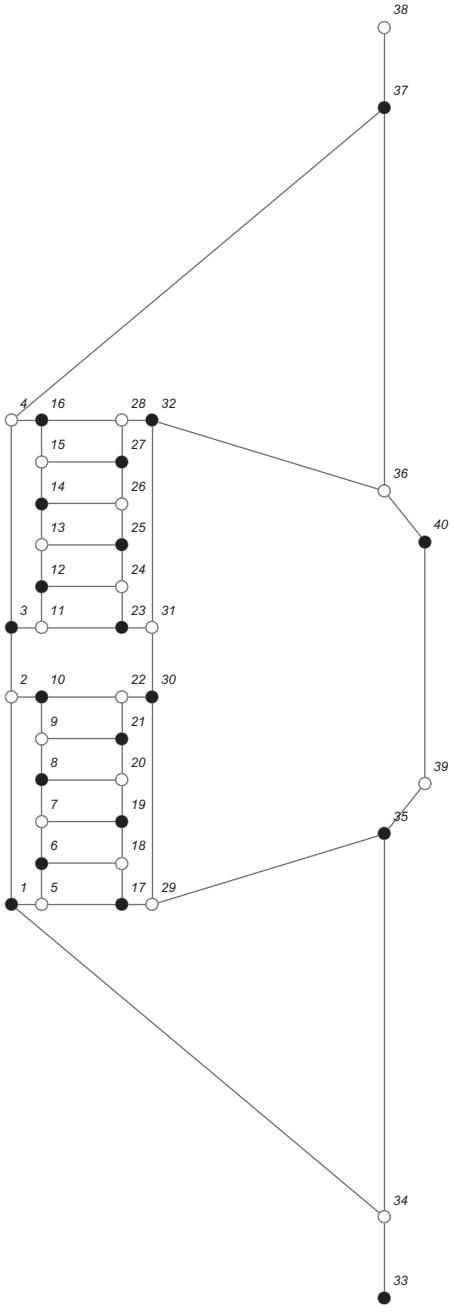
Edge list (SAGE 7.2):
[(0,1),(0,2),(0,17),(1,7),(1,20),(2,3),(2,8),(3,4),(3,9),(4,5),(4,10),(5,6),(5,11),(6,7),(6,12),(7,13),(8,9),(8,14),(9,10),(10,11),(11,12),(12,13),(13,15),(14,15),(14,18),(15,19),(16,17),(17,18),(18,22),(19,20),(19,23),(20,21),(22,23)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-8.6667,-12.],1:[-8.6667,12.],2:[-7.1667,-12.],3:[-7.1667,-7.2],4:[-7.1667,-2.4],5:[-7.1667,2.4],6:[-7.1667,7.2],7:[-7.1667,12.],8:[-3.1667,-12.],9:[-3.1667,-7.2],10:[-3.1667,-2.4],11:[-3.1667,2.4],12:[-3.1667,7.2],13:[-3.1667,12.],14:[-1.6667,-12.],15:[-1.6667,12.],16:[9.8333,-31.5],17:[9.8333,-27.5],18:[9.8333,-8.5],19:[9.8333,8.5],20:[9.8333,27.5],21:[9.8333,31.5],22:[11.8333,-6.],23:[11.8333,6.]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorTwo,6->colorOne,7->colorTwo,8->colorOne,9->colorOne,10->colorTwo,11->colorOne,12->colorTwo,13->colorOne,14->colorTwo,15->colorTwo,16->colorOne,17->colorOne,18->colorTwo,19->colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorTwo,24->colorOne}

8.29 (Figure 4.8.c) gadget (specifying (z = 2) blocks)



Graph Properties ::

--

Number of Vertices: ‘40’

Number of Edges: ‘57’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->2,1<->5,1<->34,2<->3,2<->10,3<->4,3<->11,4<->16,4<->37,5<->6,5<->17,6<->7,6<->18,7<->8,7<->19,8<->9,8<->20,9<->10,9<->21,10<->22,11<->12,11<->23,12<->13,12<->24,13<->14,13<->25,14<->15,14<->26,15<->16,15<->27,16<->28,17<->18,17<->29,18<->19,19<->20,20<->21,21<->22,22<->30,23<->24,23<->31,24<->25,25<->26,26<->27,27<->28,28<->32,29<->30,29<->35,30<->31,31<->32,32<->36,33<->34,34<->35,35<->39,36<->37,36<->40,37<->38,39<->40}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-6.6,-12.},2->{-6.6,-1.7142},3->{-6.6,1.7143},4->{-6.6,12.},5->{-5.1,-12.},6->{-5.1,-9.9429},7->{-5.1,-7.8857},8->{-5.1,-5.8285},9->{-5.1,-3.7714},10->{-5.1,-1.7142},11->{-5.1,1.7143},12->{-5.1,3.7715},13->{-5.1,5.8285},14->{-5.1,7.8855},15->{-5.1,9.9428},16->{-5.1,12.},17->{-1.1,-12.},18->{-1.1,-9.9429},19->{-1.1,-7.8857},20->{-1.1,-5.8285},21->{-1.1,-3.7714},22->{-1.1,-1.7142},23->{-1.1,1.7143},24->{-1.1,3.7715},25->{-1.1,5.8285},26->{-1.1,7.8855},27->{-1.1,9.9428},28->{-1.1,12.},29->{0.4,-12.},30->{0.4,-1.7142},31->{0.4,1.7143},32->{0.4,12.},33->{11.9,-31.5},34->{11.9,-27.5},35->{11.9,-8.5},36->{11.9,8.5},37->{11.9,27.5},38->{11.9,31.5},39->{13.9,-6.},40->{13.9,6.}}

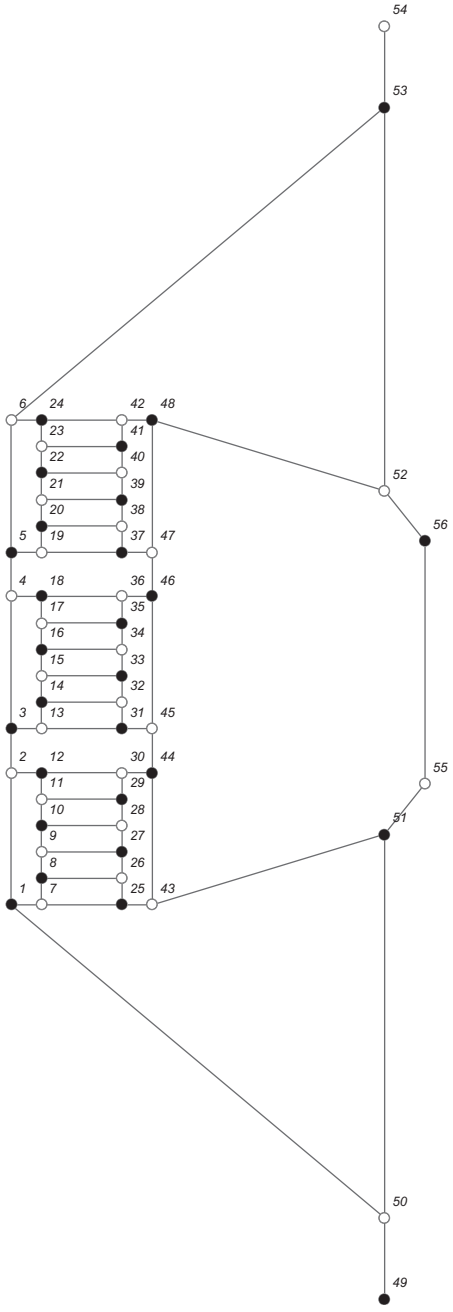
Edge list (SAGE 7.2):
[(0,1),(0,4),(0,33),(1,2),(1,9),(2,3),(2,10),(3,15),(3,36),(4,5),(4,16),(5,6),(5,17),(6,7),(6,18),(7,8),(7,19),(8,9),(8,20),(9,21),(10,11),(10,22),(11,12),(11,23),(12,13),(12,24),(13,14),(13,25),(14,15),(14,26),(15,27),(16,17),(16,28),(17,18),(18,19),(19,20),(20,21),(21,29),(22,23),(22,30),(23,24),(24,25),(25,26),(26,27),(27,31),(28,29),(28,34),(29,30),(30,31),(31,35),(32,33),(33,34),(34,38),(35,36),(35,39),(36,37),(38,39)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-6.6,-12.],1:[-6.6,-1.7142],2:[-6.6,1.7143],3:[-6.6,12.],4:[-5.1,-12.],5:[-5.1,-9.9429],6:[-5.1,-7.8857],7:[-5.1,-5.8285],8:[-5.1,-3.7714],9:[-5.1,-1.7142],10:[-5.1,1.7143],11:[-5.1,3.7715],12:[-5.1,5.8285],13:[-5.1,7.8855],14:[-5.1,9.9428],15:[-5.1,12.],16:[-1.1,-12.],17:[-1.1,-9.9429],18:[-1.1,-7.8857],19:[-1.1,-5.8285],20:[-1.1,-3.7714],21:[-1.1,-1.7142],22:[-1.1,1.7143],23:[-1.1,3.7715],24:[-1.1,5.8285],25:[-1.1,7.8855],26:[-1.1,9.9428],27:[-1.1,12.],28:[0.4,-12.],29:[0.4,-1.7142],30:[0.4,1.7143],31:[0.4,12.],32:[11.9,-31.5],33:[11.9,-27.5],34:[11.9,-8.5],35:[11.9,8.5],36:[11.9,27.5],37:[11.9,31.5],38:[13.9,-6.],39:[13.9,6.]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorTwo,3->colorOne,4->colorTwo,5->colorTwo,6->colorOne,7->colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13->colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorOne,18->colorTwo,19->colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25->colorOne,26->colorTwo,27->colorOne,28->colorTwo,29->colorTwo,30->colorOne,31->colorTwo,32->colorOne,33->colorOne,34->colorTwo,35->colorOne,36->colorTwo,37->colorOne,38->colorTwo,39->colorTwo,40->colorOne}

8.30 (Figure 4.8.c) gadget (specifying (z = 3) blocks)



Graph Properties ::

--

Number of Vertices: ‘56’

Number of Edges: ‘81’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->2,1<->7,1<->50,2<->3,2<->12,3<->4,3<->13,4<->5,4<->18,5<->6,5<->19,6<->24,6<->53,7<->8,7<->25,8<->9,8<->26,9<->10,9<->27,10<->11,10<->28,11<->12,11<->29,12<->30,13<->14,13<->31,14<->15,14<->32,15<->16,15<->33,16<->17,16<->34,17<->18,17<->35,18<->36,19<->20,19<->37,20<->21,20<->38,21<->22,21<->39,22<->23,22<->40,23<->24,23<->41,24<->42,25<->26,25<->43,26<->27,27<->28,28<->29,29<->30,30<->44,31<->32,31<->45,32<->33,33<->34,34<->35,35<->36,36<->46,37<->38,37<->47,38<->39,39<->40,40<->41,41<->42,42<->48,43<->44,43<->51,44<->45,45<->46,46<->47,47<->48,48<->52,49<->50,50<->51,51<->55,52<->53,52<->56,53<->54,55<->56}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-5.7143,-12.},2->{-5.7143,-5.4545},3->{-5.7143,-3.2727},4->{-5.7143,3.2727},5->{-5.7143,5.4546},6->{-5.7143,12.},7->{-4.2143,-12.},8->{-4.2143,-10.6909},9->{-4.2143,-9.3818},10->{-4.2143,-8.0727},11->{-4.2143,-6.7636},12->{-4.2143,-5.4545},13->{-4.2143,-3.2727},14->{-4.2143,-1.9636},15->{-4.2143,-0.6546},16->{-4.2143,0.6544},17->{-4.2143,1.9636},18->{-4.2143,3.2727},19->{-4.2143,5.4546},20->{-4.2143,6.7635},21->{-4.2143,8.0727},22->{-4.2143,9.3818},23->{-4.2143,10.6909},24->{-4.2143,12.},25->{-0.2143,-12.},26->{-0.2143,-10.6909},27->{-0.2143,-9.3818},28->{-0.2143,-8.0727},29->{-0.2143,-6.7636},30->{-0.2143,-5.4545},31->{-0.2143,-3.2727},32->{-0.2143,-1.9636},33->{-0.2143,-0.6546},34->{-0.2143,0.6544},35->{-0.2143,1.9636},36->{-0.2143,3.2727},37->{-0.2143,5.4546},38->{-0.2143,6.7635},39->{-0.2143,8.0727},40->{-0.2143,9.3818},41->{-0.2143,10.6909},42->{-0.2143,12.},43->{1.2857,-12.},44->{1.2857,-5.4545},45->{1.2857,-3.2727},46->{1.2857,3.2727},47->{1.2857,5.4546},48->{1.2857,12.},49->{12.7857,-31.5},50->{12.7857,-27.5},51->{12.7857,-8.5},52->{12.7857,8.5},53->{12.7857,27.5},54->{12.7857,31.5},55->{14.7857,-6.},56->{14.7857,6.}}

Edge list (SAGE 7.2):

[(0,1),(0,6),(0,49),(1,2),(1,11),(2,3),(2,12),(3,4),(3,17),(4,5),(4,18),(5,23),(5,52),(6,7),(6,24),(7,8),(7,25),(8,9),(8,26),(9,10),(9,27),(10,11),(10,28),(11,29),(12,13),(12,30),(13,14),(13,31),(14,15),(14,32),(15,16),(15,33),(16,17),(16,34),(17,35),(18,19),(18,36),(19,20),(19,37),(20,21),(20,38),(21,22),(21,39),(22,23),(22,40),(23,41),(24,25),(24,42),(25,26),(26,27),(27,28),(28,29),(29,43),(30,31),(30,44),(31,32),(32,33),(33,34),(34,35),(35,45),(36,37),(36,46),(37,38),(38,39),(39,40),(40,41),(41,47),(42,43),(42,50),(43,44),(44,45),(45,46),(46,47),(47,51),(48,49),(49,50),(50,54),(51,52),(51,55),(52,53),(54,55)]

-

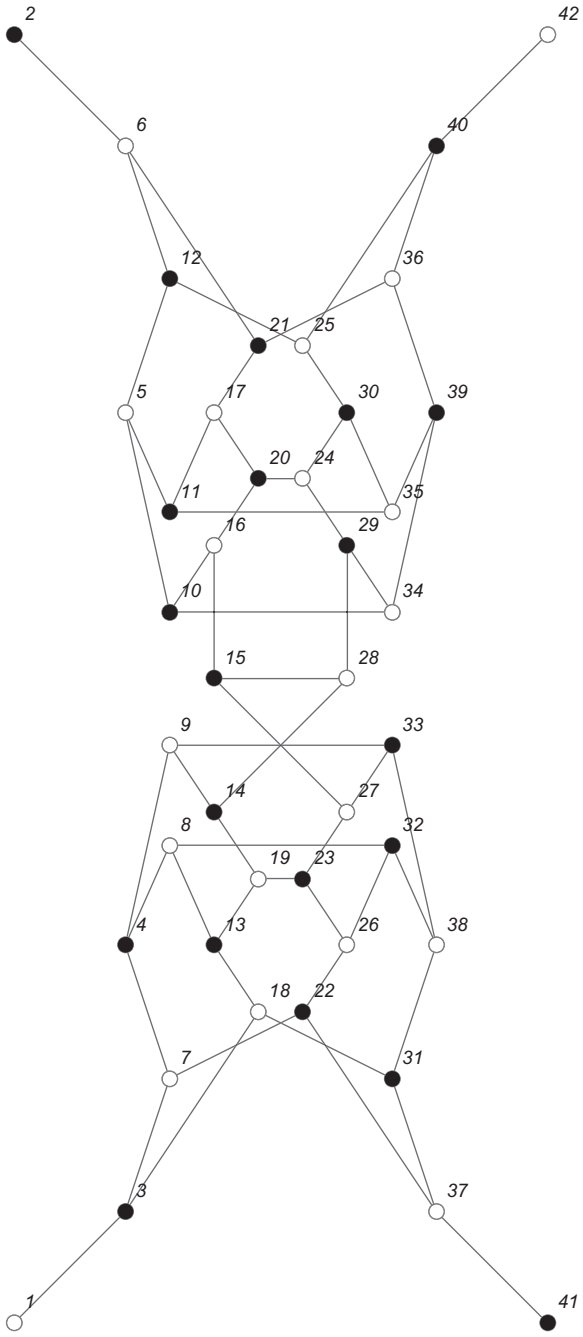
Example embedding coordinates (SAGE 7.2):

{0:[-5.7143,-12.],1:[-5.7143,-5.4545],2:[-5.7143,-3.2727],3:[-5.7143,3.2727],4:[-5.7143,5.4546],5:[-5.7143,12.],6:[-4.2143,-12.],7:[-4.2143,-10.6909],8:[-4.2143,-9.3818],9:[-4.2143,-8.0727],10:[-4.2143,-6.7636],11:[-4.2143,-5.4545],12:[-4.2143,-3.2727],13:[-4.2143,-1.9636],14:[-4.2143,-0.6546],15:[-4.2143,0.6544],16:[-4.2143,1.9636],17:[-4.2143,3.2727],18:[-4.2143,5.4546],19:[-4.2143,6.7635],20:[-4.2143,8.0727],21:[-4.2143,9.3818],22:[-4.2143,10.6909],23:[-4.2143,12.],24:[-0.2143,-12.],25:[-0.2143,-10.6909],26:[-0.2143,-9.3818],27:[-0.2143,-8.0727],28:[-0.2143,-6.7636],29:[-0.2143,-5.4545],30:[-0.2143,-3.2727],31:[-0.2143,-1.9636],32:[-0.2143,-0.6546],33:[-0.2143,0.6544],34:[-0.2143,1.9636],35:[-0.2143,3.2727],36:[-0.2143,5.4546],37:[-0.2143,6.7635],38:[-0.2143,8.0727],39:[-0.2143,9.3818],40:[-0.2143,10.6909],41:[-0.2143,12.],42:[1.2857,-12.],43:[1.2857,-5.4545],44:[1.2857,-3.2727],45:[1.2857,3.2727],46:[1.2857,5.4546],47:[1.2857,12.],48:[12.7857,-31.5],49:[12.7857,-27.5],50:[12.7857,-8.5],51:[12.7857,8.5],52:[12.7857,27.5],53:[12.7857,31.5],54:[14.7857,-6.],55:[14.7857,6.]}

Canonical vertex ($k = 2$)-coloring ::

{1->colorOne,2->colorTwo,3->colorOne,4->colorTwo,5->colorOne,6->colorTwo,7->colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13->colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorTwo,18->colorOne,19->colorTwo,20->colorOne,21->colorTwo,22->colorOne,23->colorTwo,24->colorOne,25->colorOne,26->colorTwo,27->colorOne,28->colorTwo,29->colorOne,30->colorTwo,31->colorOne,32->colorTwo,33->colorOne,34->colorTwo,35->colorOne,36->colorTwo,37->colorOne,38->colorTwo,39->colorOne,40->colorTwo,41->colorOne,42->colorTwo,43->colorTwo,44->colorOne,45->colorTwo,46->colorOne,47->colorTwo,48->colorOne,49->colorOne,50->colorTwo,51->colorOne,52->colorTwo,53->colorOne,54->colorTwo,55->colorTwo,56->colorOne}

8.31 (Figure 4.9.b) gadget (subgraph of the "Fig. 4" 54 vertex cubic 3-connected bipartite non-Hamiltonian graph from ref. [54])



Graph Properties ::

--

Number of Vertices: ‘42’

Number of Edges: ‘59’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘False’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘False’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘False’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘6’

Output of SAGE 7.2’s ‘girth()’ function: ‘6’

Output for the ‘igraph’ R package ‘girth()’ function: ‘6’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: '2'

(Embedding in the orientable surface S_2 --- FORMAT: "vertex v_i: [clockwise ordering of vertices about v_i in $S_g = S_2$ embedding]) ::

```
{0: [2],
 1: [5],
 2: [0, 6, 17],
 3: [6, 7, 8],
 4: [9, 10, 11],
 5: [1, 11, 20],
 6: [2, 3, 21],
 7: [3, 12, 31],
 8: [3, 32, 13],
 9: [4, 33, 15],
 10: [4, 16, 34],
 11: [4, 5, 24],
 12: [7, 17, 18],
 13: [8, 27, 18],
 14: [15, 27, 26],
 15: [9, 14, 19],
 16: [10, 19, 20],
 17: [2, 30, 12],
 18: [12, 13, 22],
 19: [15, 23, 16],
 20: [5, 35, 16],
 21: [6, 36, 25],
 22: [18, 26, 25],
 23: [19, 28, 29],
 24: [11, 29, 39],
 25: [21, 31, 22],
 26: [14, 32, 22],
 27: [13, 14, 28],
 28: [23, 27, 33],
 29: [23, 34, 24],
 30: [17, 37, 36],
 31: [7, 25, 37],
 32: [8, 37, 26],
 33: [9, 38, 28],
 34: [10, 29, 38],
 35: [20, 38, 39],
 36: [21, 40, 30],
 37: [30, 32, 31],
 38: [33, 35, 34],
 39: [24, 35, 41],
 40: [36],
 41: [39]}
```

Edge list (Mathematica 10.4.1):

{1<->3,2<->6,3<->7,3<->18,4<->7,4<->8,4<->9,5<->10,5<->11,5<->12,6<->12,6<->21,7<->22,8<->13,8<->32,9<->14,9<->33,10<->16,10<->34,11<->17,11<->35,12<->25,13<->18,13<->19,14<->19,14<->28,15<->16,15<->27,15<->28,16<->20,17<->20,17<->21,18<->31,19<->23,20<->24,21<->36,22<->26,22<->37,23<->26,23<->27,24<->29,24<->30,25<->30,25<->40,26<->32,27<->33,28<->29,29<->34,30<->35,31<->37,31<->38,32<->38,33<->38,34<->39,35<->39,36<->39,36<->40,37<->41,40<->42}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-6.,-14.5},2->{-6.,14.5},3->{-3.5,-12.},4->{-3.5,-6.},5->{-3.5,6.},6->{-3.5,12.},7->{-2.5,-9.},8->{-2.5,-3.75},9->{-2.5,-1.5},10->{-2.5,1.5},11->{-2.5,3.75},12->{-2.5,9.},13->{-1.5,-6.},14->{-1.5,-3.},15->{-1.5,0.},16->{-1.5,3.},17->{-1.5,6.},18->{-0.5,-7.5},19->{-0.5,-4.5},20->{-0.5,4.5},21->{-0.5,7.5},22->{0.5,-7.5},23->{0.5,-4.5},24->{0.5,4.5},25->{0.5,7.5},26->{1.5,-6.},27->{1.5,-3.},28->{1.5,0.},29->{1.5,3.},30->{1.5,6.},31->{2.5,-9.},32->{2.5,-3.75},33->{2.5,-1.5},34->{2.5,1.5},35->{2.5,3.75},36->{2.5,9.},37->{3.5,-12.},38->{3.5,-6.},39->{3.5,6.},40->{3.5,12.},41->{6.,-14.5},42->{6.,14.5}}

Edge list (SAGE 7.2):

[(0,2),(1,5),(2,6),(2,17),(3,6),(3,7),(3,8),(4,9),(4,10),(4,11),(5,11),(5,20),(6,21),(7,12),(7,31),(8,13),(8,32),(9,15),(9,33),(10,16),(10,34),(11,24),(12,17),(12,18),(13,18),(13,27),(14,15),(14,26),(14,27),(15,19),(16,19),(16,20),(17,30),(18,22),(19,23),(20,35),(21,25),(21,36),(22,25),(22,26),(23,28),(23,29),(24,29),(24,39),(25,31),(26,32),(27,28),(28,33),(29,34),(30,36),(30,37),(31,37),(32,37),(33,38),(34,38),(35,38),(35,39),(36,40),(39,41)]

-

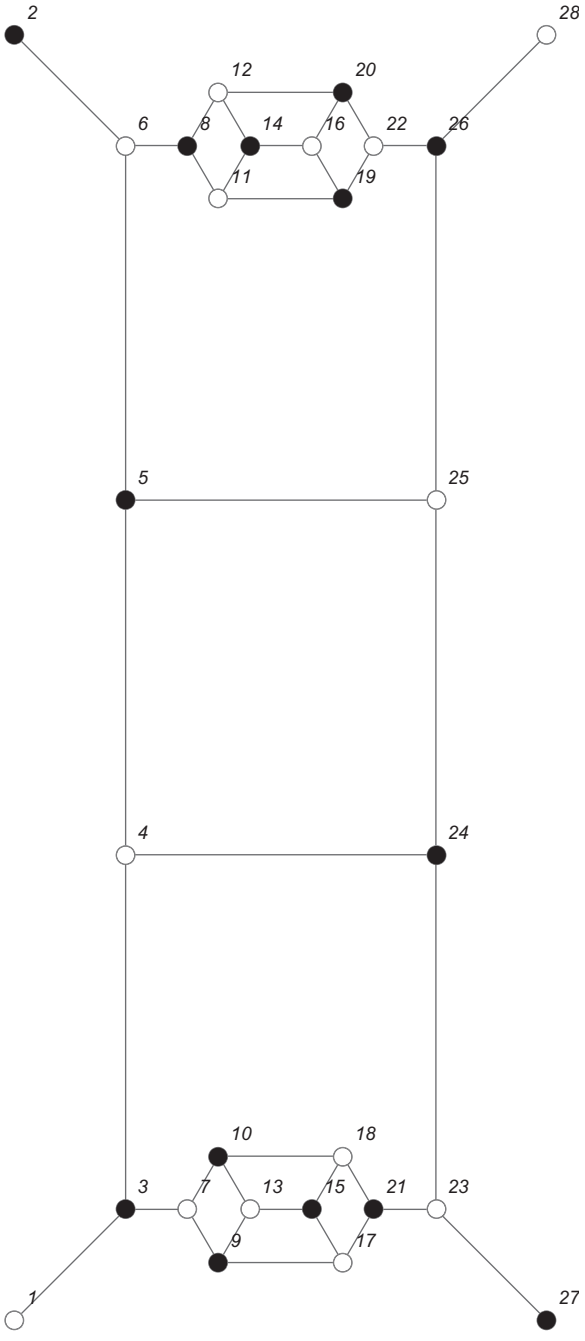
Example embedding coordinates (SAGE 7.2):

{0:[-4.75,-13.875],1:[-4.75,13.875],2:[-3.5,-12.],3:[-3.5,-6.],4:[-3.5,6.],5:[-3.5,12.],6:[-2.5,-9.],7:[-2.5,-3.75],8:[-2.5,-1.5],9:[-2.5,1.5],10:[-2.5,3.75],11:[-2.5,9.],12:[-1.5,-6.],13:[-1.5,-3.],14:[-1.5,0.],15:[-1.5,3.],16:[-1.5,6.],17:[-0.5,-7.5],18:[-0.5,-4.5],19:[-0.5,4.5],20:[-0.5,7.5],21:[0.5,-7.5],22:[0.5,-4.5],23:[0.5,4.5],24:[0.5,7.5],25:[1.5,-6.],26:[1.5,-3.],27:[1.5,0.],28:[1.5,3.],29:[1.5,6.],30:[2.5,-9.],31:[2.5,-3.75],32:[2.5,-1.5],33:[2.5,1.5],34:[2.5,3.75],35:[2.5,9.],36:[3.5,-12.],37:[3.5,-6.],38:[3.5,6.],39:[3.5,12.],40:[4.75,-13.875],41:[4.75,13.875]}

Canonical vertex $(k = 2)$-coloring ::

{1->colorTwo,2->colorOne,3->colorOne,4->colorOne,5->colorTwo,6->colorTwo,7->colorTwo,8->colorTwo,9->colorTwo,10->colorOne,11->colorOne,12->colorOne,13->colorOne,14->colorOne,15->colorOne,16->colorTwo,17->colorTwo,18->colorTwo,19->colorTwo,20->colorOne,21->colorOne,22->colorOne,23->colorOne,24->colorTwo,25->colorTwo,26->colorTwo,27->colorTwo,28->colorTwo,29->colorOne,30->colorOne,31->colorOne,32->colorOne,33->colorOne,34->colorTwo,35->colorTwo,36->colorTwo,37->colorTwo,38->colorTwo,39->colorOne,40->colorOne,41->colorOne,42->colorTwo}

8.32 (Figure 4.9.c) gadget



Graph Properties ::

--

Number of Vertices: ‘28’

Number of Edges: ‘38’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: '0'

Edge list (Mathematica 10.4.1):
{1<->3,2<->6,3<->4,3<->7,4<->5,4<->24,5<->6,5<->25,6<->8,7<->9,7<->10,8<->11,8<->12,9<->13,9<->17,10<->13,10<->18,11<->14,11<->19,12<->14,12<->20,13<->15,14<->16,15<->17,15<->18,16<->19,16<->20,17<->21,18<->21,19<->22,20<->22,21<->23,22<->26,23<->24,23<->27,24<->25,25<->26,26<->28}

-

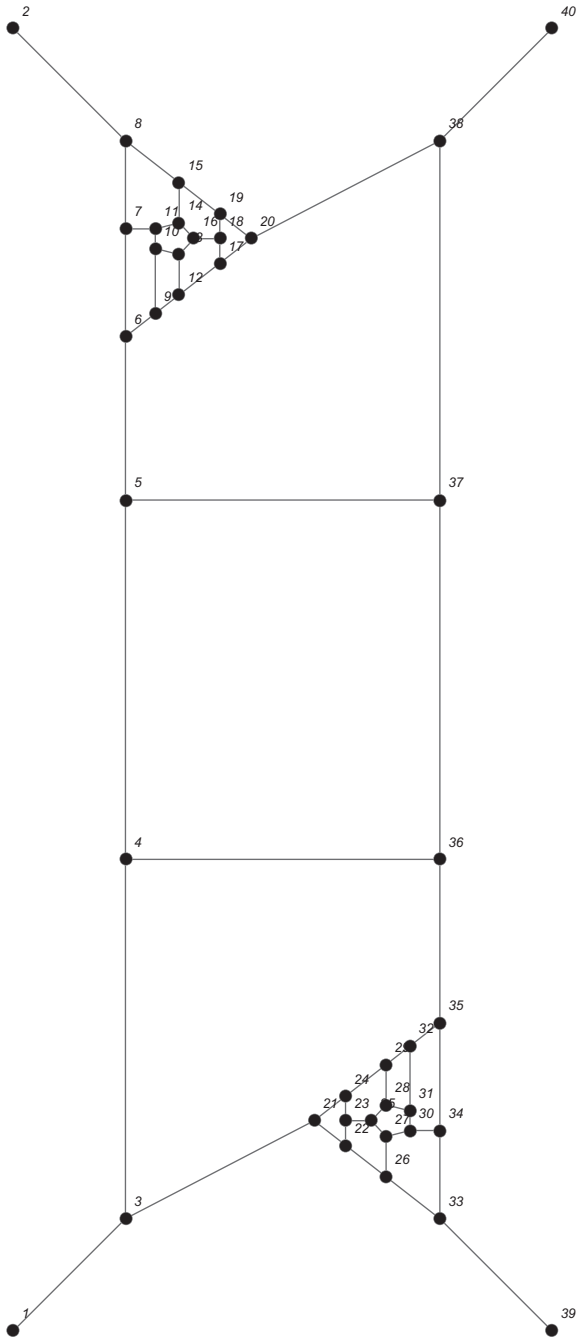
Example embedding coordinates (Mathematica 10.4.1):
{1->{-6.,-14.5},2->{-6.,14.5},3->{-3.5,-12.},4->{-3.5,-4.},5->{-3.5,4.},6->{-3.5,12.},7->{-2.1,-12.},8->{-2.1,12.},9->{-1.4,-13.2},10->{-1.4,-10.8},11->{-1.4,10.8},12->{-1.4,13.2},13->{-0.7,-12.},14->{-0.7,12.},15->{0.7,-12.},16->{0.7,12.},17->{1.4,-13.2},18->{1.4,-10.8},19->{1.4,10.8},20->{1.4,13.2},21->{2.1,-12.},22->{2.1,12.},23->{3.5,-12.},24->{3.5,-4.},25->{3.5,4.},26->{3.5,12.},27->{6.,-14.5},28->{6.,14.5}}

Edge list (SAGE 7.2):
[(0,2),(1,5),(2,3),(2,6),(3,4),(3,23),(4,5),(4,24),(5,7),(6,8),(6,9),(7,10),(7,11),(8,12),(8,16),(9,12),(9,17),(10,13),(10,18),(11,13),(11,19),(12,14),(13,15),(14,16),(14,17),(15,18),(15,19),(16,20),(17,20),(18,21),(19,21),(20,22),(21,25),(22,23),(22,26),(23,24),(24,25),(25,27)]

Example embedding coordinates (SAGE 7.2):
{0:[-6.,-14.5],1:[-6.,14.5],2:[-3.5,-12.],3:[-3.5,-4.],4:[-3.5,4.],5:[-3.5,12.],6:[-2.1,-12.],7:[-2.1,12.],8:[-1.4,-13.2],9:[-1.4,-10.8],10:[-1.4,10.8],11:[-1.4,13.2],12:[-0.7,-12.],13:[-0.7,12.],14:[0.7,-12.],15:[0.7,12.],16:[1.4,-13.2],17:[1.4,-10.8],18:[1.4,10.8],19:[1.4,13.2],20:[2.1,-12.],21:[2.1,12.],22:[3.5,-12.],23:[3.5,-4.],24:[3.5,4.],25:[3.5,12.],26:[6.,-14.5],27:[6.,14.5]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorTwo,2->colorOne,3->colorOne,4->colorTwo,5->colorOne,6->colorTwo,7->colorTwo,8->colorOne,9->colorOne,10->colorOne,11->colorTwo,12->colorTwo,13->colorTwo,14->colorOne,15->colorOne,16->colorTwo,17->colorTwo,18->colorTwo,19->colorOne,20->colorOne,21->colorOne,22->colorTwo,23->colorTwo,24->colorOne,25->colorTwo,26->colorOne,27->colorOne,28->colorTwo}

8.33 (Figure 4.9.d) gadget (nearly identical to the "Fig. 2.a" gadget from ref. [67])



Graph Properties ::

--

Number of Vertices: ‘40’

Number of Edges: ‘56’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘3’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘3’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘False’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘False’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘False’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: '0'

Edge list (Mathematica 10.4.1):
{1<->3,2<->8,3<->4,3<->21,4<->5,4<->36,5<->6,5<->37,6<->7,6<->9,7<->8,7<->11,8<->15,9<->10,9<->12,10<->11,10<->13,11<->14,12<->13,12<->17,13<->16,14<->15,14<->16,15<->19,16<->18,17<->18,17<->20,18<->19,19<->20,20<->38,21<->22,21<->24,22<->23,22<->26,23<->24,23<->25,24<->29,25<->27,25<->28,26<->27,26<->33,27<->30,28<->29,28<->31,29<->32,30<->31,30<->34,31<->32,32<->35,33<->34,33<->39,34<->35,35<->36,36<->37,37<->38,38<->40}

-

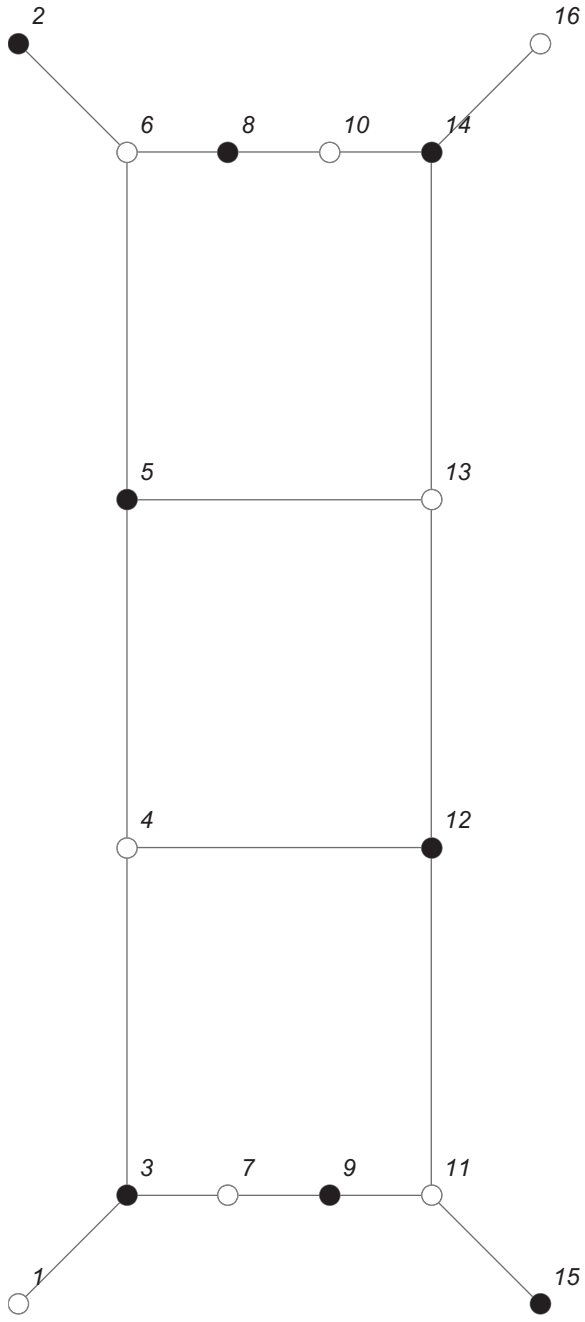
Example embedding coordinates (Mathematica 10.4.1):
{1->{-6.,-14.5},2->{-6.,14.5},3->{-3.5,-12.},4->{-3.5,-4.},5->{-3.5,4.},6->{-3.5,7.6364},7->{-3.5,10.0364},8->{-3.5,12.},9->{-2.8354,8.1542},10->{-2.8354,9.6},11->{-2.8354,10.0364},12->{-2.3028,8.5693},13->{-2.3028,9.4652},14->{-2.3028,10.1712},15->{-2.3028,11.0671},16->{-1.9736,9.8182},17->{-1.4,9.2727},18->{-1.4,9.8182},19->{-1.4,10.3636},20->{-0.7,9.8182},21->{0.7,-9.8182},22->{1.4,-10.3636},23->{1.4,-9.8182},24->{1.4,-9.2727},25->{1.9736,-9.8182},26->{2.3028,-11.0671},27->{2.3028,-10.1712},28->{2.3028,-9.4652},29->{2.3028,-8.5693},30->{2.8354,-10.0364},31->{2.8354,-9.6},32->{2.8354,-8.1542},33->{3.5,-12.},34->{3.5,-10.0364},35->{3.5,-7.6364},36->{3.5,-4.},37->{3.5,4.},38->{3.5,12.},39->{6.,-14.5},40->{6.,14.5}}

Edge list (SAGE 7.2):
[(0,2),(1,7),(2,3),(2,20),(3,4),(3,35),(4,5),(4,36),(5,6),(5,8),(6,7),(6,10),(7,14),(8,9),(8,11),(9,10),(9,12),(10,13),(11,12),(11,16),(12,15),(13,14),(13,15),(14,18),(15,17),(16,17),(16,19),(17,18),(18,19),(19,37),(20,21),(20,23),(21,22),(21,25),(22,23),(22,24),(23,28),(24,26),(24,27),(25,26),(25,32),(26,29),(27,28),(27,30),(28,31),(29,30),(29,33),(30,31),(31,34),(32,33),(32,38),(33,34),(34,35),(35,36),(36,37),(37,39)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-6.,-14.5],1:[-6.,14.5],2:[-3.5,-12.],3:[-3.5,-4.],4:[-3.5,4.],5:[-3.5,7.6364],6:[-3.5,10.0364],7:[-3.5,12.],8:[-2.8354,8.1542],9:[-2.8354,9.6],10:[-2.8354,10.0364],11:[-2.3028,8.5693],12:[-2.3028,9.4652],13:[-2.3028,10.1712],14:[-2.3028,11.0671],15:[-1.9736,9.8182],16:[-1.4,9.2727],17:[-1.4,9.8182],18:[-1.4,10.3636],19:[-0.7,9.8182],20:[0.7,-9.8182],21:[1.4,-10.3636],22:[1.4,-9.8182],23:[1.4,-9.2727],24:[1.9736,-9.8182],25:[2.3028,-11.0671],26:[2.3028,-10.1712],27:[2.3028,-9.4652],28:[2.3028,-8.5693],29:[2.8354,-10.0364],30:[2.8354,-9.6],31:[2.8354,-8.1542],32:[3.5,-12.],33:[3.5,-10.0364],34:[3.5,-7.6364],35:[3.5,-4.],36:[3.5,4.],37:[3.5,12.],38:[6.,-14.5],39:[6.,14.5]}

8.34 (Figure 4.9.e) gadget



Graph Properties ::

--

Number of Vertices: ‘16’

Number of Edges: ‘18’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: '0'

Edge list (Mathematica 10.4.1):
{1<->3,2<->6,3<->4,3<->7,4<->5,4<->12,5<->6,5<->13,6<->8,7<->9,8<->10,9<->11,10<->14,11<->12,11<->15,12<->13,13<->14,14<->16}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-6.,-14.5},2->{-6.,14.5},3->{-3.5,-12.},4->{-3.5,-4.},5->{-3.5,4.},6->{-3.5,12.},7->{-1.1667,-12.},8->{-1.1667,12.},9->{1.1667,-12.},10->{1.1667,12.},11->{3.5,-12.},12->{3.5,-4.},13->{3.5,4.},14->{3.5,12.},15->{6.,-14.5},16->{6.,14.5}}

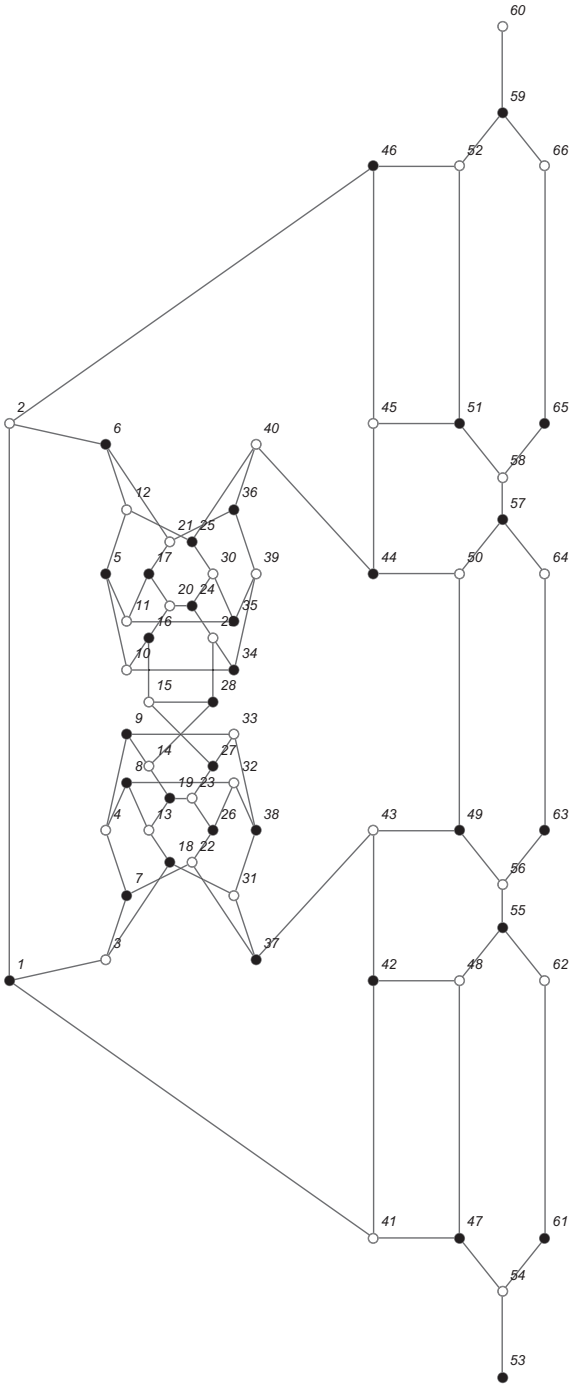
Edge list (SAGE 7.2):
[(0,2),(1,5),(2,3),(2,6),(3,4),(3,11),(4,5),(4,12),(5,7),(6,8),(7,9),(8,10),(9,13),(10,11),(10,14),(11,12),(12,13),(13,15)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-6.,-14.5],1:[-6.,14.5],2:[-3.5,-12.],3:[-3.5,-4.],4:[-3.5,4.],5:[-3.5,12.],6:[-1.1667,-12.],7:[-1.1667,12.],8:[1.1667,-12.],9:[1.1667,12.],10:[3.5,-12.],11:[3.5,-4.],12:[3.5,4.],13:[3.5,12.],14:[6.,-14.5],15:[6.,14.5]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorTwo,2->colorOne,3->colorOne,4->colorTwo,5->colorOne,6->colorTwo,7->colorTwo,8->colorOne,9->colorOne,10->colorTwo,11->colorTwo,12->colorOne,13->colorTwo,14->colorOne,15->colorOne,16->colorTwo}

8.35 (Figure 4.10.b) gadget



Graph Properties ::

--

Number of Vertices: ‘66’

Number of Edges: ‘94’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘False’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘False’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘False’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):


```
{1<->2,1<->3,1<->41,2<->6,2<->46,3<->7,3<->18,4<->7,4<->8,4<->9,5<->10,5<->11,5<-
>12,6<->12,6<->21,7<->22,8<->13,8<->32,9<->14,9<->33,10<->16,10<->34,11<->17,11<-
>35,12<->25,13<->18,13<->19,14<->19,14<->28,15<->16,15<->27,15<->28,16<->20,17<-
>20,17<->21,18<->31,19<->23,20<->24,21<->36,22<->26,22<->37,23<->26,23<->27,24<-
>29,24<->30,25<->30,25<->40,26<->32,27<->33,28<->29,29<->34,30<->35,31<->37,31<-
>38,32<->38,33<->38,34<->39,35<->39,36<->39,36<->40,37<->43,40<->44,41<->42,41<-
>47,42<->43,42<->48,43<->49,44<->45,44<->50,45<->46,45<->51,46<->52,47<->48,47<-
>54,48<->55,49<->50,49<->56,50<->57,51<->52,51<->58,52<->59,53<->54,54<->61,55<-
>56,55<->62,56<->63,57<->58,57<->64,58<->65,59<->60,59<->66,61<->62,63<->64,65<->66}
```


-

Example embedding coordinates (Mathematica 10.4.1):


```
{1->{-13.1212,-13.},2->{-13.1212,13.},3->{-8.6212,-12.},4->{-8.6212,-6.},5->{-
8.6212,6.},6->{-8.6212,12.},7->{-7.6212,-9.},8->{-7.6212,-3.75},9->{-7.6212,-1.5},10-
>{-7.6212,1.5},11->{-7.6212,3.75},12->{-7.6212,9.},13->{-6.6212,-6.},14->{-6.6212,-
3.},15->{-6.6212,0.},16->{-6.6212,3.},17->{-6.6212,6.},18->{-5.6212,-7.5},19->{-
5.6212,-4.5},20->{-5.6212,4.5},21->{-5.6212,7.5},22->{-4.6212,-7.5},23->{-4.6212,-
4.5},24->{-4.6212,4.5},25->{-4.6212,7.5},26->{-3.6212,-6.},27->{-3.6212,-3.},28->{-
3.6212,0.},29->{-3.6212,3.},30->{-3.6212,6.},31->{-2.6212,-9.},32->{-2.6212,-3.75},33-
>{-2.6212,-1.5},34->{-2.6212,1.5},35->{-2.6212,3.75},36->{-2.6212,9.},37->{-1.6212,-
12.},38->{-1.6212,-6.},39->{-1.6212,6.},40->{-1.6212,12.},41->{3.8788,-25.},42-
>{3.8788,-13.},43->{3.8788,-6.},44->{3.8788,6.},45->{3.8788,13.},46->{3.8788,25.},47-
>{7.8788,-25.},48->{7.8788,-13.},49->{7.8788,-6.},50->{7.8788,6.},51->{7.8788,13.},52-
>{7.8788,25.},53->{9.8788,-31.5},54->{9.8788,-27.5},55->{9.8788,-10.5},56->{9.8788,-
8.5},57->{9.8788,8.5},58->{9.8788,10.5},59->{9.8788,27.5},60->{9.8788,31.5},61-
>{11.8788,-25.},62->{11.8788,-13.},63->{11.8788,-6.},64->{11.8788,6.},65-
>{11.8788,13.},66->{11.8788,25.}}
```

Edge list (SAGE 7.2):


```
[(0,1),(0,2),(0,40),(1,5),(1,45),(2,6),(2,17),(3,6),(3,7),(3,8),(4,9),(4,10),(4,11),(5
,11),(5,20),(6,21),(7,12),(7,31),(8,13),(8,32),(9,15),(9,33),(10,16),(10,34),(11,24),(
12,17),(12,18),(13,18),(13,27),(14,15),(14,26),(14,27),(15,19),(16,19),(16,20),(17,30)
,(18,22),(19,23),(20,35),(21,25),(21,36),(22,25),(22,26),(23,28),(23,29),(24,29),(24,3
9),(25,31),(26,32),(27,28),(28,33),(29,34),(30,36),(30,37),(31,37),(32,37),(33,38),(34
,38),(35,38),(35,39),(36,42),(39,43),(40,41),(40,46),(41,42),(41,47),(42,48),(43,44),(
43,49),(44,45),(44,50),(45,51),(46,47),(46,53),(47,54),(48,49),(48,55),(49,56),(50,51)
,(50,57),(51,58),(52,53),(53,60),(54,55),(54,61),(55,62),(56,57),(56,63),(57,64),(58,5
9),(58,65),(60,61),(62,63),(64,65)]
```

-

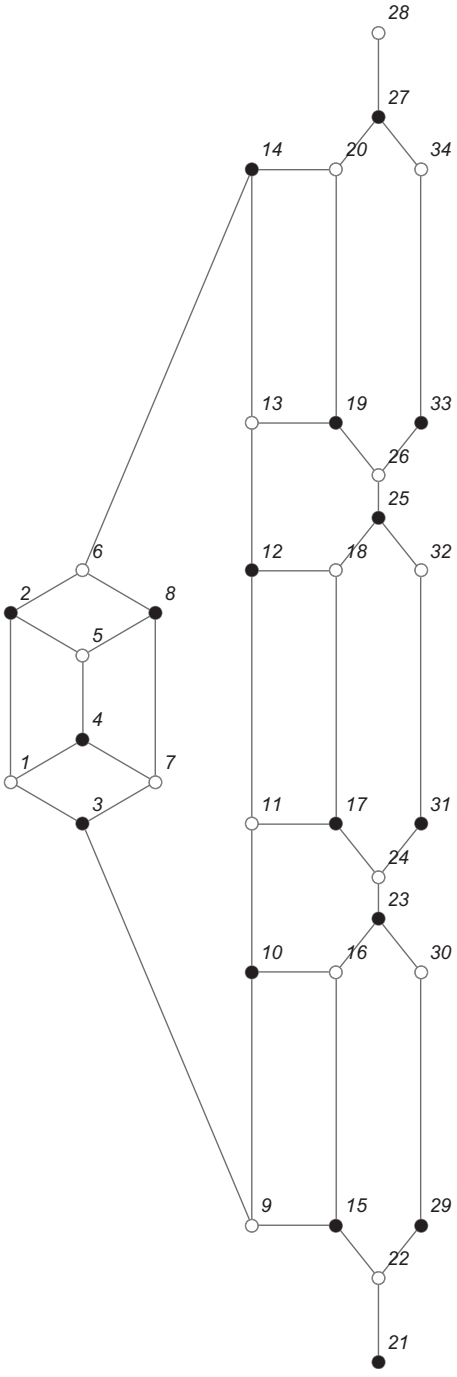
Example embedding coordinates (SAGE 7.2):

{0:[-13.1212,-13.],1:[-13.1212,13.],2:[-8.6212,-12.],3:[-8.6212,-6.],4:[-
8.6212,6.],5:[-8.6212,12.],6:[-7.6212,-9.],7:[-7.6212,-3.75],8:[-7.6212,-1.5],9:[-
7.6212,1.5],10:[-7.6212,3.75],11:[-7.6212,9.],12:[-6.6212,-6.],13:[-6.6212,-3.],14:[-
6.6212,0.],15:[-6.6212,3.],16:[-6.6212,6.],17:[-5.6212,-7.5],18:[-5.6212,-4.5],19:[-
5.6212,4.5],20:[-5.6212,7.5],21:[-4.6212,-7.5],22:[-4.6212,-4.5],23:[-
4.6212,4.5],24:[-4.6212,7.5],25:[-3.6212,-6.],26:[-3.6212,-3.],27:[-3.6212,0.],28:[-
3.6212,3.],29:[-3.6212,6.],30:[-2.6212,-9.],31:[-2.6212,-3.75],32:[-2.6212,-1.5],33:[-
2.6212,1.5],34:[-2.6212,3.75],35:[-2.6212,9.],36:[-1.6212,-12.],37:[-1.6212,-6.],38:[-
1.6212,6.],39:[-1.6212,12.],40:[3.8788,-25.],41:[3.8788,-13.],42:[3.8788,-
6.],43:[3.8788,6.],44:[3.8788,13.],45:[3.8788,25.],46:[7.8788,-25.],47:[7.8788,-
13.],48:[7.8788,-6.],49:[7.8788,6.],50:[7.8788,13.],51:[7.8788,25.],52:[9.8788,-
31.5],53:[9.8788,-27.5],54:[9.8788,-10.5],55:[9.8788,-
8.5],56:[9.8788,8.5],57:[9.8788,10.5],58:[9.8788,27.5],59:[9.8788,31.5],60:[11.8788,-
25.],61:[11.8788,-13.],62:[11.8788,-
6.],63:[11.8788,6.],64:[11.8788,13.],65:[11.8788,25.]}

Canonical vertex ($k = 2$)-coloring ::

{1->colorOne,2->colorTwo,3->colorTwo,4->colorTwo,5->colorOne,6->colorOne,7-
>colorOne,8->colorOne,9->colorOne,10->colorTwo,11->colorTwo,12->colorTwo,13-
>colorTwo,14->colorTwo,15->colorTwo,16->colorOne,17->colorOne,18->colorOne,19-
>colorOne,20->colorTwo,21->colorTwo,22->colorTwo,23->colorTwo,24->colorOne,25-
>colorOne,26->colorOne,27->colorOne,28->colorOne,29->colorTwo,30->colorTwo,31-
>colorTwo,32->colorTwo,33->colorTwo,34->colorOne,35->colorOne,36->colorOne,37-
>colorOne,38->colorOne,39->colorTwo,40->colorTwo,41->colorTwo,42->colorOne,43-
>colorTwo,44->colorOne,45->colorTwo,46->colorOne,47->colorOne,48->colorTwo,49-
>colorOne,50->colorTwo,51->colorOne,52->colorTwo,53->colorOne,54->colorTwo,55-
>colorOne,56->colorTwo,57->colorOne,58->colorTwo,59->colorOne,60->colorTwo,61-
>colorOne,62->colorTwo,63->colorOne,64->colorTwo,65->colorOne,66->colorTwo}

8.36 (Figure 4.10.c) gadget



Graph Properties ::

--

Number of Vertices: ‘34’

Number of Edges: ‘46’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: '0'

Edge list (Mathematica 10.4.1):
{1<->2,1<->3,1<->4,2<->5,2<->6,3<->7,3<->9,4<->5,4<->7,5<->8,6<->8,6<->14,7<->8,9<->10,9<->15,10<->11,10<->16,11<->12,11<->17,12<->13,12<->18,13<->14,13<->19,14<->20,15<->16,15<->22,16<->23,17<->18,17<->24,18<->25,19<->20,19<->26,20<->27,21<->22,22<->29,23<->24,23<->30,24<->31,25<->26,25<->32,26<->33,27<->28,27<->34,29<->30,31<->32,33<->34}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-13.0757,-4.},2->{-13.0757,4.},3->{-9.6471,-6.},4->{-9.6471,-2.},5->{-9.6471,2.},6->{-9.6471,6.},7->{-6.2185,-4.},8->{-6.2185,4.},9->{-1.6471,-25.},10->{-1.6471,-13.},11->{-1.6471,-6.},12->{-1.6471,6.},13->{-1.6471,13.},14->{-1.6471,25.},15->{2.3529,-25.},16->{2.3529,-13.},17->{2.3529,-6.},18->{2.3529,6.},19->{2.3529,13.},20->{2.3529,25.},21->{4.3529,-31.5},22->{4.3529,-27.5},23->{4.3529,-10.5},24->{4.3529,-8.5},25->{4.3529,8.5},26->{4.3529,10.5},27->{4.3529,27.5},28->{4.3529,31.5},29->{6.3529,-25.},30->{6.3529,-13.},31->{6.3529,-6.},32->{6.3529,6.},33->{6.3529,13.},34->{6.3529,25.}}

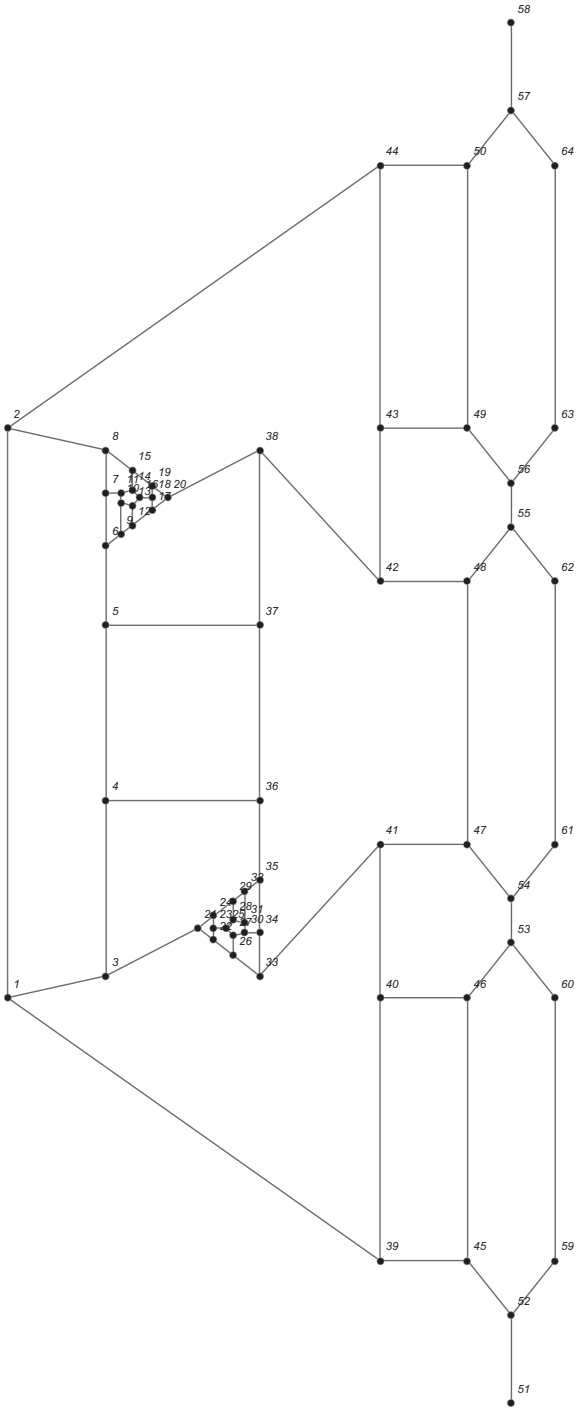
Edge list (SAGE 7.2):
[(0,1),(0,2),(0,3),(1,4),(1,5),(2,6),(2,8),(3,4),(3,6),(4,7),(5,7),(5,13),(6,7),(8,9),(8,14),(9,10),(9,15),(10,11),(10,16),(11,12),(11,17),(12,13),(12,18),(13,19),(14,15),(14,21),(15,22),(16,17),(16,23),(17,24),(18,19),(18,25),(19,26),(20,21),(21,28),(22,23),(22,29),(23,30),(24,25),(24,31),(25,32),(26,27),(26,33),(28,29),(30,31),(32,33)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-13.0757,-4.],1:[-13.0757,4.],2:[-9.6471,-6.],3:[-9.6471,-2.],4:[-9.6471,2.],5:[-9.6471,6.],6:[-6.2185,-4.],7:[-6.2185,4.],8:[-1.6471,-25.],9:[-1.6471,-13.],10:[-1.6471,-6.],11:[-1.6471,6.],12:[-1.6471,13.],13:[-1.6471,25.],14:[2.3529,-25.],15:[2.3529,-13.],16:[2.3529,-6.],17:[2.3529,6.],18:[2.3529,13.],19:[2.3529,25.],20:[4.3529,-31.5],21:[4.3529,-27.5],22:[4.3529,-10.5],23:[4.3529,-8.5],24:[4.3529,8.5],25:[4.3529,10.5],26:[4.3529,27.5],27:[4.3529,31.5],28:[6.3529,-25.],29:[6.3529,-13.],30:[6.3529,-6.],31:[6.3529,6.],32:[6.3529,13.],33:[6.3529,25.]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorTwo,2->colorOne,3->colorOne,4->colorOne,5->colorTwo,6->colorTwo,7->colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13->colorTwo,14->colorOne,15->colorOne,16->colorTwo,17->colorOne,18->colorTwo,19->colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25->colorOne,26->colorTwo,27->colorOne,28->colorTwo,29->colorOne,30->colorTwo,31->colorOne,32->colorTwo,33->colorOne,34->colorTwo}

8.37 (Figure 4.10.d) gadget



Graph Properties ::

--

Number of Vertices: ‘64’

Number of Edges: ‘91’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘3’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘3’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘False’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘False’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘False’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->2,1<->3,1<->39,2<->8,2<->44,3<->4,3<->21,4<->5,4<->36,5<->6,5<->37,6<->7,6<->9,7<->8,7<->11,8<->15,9<->10,9<->12,10<->11,10<->13,11<->14,12<->13,12<->17,13<->16,14<->15,14<->16,15<->19,16<->18,17<->18,17<->20,18<->19,19<->20,20<->38,21<->22,21<->24,22<->23,22<->26,23<->24,23<->25,24<->29,25<->27,25<->28,26<->27,26<->33,27<->30,28<->29,28<->31,29<->32,30<->31,30<->34,31<->32,32<->35,33<->34,33<->41,34<->35,35<->36,36<->37,37<->38,38<->42,39<->40,39<->45,40<->41,40<->46,41<->47,42<->43,42<->48,43<->44,43<->49,44<->50,45<->46,45<->52,46<->53,47<->48,47<->54,48<->55,49<->50,49<->56,50<->57,51<->52,52<->59,53<->54,53<->60,54<->61,55<->56,55<->62,56<->63,57<->58,57<->64,59<->60,61<->62,63<->64}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-13.2812,-13.},2->{-13.2812,13.},3->{-8.7812,-12.},4->{-8.7812,-4.},5->{-8.7812,4.},6->{-8.7812,7.6364},7->{-8.7812,10.0364},8->{-8.7812,12.},9->{-8.1166,8.1542},10->{-8.1166,9.6},11->{-8.1166,10.0364},12->{-7.584,8.5693},13->{-7.584,9.4652},14->{-7.584,10.1712},15->{-7.584,11.0671},16->{-7.2548,9.8182},17->{-6.6812,9.2727},18->{-6.6812,9.8182},19->{-6.6812,10.3636},20->{-5.9813,9.8182},21->{-4.5813,-9.8182},22->{-3.8812,-10.3636},23->{-3.8812,-9.8182},24->{-3.8812,-9.2727},25->{-3.3077,-9.8182},26->{-2.9785,-11.0671},27->{-2.9785,-10.1712},28->{-2.9785,-9.4652},29->{-2.9785,-8.5693},30->{-2.4459,-10.0364},31->{-2.4459,-9.6},32->{-2.4459,-8.1542},33->{-1.7813,-12.},34->{-1.7813,-10.0364},35->{-1.7813,-7.6364},36->{-1.7813,-4.},37->{-1.7813,4.},38->{-1.7813,12.},39->{3.7187,-25.},40->{3.7187,-13.},41->{3.7187,-6.},42->{3.7187,6.},43->{3.7187,13.},44->{3.7187,25.},45->{7.7188,-25.},46->{7.7188,-13.},47->{7.7188,-6.},48->{7.7188,6.},49->{7.7188,13.},50->{7.7188,25.},51->{9.7188,-31.5},52->{9.7188,-27.5},53->{9.7188,-10.5},54->{9.7188,-8.5},55->{9.7188,8.5},56->{9.7188,10.5},57->{9.7188,27.5},58->{9.7188,31.5},59->{11.7188,-25.},60->{11.7188,-13.},61->{11.7188,-6.},62->{11.7188,6.},63->{11.7188,13.},64->{11.7188,25.}}

Edge list (SAGE 7.2):

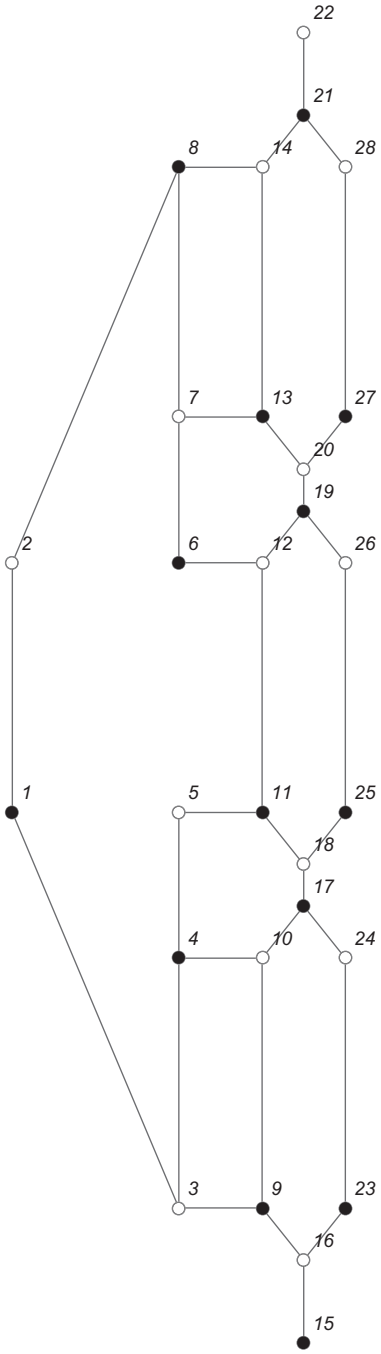
[(0,1),(0,2),(0,38),(1,7),(1,43),(2,3),(2,20),(3,4),(3,35),(4,5),(4,36),(5,6),(5,8),(6,7),(6,10),(7,14),(8,9),(8,11),(9,10),(9,12),(10,13),(11,12),(11,16),(12,15),(13,14),(13,15),(14,18),(15,17),(16,17),(16,19),(17,18),(18,19),(19,37),(20,21),(20,23),(21,22),(21,25),(22,23),(22,24),(23,28),(24,26),(24,27),(25,26),(25,32),(26,29),(27,28),(27,30),(28,31),(29,30),(29,33),(30,31),(31,34),(32,33),(32,40),(33,34),(34,35),(35,36),(36,37),(37,41),(38,39),(38,44),(39,40),(39,45),(40,46),(41,42),(41,47),(42,43),(42,48),(43,49),(44,45),(44,51),(45,52),(46,47),(46,53),(47,54),(48,49),(48,55),(49,56),(50,51),(51,58),(52,53),(52,59),(53,60),(54,55),(54,61),(55,62),(56,57),(56,63),(58,59),(60,61),(62,63)]

-

Example embedding coordinates (SAGE 7.2):

{0:[-13.2812,-13.],1:[-13.2812,13.],2:[-8.7812,-12.],3:[-8.7812,-4.],4:[-
8.7812,4.],5:[-8.7812,7.6364],6:[-8.7812,10.0364],7:[-8.7812,12.],8:[-
8.1166,8.1542],9:[-8.1166,9.6],10:[-8.1166,10.0364],11:[-7.584,8.5693],12:[-
7.584,9.4652],13:[-7.584,10.1712],14:[-7.584,11.0671],15:[-7.2548,9.8182],16:[-
6.6812,9.2727],17:[-6.6812,9.8182],18:[-6.6812,10.3636],19:[-5.9813,9.8182],20:[-
4.5813,-9.8182],21:[-3.8812,-10.3636],22:[-3.8812,-9.8182],23:[-3.8812,-9.2727],24:[-
3.3077,-9.8182],25:[-2.9785,-11.0671],26:[-2.9785,-10.1712],27:[-2.9785,-9.4652],28:[-
2.9785,-8.5693],29:[-2.4459,-10.0364],30:[-2.4459,-9.6],31:[-2.4459,-8.1542],32:[-
1.7813,-12.],33:[-1.7813,-10.0364],34:[-1.7813,-7.6364],35:[-1.7813,-4.],36:[-
1.7813,4.],37:[-1.7813,12.],38:[3.7187,-25.],39:[3.7187,-13.],40:[3.7187,-
6.],41:[3.7187,6.],42:[3.7187,13.],43:[3.7187,25.],44:[7.7188,-25.],45:[7.7188,-
13.],46:[7.7188,-6.],47:[7.7188,6.],48:[7.7188,13.],49:[7.7188,25.],50:[9.7188,-
31.5],51:[9.7188,-27.5],52:[9.7188,-10.5],53:[9.7188,-
8.5],54:[9.7188,8.5],55:[9.7188,10.5],56:[9.7188,27.5],57:[9.7188,31.5],58:[11.7188,-
25.],59:[11.7188,-13.],60:[11.7188,-
6.],61:[11.7188,6.],62:[11.7188,13.],63:[11.7188,25.]}

8.38 (Figure 4.10.e) gadget



Graph Properties ::

--

Number of Vertices: ‘28’

Number of Edges: ‘35’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: '0'

Edge list (Mathematica 10.4.1):
{1<->2,1<->3,2<->8,3<->4,3<->9,4<->5,4<->10,5<->11,6<->7,6<->12,7<->8,7<->13,8<->14,9<->10,9<->16,10<->17,11<->12,11<->18,12<->19,13<->14,13<->20,14<->21,15<->16,16<->23,17<->18,17<->24,18<->25,19<->20,19<->26,20<->27,21<->22,21<->28,23<->24,25<->26,27<->28}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-11.7143,-6.},2->{-11.7143,6.},3->{-3.7143,-25.},4->{-3.7143,-13.},5->{-3.7143,-6.},6->{-3.7143,6.},7->{-3.7143,13.},8->{-3.7143,25.},9->{0.2857,-25.},10->{0.2857,-13.},11->{0.2857,-6.},12->{0.2857,6.},13->{0.2857,13.},14->{0.2857,25.},15->{2.2857,-31.5},16->{2.2857,-27.5},17->{2.2857,-10.5},18->{2.2857,-8.5},19->{2.2857,8.5},20->{2.2857,10.5},21->{2.2857,27.5},22->{2.2857,31.5},23->{4.2857,-25.},24->{4.2857,-13.},25->{4.2857,-6.},26->{4.2857,6.},27->{4.2857,13.},28->{4.2857,25.}}

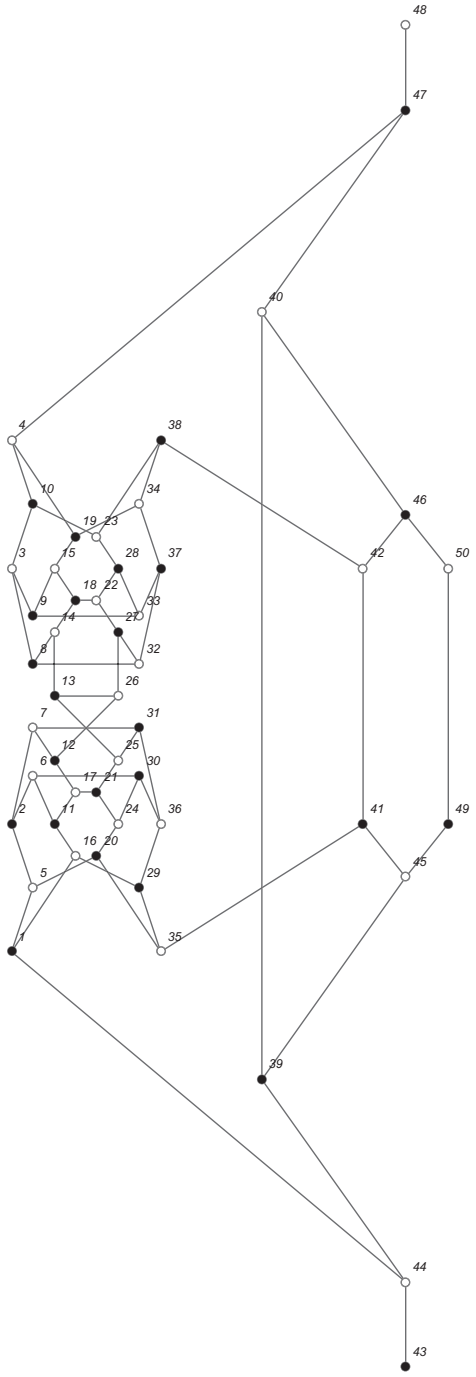
Edge list (SAGE 7.2):
[(0,1),(0,2),(1,7),(2,3),(2,8),(3,4),(3,9),(4,10),(5,6),(5,11),(6,7),(6,12),(7,13),(8,9),(8,15),(9,16),(10,11),(10,17),(11,18),(12,13),(12,19),(13,20),(14,15),(15,22),(16,17),(16,23),(17,24),(18,19),(18,25),(19,26),(20,21),(20,27),(22,23),(24,25),(26,27)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-11.7143,-6.],1:[-11.7143,6.],2:[-3.7143,-25.],3:[-3.7143,-13.],4:[-3.7143,-6.],5:[-3.7143,6.],6:[-3.7143,13.],7:[-3.7143,25.],8:[0.2857,-25.],9:[0.2857,-13.],10:[0.2857,-6.],11:[0.2857,6.],12:[0.2857,13.],13:[0.2857,25.],14:[2.2857,-31.5],15:[2.2857,-27.5],16:[2.2857,-10.5],17:[2.2857,-8.5],18:[2.2857,8.5],19:[2.2857,10.5],20:[2.2857,27.5],21:[2.2857,31.5],22:[4.2857,-25.],23:[4.2857,-13.],24:[4.2857,-6.],25:[4.2857,6.],26:[4.2857,13.],27:[4.2857,25.]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorTwo,6->colorOne,7->colorTwo,8->colorOne,9->colorOne,10->colorTwo,11->colorOne,12->colorTwo,13->colorOne,14->colorTwo,15->colorOne,16->colorTwo,17->colorOne,18->colorTwo,19->colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25->colorOne,26->colorTwo,27->colorOne,28->colorTwo}

8.39 (Figure 4.11.b) gadget



Graph Properties ::

--

Number of Vertices: ‘50’

Number of Edges: ‘72’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘False’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘False’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘False’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘6’

Output of SAGE 7.2’s ‘girth()’ function: ‘6’

Output for the ‘igraph’ R package ‘girth()’ function: ‘6’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->5,1<->16,1<->44,2<->5,2<->6,2<->7,3<->8,3<->9,3<->10,4<->10,4<->19,4<->47,5<->20,6<->11,6<->30,7<->12,7<->31,8<->14,8<->32,9<->15,9<->33,10<->23,11<->16,11<->17,12<->17,12<->26,13<->14,13<->25,13<->26,14<->18,15<->18,15<->19,16<->29,17<->21,18<->22,19<->34,20<->24,20<->35,21<->24,21<->25,22<->27,22<->28,23<->28,23<->38,24<->30,25<->31,26<->27,27<->32,28<->33,29<->35,29<->36,30<->36,31<->36,32<->37,33<->37,34<->37,34<->38,35<->41,38<->42,39<->40,39<->44,39<->45,40<->46,40<->47,41<->42,41<->45,42<->46,43<->44,45<->49,46<->50,47<->48,49<->50}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-6.83,-12.},2->{-6.83,-6.},3->{-6.83,6.},4->{-6.83,12.},5->{-5.83,-9.},6->{-5.83,-3.75},7->{-5.83,-1.5},8->{-5.83,1.5},9->{-5.83,3.75},10->{-5.83,9.},11->{-4.83,-6.},12->{-4.83,-3.},13->{-4.83,0.},14->{-4.83,3.},15->{-4.83,6.},16->{-3.83,-7.5},17->{-3.83,-4.5},18->{-3.83,4.5},19->{-3.83,7.5},20->{-2.83,-7.5},21->{-2.83,-4.5},22->{-2.83,4.5},23->{-2.83,7.5},24->{-1.83,-6.},25->{-1.83,-3.},26->{-1.83,0.},27->{-1.83,3.},28->{-1.83,6.},29->{-0.83,-9.},30->{-0.83,-3.75},31->{-0.83,-1.5},32->{-0.83,1.5},33->{-0.83,3.75},34->{-0.83,9.},35->{0.17,-12.},36->{0.17,-6.},37->{0.17,6.},38->{0.17,12.},39->{4.92,-18.},40->{4.92,18.},41->{9.67,-6.},42->{9.67,6.},43->{11.67,-31.5},44->{11.67,-27.5},45->{11.67,-8.5},46->{11.67,8.5},47->{11.67,27.5},48->{11.67,31.5},49->{13.67,-6.},50->{13.67,6.}}

Edge list (SAGE 7.2):

[(0,4),(0,15),(0,43),(1,4),(1,5),(1,6),(2,7),(2,8),(2,9),(3,9),(3,18),(3,46),(4,19),(5,10),(5,29),(6,11),(6,30),(7,13),(7,31),(8,14),(8,32),(9,22),(10,15),(10,16),(11,16),(11,25),(12,13),(12,24),(12,25),(13,17),(14,17),(14,18),(15,28),(16,20),(17,21),(18,33),(19,23),(19,34),(20,23),(20,24),(21,26),(21,27),(22,27),(22,37),(23,29),(24,30),(25,26),(26,31),(27,32),(28,34),(28,35),(29,35),(30,35),(31,36),(32,36),(33,36),(33,37),(34,40),(37,41),(38,39),(38,43),(38,44),(39,45),(39,46),(40,41),(40,44),(41,45),(42,43),(44,48),(45,49),(46,47),(48,49)]

-

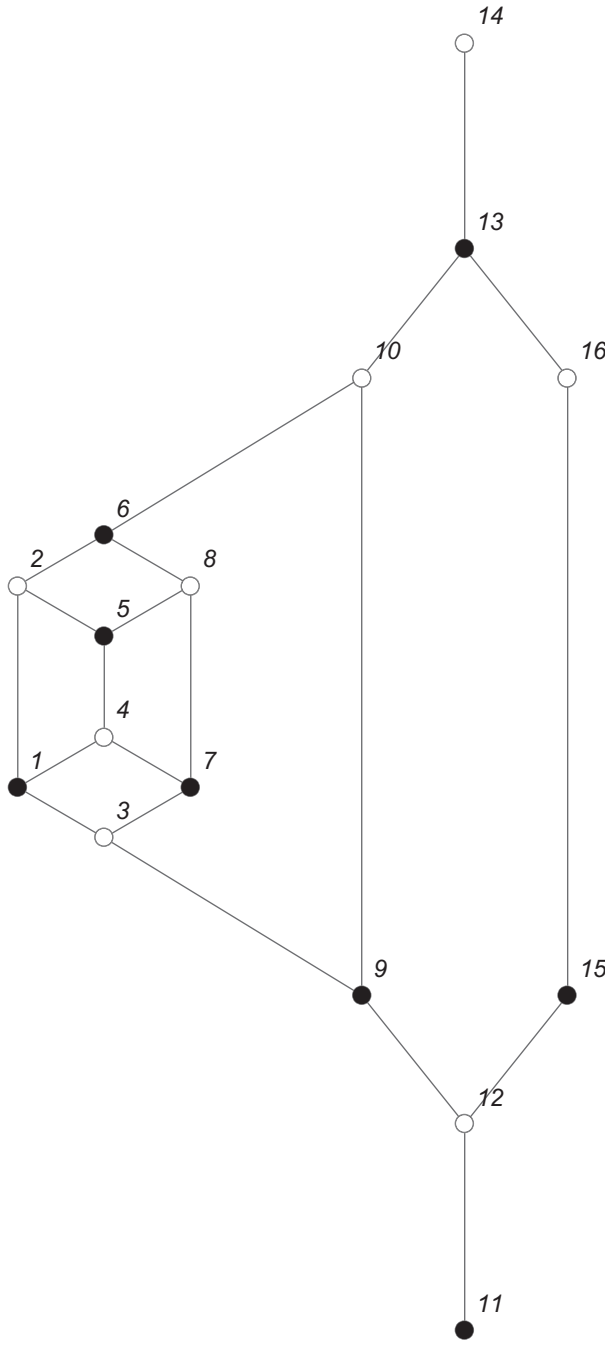
Example embedding coordinates (SAGE 7.2):

{0:[-6.83,-12.],1:[-6.83,-6.],2:[-6.83,6.],3:[-6.83,12.],4:[-5.83,-9.],5:[-5.83,-3.75],6:[-5.83,-1.5],7:[-5.83,1.5],8:[-5.83,3.75],9:[-5.83,9.],10:[-4.83,-6.],11:[-4.83,-3.],12:[-4.83,0.],13:[-4.83,3.],14:[-4.83,6.],15:[-3.83,-7.5],16:[-3.83,-4.5],17:[-3.83,4.5],18:[-3.83,7.5],19:[-2.83,-7.5],20:[-2.83,-4.5],21:[-2.83,4.5],22:[-2.83,7.5],23:[-1.83,-6.],24:[-1.83,-3.],25:[-1.83,0.],26:[-1.83,3.],27:[-1.83,6.],28:[-0.83,-9.],29:[-0.83,-3.75],30:[-0.83,-1.5],31:[-0.83,1.5],32:[-0.83,3.75],33:[-0.83,9.],34:[0.17,-12.],35:[0.17,-6.],36:[0.17,6.],37:[0.17,12.],38:[4.92,-18.],39:[4.92,18.],40:[9.67,-6.],41:[9.67,6.],42:[11.67,-31.5],43:[11.67,-27.5],44:[11.67,-8.5],45:[11.67,8.5],46:[11.67,27.5],47:[11.67,31.5],48:[13.67,-6.],49:[13.67,6.]}

Canonical vertex $(k = 2)$-coloring ::

{1->colorOne,2->colorOne,3->colorTwo,4->colorTwo,5->colorTwo,6->colorTwo,7->colorTwo,8->colorOne,9->colorOne,10->colorOne,11->colorOne,12->colorOne,13->colorOne,14->colorTwo,15->colorTwo,16->colorTwo,17->colorTwo,18->colorOne,19->colorOne,20->colorOne,21->colorOne,22->colorTwo,23->colorTwo,24->colorTwo,25->colorTwo,26->colorTwo,27->colorOne,28->colorOne,29->colorOne,30->colorOne,31->colorOne,32->colorTwo,33->colorTwo,34->colorTwo,35->colorTwo,36->colorTwo,37->colorOne,38->colorOne,39->colorOne,40->colorTwo,41->colorOne,42->colorTwo,43->colorOne,44->colorTwo,45->colorTwo,46->colorOne,47->colorOne,48->colorTwo,49->colorOne,50->colorTwo}

8.40 (Figure 4.11.c) gadget



Graph Properties ::

--

Number of Vertices: ‘16’

Number of Edges: ‘21’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: '0'

Edge list (Mathematica 10.4.1):
{1<->2,1<->3,1<->4,2<->5,2<->6,3<->7,3<->9,4<->5,4<->7,5<->8,6<->8,6<->10,7<->8,9<->10,9<->12,10<->13,11<->12,12<->15,13<->14,13<->16,15<->16}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-5.18,-1.96},2->{-5.18,1.96},3->{-3.5,-2.94},4->{-3.5,-0.98},5->{-3.5,0.98},6->{-3.5,2.94},7->{-1.82,-1.96},8->{-1.82,1.96},9->{1.5,-6.},10->{1.5,6.},11->{3.5,-12.5},12->{3.5,-8.5},13->{3.5,8.5},14->{3.5,12.5},15->{5.5,-6.},16->{5.5,6.}}

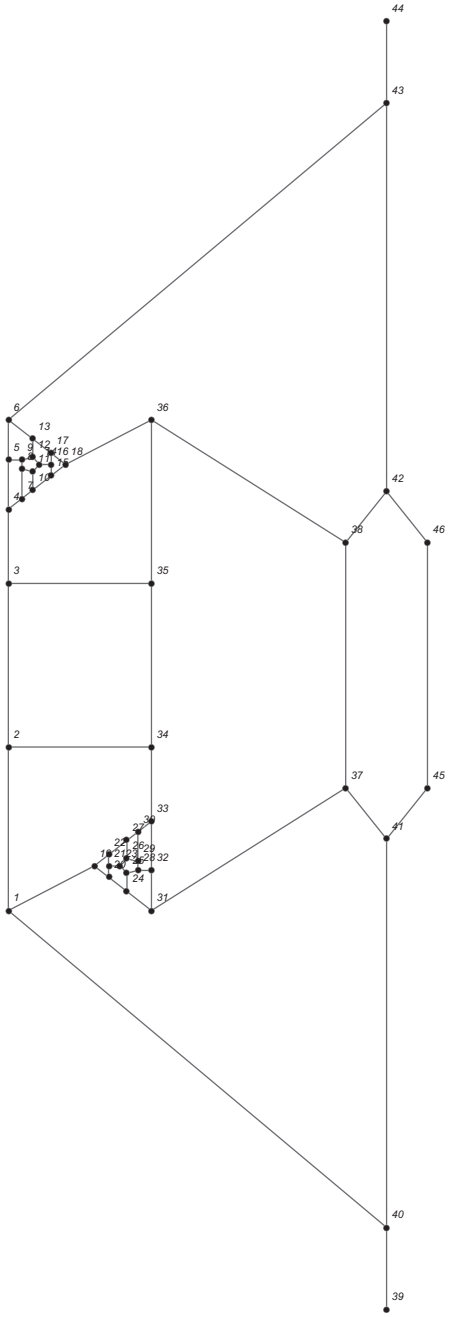
Edge list (SAGE 7.2):
[(0,1),(0,2),(0,3),(1,4),(1,5),(2,6),(2,8),(3,4),(3,6),(4,7),(5,7),(5,9),(6,7),(8,9),(8,11),(9,12),(10,11),(11,14),(12,13),(12,15),(14,15)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-5.18,-1.96],1:[-5.18,1.96],2:[-3.5,-2.94],3:[-3.5,-0.98],4:[-3.5,0.98],5:[-3.5,2.94],6:[-1.82,-1.96],7:[-1.82,1.96],8:[1.5,-6.],9:[1.5,6.],10:[3.5,-12.5],11:[3.5,-8.5],12:[3.5,8.5],13:[3.5,12.5],14:[5.5,-6.],15:[5.5,6.]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorTwo,3->colorTwo,4->colorTwo,5->colorOne,6->colorOne,7->colorOne,8->colorTwo,9->colorOne,10->colorTwo,11->colorOne,12->colorTwo,13->colorOne,14->colorTwo,15->colorOne,16->colorTwo}

8.41 (Figure 4.11.d) gadget



Graph Properties ::

--

Number of Vertices: ‘46’

Number of Edges: ‘66’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘3’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘3’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘False’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘False’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘False’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->2,1<->19,1<->40,2<->3,2<->34,3<->4,3<->35,4<->5,4<->7,5<->6,5<->9,6<->13,6<->43,7<->8,7<->10,8<->9,8<->11,9<->12,10<->11,10<->15,11<->14,12<->13,12<->14,13<->17,14<->16,15<->16,15<->18,16<->17,17<->18,18<->36,19<->20,19<->22,20<->21,20<->24,21<->22,21<->23,22<->27,23<->25,23<->26,24<->25,24<->31,25<->28,26<->27,26<->29,27<->30,28<->29,28<->32,29<->30,30<->33,31<->32,31<->37,32<->33,33<->34,34<->35,35<->36,36<->38,37<->38,37<->41,38<->42,39<->40,40<->41,41<->45,42<->43,42<->46,43<->44,45<->46}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-6.7609,-12.},2->{-6.7609,-4.},3->{-6.7609,4.},4->{-6.7609,7.6364},5->{-6.7609,10.0364},6->{-6.7609,12.},7->{-6.0963,8.1542},8->{-6.0963,9.6},9->{-6.0963,10.0364},10->{-5.5637,8.5693},11->{-5.5637,9.4652},12->{-5.5637,10.1712},13->{-5.5637,11.0671},14->{-5.2345,9.8182},15->{-4.6609,9.2727},16->{-4.6609,9.8182},17->{-4.6609,10.3636},18->{-3.9609,9.8182},19->{-2.5609,-9.8182},20->{-1.8609,-10.3636},21->{-1.8609,-9.8182},22->{-1.8609,-9.2727},23->{-1.2873,-9.8182},24->{-0.9581,-11.0671},25->{-0.9581,-10.1712},26->{-0.9581,-9.4652},27->{-0.9581,-8.5693},28->{-0.4255,-10.0364},29->{-0.4255,-9.6},30->{-0.4255,-8.1542},31->{0.2391,-12.},32->{0.2391,-10.0364},33->{0.2391,-7.6364},34->{0.2391,-4.},35->{0.2391,4.},36->{0.2391,12.},37->{9.7391,-6.},38->{9.7391,6.},39->{11.7391,-31.5},40->{11.7391,-27.5},41->{11.7391,-8.5},42->{11.7391,8.5},43->{11.7391,27.5},44->{11.7391,31.5},45->{13.7391,-6.},46->{13.7391,6.}}

Edge list (SAGE 7.2):

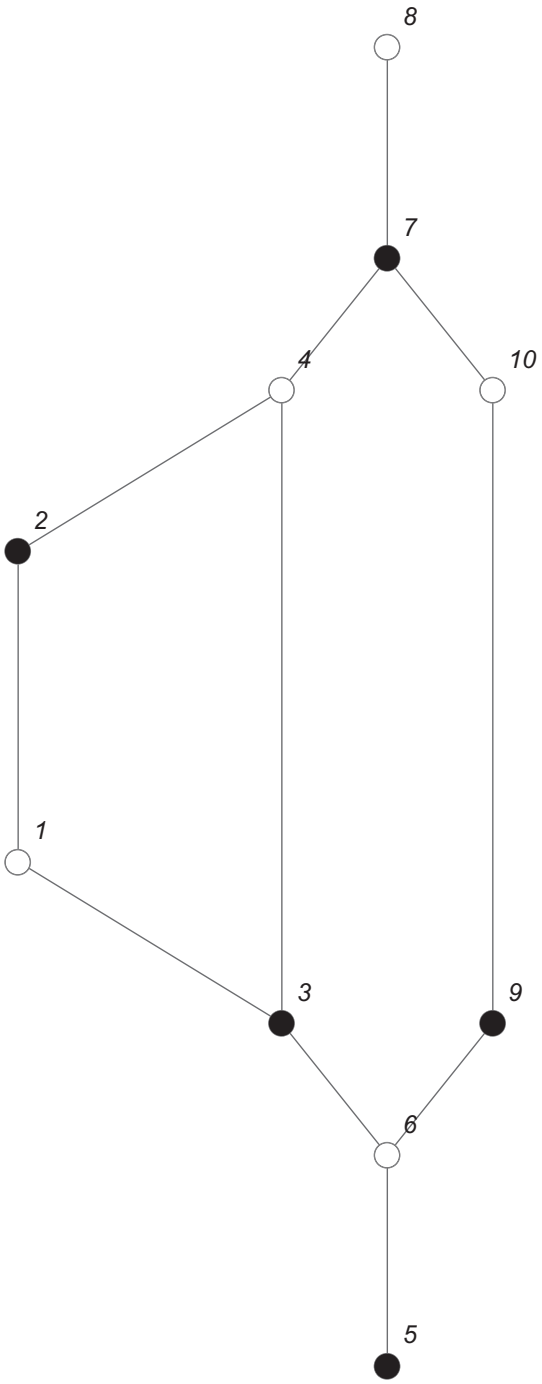
[(0,1),(0,18),(0,39),(1,2),(1,33),(2,3),(2,34),(3,4),(3,6),(4,5),(4,8),(5,12),(5,42),(6,7),(6,9),(7,8),(7,10),(8,11),(9,10),(9,14),(10,13),(11,12),(11,13),(12,16),(13,15),(14,15),(14,17),(15,16),(16,17),(17,35),(18,19),(18,21),(19,20),(19,23),(20,21),(20,22),(21,26),(22,24),(22,25),(23,24),(23,30),(24,27),(25,26),(25,28),(26,29),(27,28),(27,31),(28,29),(29,32),(30,31),(30,36),(31,32),(32,33),(33,34),(34,35),(35,37),(36,37),(36,40),(37,41),(38,39),(39,40),(40,44),(41,42),(41,45),(42,43),(44,45)]

-

Example embedding coordinates (SAGE 7.2):

{0:[-6.7609,-12.],1:[-6.7609,-4.],2:[-6.7609,4.],3:[-6.7609,7.6364],4:[-6.7609,10.0364],5:[-6.7609,12.],6:[-6.0963,8.1542],7:[-6.0963,9.6],8:[-6.0963,10.0364],9:[-5.5637,8.5693],10:[-5.5637,9.4652],11:[-5.5637,10.1712],12:[-5.5637,11.0671],13:[-5.2345,9.8182],14:[-4.6609,9.2727],15:[-4.6609,9.8182],16:[-4.6609,10.3636],17:[-3.9609,9.8182],18:[-2.5609,-9.8182],19:[-1.8609,-10.3636],20:[-1.8609,-9.8182],21:[-1.8609,-9.2727],22:[-1.2873,-9.8182],23:[-0.9581,-11.0671],24:[-0.9581,-10.1712],25:[-0.9581,-9.4652],26:[-0.9581,-8.5693],27:[-0.4255,-10.0364],28:[-0.4255,-9.6],29:[-0.4255,-8.1542],30:[0.2391,-12.],31:[0.2391,-10.0364],32:[0.2391,-7.6364],33:[0.2391,-4.],34:[0.2391,4.],35:[0.2391,12.],36:[9.7391,-6.],37:[9.7391,6.],38:[11.7391,-31.5],39:[11.7391,-27.5],40:[11.7391,-8.5],41:[11.7391,8.5],42:[11.7391,27.5],43:[11.7391,31.5],44:[13.7391,-6.],45:[13.7391,6.]}

8.42 (Figure 4.11.e) gadget



Graph Properties ::

--

Number of Vertices: ‘10’

Number of Edges: ‘11’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: '0'

Edge list (Mathematica 10.4.1):
{1<->2,1<->3,2<->4,3<->4,3<->6,4<->7,5<->6,6<->9,7<->8,7<->10,9<->10}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-5.6,-2.94},2->{-5.6,2.94},3->{-0.6,-6.},4->{-0.6,6.},5->{1.4,-12.5},6->{1.4,-8.5},7->{1.4,8.5},8->{1.4,12.5},9->{3.4,-6.},10->{3.4,6.}}

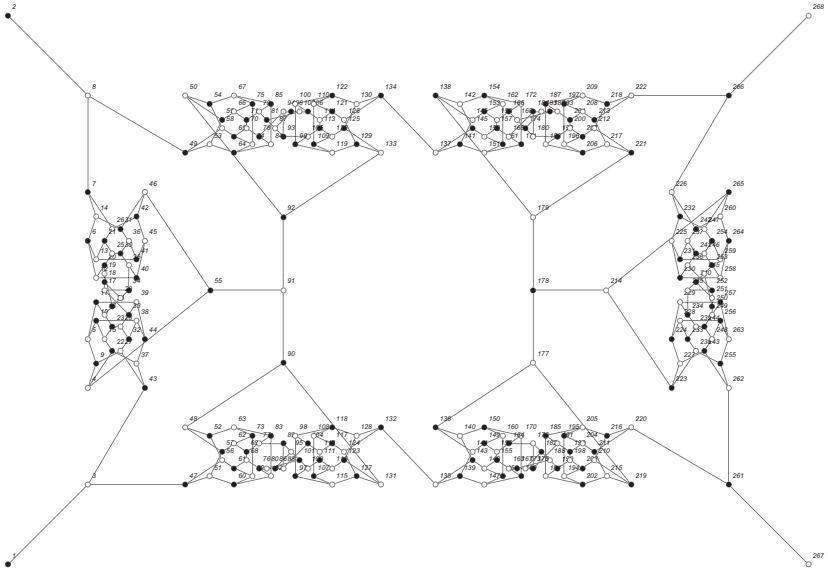
Edge list (SAGE 7.2):
[(0,1),(0,2),(1,3),(2,3),(2,5),(3,6),(4,5),(5,8),(6,7),(6,9),(8,9)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-5.6,-2.94],1:[-5.6,2.94],2:[-0.6,-6.],3:[-0.6,6.],4:[1.4,-12.5],5:[1.4,-8.5],6:[1.4,8.5],7:[1.4,12.5],8:[3.4,-6.],9:[3.4,6.]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorTwo,2->colorOne,3->colorOne,4->colorTwo,5->colorOne,6->colorTwo,7->colorOne,8->colorTwo,9->colorOne,10->colorTwo}

8.43 (Figure 4.13) gadget



Graph Properties ::

--

Number of Vertices: ‘268’

Number of Edges: ‘398’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘False’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘False’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘False’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):


```
{1<->3,2<->8,3<->43,3<->47,4<->9,4<->22,4<->55,5<->9,5<->10,5<->11,6<->12,6<->13,6<-
>14,7<->8,7<->14,7<->26,8<->49,9<->27,10<->15,10<->38,11<->16,11<->39,12<->20,12<-
>40,13<->21,13<->41,14<->31,15<->22,15<->23,16<->23,16<->24,17<->18,17<->33,17<-
>34,18<->19,18<->29,19<->20,19<->24,20<->25,21<->25,21<->26,22<->37,23<->28,24<-
>29,25<->30,26<->42,27<->32,27<->43,28<->32,28<->33,29<->34,30<->35,30<->36,31<-
>36,31<->46,32<->38,33<->39,34<->35,35<->40,36<->41,37<->43,37<->44,38<->44,39<-
>44,40<->45,41<->45,42<->45,42<->46,46<->55,47<->51,47<->56,48<->52,48<->57,48<-
>90,49<->53,49<->58,50<->54,50<->59,50<->92,51<->57,51<->60,52<->56,52<->63,53<-
>59,53<->64,54<->58,54<->67,55<->91,56<->61,57<->62,58<->65,59<->66,60<->72,60<-
>82,61<->68,61<->72,62<->69,62<->73,63<->73,63<->83,64<->74,64<->84,65<->70,65<-
>74,66<->71,66<->75,67<->75,67<->85,68<->69,68<->76,69<->77,70<->71,70<->78,71<-
>79,72<->73,74<->75,76<->80,76<->82,77<->83,77<->89,78<->84,78<->94,79<->81,79<-
>85,80<->86,80<->101,81<->87,81<->102,82<->83,84<->85,86<->88,86<->95,87<->93,87<-
>96,88<->89,88<->104,89<->95,90<->91,90<->131,91<->92,92<->133,93<->94,93<->105,94<-
>96,95<->101,96<->102,97<->98,97<->103,97<->115,98<->104,98<->118,99<->100,99<-
>105,99<->119,100<->106,100<->122,101<->103,102<->106,103<->111,104<->112,105<-
>113,106<->114,107<->108,107<->115,107<->116,108<->117,108<->118,109<->110,109<-
>119,109<->120,110<->121,110<->122,111<->112,111<->116,112<->117,113<->114,113<-
>120,114<->121,115<->127,116<->123,117<->124,118<->128,119<->129,120<->125,121<-
>126,122<->130,123<->128,123<->131,124<->127,124<->132,125<->130,125<->133,126<-
>129,126<->134,127<->131,128<->132,129<->133,130<->134,132<->135,134<->137,135<-
>139,135<->143,136<->140,136<->144,136<->177,137<->141,137<->145,138<->142,138<-
>146,138<->179,139<->144,139<->147,140<->143,140<->150,141<->146,141<->151,142<-
>145,142<->154,143<->148,144<->149,145<->152,146<->153,147<->159,147<->169,148<-
>155,148<->159,149<->156,149<->160,150<->160,150<->170,151<->161,151<->171,152<-
>157,152<->161,153<->158,153<->162,154<->162,154<->172,155<->156,155<->163,156<-
>164,157<->158,157<->165,158<->166,159<->160,161<->162,163<->167,163<->169,164<-
>170,164<->176,165<->171,165<->181,166<->168,166<->172,167<->173,167<->188,168<-
>174,168<->189,169<->170,171<->172,173<->175,173<->182,174<->180,174<->183,175<-
>176,175<->191,176<->182,177<->178,177<->219,178<->179,178<->214,179<->221,180<-
>181,180<->192,181<->183,182<->188,183<->189,184<->185,184<->190,184<->202,185<-
>191,185<->205,186<->187,186<->192,186<->206,187<->193,187<->209,188<->190,189<-
>193,190<->198,191<->199,192<->200,193<->201,194<->195,194<->202,194<->203,195<-
>204,195<->205,196<->197,196<->206,196<->207,197<->208,197<->209,198<->199,198<-
>203,199<->204,200<->201,200<->207,201<->208,202<->215,203<->210,204<->211,205<-
>216,206<->217,207<->212,208<->213,209<->218,210<->216,210<->219,211<->215,211<-
>220,212<->218,212<->221,213<->217,213<->222,214<->223,214<->265,215<->219,216<-
>220,217<->221,218<->222,220<->261,222<->266,223<->227,223<->238,224<->227,224<-
>228,224<->229,225<->230,225<->231,225<->232,226<->232,226<->242,226<->266,227<-
>243,228<->233,228<->256,229<->234,229<->257,230<->236,230<->258,231<->237,231<-
>259,232<->247,233<->238,233<->239,234<->235,234<->239,235<->240,235<->252,236<-
>241,236<->252,237<->241,237<->242,238<->255,239<->244,240<->245,240<->251,241<-
>246,242<->260,243<->248,243<->262,244<->248,244<->249,245<->250,245<->253,246<-
>253,246<->254,247<->254,247<->265,248<->256,249<->250,249<->257,250<->251,251<-
>252,253<->258,254<->259,255<->262,255<->263,256<->263,257<->263,258<->264,259<-
>264,260<->264,260<->265,261<->262,261<->267,266<->268}
```


-

Example embedding coordinates (Mathematica 10.4.1):


```
{1->{-17.5,-12.},2->{-17.5,12.},3->{-14.,-8.5},4->{-14.,-4.274},5->{-14.,-2.137},6->{-
14.,2.137},7->{-14.,4.274},8->{-14.,8.5},9->{-13.6438,-3.2055},10->{-13.6438,-
1.3356},11->{-13.6438,-0.5343},12->{-13.6438,0.5342},13->{-13.6438,1.3356},14->{-
13.6438,3.2055},15->{-13.2877,-2.137},16->{-13.2877,-1.0685},17->{-13.2877,0.},18->{-
```

13.2877,0.3561},19->{-13.2877,0.7123},20->{-13.2877,1.0685},21->{-13.2877,2.137},22-
>{-12.9315,-2.6713},23->{-12.9315,-1.6028},24->{-12.9315,-0.7123},25->{-
12.9315,1.6027},26->{-12.9315,2.6712},27->{-12.5753,-2.6713},28->{-12.5753,-
1.6028},29->{-12.5753,-0.3562},30->{-12.5753,1.6027},31->{-12.5753,2.6712},32->{-
12.2192,-2.137},33->{-12.2192,-1.0685},34->{-12.2192,0.},35->{-12.2192,1.0685},36->{-
12.2192,2.137},37->{-11.863,-3.2055},38->{-11.863,-1.3356},39->{-11.863,-0.5343},40-
>{-11.863,0.5342},41->{-11.863,1.3356},42->{-11.863,3.2055},43->{-11.5068,-4.274},44-
>{-11.5068,-2.137},45->{-11.5068,2.137},46->{-11.5068,4.274},47->{-9.7415,-8.5},48->{-
9.7415,-6.0069},49->{-9.7415,6.0068},50->{-9.7415,8.5},51->{-8.673,-8.1439},52->{-
8.673,-6.363},53->{-8.673,6.363},54->{-8.673,8.1438},55->{-8.6575,0.},56->{-8.1388,-
7.4315},57->{-8.1388,-7.0754},58->{-8.1388,7.0753},59->{-8.1388,7.4315},60->{-7.6045,-
8.5},61->{-7.6045,-7.7877},62->{-7.6045,-6.7192},63->{-7.6045,-6.0069},64->{-
7.6045,6.0068},65->{-7.6045,6.7192},66->{-7.6045,7.7877},67->{-7.6045,8.5},68->{-
7.0703,-7.4315},69->{-7.0703,-7.0754},70->{-7.0703,7.0753},71->{-7.0703,7.4315},72->{-
6.8031,-8.1439},73->{-6.8031,-6.363},74->{-6.8031,6.363},75->{-6.8031,8.1438},76->{-
6.536,-7.7877},77->{-6.536,-6.7192},78->{-6.536,6.7192},79->{-6.536,7.7877},80->{-
6.1798,-7.7877},81->{-6.1798,7.4315},82->{-6.0018,-8.1439},83->{-6.0018,-6.363},84->{-
6.0018,6.363},85->{-6.0018,8.1438},86->{-5.8237,-7.7877},87->{-5.8237,7.0753},88->{-
5.4675,-7.7877},89->{-5.4675,-6.7192},90->{-5.4675,-3.19},91->{-5.4675,0.},92->{-
5.4675,3.19},93->{-5.4675,6.7192},94->{-5.4675,7.7877},95->{-5.1113,-7.0754},96->{-
5.1113,7.7877},97->{-4.9333,-8.1439},98->{-4.9333,-6.363},99->{-4.9333,6.363},100->{-
4.9333,8.1438},101->{-4.7552,-7.4315},102->{-4.7552,7.7877},103->{-4.399,-7.7877},104-
>{-4.399,-6.7192},105->{-4.399,6.7192},106->{-4.399,7.7877},107->{-4.1319,-
8.1439},108->{-4.1319,-6.363},109->{-4.1319,6.363},110->{-4.1319,8.1438},111->{-
3.8648,-7.4315},112->{-3.8648,-7.0754},113->{-3.8648,7.0753},114->{-
3.8648,7.4315},115->{-3.3305,-8.5},116->{-3.3305,-7.7877},117->{-3.3305,-6.7192},118-
>{-3.3305,-6.0069},119->{-3.3305,6.0068},120->{-3.3305,6.7192},121->{-
3.3305,7.7877},122->{-3.3305,8.5},123->{-2.7963,-7.4315},124->{-2.7963,-7.0754},125-
>{-2.7963,7.0753},126->{-2.7963,7.4315},127->{-2.262,-8.1439},128->{-2.262,-
6.363},129->{-2.262,6.363},130->{-2.262,8.1438},131->{-1.1935,-8.5},132->{-1.1935,-
6.0069},133->{-1.1935,6.0068},134->{-1.1935,8.5},135->{1.1935,-8.5},136->{1.1935,-
6.0069},137->{1.1935,6.0068},138->{1.1935,8.5},139->{2.262,-8.1439},140->{2.262,-
6.363},141->{2.262,6.363},142->{2.262,8.1438},143->{2.7962,-7.4315},144->{2.7962,-
7.0754},145->{2.7962,7.0753},146->{2.7962,7.4315},147->{3.3305,-8.5},148->{3.3305,-
7.7877},149->{3.3305,-6.7192},150->{3.3305,-6.0069},151->{3.3305,6.0068},152-
>{3.3305,6.7192},153->{3.3305,7.7877},154->{3.3305,8.5},155->{3.8647,-7.4315},156-
>{3.8647,-7.0754},157->{3.8647,7.0753},158->{3.8647,7.4315},159->{4.1319,-8.1439},160-
>{4.1319,-6.363},161->{4.1319,6.363},162->{4.1319,8.1438},163->{4.399,-7.7877},164-
>{4.399,-6.7192},165->{4.399,6.7192},166->{4.399,7.7877},167->{4.7552,-7.7877},168-
>{4.7552,7.4315},169->{4.9332,-8.1439},170->{4.9332,-6.363},171->{4.9332,6.363},172-
>{4.9332,8.1438},173->{5.1113,-7.7877},174->{5.1113,7.0753},175->{5.4675,-7.7877},176-
>{5.4675,-6.7192},177->{5.4675,-3.19},178->{5.4675,0.},179->{5.4675,3.19},180-
>{5.4675,6.7192},181->{5.4675,7.7877},182->{5.8237,-7.0754},183->{5.8237,7.7877},184-
>{6.0017,-8.1439},185->{6.0017,-6.363},186->{6.0017,6.363},187->{6.0017,8.1438},188-
>{6.1798,-7.4315},189->{6.1798,7.7877},190->{6.536,-7.7877},191->{6.536,-6.7192},192-
>{6.536,6.7192},193->{6.536,7.7877},194->{6.8031,-8.1439},195->{6.8031,-6.363},196-
>{6.8031,6.363},197->{6.8031,8.1438},198->{7.0702,-7.4315},199->{7.0702,-7.0754},200-
>{7.0702,7.0753},201->{7.0702,7.4315},202->{7.6045,-8.5},203->{7.6045,-7.7877},204-
>{7.6045,-6.7192},205->{7.6045,-6.0069},206->{7.6045,6.0068},207->{7.6045,6.7192},208-
>{7.6045,7.7877},209->{7.6045,8.5},210->{8.1387,-7.4315},211->{8.1387,-7.0754},212-
>{8.1387,7.0753},213->{8.1387,7.4315},214->{8.6575,0.},215->{8.673,-8.1439},216-
>{8.673,-6.363},217->{8.673,6.363},218->{8.673,8.1438},219->{9.7415,-8.5},220-
>{9.7415,-6.0069},221->{9.7415,6.0068},222->{9.7415,8.5},223->{11.5068,-4.274},224-
>{11.5068,-2.137},225->{11.5068,2.137},226->{11.5068,4.274},227->{11.863,-3.2055},228-
>{11.863,-1.3356},229->{11.863,-0.5343},230->{11.863,0.5342},231->{11.863,1.3356},232-
>{11.863,3.2055},233->{12.2192,-2.137},234->{12.2192,-1.0685},235->{12.2192,0.},236-
>{12.2192,1.0685},237->{12.2192,2.137},238->{12.5753,-2.6713},239->{12.5753,-
1.6028},240->{12.5753,0.3561},241->{12.5753,1.6027},242->{12.5753,2.6712},243-
>{12.9315,-2.6713},244->{12.9315,-1.6028},245->{12.9315,0.7123},246-
>{12.9315,1.6027},247->{12.9315,2.6712},248->{13.2877,-2.137},249->{13.2877,-
1.0685},250->{13.2877,-0.7123},251->{13.2877,-0.3562},252->{13.2877,0.},253-
>{13.2877,1.0685},254->{13.2877,2.137},255->{13.6438,-3.2055},256->{13.6438,-
1.3356},257->{13.6438,-0.5343},258->{13.6438,0.5342},259->{13.6438,1.3356},260-

```
>{13.6438,3.2055},261->{14.,-8.5},262->{14.,-4.274},263->{14.,-2.137},264-
>{14.,2.137},265->{14.,4.274},266->{14.,8.5},267->{17.5,-12.},268->{17.5,12.}}
```

Edge list (SAGE 7.2):

```
[(0,2),(1,7),(2,42),(2,46),(3,8),(3,21),(3,54),(4,8),(4,9),(4,10),(5,11),(5,12),(5,13)
,(6,7),(6,13),(6,25),(7,48),(8,26),(9,14),(9,37),(10,15),(10,38),(11,19),(11,39),(12,2
0),(12,40),(13,30),(14,21),(14,22),(15,22),(15,23),(16,17),(16,32),(16,33),(17,18),(17
,28),(18,19),(18,23),(19,24),(20,24),(20,25),(21,36),(22,27),(23,28),(24,29),(25,41),(
26,31),(26,42),(27,31),(27,32),(28,33),(29,34),(29,35),(30,35),(30,45),(31,37),(32,38)
,(33,34),(34,39),(35,40),(36,42),(36,43),(37,43),(38,43),(39,44),(40,44),(41,44),(41,4
5),(45,54),(46,50),(46,55),(47,51),(47,56),(47,89),(48,52),(48,57),(49,53),(49,58),(49
,91),(50,56),(50,59),(51,55),(51,62),(52,58),(52,63),(53,57),(53,66),(54,90),(55,60),(
56,61),(57,64),(58,65),(59,71),(59,81),(60,67),(60,71),(61,68),(61,72),(62,72),(62,82)
,(63,73),(63,83),(64,69),(64,73),(65,70),(65,74),(66,74),(66,84),(67,68),(67,75),(68,7
6),(69,70),(69,77),(70,78),(71,72),(73,74),(75,79),(75,81),(76,82),(76,88),(77,83),(77
,93),(78,80),(78,84),(79,85),(79,100),(80,86),(80,101),(81,82),(83,84),(85,87),(85,94)
,(86,92),(86,95),(87,88),(87,103),(88,94),(89,90),(89,130),(90,91),(91,132),(92,93),(9
2,104),(93,95),(94,100),(95,101),(96,97),(96,102),(96,114),(97,103),(97,117),(98,99),(
98,104),(98,118),(99,105),(99,121),(100,102),(101,105),(102,110),(103,111),(104,112),(
105,113),(106,107),(106,114),(106,115),(107,116),(107,117),(108,109),(108,118),(108,11
9),(109,120),(109,121),(110,111),(110,115),(111,116),(112,113),(112,119),(113,120),(11
4,126),(115,122),(116,123),(117,127),(118,128),(119,124),(120,125),(121,129),(122,127)
,(122,130),(123,126),(123,131),(124,129),(124,132),(125,128),(125,133),(126,130),(127,
131),(128,132),(129,133),(131,134),(133,136),(134,138),(134,142),(135,139),(135,143),(
135,176),(136,140),(136,144),(137,141),(137,145),(137,178),(138,143),(138,146),(139,14
2),(139,149),(140,145),(140,150),(141,144),(141,153),(142,147),(143,148),(144,151),(14
5,152),(146,158),(146,168),(147,154),(147,158),(148,155),(148,159),(149,159),(149,169)
,(150,160),(150,170),(151,156),(151,160),(152,157),(152,161),(153,161),(153,171),(154,
155),(154,162),(155,163),(156,157),(156,164),(157,165),(158,159),(160,161),(162,166),(
162,168),(163,169),(163,175),(164,170),(164,180),(165,167),(165,171),(166,172),(166,18
7),(167,173),(167,188),(168,169),(170,171),(172,174),(172,181),(173,179),(173,182),(17
4,175),(174,190),(175,181),(176,177),(176,218),(177,178),(177,213),(178,220),(179,180)
,(179,191),(180,182),(181,187),(182,188),(183,184),(183,189),(183,201),(184,190),(184,
204),(185,186),(185,191),(185,205),(186,192),(186,208),(187,189),(188,192),(189,197),(
190,198),(191,199),(192,200),(193,194),(193,201),(193,202),(194,203),(194,204),(195,19
6),(195,205),(195,206),(196,207),(196,208),(197,198),(197,202),(198,203),(199,200),(19
9,206),(200,207),(201,214),(202,209),(203,210),(204,215),(205,216),(206,211),(207,212)
,(208,217),(209,215),(209,218),(210,214),(210,219),(211,217),(211,220),(212,216),(212,
221),(213,222),(213,264),(214,218),(215,219),(216,220),(217,221),(219,260),(221,265),(
222,226),(222,237),(223,226),(223,227),(223,228),(224,229),(224,230),(224,231),(225,23
1),(225,241),(225,265),(226,242),(227,232),(227,255),(228,233),(228,256),(229,235),(22
9,257),(230,236),(230,258),(231,246),(232,237),(232,238),(233,234),(233,238),(234,239)
,(234,251),(235,240),(235,251),(236,240),(236,241),(237,254),(238,243),(239,244),(239,
250),(240,245),(241,259),(242,247),(242,261),(243,247),(243,248),(244,249),(244,252),(
245,252),(245,253),(246,253),(246,264),(247,255),(248,249),(248,256),(249,250),(250,25
1),(252,257),(253,258),(254,261),(254,262),(255,262),(256,262),(257,263),(258,263),(25
9,263),(259,264),(260,261),(260,266),(265,267)]
```

-

Example embedding coordinates (SAGE 7.2):

```
{0:[-17.5,-12.],1:[-17.5,12.],2:[-14.,-8.5],3:[-14.,-4.274],4:[-14.,-2.137],5:[-
14.,2.137],6:[-14.,4.274],7:[-14.,8.5],8:[-13.6438,-3.2055],9:[-13.6438,-1.3356],10:[-
13.6438,-0.5343],11:[-13.6438,0.5342],12:[-13.6438,1.3356],13:[-13.6438,3.2055],14:[-
13.2877,-2.137],15:[-13.2877,-1.0685],16:[-13.2877,0.],17:[-13.2877,0.3561],18:[-
13.2877,0.7123],19:[-13.2877,1.0685],20:[-13.2877,2.137],21:[-12.9315,-2.6713],22:[-
12.9315,-1.6028],23:[-12.9315,-0.7123],24:[-12.9315,1.6027],25:[-12.9315,2.6712],26:[-
12.5753,-2.6713],27:[-12.5753,-1.6028],28:[-12.5753,-0.3562],29:[-
12.5753,1.6027],30:[-12.5753,2.6712],31:[-12.2192,-2.137],32:[-12.2192,-1.0685],33:[-
12.2192,0.],34:[-12.2192,1.0685],35:[-12.2192,2.137],36:[-11.863,-3.2055],37:[-
11.863,-1.3356],38:[-11.863,-0.5343],39:[-11.863,0.5342],40:[-11.863,1.3356],41:[-
```

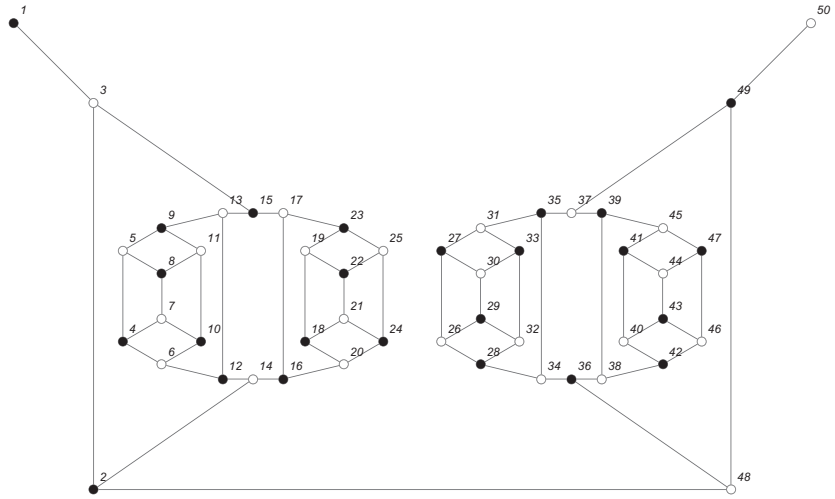
11.863,3.2055],42:[-11.5068,-4.274],43:[-11.5068,-2.137],44:[-11.5068,2.137],45:[-
11.5068,4.274],46:[-9.7415,-8.5],47:[-9.7415,-6.0069],48:[-9.7415,6.0068],49:[-
9.7415,8.5],50:[-8.673,-8.1439],51:[-8.673,-6.363],52:[-8.673,6.363],53:[-
8.673,8.1438],54:[-8.6575,0.],55:[-8.1388,-7.4315],56:[-8.1388,-7.0754],57:[-
8.1388,7.0753],58:[-8.1388,7.4315],59:[-7.6045,-8.5],60:[-7.6045,-7.7877],61:[-
7.6045,-6.7192],62:[-7.6045,-6.0069],63:[-7.6045,6.0068],64:[-7.6045,6.7192],65:[-
7.6045,7.7877],66:[-7.6045,8.5],67:[-7.0703,-7.4315],68:[-7.0703,-7.0754],69:[-
7.0703,7.0753],70:[-7.0703,7.4315],71:[-6.8031,-8.1439],72:[-6.8031,-6.363],73:[-
6.8031,6.363],74:[-6.8031,8.1438],75:[-6.536,-7.7877],76:[-6.536,-6.7192],77:[-
6.536,6.7192],78:[-6.536,7.7877],79:[-6.1798,-7.7877],80:[-6.1798,7.4315],81:[-
6.0018,-8.1439],82:[-6.0018,-6.363],83:[-6.0018,6.363],84:[-6.0018,8.1438],85:[-
5.8237,-7.7877],86:[-5.8237,7.0753],87:[-5.4675,-7.7877],88:[-5.4675,-6.7192],89:[-
5.4675,-3.19],90:[-5.4675,0.],91:[-5.4675,3.19],92:[-5.4675,6.7192],93:[-
5.4675,7.7877],94:[-5.1113,-7.0754],95:[-5.1113,7.7877],96:[-4.9333,-8.1439],97:[-
4.9333,-6.363],98:[-4.9333,6.363],99:[-4.9333,8.1438],100:[-4.7552,-7.4315],101:[-
4.7552,7.7877],102:[-4.399,-7.7877],103:[-4.399,-6.7192],104:[-4.399,6.7192],105:[-
4.399,7.7877],106:[-4.1319,-8.1439],107:[-4.1319,-6.363],108:[-4.1319,6.363],109:[-
4.1319,8.1438],110:[-3.8648,-7.4315],111:[-3.8648,-7.0754],112:[-3.8648,7.0753],113:[-
3.8648,7.4315],114:[-3.3305,-8.5],115:[-3.3305,-7.7877],116:[-3.3305,-6.7192],117:[-
3.3305,-6.0069],118:[-3.3305,6.0068],119:[-3.3305,6.7192],120:[-3.3305,7.7877],121:[-
3.3305,8.5],122:[-2.7963,-7.4315],123:[-2.7963,-7.0754],124:[-2.7963,7.0753],125:[-
2.7963,7.4315],126:[-2.262,-8.1439],127:[-2.262,-6.363],128:[-2.262,6.363],129:[-
2.262,8.1438],130:[-1.1935,-8.5],131:[-1.1935,-6.0069],132:[-1.1935,6.0068],133:[-
1.1935,8.5],134:[1.1935,-8.5],135:[1.1935,-
6.0069],136:[1.1935,6.0068],137:[1.1935,8.5],138:[2.262,-8.1439],139:[2.262,-
6.363],140:[2.262,6.363],141:[2.262,8.1438],142:[2.7962,-7.4315],143:[2.7962,-
7.0754],144:[2.7962,7.0753],145:[2.7962,7.4315],146:[3.3305,-8.5],147:[3.3305,-
7.7877],148:[3.3305,-6.7192],149:[3.3305,-
6.0069],150:[3.3305,6.0068],151:[3.3305,6.7192],152:[3.3305,7.7877],153:[3.3305,8.5],1
54:[3.8647,-7.4315],155:[3.8647,-
7.0754],156:[3.8647,7.0753],157:[3.8647,7.4315],158:[4.1319,-8.1439],159:[4.1319,-
6.363],160:[4.1319,6.363],161:[4.1319,8.1438],162:[4.399,-7.7877],163:[4.399,-
6.7192],164:[4.399,6.7192],165:[4.399,7.7877],166:[4.7552,-
7.7877],167:[4.7552,7.4315],168:[4.9332,-8.1439],169:[4.9332,-
6.363],170:[4.9332,6.363],171:[4.9332,8.1438],172:[5.1113,-
7.7877],173:[5.1113,7.0753],174:[5.4675,-7.7877],175:[5.4675,-6.7192],176:[5.4675,-
3.19],177:[5.4675,0.],178:[5.4675,3.19],179:[5.4675,6.7192],180:[5.4675,7.7877],181:[5
.8237,-7.0754],182:[5.8237,7.7877],183:[6.0017,-8.1439],184:[6.0017,-
6.363],185:[6.0017,6.363],186:[6.0017,8.1438],187:[6.1798,-
7.4315],188:[6.1798,7.7877],189:[6.536,-7.7877],190:[6.536,-
6.7192],191:[6.536,6.7192],192:[6.536,7.7877],193:[6.8031,-8.1439],194:[6.8031,-
6.363],195:[6.8031,6.363],196:[6.8031,8.1438],197:[7.0702,-7.4315],198:[7.0702,-
7.0754],199:[7.0702,7.0753],200:[7.0702,7.4315],201:[7.6045,-8.5],202:[7.6045,-
7.7877],203:[7.6045,-6.7192],204:[7.6045,-
6.0069],205:[7.6045,6.0068],206:[7.6045,6.7192],207:[7.6045,7.7877],208:[7.6045,8.5],2
09:[8.1387,-7.4315],210:[8.1387,-
7.0754],211:[8.1387,7.0753],212:[8.1387,7.4315],213:[8.6575,0.],214:[8.673,-
8.1439],215:[8.673,-6.363],216:[8.673,6.363],217:[8.673,8.1438],218:[9.7415,-
8.5],219:[9.7415,-6.0069],220:[9.7415,6.0068],221:[9.7415,8.5],222:[11.5068,-
4.274],223:[11.5068,-2.137],224:[11.5068,2.137],225:[11.5068,4.274],226:[11.863,-
3.2055],227:[11.863,-1.3356],228:[11.863,-
0.5343],229:[11.863,0.5342],230:[11.863,1.3356],231:[11.863,3.2055],232:[12.2192,-
2.137],233:[12.2192,-
1.0685],234:[12.2192,0.],235:[12.2192,1.0685],236:[12.2192,2.137],237:[12.5753,-
2.6713],238:[12.5753,-
1.6028],239:[12.5753,0.3561],240:[12.5753,1.6027],241:[12.5753,2.6712],242:[12.9315,-
2.6713],243:[12.9315,-
1.6028],244:[12.9315,0.7123],245:[12.9315,1.6027],246:[12.9315,2.6712],247:[13.2877,-
2.137],248:[13.2877,-1.0685],249:[13.2877,-0.7123],250:[13.2877,-
0.3562],251:[13.2877,0.],252:[13.2877,1.0685],253:[13.2877,2.137],254:[13.6438,-
3.2055],255:[13.6438,-1.3356],256:[13.6438,-
0.5343],257:[13.6438,0.5342],258:[13.6438,1.3356],259:[13.6438,3.2055],260:[14.,-

8.5],261:[14.,-4.274],262:[14.,-
2.137],263:[14.,2.137],264:[14.,4.274],265:[14.,8.5],266:[17.5,-12.],267:[17.5,12.]}

Canonical vertex $(k = 2)$-coloring ::

{1->colorOne,2->colorOne,3->colorTwo,4->colorTwo,5->colorTwo,6->colorOne,7-
>colorOne,8->colorTwo,9->colorOne,10->colorOne,11->colorOne,12->colorTwo,13-
>colorTwo,14->colorTwo,15->colorTwo,16->colorTwo,17->colorTwo,18->colorOne,19-
>colorTwo,20->colorOne,21->colorOne,22->colorOne,23->colorOne,24->colorOne,25-
>colorTwo,26->colorTwo,27->colorTwo,28->colorTwo,29->colorTwo,30->colorOne,31-
>colorOne,32->colorOne,33->colorOne,34->colorOne,35->colorTwo,36->colorTwo,37-
>colorTwo,38->colorTwo,39->colorTwo,40->colorOne,41->colorOne,42->colorOne,43-
>colorOne,44->colorOne,45->colorTwo,46->colorTwo,47->colorOne,48->colorTwo,49-
>colorOne,50->colorTwo,51->colorTwo,52->colorOne,53->colorTwo,54->colorOne,55-
>colorOne,56->colorTwo,57->colorOne,58->colorTwo,59->colorOne,60->colorOne,61-
>colorOne,62->colorTwo,63->colorTwo,64->colorOne,65->colorOne,66->colorTwo,67-
>colorTwo,68->colorTwo,69->colorOne,70->colorTwo,71->colorOne,72->colorTwo,73-
>colorOne,74->colorTwo,75->colorOne,76->colorOne,77->colorTwo,78->colorOne,79-
>colorTwo,80->colorTwo,81->colorOne,82->colorTwo,83->colorOne,84->colorTwo,85-
>colorOne,86->colorOne,87->colorTwo,88->colorTwo,89->colorOne,90->colorOne,91-
>colorTwo,92->colorOne,93->colorOne,94->colorTwo,95->colorTwo,96->colorOne,97-
>colorOne,98->colorTwo,99->colorOne,100->colorTwo,101->colorOne,102->colorTwo,103-
>colorTwo,104->colorOne,105->colorTwo,106->colorOne,107->colorOne,108->colorTwo,109-
>colorOne,110->colorTwo,111->colorOne,112->colorTwo,113->colorOne,114->colorTwo,115-
>colorTwo,116->colorTwo,117->colorOne,118->colorOne,119->colorTwo,120->colorTwo,121-
>colorOne,122->colorOne,123->colorOne,124->colorTwo,125->colorOne,126->colorTwo,127-
>colorOne,128->colorTwo,129->colorOne,130->colorTwo,131->colorTwo,132->colorOne,133-
>colorTwo,134->colorOne,135->colorTwo,136->colorOne,137->colorTwo,138->colorOne,139-
>colorOne,140->colorTwo,141->colorOne,142->colorTwo,143->colorOne,144->colorTwo,145-
>colorOne,146->colorTwo,147->colorTwo,148->colorTwo,149->colorOne,150->colorOne,151-
>colorTwo,152->colorTwo,153->colorOne,154->colorOne,155->colorOne,156->colorTwo,157-
>colorOne,158->colorTwo,159->colorOne,160->colorTwo,161->colorOne,162->colorTwo,163-
>colorTwo,164->colorOne,165->colorTwo,166->colorOne,167->colorOne,168->colorTwo,169-
>colorOne,170->colorTwo,171->colorOne,172->colorTwo,173->colorTwo,174->colorOne,175-
>colorOne,176->colorTwo,177->colorTwo,178->colorOne,179->colorTwo,180->colorTwo,181-
>colorOne,182->colorOne,183->colorTwo,184->colorTwo,185->colorOne,186->colorTwo,187-
>colorOne,188->colorTwo,189->colorOne,190->colorOne,191->colorTwo,192->colorOne,193-
>colorTwo,194->colorTwo,195->colorOne,196->colorTwo,197->colorOne,198->colorTwo,199-
>colorOne,200->colorTwo,201->colorOne,202->colorOne,203->colorOne,204->colorTwo,205-
>colorTwo,206->colorOne,207->colorOne,208->colorTwo,209->colorTwo,210->colorTwo,211-
>colorOne,212->colorTwo,213->colorOne,214->colorTwo,215->colorTwo,216->colorOne,217-
>colorTwo,218->colorOne,219->colorOne,220->colorTwo,221->colorOne,222->colorTwo,223-
>colorOne,224->colorOne,225->colorTwo,226->colorTwo,227->colorTwo,228->colorTwo,229-
>colorTwo,230->colorOne,231->colorOne,232->colorOne,233->colorOne,234->colorOne,235-
>colorTwo,236->colorTwo,237->colorTwo,238->colorTwo,239->colorTwo,240->colorOne,241-
>colorOne,242->colorOne,243->colorOne,244->colorOne,245->colorTwo,246->colorTwo,247-
>colorTwo,248->colorTwo,249->colorTwo,250->colorOne,251->colorTwo,252->colorOne,253-
>colorOne,254->colorOne,255->colorOne,256->colorOne,257->colorOne,258->colorTwo,259-
>colorTwo,260->colorTwo,261->colorOne,262->colorTwo,263->colorTwo,264->colorOne,265-
>colorOne,266->colorOne,267->colorTwo,268->colorTwo}

8.44 (Figure 4.14) gadget



Graph Properties ::

--

Number of Vertices: ‘50’

Number of Edges: ‘73’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->3,2<->3,2<->14,2<->48,3<->15,4<->5,4<->6,4<->7,5<->8,5<->9,6<->10,6<->12,7<->8,7<->10,8<->11,9<->11,9<->13,10<->11,12<->13,12<->14,13<->15,14<->16,15<->17,16<->17,16<->20,17<->23,18<->19,18<->20,18<->21,19<->22,19<->23,20<->24,21<->22,21<->24,22<->25,23<->25,24<->25,26<->27,26<->28,26<->29,27<->30,27<->31,28<->32,28<->34,29<->30,29<->32,30<->33,31<->33,31<->35,32<->33,34<->35,34<->36,35<->37,36<->38,36<->48,37<->39,37<->49,38<->39,38<->42,39<->45,40<->41,40<->42,40<->43,41<->44,41<->45,42<->46,43<->44,43<->46,44<->47,45<->47,46<->47,48<->49,49<->50}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-17.5,11.52},2->{-14.,-8.98},3->{-14.,8.02},4->{-12.7143,-2.48},5->{-12.7143,1.52},6->{-11.,-3.48},7->{-11.,-1.48},8->{-11.,0.52},9->{-11.,2.52},10->{-9.2857,-2.48},11->{-9.2857,1.52},12->{-8.33,-4.1229},13->{-8.33,3.1628},14->{-7.,-4.1229},15->{-7.,3.1628},16->{-5.67,-4.1229},17->{-5.67,3.1628},18->{-4.7143,-2.48},19->{-4.7143,1.52},20->{-3.,-3.48},21->{-3.,-1.48},22->{-3.,0.52},23->{-3.,2.52},24->{-1.2857,-2.48},25->{-1.2857,1.52},26->{1.2857,-2.48},27->{1.2857,1.52},28->{3.,-3.48},29->{3.,-1.48},30->{3.,0.52},31->{3.,2.52},32->{4.7143,-2.48},33->{4.7143,1.52},34->{5.67,-4.1229},35->{5.67,3.1628},36->{7.,-4.1229},37->{7.,3.1628},38->{8.33,-4.1229},39->{8.33,3.1628},40->{9.2857,-2.48},41->{9.2857,1.52},42->{11.,-3.48},43->{11.,-1.48},44->{11.,0.52},45->{11.,2.52},46->{12.7143,-2.48},47->{12.7143,1.52},48->{14.,-8.98},49->{14.,8.02},50->{17.5,11.52}}

Edge list (SAGE 7.2):

[(0,2),(1,2),(1,13),(1,47),(2,14),(3,4),(3,5),(3,6),(4,7),(4,8),(5,9),(5,11),(6,7),(6,9),(7,10),(8,10),(8,12),(9,10),(11,12),(11,13),(12,14),(13,15),(14,16),(15,16),(15,19),(16,22),(17,18),(17,19),(17,20),(18,21),(18,22),(19,23),(20,21),(20,23),(21,24),(22,24),(23,24),(25,26),(25,27),(25,28),(26,29),(26,30),(27,31),(27,33),(28,29),(28,31),(29,32),(30,32),(30,34),(31,32),(33,34),(33,35),(34,36),(35,37),(35,47),(36,38),(36,48),(37,38),(37,41),(38,44),(39,40),(39,41),(39,42),(40,43),(40,44),(41,45),(42,43),(42,45),(43,46),(44,46),(45,46),(47,48),(48,49)]

-

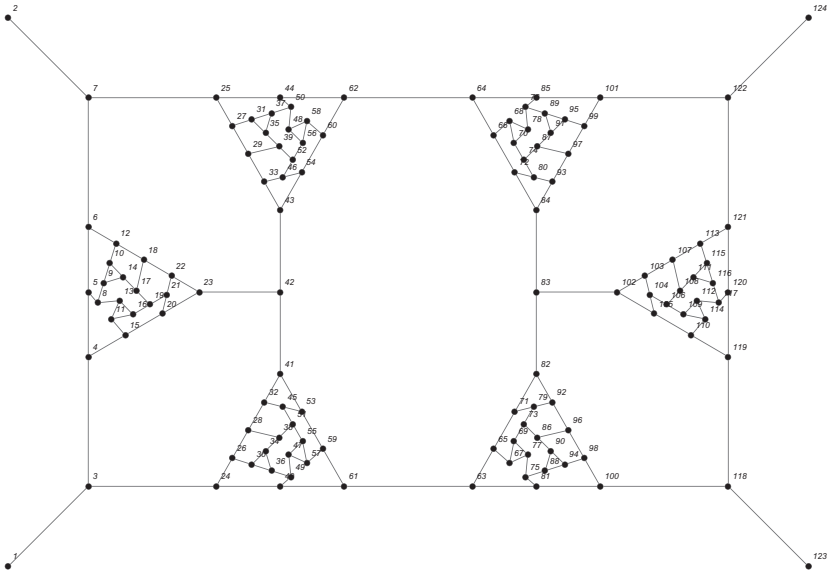
Example embedding coordinates (SAGE 7.2):

{0:[-17.5,11.52],1:[-14.,-8.98],2:[-14.,8.02],3:[-12.7143,-2.48],4:[-12.7143,1.52],5:[-11.,-3.48],6:[-11.,-1.48],7:[-11.,0.52],8:[-11.,2.52],9:[-9.2857,-2.48],10:[-9.2857,1.52],11:[-8.33,-4.1229],12:[-8.33,3.1628],13:[-7.,-4.1229],14:[-7.,3.1628],15:[-5.67,-4.1229],16:[-5.67,3.1628],17:[-4.7143,-2.48],18:[-4.7143,1.52],19:[-3.,-3.48],20:[-3.,-1.48],21:[-3.,0.52],22:[-3.,2.52],23:[-1.2857,-2.48],24:[-1.2857,1.52],25:[1.2857,-2.48],26:[1.2857,1.52],27:[3.,-3.48],28:[3.,-1.48],29:[3.,0.52],30:[3.,2.52],31:[4.7143,-2.48],32:[4.7143,1.52],33:[5.67,-4.1229],34:[5.67,3.1628],35:[7.,-4.1229],36:[7.,3.1628],37:[8.33,-4.1229],38:[8.33,3.1628],39:[9.2857,-2.48],40:[9.2857,1.52],41:[11.,-3.48],42:[11.,-1.48],43:[11.,0.52],44:[11.,2.52],45:[12.7143,-2.48],46:[12.7143,1.52],47:[14.,-8.98],48:[14.,8.02],49:[17.5,11.52]}

Canonical vertex $(k = 2)$-coloring ::

{1->colorOne,2->colorOne,3->colorTwo,4->colorOne,5->colorTwo,6->colorTwo,7-
>colorTwo,8->colorOne,9->colorOne,10->colorOne,11->colorTwo,12->colorOne,13-
>colorTwo,14->colorTwo,15->colorOne,16->colorOne,17->colorTwo,18->colorOne,19-
>colorTwo,20->colorTwo,21->colorTwo,22->colorOne,23->colorOne,24->colorOne,25-
>colorTwo,26->colorTwo,27->colorOne,28->colorOne,29->colorOne,30->colorTwo,31-
>colorTwo,32->colorTwo,33->colorOne,34->colorTwo,35->colorOne,36->colorOne,37-
>colorTwo,38->colorTwo,39->colorOne,40->colorTwo,41->colorOne,42->colorOne,43-
>colorOne,44->colorTwo,45->colorTwo,46->colorTwo,47->colorOne,48->colorTwo,49-
>colorOne,50->colorTwo}

8.45 (Figure 4.15) gadget



Graph Properties ::

--

Number of Vertices: ‘124’

Number of Edges: ‘182’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘3’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘3’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘False’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘False’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘False’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘3’

Output of SAGE 7.2’s ‘girth()’ function: ‘3’

Output for the ‘igraph’ R package ‘girth()’ function: ‘3’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->3,2<->7,3<->4,3<->24,4<->5,4<->15,5<->6,5<->8,6<->7,6<->12,7<->25,8<->9,8<->13,9<->10,9<->14,10<->12,10<->14,11<->13,11<->15,11<->16,12<->18,13<->16,14<->17,15<->20,16<->19,17<->18,17<->19,18<->22,19<->21,20<->21,20<->23,21<->22,22<->23,23<->42,24<->26,24<->40,25<->27,25<->44,26<->28,26<->30,27<->29,27<->31,28<->32,28<->38,29<->33,29<->39,30<->34,30<->36,31<->35,31<->37,32<->41,32<->45,33<->43,33<->46,34<->36,34<->38,35<->37,35<->39,36<->49,37<->50,38<->51,39<->52,40<->49,40<->61,41<->42,41<->53,42<->43,43<->54,44<->50,44<->62,45<->51,45<->53,46<->52,46<->54,47<->49,47<->55,47<->57,48<->50,48<->56,48<->58,51<->55,52<->56,53<->59,54<->60,55<->57,56<->58,57<->59,58<->60,59<->61,60<->62,61<->63,62<->64,63<->65,63<->81,64<->66,64<->85,65<->67,65<->71,66<->68,66<->72,67<->69,67<->77,68<->70,68<->78,69<->73,69<->77,70<->74,70<->78,71<->79,71<->82,72<->80,72<->84,73<->79,73<->86,74<->80,74<->87,75<->77,75<->81,75<->88,76<->78,76<->85,76<->89,79<->92,80<->93,81<->100,82<->83,82<->92,83<->84,83<->102,84<->93,85<->101,86<->90,86<->96,87<->91,87<->97,88<->90,88<->94,89<->91,89<->95,90<->94,91<->95,92<->96,93<->97,94<->98,95<->99,96<->98,97<->99,98<->100,99<->101,100<->118,101<->122,102<->103,102<->105,103<->104,103<->107,104<->105,104<->106,105<->110,106<->108,106<->109,107<->108,107<->113,108<->111,109<->112,109<->114,110<->114,110<->119,111<->115,111<->116,112<->114,112<->117,113<->115,113<->121,115<->116,116<->117,117<->120,118<->119,118<->123,119<->120,120<->121,121<->122,122<->124}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-17.5,-12.0015},2->{-17.5,11.9985},3->{-14.,-8.5015},4->{-14.,-2.8349},5->{-14.,-0.0015},6->{-14.,2.8318},7->{-14.,8.4985},8->{-13.5957,-0.4736},9->{-13.3264,0.3917},10->{-13.057,1.2576},11->{-12.9898,-1.1819},12->{-12.7876,2.1235},13->{-12.632,-0.3795},14->{-12.4735,0.6541},15->{-12.3833,-1.8902},16->{-12.0468,-0.9853},17->{-11.8894,0.0512},18->{-11.5752,1.4151},19->{-11.3059,-0.5523},20->{-10.7666,-0.9462},21->{-10.565,-0.1194},22->{-10.3628,0.7068},23->{-9.1504,-0.0015},24->{-8.4,-8.5015},25->{-8.4,8.4985},26->{-7.7,-7.2747},27->{-7.7,7.2717},28->{-7.,-6.0479},29->{-7.,6.0448},30->{-6.8443,-7.5473},31->{-6.8443,7.5442},32->{-6.3,-4.821},33->{-6.3,4.818},34->{-6.2479,-6.9568},35->{-6.2479,6.9538},36->{-5.9886,-7.8198},37->{-5.9886,7.8168},38->{-5.6521,-6.3658},39->{-5.6521,6.3627},40->{-5.6,-8.5015},41->{-5.6,-3.5942},42->{-5.6,-0.0015},43->{-5.6,3.5912},44->{-5.6,8.4985},45->{-5.4835,-5.0256},46->{-5.4835,5.0226},47->{-5.2265,-7.1172},48->{-5.2265,7.1141},49->{-5.1335,-8.0924},50->{-5.1335,8.0894},51->{-5.0557,-5.7753},52->{-5.0557,5.7723},53->{-4.6665,-5.2296},54->{-4.6665,5.2266},55->{-4.6278,-6.525},56->{-4.6278,6.522},57->{-4.4335,-7.4793},58->{-4.4335,7.4762},59->{-3.7335,-6.8656},60->{-3.7335,6.8625},61->{-2.8,-8.5015},62->{-2.8,8.4985},63->{2.8,-8.5015},64->{2.8,8.4985},65->{3.7335,-6.8656},66->{3.7335,6.8625},67->{4.4335,-7.4793},68->{4.4335,7.4762},69->{4.6278,-6.525},70->{4.6278,6.522},71->{4.6665,-5.2296},72->{4.6665,5.2266},73->{5.0557,-5.7753},74->{5.0557,5.7723},75->{5.1335,-8.0924},76->{5.1335,8.0894},77->{5.2265,-7.1172},78->{5.2265,7.1141},79->{5.4835,-5.0256},80->{5.4835,5.0226},81->{5.6,-8.5015},82->{5.6,-3.5942},83->{5.6,-0.0015},84->{5.6,3.5912},85->{5.6,8.4985},86->{5.6521,-6.3658},87->{5.6521,6.3627},88->{5.9886,-7.8198},89->{5.9886,7.8168},90->{6.2479,-6.9568},91->{6.2479,6.9538},92->{6.3,-4.821},93->{6.3,4.818},94->{6.8443,-7.5473},95->{6.8443,7.5442},96->{7.,-6.0479},97->{7.,6.0448},98->{7.7,-7.2747},99->{7.7,7.2717},100->{8.4,-8.5015},101->{8.4,8.4985},102->{9.1504,-0.0015},103->{10.3628,0.7069},104->{10.565,-0.1193},105->{10.7666,-0.9461},106->{11.3058,-0.5523},107->{11.5752,1.4152},108->{11.8893,0.0512},109->{12.0467,-0.9852},110->{12.3832,-1.8902},111->{12.4734,0.6542},112->{12.6319,-0.3794},113->{12.7876,2.1235},114->{12.9897,-1.1818},115->{13.0569,1.2577},116->{13.3263,0.3918},117->{13.5956,-0.4735},118->{14.,-8.5015},119->{14.,-2.8348},120-

>{14.,-0.0015},121->{14.,2.8319},122->{14.,8.4985},123->{17.5,-12.0015},124-
>{17.5,11.9985}}

Edge list (SAGE 7.2):

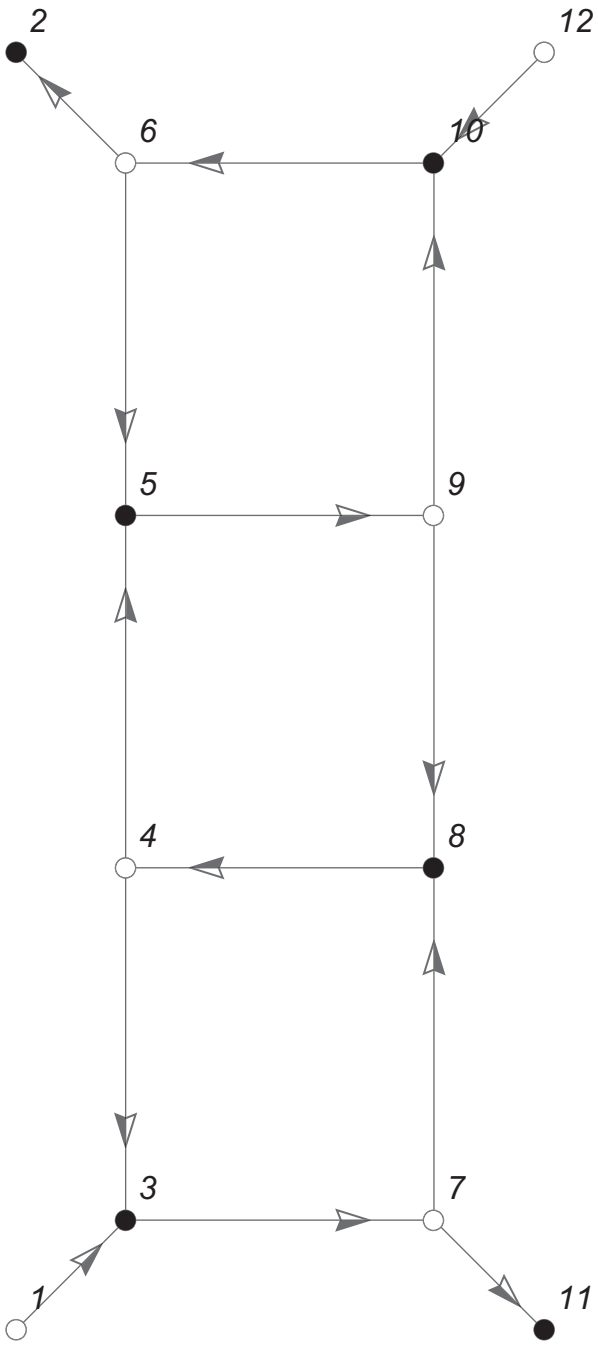
```
[(0,2),(1,6),(2,3),(2,23),(3,4),(3,14),(4,5),(4,7),(5,6),(5,11),(6,24),(7,8),(7,12),(8
,9),(8,13),(9,11),(9,13),(10,12),(10,14),(10,15),(11,17),(12,15),(13,16),(14,19),(15,1
8),(16,17),(16,18),(17,21),(18,20),(19,20),(19,22),(20,21),(21,22),(22,41),(23,25),(23
,39),(24,26),(24,43),(25,27),(25,29),(26,28),(26,30),(27,31),(27,37),(28,32),(28,38),(
29,33),(29,35),(30,34),(30,36),(31,40),(31,44),(32,42),(32,45),(33,35),(33,37),(34,36)
,(34,38),(35,48),(36,49),(37,50),(38,51),(39,48),(39,60),(40,41),(40,52),(41,42),(42,5
3),(43,49),(43,61),(44,50),(44,52),(45,51),(45,53),(46,48),(46,54),(46,56),(47,49),(47
,55),(47,57),(50,54),(51,55),(52,58),(53,59),(54,56),(55,57),(56,58),(57,59),(58,60),(
59,61),(60,62),(61,63),(62,64),(62,80),(63,65),(63,84),(64,66),(64,70),(65,67),(65,71)
,(66,68),(66,76),(67,69),(67,77),(68,72),(68,76),(69,73),(69,77),(70,78),(70,81),(71,7
9),(71,83),(72,78),(72,85),(73,79),(73,86),(74,76),(74,80),(74,87),(75,77),(75,84),(75
,88),(78,91),(79,92),(80,99),(81,82),(81,91),(82,83),(82,101),(83,92),(84,100),(85,89)
,(85,95),(86,90),(86,96),(87,89),(87,93),(88,90),(88,94),(89,93),(90,94),(91,95),(92,9
6),(93,97),(94,98),(95,97),(96,98),(97,99),(98,100),(99,117),(100,121),(101,102),(101,
104),(102,103),(102,106),(103,104),(103,105),(104,109),(105,107),(105,108),(106,107),(
106,112),(107,110),(108,111),(108,113),(109,113),(109,118),(110,114),(110,115),(111,11
3),(111,116),(112,114),(112,120),(114,115),(115,116),(116,119),(117,118),(117,122),(11
8,119),(119,120),(120,121),(121,123)]
```

-

Example embedding coordinates (SAGE 7.2):

```
{0:[-17.5,-12.0015],1:[-17.5,11.9985],2:[-14.,-8.5015],3:[-14.,-2.8349],4:[-14.,-
0.0015],5:[-14.,2.8318],6:[-14.,8.4985],7:[-13.5957,-0.4736],8:[-13.3264,0.3917],9:[-
13.057,1.2576],10:[-12.9898,-1.1819],11:[-12.7876,2.1235],12:[-12.632,-0.3795],13:[-
12.4735,0.6541],14:[-12.3833,-1.8902],15:[-12.0468,-0.9853],16:[-11.8894,0.0512],17:[-
11.5752,1.4151],18:[-11.3059,-0.5523],19:[-10.7666,-0.9462],20:[-10.565,-0.1194],21:[-
10.3628,0.7068],22:[-9.1504,-0.0015],23:[-8.4,-8.5015],24:[-8.4,8.4985],25:[-7.7,-
7.2747],26:[-7.7,7.2717],27:[-7.,-6.0479],28:[-7.,6.0448],29:[-6.8443,-7.5473],30:[-
6.8443,7.5442],31:[-6.3,-4.821],32:[-6.3,4.818],33:[-6.2479,-6.9568],34:[-
6.2479,6.9538],35:[-5.9886,-7.8198],36:[-5.9886,7.8168],37:[-5.6521,-6.3658],38:[-
5.6521,6.3627],39:[-5.6,-8.5015],40:[-5.6,-3.5942],41:[-5.6,-0.0015],42:[-
5.6,3.5912],43:[-5.6,8.4985],44:[-5.4835,-5.0256],45:[-5.4835,5.0226],46:[-5.2265,-
7.1172],47:[-5.2265,7.1141],48:[-5.1335,-8.0924],49:[-5.1335,8.0894],50:[-5.0557,-
5.7753],51:[-5.0557,5.7723],52:[-4.6665,-5.2296],53:[-4.6665,5.2266],54:[-4.6278,-
6.525],55:[-4.6278,6.522],56:[-4.4335,-7.4793],57:[-4.4335,7.4762],58:[-3.7335,-
6.8656],59:[-3.7335,6.8625],60:[-2.8,-8.5015],61:[-2.8,8.4985],62:[2.8,-
8.5015],63:[2.8,8.4985],64:[3.7335,-6.8656],65:[3.7335,6.8625],66:[4.4335,-
7.4793],67:[4.4335,7.4762],68:[4.6278,-6.525],69:[4.6278,6.522],70:[4.6665,-
5.2296],71:[4.6665,5.2266],72:[5.0557,-5.7753],73:[5.0557,5.7723],74:[5.1335,-
8.0924],75:[5.1335,8.0894],76:[5.2265,-7.1172],77:[5.2265,7.1141],78:[5.4835,-
5.0256],79:[5.4835,5.0226],80:[5.6,-8.5015],81:[5.6,-3.5942],82:[5.6,-
0.0015],83:[5.6,3.5912],84:[5.6,8.4985],85:[5.6521,-
6.3658],86:[5.6521,6.3627],87:[5.9886,-7.8198],88:[5.9886,7.8168],89:[6.2479,-
6.9568],90:[6.2479,6.9538],91:[6.3,-4.821],92:[6.3,4.818],93:[6.8443,-
7.5473],94:[6.8443,7.5442],95:[7.,-6.0479],96:[7.,6.0448],97:[7.7,-
7.2747],98:[7.7,7.2717],99:[8.4,-8.5015],100:[8.4,8.4985],101:[9.1504,-
0.0015],102:[10.3628,0.7069],103:[10.565,-0.1193],104:[10.7666,-0.9461],105:[11.3058,-
0.5523],106:[11.5752,1.4152],107:[11.8893,0.0512],108:[12.0467,-0.9852],109:[12.3832,-
1.8902],110:[12.4734,0.6542],111:[12.6319,-0.3794],112:[12.7876,2.1235],113:[12.9897,-
1.1818],114:[13.0569,1.2577],115:[13.3263,0.3918],116:[13.5956,-0.4735],117:[14.,-
8.5015],118:[14.,-2.8348],119:[14.,-
0.0015],120:[14.,2.8319],121:[14.,8.4985],122:[17.5,-12.0015],123:[17.5,11.9985]}
```

8.46 (Figure 5.3.a) gadget (equivalent to the "Fig. 4" gadget from ref. [146])



Graph Properties ::

(NOTE: Properties in this section are computed assuming that the digraph is undirected.)

--

Number of Vertices: '12'

Number of Edges: '14'

Minimum vertex degree $\delta(G)$: '1'

Maximum vertex degree $\Delta(G)$: '3'

Output of SAGE 7.2's 'is_regular(k=3)' function: 'False'

--

Planarity:

Output of Mathematica 10.4.1's 'PlanarGraphQ[]' function: 'True'

Output of Combinatorica's 'PlanarQ[]' function: 'True'

Output of SAGE 7.2's 'is_planar(sageTargetGraph)' function: 'True'

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1's 'VertexConnectivity[]' function: '1'

Output of Combinatorica's 'VertexConnectivity[]' function: '1'

Output of SAGE 7.2's 'vertex_connectivity()' function: '1'

Output for the 'igraph' R package command 'vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)': '1'

--

Chromatic Number $\chi(G)$:

Output of Combinatorica's 'ChromaticNumber[]' function: '2'

Output of SAGE 7.2's 'chromatic_number(sageTargetGraph)' function: '2'

Output of Mathematica 10.4.1's 'BipartiteGraphQ[]' function: 'True'

Output of Combinatorica's 'BipartiteQ[]' function: 'True'

Output of SAGE 7.2's 'is_bipartite()' function: 'True'

--

Girth:

Output of Combinatorica's 'Girth[]' function: '4'

Output of SAGE 7.2's 'girth()' function: '4'

Output for the 'igraph' R package 'girth()' function: '4'

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: '0'

Edge list (Mathematica 10.4.1):
{1\[DirectedEdge]3,3\[DirectedEdge]7,4\[DirectedEdge]3,4\[DirectedEdge]5,5\[DirectedEd
ge]9,6\[DirectedEdge]2,6\[DirectedEdge]5,7\[DirectedEdge]8,7\[DirectedEdge]11,8\[Direc
tedEdge]4,9\[DirectedEdge]8,9\[DirectedEdge]10,10\[DirectedEdge]6,12\[DirectedEdge]10}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-6.,-14.5},2->{-6.,14.5},3->{-3.5,-12.},4->{-3.5,-4.},5->{-3.5,4.},6->{-
3.5,12.},7->{3.5,-12.},8->{3.5,-4.},9->{3.5,4.},10->{3.5,12.},11->{6.,-14.5},12-
>{6.,14.5}}

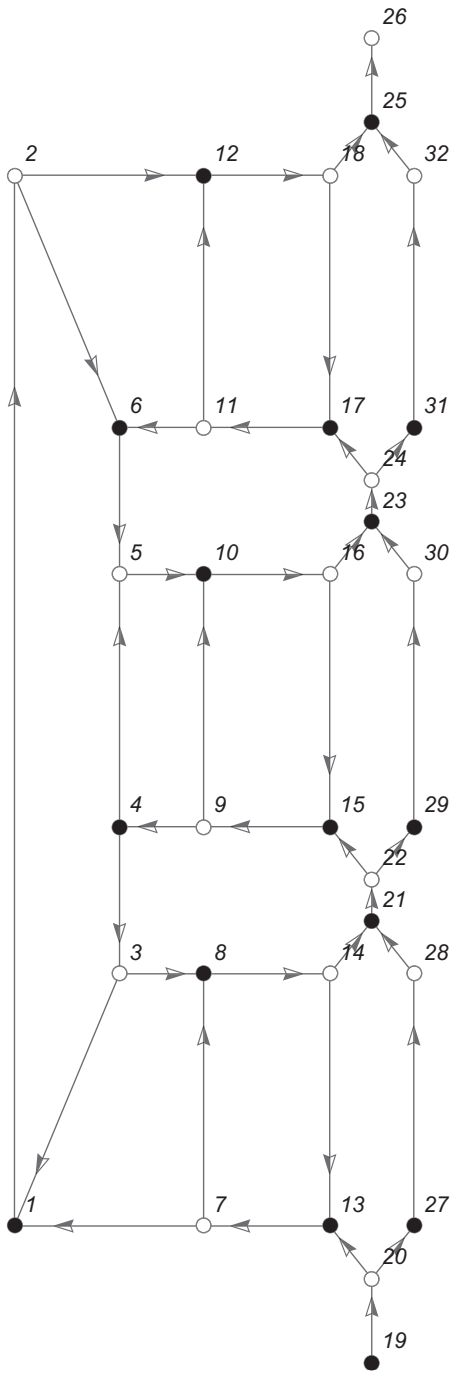
Edge list (SAGE 7.2):
({0:[2],2:[6],6:[7,10],3:[2,4],4:[8],8:[7,9],5:[4,1],1:[],7:[3],10:[],9:[5],11:[9]})

-

Example embedding coordinates (SAGE 7.2):
{0:[-6.,-14.5],1:[-6.,14.5],2:[-3.5,-12.],3:[-3.5,-4.],4:[-3.5,4.],5:[-
3.5,12.],6:[3.5,-12.],7:[3.5,-4.],8:[3.5,4.],9:[3.5,12.],10:[6.,-14.5],11:[6.,14.5]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorTwo,2->colorOne,3->colorOne,4->colorTwo,5->colorOne,6->colorTwo,7-
>colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorOne,12->colorTwo}

8.47 (Figure 5.3.b) gadget (equivalent to the "Fig. 2" gadget from ref. [146])



Graph Properties ::

(NOTE: Properties in this section are computed assuming that the digraph is undirected.)

--

Number of Vertices: '32'

Number of Edges: '43'

Minimum vertex degree $\delta(G)$: '1'

Maximum vertex degree $\Delta(G)$: '3'

Output of SAGE 7.2's 'is_regular(k=3)' function: 'False'

--

Planarity:

Output of Mathematica 10.4.1's 'PlanarGraphQ[]' function: 'True'

Output of Combinatorica's 'PlanarQ[]' function: 'True'

Output of SAGE 7.2's 'is_planar(sageTargetGraph)' function: 'True'

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1's 'VertexConnectivity[]' function: '1'

Output of Combinatorica's 'VertexConnectivity[]' function: '1'

Output of SAGE 7.2's 'vertex_connectivity()' function: '1'

Output for the 'igraph' R package command 'vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)': '1'

--

Chromatic Number $\chi(G)$:

Output of Combinatorica's 'ChromaticNumber[]' function: '2'

Output of SAGE 7.2's 'chromatic_number(sageTargetGraph)' function: '2'

Output of Mathematica 10.4.1's 'BipartiteGraphQ[]' function: 'True'

Output of Combinatorica's 'BipartiteQ[]' function: 'True'

Output of SAGE 7.2's 'is_bipartite()' function: 'True'

--

Girth:

Output of Combinatorica's 'Girth[]' function: '4'

Output of SAGE 7.2's 'girth()' function: '4'

Output for the 'igraph' R package 'girth()' function: '4'

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: '0'

Edge list (Mathematica 10.4.1):
{1\[DirectedEdge]2,2\[DirectedEdge]6,2\[DirectedEdge]12,3\[DirectedEdge]1,3\[DirectedEdge]8,4\[DirectedEdge]3,4\[DirectedEdge]5,5\[DirectedEdge]10,6\[DirectedEdge]5,7\[DirectedEdge]1,7\[DirectedEdge]8,8\[DirectedEdge]14,9\[DirectedEdge]4,9\[DirectedEdge]10,10\[DirectedEdge]16,11\[DirectedEdge]6,11\[DirectedEdge]12,12\[DirectedEdge]18,13\[DirectedEdge]7,14\[DirectedEdge]13,14\[DirectedEdge]21,15\[DirectedEdge]9,16\[DirectedEdge]15,16\[DirectedEdge]23,17\[DirectedEdge]11,18\[DirectedEdge]17,18\[DirectedEdge]25,19\[DirectedEdge]20,20\[DirectedEdge]13,20\[DirectedEdge]27,21\[DirectedEdge]22,22\[DirectedEdge]15,22\[DirectedEdge]29,23\[DirectedEdge]24,24\[DirectedEdge]17,24\[DirectedEdge]31,25\[DirectedEdge]26,27\[DirectedEdge]28,28\[DirectedEdge]21,29\[DirectedEdge]30,30\[DirectedEdge]23,31\[DirectedEdge]32,32\[DirectedEdge]25}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-12.9375,-25.},2->{-12.9375,25.},3->{-7.9375,-13.},4->{-7.9375,-6.},5->{-7.9375,6.},6->{-7.9375,13.},7->{-3.9375,-25.},8->{-3.9375,-13.},9->{-3.9375,-6.},10->{-3.9375,6.},11->{-3.9375,13.},12->{-3.9375,25.},13->{2.0625,-25.},14->{2.0625,-13.},15->{2.0625,-6.},16->{2.0625,6.},17->{2.0625,13.},18->{2.0625,25.},19->{4.0625,-31.5},20->{4.0625,-27.5},21->{4.0625,-10.5},22->{4.0625,-8.5},23->{4.0625,8.5},24->{4.0625,10.5},25->{4.0625,27.5},26->{4.0625,31.5},27->{6.0625,-25.},28->{6.0625,-13.},29->{6.0625,-6.},30->{6.0625,6.},31->{6.0625,13.},32->{6.0625,25.}}

Edge list (SAGE 7.2):
({0:[1],1:[5,11],5:[4],11:[17],2:[0,7],7:[13],3:[2,4],4:[9],9:[15],6:[0,7],13:[12,20],8:[3,9],15:[14,22],10:[5,11],17:[16,24],12:[6],20:[21],14:[8],22:[23],16:[10],24:[25],18:[19],19:[12,26],26:[27],21:[14,28],28:[29],23:[16,30],30:[31],25:[],27:[20],29:[22],31:[24]})

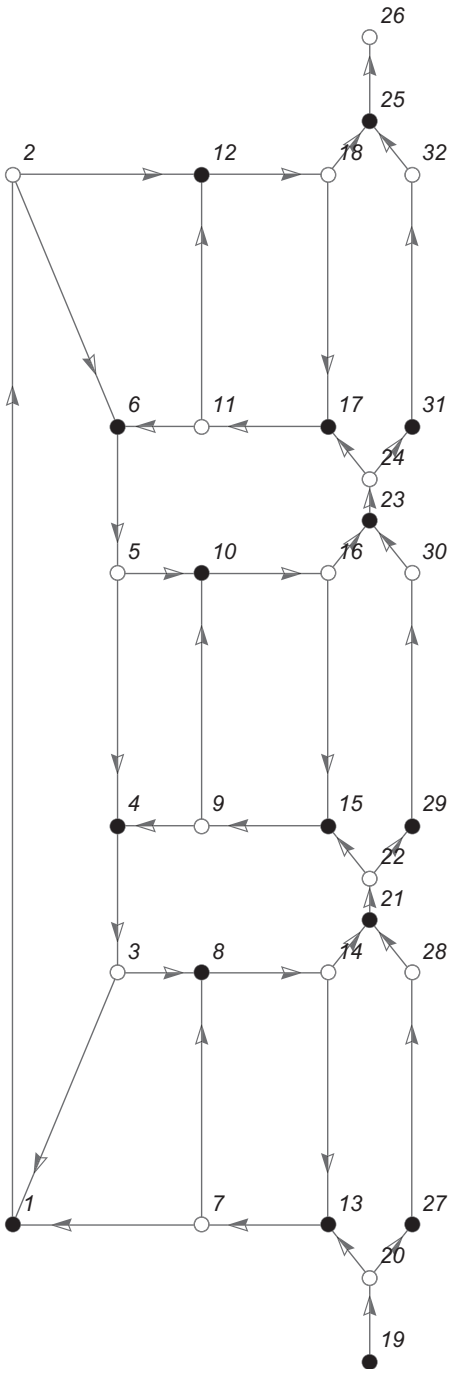
-

Example embedding coordinates (SAGE 7.2):
{0:[-12.9375,-25.],1:[-12.9375,25.],2:[-7.9375,-13.],3:[-7.9375,-6.],4:[-7.9375,6.],5:[-7.9375,13.],6:[-3.9375,-25.],7:[-3.9375,-13.],8:[-3.9375,-6.],9:[-3.9375,6.],10:[-3.9375,13.],11:[-3.9375,25.],12:[2.0625,-25.],13:[2.0625,-13.],14:[2.0625,-6.],15:[2.0625,6.],16:[2.0625,13.],17:[2.0625,25.],18:[4.0625,-31.5],19:[4.0625,-27.5],20:[4.0625,-10.5],21:[4.0625,-8.5],22:[4.0625,8.5],23:[4.0625,10.5],24:[4.0625,27.5],25:[4.0625,31.5],26:[6.0625,-25.],27:[6.0625,-13.],28:[6.0625,-6.],29:[6.0625,6.],30:[6.0625,13.],31:[6.0625,25.]}

Canonical vertex $(k = 2)$-coloring ::

{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorTwo,6->colorOne,7-
>colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13-
>colorOne,14->colorTwo,15->colorOne,16->colorTwo,17->colorOne,18->colorTwo,19-
>colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25-
>colorOne,26->colorTwo,27->colorOne,28->colorTwo,29->colorOne,30->colorTwo,31-
>colorOne,32->colorTwo}

8.48 (Figure 5.3.c) gadget



Graph Properties ::

(NOTE: Properties in this section are computed assuming that the digraph is undirected.)

--

Number of Vertices: '32'

Number of Edges: '43'

Minimum vertex degree $\delta(G)$: '1'

Maximum vertex degree $\Delta(G)$: '3'

Output of SAGE 7.2's 'is_regular(k=3)' function: 'False'

--

Planarity:

Output of Mathematica 10.4.1's 'PlanarGraphQ[]' function: 'True'

Output of Combinatorica's 'PlanarQ[]' function: 'True'

Output of SAGE 7.2's 'is_planar(sageTargetGraph)' function: 'True'

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1's 'VertexConnectivity[]' function: '1'

Output of Combinatorica's 'VertexConnectivity[]' function: '1'

Output of SAGE 7.2's 'vertex_connectivity()' function: '1'

Output for the 'igraph' R package command 'vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)': '1'

--

Chromatic Number $\chi(G)$:

Output of Combinatorica's 'ChromaticNumber[]' function: '2'

Output of SAGE 7.2's 'chromatic_number(sageTargetGraph)' function: '2'

Output of Mathematica 10.4.1's 'BipartiteGraphQ[]' function: 'True'

Output of Combinatorica's 'BipartiteQ[]' function: 'True'

Output of SAGE 7.2's 'is_bipartite()' function: 'True'

--

Girth:

Output of Combinatorica's 'Girth[]' function: '4'

Output of SAGE 7.2's 'girth()' function: '4'

Output for the 'igraph' R package 'girth()' function: '4'

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: '0'

Edge list (Mathematica 10.4.1):

{1\[DirectedEdge]2,2\[DirectedEdge]6,2\[DirectedEdge]12,3\[DirectedEdge]1,3\[DirectedEdge]8,4\[DirectedEdge]3,5\[DirectedEdge]4,5\[DirectedEdge]10,6\[DirectedEdge]5,7\[DirectedEdge]1,7\[DirectedEdge]8,8\[DirectedEdge]14,9\[DirectedEdge]4,9\[DirectedEdge]10,10\[DirectedEdge]16,11\[DirectedEdge]6,11\[DirectedEdge]12,12\[DirectedEdge]18,13\[DirectedEdge]7,14\[DirectedEdge]13,14\[DirectedEdge]21,15\[DirectedEdge]9,16\[DirectedEdge]15,16\[DirectedEdge]23,17\[DirectedEdge]11,18\[DirectedEdge]17,18\[DirectedEdge]25,19\[DirectedEdge]20,20\[DirectedEdge]13,20\[DirectedEdge]27,21\[DirectedEdge]22,22\[DirectedEdge]15,22\[DirectedEdge]29,23\[DirectedEdge]24,24\[DirectedEdge]17,24\[DirectedEdge]31,25\[DirectedEdge]26,27\[DirectedEdge]28,28\[DirectedEdge]21,29\[DirectedEdge]30,30\[DirectedEdge]23,31\[DirectedEdge]32,32\[DirectedEdge]25}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-12.9375,-25.},2->{-12.9375,25.},3->{-7.9375,-13.},4->{-7.9375,-6.},5->{-7.9375,6.},6->{-7.9375,13.},7->{-3.9375,-25.},8->{-3.9375,-13.},9->{-3.9375,-6.},10->{-3.9375,6.},11->{-3.9375,13.},12->{-3.9375,25.},13->{2.0625,-25.},14->{2.0625,-13.},15->{2.0625,-6.},16->{2.0625,6.},17->{2.0625,13.},18->{2.0625,25.},19->{4.0625,-31.5},20->{4.0625,-27.5},21->{4.0625,-10.5},22->{4.0625,-8.5},23->{4.0625,8.5},24->{4.0625,10.5},25->{4.0625,27.5},26->{4.0625,31.5},27->{6.0625,-25.},28->{6.0625,-13.},29->{6.0625,-6.},30->{6.0625,6.},31->{6.0625,13.},32->{6.0625,25.}}

Edge list (SAGE 7.2):

({0:[1],1:[5,11],5:[4],11:[17],2:[0,7],7:[13],3:[2],4:[3,9],9:[15],6:[0,7],13:[12,20],8:[3,9],15:[14,22],10:[5,11],17:[16,24],12:[6],20:[21],14:[8],22:[23],16:[10],24:[25],18:[19],19:[12,26],26:[27],21:[14,28],28:[29],23:[16,30],30:[31],25:[],27:[20],29:[22],31:[24]})

-

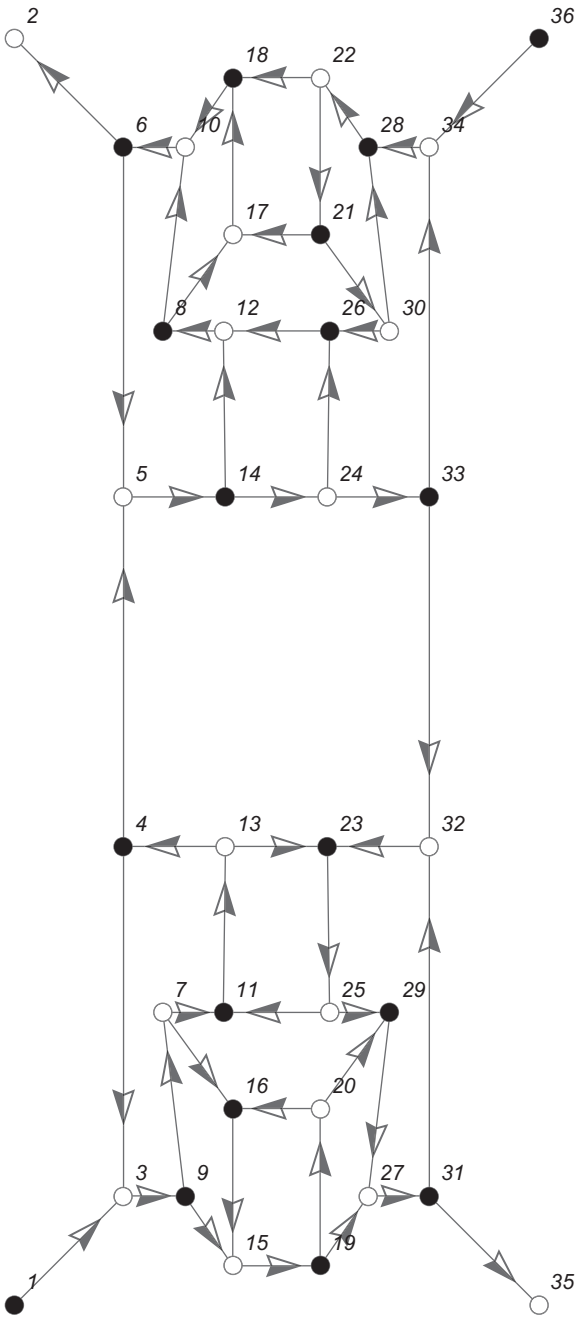
Example embedding coordinates (SAGE 7.2):

{0:[-12.9375,-25.],1:[-12.9375,25.],2:[-7.9375,-13.],3:[-7.9375,-6.],4:[-7.9375,6.],5:[-7.9375,13.],6:[-3.9375,-25.],7:[-3.9375,-13.],8:[-3.9375,-6.],9:[-3.9375,6.],10:[-3.9375,13.],11:[-3.9375,25.],12:[2.0625,-25.],13:[2.0625,-13.],14:[2.0625,-6.],15:[2.0625,6.],16:[2.0625,13.],17:[2.0625,25.],18:[4.0625,-31.5],19:[4.0625,-27.5],20:[4.0625,-10.5],21:[4.0625,-8.5],22:[4.0625,8.5],23:[4.0625,10.5],24:[4.0625,27.5],25:[4.0625,31.5],26:[6.0625,-25.],27:[6.0625,-13.],28:[6.0625,-6.],29:[6.0625,6.],30:[6.0625,13.],31:[6.0625,25.]}

Canonical vertex $(k = 2)$-coloring ::

{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorTwo,6->colorOne,7-
>colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13-
>colorOne,14->colorTwo,15->colorOne,16->colorTwo,17->colorOne,18->colorTwo,19-
>colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25-
>colorOne,26->colorTwo,27->colorOne,28->colorTwo,29->colorOne,30->colorTwo,31-
>colorOne,32->colorTwo}

8.49 (Figure 5.4.a) gadget



Graph Properties ::

(NOTE: Properties in this section are computed assuming that the digraph is undirected.)

--

Number of Vertices: '36'

Number of Edges: '50'

Minimum vertex degree $\delta(G)$: '1'

Maximum vertex degree $\Delta(G)$: '3'

Output of SAGE 7.2's 'is_regular(k=3)' function: 'False'

--

Planarity:

Output of Mathematica 10.4.1's 'PlanarGraphQ[]' function: 'True'

Output of Combinatorica's 'PlanarQ[]' function: 'True'

Output of SAGE 7.2's 'is_planar(sageTargetGraph)' function: 'True'

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1's 'VertexConnectivity[]' function: '1'

Output of Combinatorica's 'VertexConnectivity[]' function: '1'

Output of SAGE 7.2's 'vertex_connectivity()' function: '1'

Output for the 'igraph' R package command 'vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)': '1'

--

Chromatic Number $\chi(G)$:

Output of Combinatorica's 'ChromaticNumber[]' function: '2'

Output of SAGE 7.2's 'chromatic_number(sageTargetGraph)' function: '2'

Output of Mathematica 10.4.1's 'BipartiteGraphQ[]' function: 'True'

Output of Combinatorica's 'BipartiteQ[]' function: 'True'

Output of SAGE 7.2's 'is_bipartite()' function: 'True'

--

Girth:

Output of Combinatorica's 'Girth[]' function: '4'

Output of SAGE 7.2's 'girth()' function: '4'

Output for the 'igraph' R package 'girth()' function: '4'

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1\[DirectedEdge]3,3\[DirectedEdge]9,4\[DirectedEdge]3,4\[DirectedEdge]5,5\[DirectedEdge]14,6\[DirectedEdge]2,6\[DirectedEdge]5,7\[DirectedEdge]11,7\[DirectedEdge]16,8\[DirectedEdge]10,8\[DirectedEdge]17,9\[DirectedEdge]7,9\[DirectedEdge]15,10\[DirectedEdge]6,11\[DirectedEdge]13,12\[DirectedEdge]8,13\[DirectedEdge]4,13\[DirectedEdge]23,14\[DirectedEdge]12,14\[DirectedEdge]24,15\[DirectedEdge]19,16\[DirectedEdge]15,17\[DirectedEdge]18,18\[DirectedEdge]10,19\[DirectedEdge]20,19\[DirectedEdge]27,20\[DirectedEdge]16,20\[DirectedEdge]29,21\[DirectedEdge]17,21\[DirectedEdge]30,22\[DirectedEdge]18,22\[DirectedEdge]21,23\[DirectedEdge]25,24\[DirectedEdge]26,24\[DirectedEdge]33,25\[DirectedEdge]11,25\[DirectedEdge]29,26\[DirectedEdge]12,27\[DirectedEdge]31,28\[DirectedEdge]22,29\[DirectedEdge]27,30\[DirectedEdge]26,30\[DirectedEdge]28,31\[DirectedEdge]32,31\[DirectedEdge]35,32\[DirectedEdge]23,33\[DirectedEdge]32,33\[DirectedEdge]34,34\[DirectedEdge]28,36\[DirectedEdge]34}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-6.,-14.5},2->{-6.,14.5},3->{-3.5,-12.},4->{-3.5,-4.},5->{-3.5,4.},6->{-3.5,12.},7->{-2.6,-7.8},8->{-2.6,7.8},9->{-2.1,-12.},10->{-2.1,12.},11->{-1.2167,-7.8},12->{-1.2167,7.8},13->{-1.1667,-4.},14->{-1.1667,4.},15->{-1.,-13.6},16->{-1.,-10.},17->{-1.,10.},18->{-1.,13.6},19->{1.,-13.6},20->{1.,-10.},21->{1.,10.},22->{1.,13.6},23->{1.1667,-4.},24->{1.1667,4.},25->{1.2167,-7.8},26->{1.2167,7.8},27->{2.1,-12.},28->{2.1,12.},29->{2.6,-7.8},30->{2.6,7.8},31->{3.5,-12.},32->{3.5,-4.},33->{3.5,4.},34->{3.5,12.},35->{6.,-14.5},36->{6.,14.5}}

Edge list (SAGE 7.2):

[(0,2),(2,8),(3,2),(3,4),(4,13),(5,1),(5,4),(6,10),(6,15),(7,9),(7,16),(8,6),(8,14),(9,5),(10,12),(11,7),(12,3),(12,22),(13,11),(13,23),(14,18),(15,14),(16,17),(17,9),(18,19),(18,26),(19,15),(19,28),(20,16),(20,29),(21,17),(21,20),(22,24),(23,25),(23,32),(24,10),(24,28),(25,11),(26,30),(27,21),(28,26),(29,25),(29,27),(30,31),(30,34),(31,22),(32,31),(32,33),(33,27),(35,33)]

-

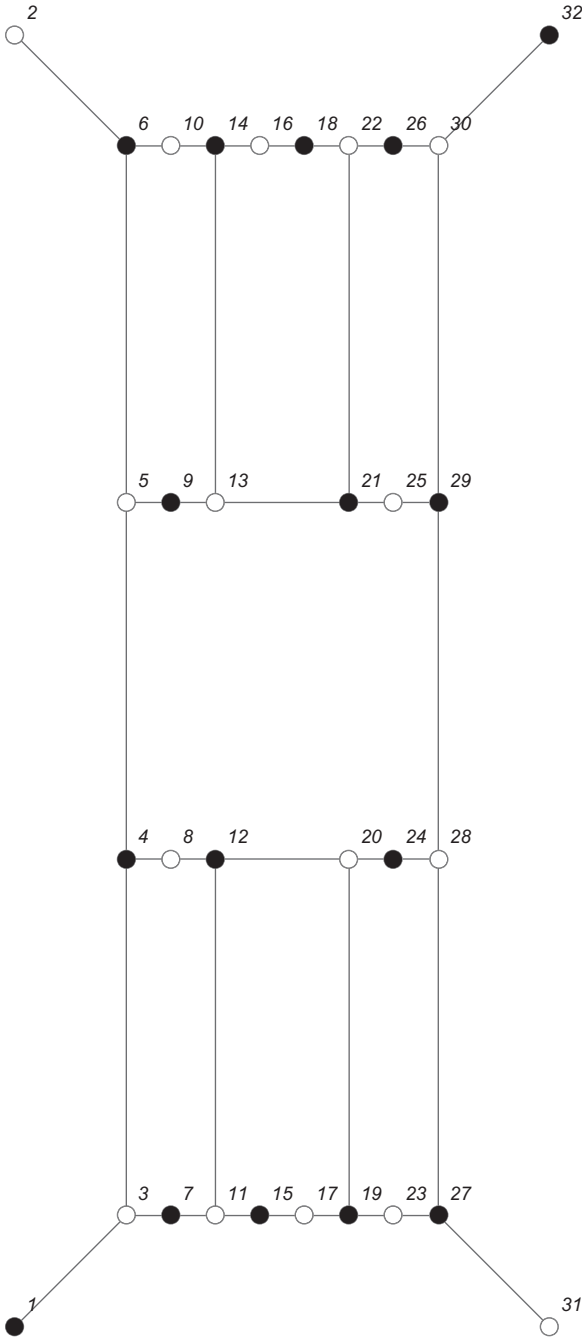
Example embedding coordinates (SAGE 7.2):

{0:[-6.,-14.5],1:[-6.,14.5],2:[-3.5,-12.],3:[-3.5,-4.],4:[-3.5,4.],5:[-3.5,12.],6:[-2.6,-7.8],7:[-2.6,7.8],8:[-2.1,-12.],9:[-2.1,12.],10:[-1.2167,-7.8],11:[-1.2167,7.8],12:[-1.1667,-4.],13:[-1.1667,4.],14:[-1.,-13.6],15:[-1.,-10.],16:[-1.,10.],17:[-1.,13.6],18:[1.,-13.6],19:[1.,-10.],20:[1.,10.],21:[1.,13.6],22:[1.1667,-4.],23:[1.1667,4.],24:[1.2167,-7.8],25:[1.2167,7.8],26:[2.1,-12.],27:[2.1,12.],28:[2.6,-7.8],29:[2.6,7.8],30:[3.5,-12.],31:[3.5,-4.],32:[3.5,4.],33:[3.5,12.],34:[6.,-14.5],35:[6.,14.5]}

Canonical vertex $(k = 2)$-coloring ::

{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorTwo,6->colorOne,7-
>colorTwo,8->colorOne,9->colorOne,10->colorTwo,11->colorOne,12->colorTwo,13-
>colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorTwo,18->colorOne,19-
>colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25-
>colorTwo,26->colorOne,27->colorTwo,28->colorOne,29->colorOne,30->colorTwo,31-
>colorOne,32->colorTwo,33->colorOne,34->colorTwo,35->colorTwo,36->colorOne}

8.50 (Figure 5.4.b) gadget



Graph Properties ::

--

Number of Vertices: ‘32’

Number of Edges: ‘38’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘6’

Output of SAGE 7.2’s ‘girth()’ function: ‘6’

Output for the ‘igraph’ R package ‘girth()’ function: ‘6’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->3,2<->6,3<->4,3<->7,4<->5,4<->8,5<->6,5<->9,6<->10,7<->11,8<->12,9<->13,10<->14,11<->12,11<->15,12<->20,13<->14,13<->21,14<->16,15<->17,16<->18,17<->19,18<->22,19<->20,19<->23,20<->24,21<->22,21<->25,22<->26,23<->27,24<->28,25<->29,26<->30,27<->28,27<->31,28<->29,29<->30,30<->32}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-6.,-14.5},2->{-6.,14.5},3->{-3.5,-12.},4->{-3.5,-4.},5->{-3.5,4.},6->{-3.5,12.},7->{-2.5,-12.},8->{-2.5,-4.},9->{-2.5,4.},10->{-2.5,12.},11->{-1.5,-12.},12->{-1.5,-4.},13->{-1.5,4.},14->{-1.5,12.},15->{-0.5,-12.},16->{-0.5,12.},17->{0.5,-12.},18->{0.5,12.},19->{1.5,-12.},20->{1.5,-4.},21->{1.5,4.},22->{1.5,12.},23->{2.5,-12.},24->{2.5,-4.},25->{2.5,4.},26->{2.5,12.},27->{3.5,-12.},28->{3.5,-4.},29->{3.5,4.},30->{3.5,12.},31->{6.,-14.5},32->{6.,14.5}}

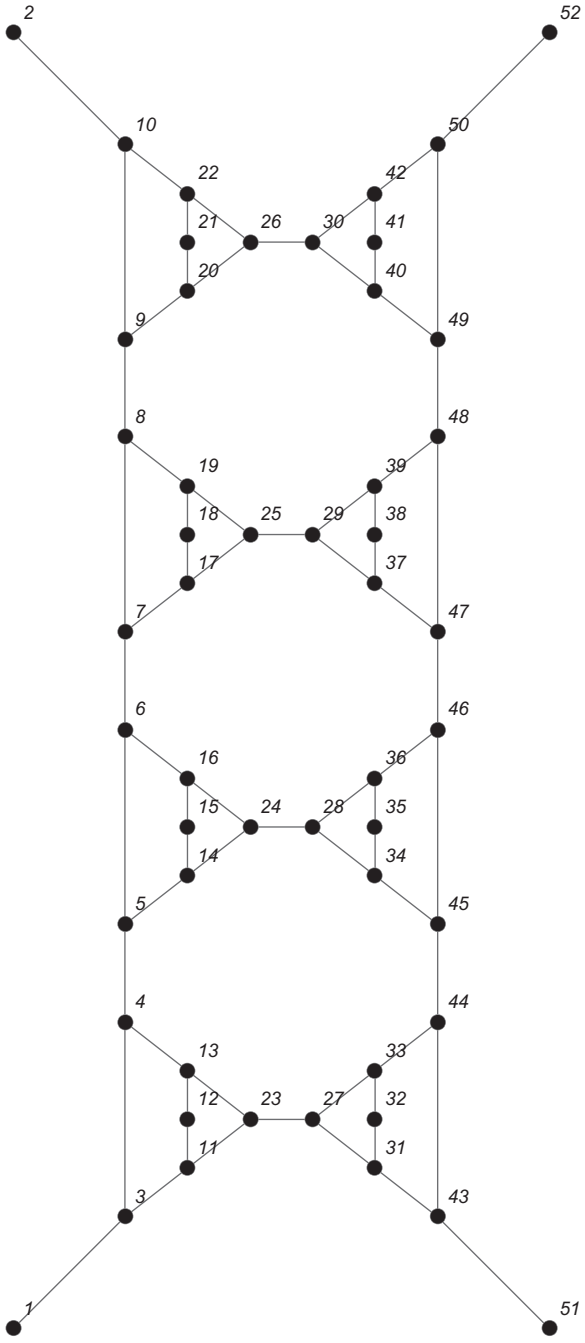
Edge list (SAGE 7.2):
[(0,2),(1,5),(2,3),(2,6),(3,4),(3,7),(4,5),(4,8),(5,9),(6,10),(7,11),(8,12),(9,13),(10,11),(10,14),(11,19),(12,13),(12,20),(13,15),(14,16),(15,17),(16,18),(17,21),(18,19),(18,22),(19,23),(20,21),(20,24),(21,25),(22,26),(23,27),(24,28),(25,29),(26,27),(26,30),(27,28),(28,29),(29,31)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-6.,-14.5],1:[-6.,14.5],2:[-3.5,-12.],3:[-3.5,-4.],4:[-3.5,4.],5:[-3.5,12.],6:[-2.5,-12.],7:[-2.5,-4.],8:[-2.5,4.],9:[-2.5,12.],10:[-1.5,-12.],11:[-1.5,-4.],12:[-1.5,4.],13:[-1.5,12.],14:[-0.5,-12.],15:[-0.5,12.],16:[0.5,-12.],17:[0.5,12.],18:[1.5,-12.],19:[1.5,-4.],20:[1.5,4.],21:[1.5,12.],22:[2.5,-12.],23:[2.5,-4.],24:[2.5,4.],25:[2.5,12.],26:[3.5,-12.],27:[3.5,-4.],28:[3.5,4.],29:[3.5,12.],30:[6.,-14.5],31:[6.,14.5]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorTwo,6->colorOne,7->colorOne,8->colorTwo,9->colorOne,10->colorTwo,11->colorTwo,12->colorOne,13->colorTwo,14->colorOne,15->colorOne,16->colorTwo,17->colorTwo,18->colorOne,19->colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorTwo,24->colorOne,25->colorTwo,26->colorOne,27->colorOne,28->colorTwo,29->colorOne,30->colorTwo,31->colorTwo,32->colorOne}

8.51 (Figure 5.4.c) gadget



Graph Properties ::

--

Number of Vertices: ‘52’

Number of Edges: ‘70’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘3’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘3’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘False’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘False’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘False’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1<->3,2<->10,3<->4,3<->11,4<->5,4<->13,5<->6,5<->14,6<->7,6<->16,7<->8,7<->17,8<->9,8<->19,9<->10,9<->20,10<->22,11<->12,11<->23,12<->13,13<->23,14<->15,14<->24,15<->16,16<->24,17<->18,17<->25,18<->19,19<->25,20<->21,20<->26,21<->22,22<->26,23<->27,24<->28,25<->29,26<->30,27<->31,27<->33,28<->34,28<->36,29<->37,29<->39,30<->40,30<->42,31<->32,31<->43,32<->33,33<->44,34<->35,34<->45,35<->36,36<->46,37<->38,37<->47,38<->39,39<->48,40<->41,40<->49,41<->42,42<->50,43<->44,43<->51,44<->45,45<->46,46<->47,47<->48,48<->49,49<->50,50<->52}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-6.,-14.5},2->{-6.,14.5},3->{-3.5,-12.},4->{-3.5,-7.6364},5->{-3.5,-5.4545},6->{-3.5,-1.0909},7->{-3.5,1.0909},8->{-3.5,5.4545},9->{-3.5,7.6364},10->{-3.5,12.},11->{-2.1,-10.9091},12->{-2.1,-9.8182},13->{-2.1,-8.7273},14->{-2.1,-4.3636},15->{-2.1,-3.2727},16->{-2.1,-2.1818},17->{-2.1,2.1819},18->{-2.1,3.2728},19->{-2.1,4.3637},20->{-2.1,8.7274},21->{-2.1,9.8183},22->{-2.1,10.9092},23->{-0.7,-9.8182},24->{-0.7,-3.2727},25->{-0.7,3.2727},26->{-0.7,9.8182},27->{0.7,-9.8182},28->{0.7,-3.2727},29->{0.7,3.2727},30->{0.7,9.8182},31->{2.1,-10.9091},32->{2.1,-9.8182},33->{2.1,-8.7273},34->{2.1,-4.3636},35->{2.1,-3.2727},36->{2.1,-2.1818},37->{2.1,2.1819},38->{2.1,3.2728},39->{2.1,4.3637},40->{2.1,8.7274},41->{2.1,9.8183},42->{2.1,10.9092},43->{3.5,-12.},44->{3.5,-7.6364},45->{3.5,-5.4545},46->{3.5,-1.0909},47->{3.5,1.0909},48->{3.5,5.4545},49->{3.5,7.6364},50->{3.5,12.},51->{6.,-14.5},52->{6.,14.5}}

Edge list (SAGE 7.2):

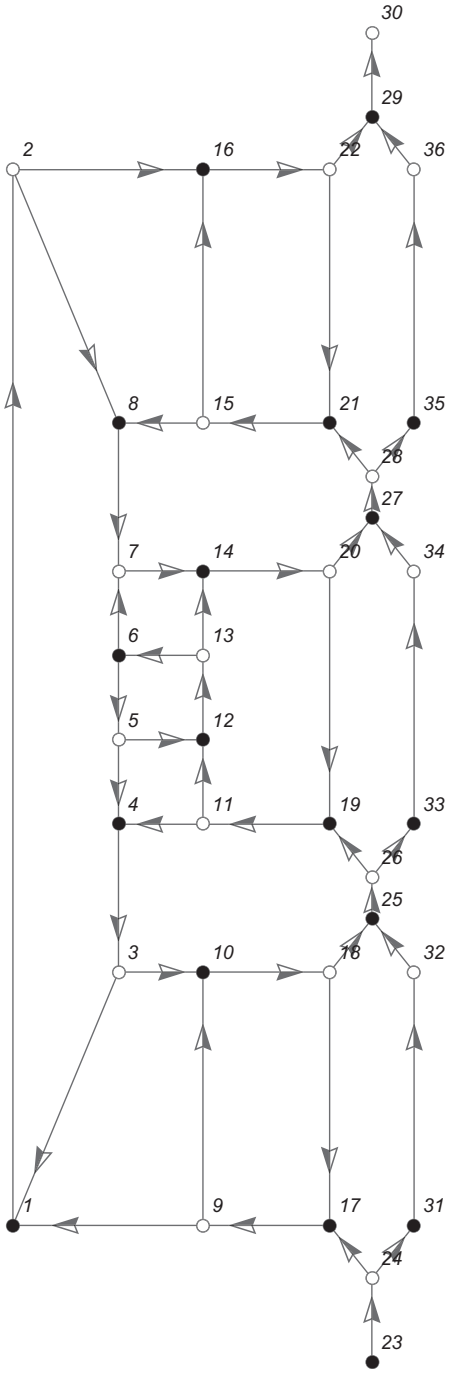
[(0,2),(1,9),(2,3),(2,10),(3,4),(3,12),(4,5),(4,13),(5,6),(5,15),(6,7),(6,16),(7,8),(7,18),(8,9),(8,19),(9,21),(10,11),(10,22),(11,12),(12,22),(13,14),(13,23),(14,15),(15,23),(16,17),(16,24),(17,18),(18,24),(19,20),(19,25),(20,21),(21,25),(22,26),(23,27),(24,28),(25,29),(26,30),(26,32),(27,33),(27,35),(28,36),(28,38),(29,39),(29,41),(30,31),(30,42),(31,32),(32,43),(33,34),(33,44),(34,35),(35,45),(36,37),(36,46),(37,38),(38,47),(39,40),(39,48),(40,41),(41,49),(42,43),(42,50),(43,44),(44,45),(45,46),(46,47),(47,48),(48,49),(49,51)]

-

Example embedding coordinates (SAGE 7.2):

{0:[-6.,-14.5],1:[-6.,14.5],2:[-3.5,-12.],3:[-3.5,-7.6364],4:[-3.5,-5.4545],5:[-3.5,-1.0909],6:[-3.5,1.0909],7:[-3.5,5.4545],8:[-3.5,7.6364],9:[-3.5,12.],10:[-2.1,-10.9091],11:[-2.1,-9.8182],12:[-2.1,-8.7273],13:[-2.1,-4.3636],14:[-2.1,-3.2727],15:[-2.1,-2.1818],16:[-2.1,2.1819],17:[-2.1,3.2728],18:[-2.1,4.3637],19:[-2.1,8.7274],20:[-2.1,9.8183],21:[-2.1,10.9092],22:[-0.7,-9.8182],23:[-0.7,-3.2727],24:[-0.7,3.2727],25:[-0.7,9.8182],26:[0.7,-9.8182],27:[0.7,-3.2727],28:[0.7,3.2727],29:[0.7,9.8182],30:[2.1,-10.9091],31:[2.1,-9.8182],32:[2.1,-8.7273],33:[2.1,-4.3636],34:[2.1,-3.2727],35:[2.1,-2.1818],36:[2.1,2.1819],37:[2.1,3.2728],38:[2.1,4.3637],39:[2.1,8.7274],40:[2.1,9.8183],41:[2.1,10.9092],42:[3.5,-12.],43:[3.5,-7.6364],44:[3.5,-5.4545],45:[3.5,-1.0909],46:[3.5,1.0909],47:[3.5,5.4545],48:[3.5,7.6364],49:[3.5,12.],50:[6.,-14.5],51:[6.,14.5]}

8.52 (Figure 5.4.d) gadget



Graph Properties ::

(NOTE: Properties in this section are computed assuming that the digraph is undirected.)

--

Number of Vertices: '36'

Number of Edges: '49'

Minimum vertex degree $\delta(G)$: '1'

Maximum vertex degree $\Delta(G)$: '3'

Output of SAGE 7.2's 'is_regular(k=3)' function: 'False'

--

Planarity:

Output of Mathematica 10.4.1's 'PlanarGraphQ[]' function: 'True'

Output of Combinatorica's 'PlanarQ[]' function: 'True'

Output of SAGE 7.2's 'is_planar(sageTargetGraph)' function: 'True'

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1's 'VertexConnectivity[]' function: '1'

Output of Combinatorica's 'VertexConnectivity[]' function: '1'

Output of SAGE 7.2's 'vertex_connectivity()' function: '1'

Output for the 'igraph' R package command 'vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)': '1'

--

Chromatic Number $\chi(G)$:

Output of Combinatorica's 'ChromaticNumber[]' function: '2'

Output of SAGE 7.2's 'chromatic_number(sageTargetGraph)' function: '2'

Output of Mathematica 10.4.1's 'BipartiteGraphQ[]' function: 'True'

Output of Combinatorica's 'BipartiteQ[]' function: 'True'

Output of SAGE 7.2's 'is_bipartite()' function: 'True'

--

Girth:

Output of Combinatorica's 'Girth[]' function: '4'

Output of SAGE 7.2's 'girth()' function: '4'

Output for the 'igraph' R package 'girth()' function: '4'

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):

{1\[DirectedEdge]2,2\[DirectedEdge]8,2\[DirectedEdge]16,3\[DirectedEdge]1,3\[DirectedEdge]10,4\[DirectedEdge]3,5\[DirectedEdge]4,5\[DirectedEdge]12,6\[DirectedEdge]5,6\[DirectedEdge]7,7\[DirectedEdge]14,8\[DirectedEdge]7,9\[DirectedEdge]1,9\[DirectedEdge]10,10\[DirectedEdge]18,11\[DirectedEdge]4,11\[DirectedEdge]12,12\[DirectedEdge]13,13\[DirectedEdge]6,13\[DirectedEdge]14,14\[DirectedEdge]20,15\[DirectedEdge]8,15\[DirectedEdge]16,16\[DirectedEdge]22,17\[DirectedEdge]9,18\[DirectedEdge]17,18\[DirectedEdge]25,19\[DirectedEdge]11,20\[DirectedEdge]19,20\[DirectedEdge]27,21\[DirectedEdge]15,22\[DirectedEdge]21,22\[DirectedEdge]29,23\[DirectedEdge]24,24\[DirectedEdge]17,24\[DirectedEdge]31,25\[DirectedEdge]26,26\[DirectedEdge]19,26\[DirectedEdge]33,27\[DirectedEdge]28,28\[DirectedEdge]21,28\[DirectedEdge]35,29\[DirectedEdge]30,31\[DirectedEdge]32,32\[DirectedEdge]25,33\[DirectedEdge]34,34\[DirectedEdge]27,35\[DirectedEdge]36,36\[DirectedEdge]29}

-

Example embedding coordinates (Mathematica 10.4.1):

{1->{-12.2778,-25.},2->{-12.2778,25.},3->{-7.2778,-13.},4->{-7.2778,-6.},5->{-7.2778,-2.},6->{-7.2778,2.},7->{-7.2778,6.},8->{-7.2778,13.},9->{-3.2778,-25.},10->{-3.2778,-13.},11->{-3.2778,-6.},12->{-3.2778,-2.},13->{-3.2778,2.},14->{-3.2778,6.},15->{-3.2778,13.},16->{-3.2778,25.},17->{2.7222,-25.},18->{2.7222,-13.},19->{2.7222,-6.},20->{2.7222,6.},21->{2.7222,13.},22->{2.7222,25.},23->{4.7222,-31.5},24->{4.7222,-27.5},25->{4.7222,-10.5},26->{4.7222,-8.5},27->{4.7222,8.5},28->{4.7222,10.5},29->{4.7222,27.5},30->{4.7222,31.5},31->{6.7222,-25.},32->{6.7222,-13.},33->{6.7222,-6.},34->{6.7222,6.},35->{6.7222,13.},36->{6.7222,25.}}

Edge list (SAGE 7.2):

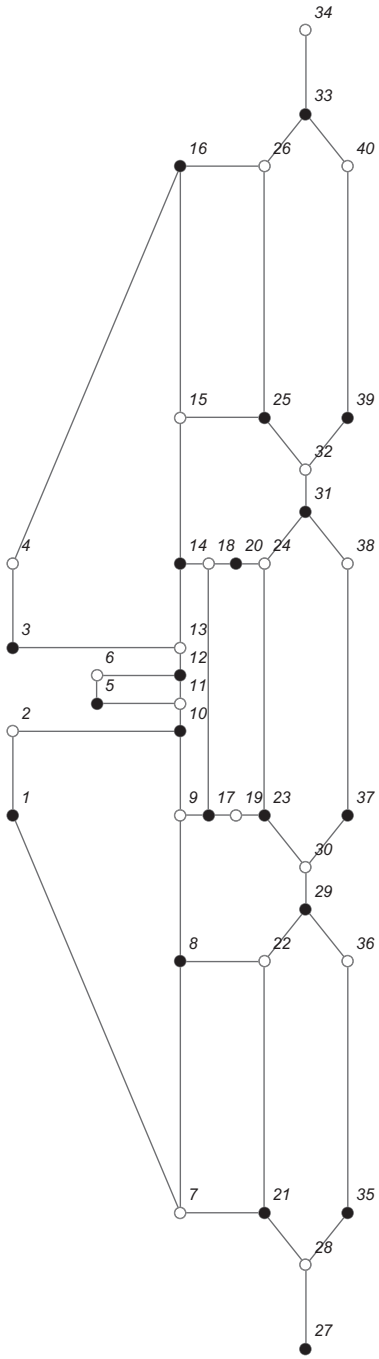
[(0,1),(1,7),(1,15),(2,0),(2,9),(3,2),(4,3),(4,11),(5,4),(5,6),(6,13),(7,6),(8,0),(8,9),(9,17),(10,3),(10,11),(11,12),(12,5),(12,13),(13,19),(14,7),(14,15),(15,21),(16,8),(17,16),(17,24),(18,10),(19,18),(19,26),(20,14),(21,20),(21,28),(22,23),(23,16),(23,30),(24,25),(25,18),(25,32),(26,27),(27,20),(27,34),(28,29),(30,31),(31,24),(32,33),(33,26),(34,35),(35,28)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-12.2778,-25.],1:[-12.2778,25.],2:[-7.2778,-13.],3:[-7.2778,-6.],4:[-7.2778,-
2.],5:[-7.2778,2.],6:[-7.2778,6.],7:[-7.2778,13.],8:[-3.2778,-25.],9:[-3.2778,-
13.],10:[-3.2778,-6.],11:[-3.2778,-2.],12:[-3.2778,2.],13:[-3.2778,6.],14:[-
3.2778,13.],15:[-3.2778,25.],16:[2.7222,-25.],17:[2.7222,-13.],18:[2.7222,-
6.],19:[2.7222,6.],20:[2.7222,13.],21:[2.7222,25.],22:[4.7222,-31.5],23:[4.7222,-
27.5],24:[4.7222,-10.5],25:[4.7222,-
8.5],26:[4.7222,8.5],27:[4.7222,10.5],28:[4.7222,27.5],29:[4.7222,31.5],30:[6.7222,-
25.],31:[6.7222,-13.],32:[6.7222,-6.],33:[6.7222,6.],34:[6.7222,13.],35:[6.7222,25.]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorTwo,3->colorTwo,4->colorOne,5->colorTwo,6->colorOne,7-
>colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13-
>colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorOne,18->colorTwo,19-
>colorOne,20->colorTwo,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25-
>colorOne,26->colorTwo,27->colorOne,28->colorTwo,29->colorOne,30->colorTwo,31-
>colorOne,32->colorTwo,33->colorOne,34->colorTwo,35->colorOne,36->colorTwo}

8.53 (Figure 5.4.e) gadget



Graph Properties ::

--

Number of Vertices: ‘40’

Number of Edges: ‘51’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘2’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘2’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘True’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘4’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->2,1<->7,2<->10,3<->4,3<->13,4<->16,5<->6,5<->11,6<->12,7<->8,7<->21,8<->9,8<->22,9<->10,9<->17,10<->11,11<->12,12<->13,13<->14,14<->15,14<->18,15<->16,15<->25,16<->26,17<->18,17<->19,18<->20,19<->23,20<->24,21<->22,21<->28,22<->29,23<->24,23<->30,24<->31,25<->26,25<->32,26<->33,27<->28,28<->35,29<->30,29<->36,30<->37,31<->32,31<->38,32<->39,33<->34,33<->40,35<->36,37<->38,39<->40}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-10.2,-6.},2->{-10.2,-2.},3->{-10.2,2.},4->{-10.2,6.},5->{-6.2,-0.6667},6->{-6.2,0.6667},7->{-2.2,-25.},8->{-2.2,-13.},9->{-2.2,-6.},10->{-2.2,-2.},11->{-2.2,-0.6667},12->{-2.2,0.6667},13->{-2.2,2.},14->{-2.2,6.},15->{-2.2,13.},16->{-2.2,25.},17->{-0.8667,-6.},18->{-0.8667,6.},19->{0.4667,-6.},20->{0.4667,6.},21->{1.8,-25.},22->{1.8,-13.},23->{1.8,-6.},24->{1.8,6.},25->{1.8,13.},26->{1.8,25.},27->{3.8,-31.5},28->{3.8,-27.5},29->{3.8,-10.5},30->{3.8,-8.5},31->{3.8,8.5},32->{3.8,10.5},33->{3.8,27.5},34->{3.8,31.5},35->{5.8,-25.},36->{5.8,-13.},37->{5.8,-6.},38->{5.8,6.},39->{5.8,13.},40->{5.8,25.}}

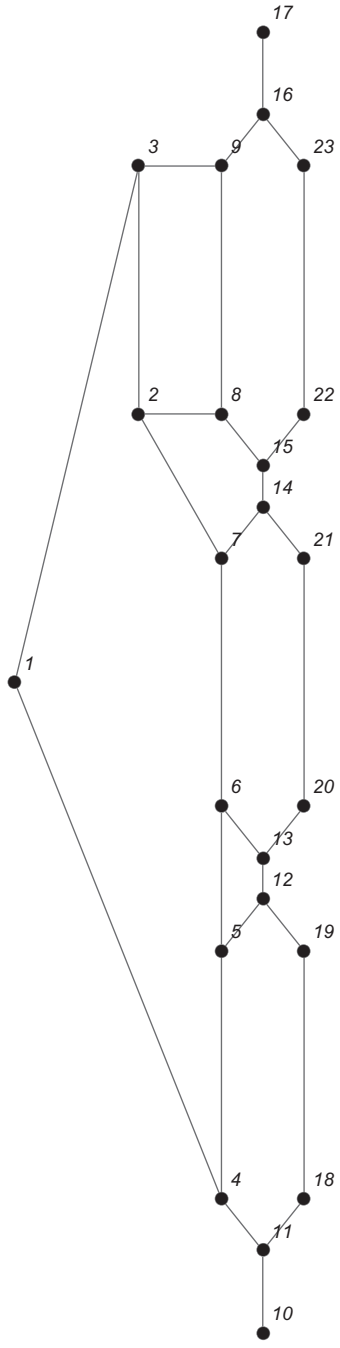
Edge list (SAGE 7.2):
[(0,1),(0,6),(1,9),(2,3),(2,12),(3,15),(4,5),(4,10),(5,11),(6,7),(6,20),(7,8),(7,21),(8,9),(8,16),(9,10),(10,11),(11,12),(12,13),(13,14),(13,17),(14,15),(14,24),(15,25),(16,17),(16,18),(17,19),(18,22),(19,23),(20,21),(20,27),(21,28),(22,23),(22,29),(23,30),(24,25),(24,31),(25,32),(26,27),(27,34),(28,29),(28,35),(29,36),(30,31),(30,37),(31,38),(32,33),(32,39),(34,35),(36,37),(38,39)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-10.2,-6.],1:[-10.2,-2.],2:[-10.2,2.],3:[-10.2,6.],4:[-6.2,-0.6667],5:[-6.2,0.6667],6:[-2.2,-25.],7:[-2.2,-13.],8:[-2.2,-6.],9:[-2.2,-2.],10:[-2.2,-0.6667],11:[-2.2,0.6667],12:[-2.2,2.],13:[-2.2,6.],14:[-2.2,13.],15:[-2.2,25.],16:[-0.8667,-6.],17:[-0.8667,6.],18:[0.4667,-6.],19:[0.4667,6.],20:[1.8,-25.],21:[1.8,-13.],22:[1.8,-6.],23:[1.8,6.],24:[1.8,13.],25:[1.8,25.],26:[3.8,-31.5],27:[3.8,-27.5],28:[3.8,-10.5],29:[3.8,-8.5],30:[3.8,8.5],31:[3.8,10.5],32:[3.8,27.5],33:[3.8,31.5],34:[5.8,-25.],35:[5.8,-13.],36:[5.8,-6.],37:[5.8,6.],38:[5.8,13.],39:[5.8,25.]}

Canonical vertex $(k = 2)$-coloring ::
{1->colorOne,2->colorTwo,3->colorOne,4->colorTwo,5->colorOne,6->colorTwo,7->colorTwo,8->colorOne,9->colorTwo,10->colorOne,11->colorTwo,12->colorOne,13->colorTwo,14->colorOne,15->colorTwo,16->colorOne,17->colorOne,18->colorTwo,19->colorTwo,20->colorOne,21->colorOne,22->colorTwo,23->colorOne,24->colorTwo,25->colorOne,26->colorTwo,27->colorOne,28->colorTwo,29->colorOne,30->colorTwo,31->colorOne,32->colorTwo,33->colorOne,34->colorTwo,35->colorOne,36->colorTwo,37->colorOne,38->colorTwo,39->colorOne,40->colorTwo}

8.54 (Figure 5.4.f) gadget



Graph Properties ::

--

Number of Vertices: ‘23’

Number of Edges: ‘29’

Minimum vertex degree $\delta(G)$: ‘1’

Maximum vertex degree $\Delta(G)$: ‘3’

Output of SAGE 7.2’s ‘is_regular(k=3)’ function: ‘False’

--

Planarity:

Output of Mathematica 10.4.1’s ‘PlanarGraphQ[]’ function: ‘True’

Output of Combinatorica’s ‘PlanarQ[]’ function: ‘True’

Output of SAGE 7.2’s ‘is_planar(sageTargetGraph)’ function: ‘True’

--

k-Vertex Connectivity / Minimum Vertex Cut:

Output of Mathematica 10.4.1’s ‘VertexConnectivity[]’ function: ‘1’

Output of Combinatorica’s ‘VertexConnectivity[]’ function: ‘1’

Output of SAGE 7.2’s ‘vertex_connectivity()’ function: ‘1’

Output for the ‘igraph’ R package command ‘vertex_connectivity(g, source = NULL, target = NULL, checks = TRUE)’: ‘1’

--

Chromatic Number $\chi(G)$:

Output of Combinatorica’s ‘ChromaticNumber[]’ function: ‘3’

Output of SAGE 7.2’s ‘chromatic_number(sageTargetGraph)’ function: ‘3’

Output of Mathematica 10.4.1’s ‘BipartiteGraphQ[]’ function: ‘False’

Output of Combinatorica’s ‘BipartiteQ[]’ function: ‘False’

Output of SAGE 7.2’s ‘is_bipartite()’ function: ‘False’

--

Girth:

Output of Combinatorica’s ‘Girth[]’ function: ‘4’

Output of SAGE 7.2’s ‘girth()’ function: ‘’

Output for the ‘igraph’ R package ‘girth()’ function: ‘4’

--

Minimum Genus $\gamma_{min}(G)$:

Output of SAGE 7.2's 'genus()' function: (--- Not Computed ---)

Edge list (Mathematica 10.4.1):
{1<->3,1<->4,2<->3,2<->7,2<->8,3<->9,4<->5,4<->11,5<->6,5<->12,6<->7,6<->13,7<->14,8<->9,8<->15,9<->16,10<->11,11<->18,12<->13,12<->19,13<->20,14<->15,14<->21,15<->22,16<->17,16<->23,18<->19,20<->21,22<->23}

-

Example embedding coordinates (Mathematica 10.4.1):
{1->{-10.9565,-1.6522},2->{-4.9565,11.3478},3->{-4.9565,23.3478},4->{-0.9565,-26.6522},5->{-0.9565,-14.6522},6->{-0.9565,-7.6522},7->{-0.9565,4.3478},8->{-0.9565,11.3478},9->{-0.9565,23.3478},10->{1.0435,-33.1522},11->{1.0435,-29.1522},12->{1.0435,-12.1522},13->{1.0435,-10.1522},14->{1.0435,6.8478},15->{1.0435,8.8478},16->{1.0435,25.8478},17->{1.0435,29.8478},18->{3.0435,-26.6522},19->{3.0435,-14.6522},20->{3.0435,-7.6522},21->{3.0435,4.3478},22->{3.0435,11.3478},23->{3.0435,23.3478}}

Edge list (SAGE 7.2):
[(0,2),(0,3),(1,2),(1,6),(1,7),(2,8),(3,4),(3,10),(4,5),(4,11),(5,6),(5,12),(6,13),(7,8),(7,14),(8,15),(9,10),(10,17),(11,12),(11,18),(12,19),(13,14),(13,20),(14,21),(15,16),(15,22),(17,18),(19,20),(21,22)]

-

Example embedding coordinates (SAGE 7.2):
{0:[-10.9565,-1.6522],1:[-4.9565,11.3478],2:[-4.9565,23.3478],3:[-0.9565,-26.6522],4:[-0.9565,-14.6522],5:[-0.9565,-7.6522],6:[-0.9565,4.3478],7:[-0.9565,11.3478],8:[-0.9565,23.3478],9:[1.0435,-33.1522],10:[1.0435,-29.1522],11:[1.0435,-12.1522],12:[1.0435,-10.1522],13:[1.0435,6.8478],14:[1.0435,8.8478],15:[1.0435,25.8478],16:[1.0435,29.8478],17:[3.0435,-26.6522],18:[3.0435,-14.6522],19:[3.0435,-7.6522],20:[3.0435,4.3478],21:[3.0435,11.3478],22:[3.0435,23.3478]}

Acknowledgements

I sincerely thank my supervisor, Professor Akira Suyama, for taking a chance on me three years ago, for his guidance in helping me to make my work coherent, and for providing me a place in which to at least try to approximate a meaningful contribution to science (and perhaps even certain areas of mathematics). To be clear, none of this would have otherwise been possible. I also wish to sincerely thank the lab members of Suyama group for their support, for sitting through my sometimes (or perhaps I should say often) lousy talks, and for sharing their knowledge and experience with me. I wish to mention here Dr. Koh-ichiroh Shohda, a superb organic chemist, and someone who had the patience to collaborate with a person who probably couldn't run a simple acid-base titration at this point. I wish to mention here Maasa Yokomori, whose extensive assistance and guidance made **(Chapter 6)** of this dissertation possible (and the citations in this section speak for themselves in this regard). I wish to express my deepest gratitude to Kazushiro Minegishi for taking care of me when I first arrived in Japan, and for getting me an honest-to-god LP of Utada Hikaru's *First Love*. Last but certainly not least, as a recipient of the Todai fellowship, I am indebted to the University of Tokyo for their generous financial support of the International Student Special Scholarship Program, and to the members of the committee who decided I was a worthwhile investment. All of these things together allowed for this dissertation to exist.

Bibliography

[1] V. Acuna, F. Chierichetti, V. Lacroix, A. Marchetti-Spaccamela, M. F. Sagot, and L. Stougie. Modes and cuts in metabolic networks: complexity and algorithms. *BioSystems*, 95(1):51–60, 2009.

[2] V. Acuna, A. Marchetti-Spaccamela, M. F. Sagot, and L. Stougie. A note on the complexity of finding and enumerating elementary modes. *BioSystems*, 99(3):210–214, 2010.

[3] K. J. Adolfsen and M. P. Brynildsen. Futile cycling increases sensitivity toward oxidative stress in Escherichia coli. *Metab. Eng.*, 29:26–35, 2015.

[4] T. Akiyama, T. Nishizeki, and N. Saito. NP-completeness of the Hamiltonian cycle problem for bipartite graphs. *Journal of Information Processing*, 3(2):73–76, 1980.

[5] R. E. L. Aldred, S. Bau, D. Holton, and B. D. McKay. Nonhamiltonian 3-connected cubic planar graphs. *SIAM J. Discrete Math.*, 13(1):25–32, 2000.

[6] J. F. Allemand, D. Bensimon, L. Jullien, A. Bensimon, and V. Croquette. pH-dependent specific binding and combing of DNA. *Biophys. J.*, 73(4):2064–2070, 1997.

[7] T. Ami, K. Ito, Y. Yoshimura, and K. Fujimoto. Sequence specific interstrand photocrosslinking for effective SNP typing. *Org. Biomol. Chem.*, 5(16):2583–2586, 2007.

[8] E. V. Anslyn and D. A. Dougherty. *Modern physical organic chemistry*. University Science Books: Sausalito, CA, 1st edition, 2005.

[9] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press: New York, NY, 1st edition, 2009.

[10] V. Arvind and P. P. Kurur. Graph isomorphism is in SPP. *Inform. Comput.*, 204(5):835–852, 2006.

[11] L. Auslander and S. Parter. On imbedding graphs in the sphere. *J. Mathematics and Mechanics*, 10(3):517–523, 1961.

[12] V. Bailly, M. Derydt, and W. G. Verly. δ-Elimination in the repair of AP (apurinic/apyrimidinic) sites in DNA. *Biochem. J.*, 261(3):707–713, 1989.

[13] V. Bailly and W. G. Verly. Possible roles of β-elimination and δ-elimination reactions in the repair of DNA containing AP (apurinic/apyrimidinic) sites in mammalian cells. *Biochem. J.*, 253(2):553–559, 1988.

[14] F. Barahona. On the computational complexity of Ising spin glass models. *J. Phys. A: Math. Gen.*, 15:3241–3253, 1982.

[15] F. Barany. Genetic disease detection and DNA amplification using cloned thermostable ligase. *PNAS*, 88(1):189–193, 1991.

[16] R. D. Barish and A. Suyama. Counting substrate cycles in topologically restricted metabolic networks. *Proceedings of the 13th conference on Computability in Europe (CiE)*, pages 129–140, 2017.

[17] R. Beigel, R. Buhrman, and L. Fortnow. NP might not be as easy as detecting unique solutions. *Proceedings of 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 203–208, 1998.

[18] B. P. Belotserkovskii and B. H. Johnston. Polypropylene tube surfaces may induce denaturation and multimerization of DNA. *Science*, 271(5246):222–223, 1996.

[19] B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *J. Comp. Biol.*, 5(1):27–40, 1998.

[20] A. Besaratinia, J. Yoon, C. Schroeder, S. E. Bradforth, M. Cockburn, and G. P. Pfeifer. Wavelength dependence of ultraviolet radiation-induced DNA damage as determined by laser irradiation suggests that cyclobutane pyrimidine dimers are the principal DNA lesions produced by terrestrial sunlight. *FASEB J.*, 25(9):3079–3091, 2011.

[21] A. Björklund. Determinant sums for undirected Hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014.

[22] A. Björklund. Below all subsets for some permutational counting problems. *Proceedings of the 15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 17:1–17:11, 2016.

[23] A. Björklund and T. Husfeldt. The parity of directed Hamiltonian cycles. *Proceedings of the 54th annual symposium on Foundations of Computer Science (FOCS)*, pages 727–735, 2013.

[24] B. Bollobás. *Modern graph theory*. Springer-Verlag: New York, NY, 1st edition, 1998.

[25] J. A. Bondy and U. S. R. Murty. *Graph theory with applications*. Macmillan Press: New York, NY, 1st edition, 1976.

[26] P. N. Borer, B. Dengler, I. Jr. Tinoco, and O. C. Uhlenbeck. Stability of ribonucleic acid double-stranded helices. *J. Mol. Biol.*, 86(4):843–853, 1974.

[27] S. N. Bose and R. J. Davies. The photoreactivity of T-A sequences in oligodeoxyribonucleotides and DNA. *Nucleic Acids Res.*, 12(20):7903–7914, 1984.

[28] S. N. Bose, R. J. Davies, S. K. Sethi, and J. A. McCloskey. Formation of an adenine-thymine photoadduct in the deoxydinucleoside monophosphate d(TpA) and in DNA. *Science*, 220(4598):723–725, 1983.

[29] S. N. Bose, S. Kumar, R. J. Davies, S. K. Sethi, and J. A. McCloskey. The photochemistry of d(T-A) in aqueous solution and in ice. *Nucleic Acids Res.*, 12(20):7929–7947, 1984.

[30] J. M. Boyer and W. J. Myrvold. On the cutting edge: simplified O(n) planarity by edge addition. *J. Graph Algorithms Appl.*, 8(3):241–273, 2004.

[31] G. Brightwell and P. Winkler. Counting linear extensions. *Order*, 8(3):225–242, 1991.

[32] G. Brightwell and P. Winkler. Counting eulerian circuits is #p-complete. *Proceedings of the 7th workshop on Algorithm Engineering and Experiments (ALENEX) and the 2nd workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 259–262, 2005.

[33] G. Brinkmann and B. D. McKay. Fast generation of planar graphs. *Match Commun. Math. Co.*, 58(2):323–357, 2007.

[34] I. Briquel and P. Koiran. A dichotomy theorem for polynomial evaluation. *Proceedings of the 34th international symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 187–198, 2009.

[35] L. G. Bunville, E. P. Geiduschek, M. A. Rawitscher, and J. Sturtevant. Kinetics and equilibria in the acid denaturation of deoxyribonucleic acids from various sources. *Biopolymers*, 3(3):213–240, 1965.

[36] J. Cadet, L. Voituriez, A. Grand, F. E. Hruska, P. Vigny, and L. S. Kan. Recent aspects of the photochemistry of nucleic acids and related model compounds. *Biochimie*, 67(3-4):277–292, 1985.

[37] N. Chiba and T. Nishizeki. The Hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs. *J. Algorithms*, 10(2):187–211, 1989.

[38] M. G. Clark, D. P. Bloxham, P. C. Holland, and H. A. Lardy. Estimation of the fructose diphosphatase–phosphofructokinase substrate cycle in the flight muscle of Bombus affinis. *Biochem. J.*, 134(2):589–597, 1973.

[39] S. Condamin, O. Bénichou, and M. Moreau. Random walks and brownian motion: a method of computation for first-passage times and related quantities in confined geometries. *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.*, 75(2):021111, 2007.

[40] S. A. Cook. The complexity of theorem-proving procedures. *Proceedings of the 3rd annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 1971.

[41] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. An improved algorithm for matching large graphs. *Proceedings of the 3rd IAPR-TC15 workshop on Graph-based Representations in Pattern Recognition (GbRPR)*, pages 149–159, 2001.

[42] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(10):1367–1372, 2004.

[43] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Inform. Comput.*, 125(1):1–12, 1996.

[44] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. *J. Comp. Biol.*, 5(3):423–465, 1998.

[45] G. Csárdi and T. Nepusz. The igraph software package for complex network research. *InterJournal, vol. Complex Systems*, 1695, 2006.

[46] M. Cygan, S. Kratsch, and J. Nederlof. Fast Hamiltonicity checking via bases of perfect matchings. *Proceedings of the 45th annual ACM Symposium on Theory of Computing (STOC)*, pages 301–310, 2013.

[47] R. J. H. Davies, J. F. Malone, and S. Neidle. High-resolution crystal structure of the intramolecular d(TpA) thymine–adenine photoadduct and its mechanistic implications. *Nucleic Acids Res.*, 35(4):1048–1053, 2007.

[48] M. de Berg and A. Khosravi. Optimal binary space partitions in the plane. *Proceedings of the 13th international Computing and Combinatorics Conference (COCOON)*, pages 216–225, 2010.

[49] K. A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1609, 1985.

[50] T. Douki, A. Reynaud-Angelin, J. Cadet, and E. Sage. Bipyrimidine photoproducts rather than oxidative lesions are the main type of DNA damage involved in the genotoxic effect of solar UVA radiation. *Biochemistry*, 42(30):9221–9226, 2003.

[51] M. Dyer, L. A. Goldberg, C. Greenhill, and M. Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2004.

[52] M. E. Dyer and A. M. Frieze. Planar 3DM is NP-complete. *J. Algorithms*, 7(2):174–184, 1986.

[53] J. Eigner, H. Boedtker, and G. Michaels. The thermal degradation of nucleic acids. *Biochim. Biophys. Acta*, 51(1):165–168, 1961.

[54] M. N. Ellingham and J. D. Horton. Non-Hamiltonian 3-connected cubic bipartite graphs. *J. Combin. Theory Ser. B*, 34(3):350–353, 1983.

[55] D. Eppstein. The traveling salesman problem for cubic graphs. *J. Graph Algorithms Appl.*, 11(1):61–81, 2007.

[56] T. Feder and C. Subi. On Barnette's conjecture. *Rep. TR06-015, Electronic Colloquium on Computational Complexity*, 2006.

[57] L. A. Frederico, T. A. Kunkel, and B. R. Shaw. A sensitive genetic assay for the detection of cytosine deamination: determination of rate constants and the activation energy. *Biochemistry*, 29(10):2532–2537, 1990.

[58] S. M. Freier, R. Kierzek, J. A. Jaeger, N. Sugimoto, M. H. Caruthers, T. Neilson, and D. H. Turner. Improved free-energy parameters for predictions of RNA duplex stability. *PNAS*, 83(24):9373–9377, 1986.

[59] K. Fujimoto, K. Hiratsuka-Konishi, T. Sakamoto, T. Ohtake, K. Shinohara, and Y. Yoshimura. Specific and reversible photochemical labeling of plasmid DNA using photoresponsive oligonucleotides containing 3-cyanovinylcarbazole. *Mol. BioSyst.*, 8(2):491–494, 2012.

[60] K. Fujimoto, S. Kishi, and T. Sakamoto. Geometric effect on the photocrosslinking reaction between 3-cyanovinylcarbazole nucleoside and pyrimidine base in DNA/RNA heteroduplex. *Photochem. Photobiol.*, 89(5):1095–1099, 2013.

[61] K. Fujimoto, K. Konishi-Hiratsuka, T. Sakamoto, and Y. Yoshimura. Site-specific cytosine to uracil transition by using reversible DNA photo-crosslinking. *ChemBioChem*, 11(12):1661–1664, 2010.

[62] K. Fujimoto, S. Matsuda, N. Takahashi, and I. Saito. Template-directed photoreversible ligation of deoxyoligonucleotides via 5-vinyldeoxyuridine. *J. Am. Chem. Soc.*, 122(23):5646–5647, 2000.

[63] K. Fujimoto, S. Matsuda, Y. Yoshimura, T. Matsumura, M. Hayashi, and I. Saito. Site-specific transition of cytosine to uracil via reversible DNA photoligation. *Chem. Commun.*, 30:3223–3225, 2006.

[64] K. Fujimoto, N. Ogawa, M. Hayashi, S. Matsuda, and I. Saito. Template directed photochemical synthesis of branched oligodeoxynucleotides via 5-carboxyvinyldeoxyuridine. *Tet. Lett.*, 41(49):9437–9440, 2000.

[65] K. Fujimoto, A. Yamada, Y. Yoshimura, T. Tsukaguchi, and T. Sakamoto. Details of the ultrafast dna photo-cross-linking

reaction of 3-cyanovinylcarbazole nucleoside: cis-trans isomeric effect and the application for SNP-based genotyping. *J. Am. Chem. Soc.*, 135(43):16161–16167, 2013.

[66] Y. Yoshimura K. Fujimoto. Ultrafast reversible photo-cross-linking reaction: toward in situ DNA manipulation. *Org. Lett.*, 10(15):3227–3230, 2008.

[67] M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar Hamiltonian circuit problem is NP-complete. *SIAM J. Comput.*, 5(4):704–714, 1976.

[68] E. R. Garrett, J. K. Seydel, and A. J. Sharpen. The acid-catalyzed solvolysis of pyrimidine nucleosides. *J. Org. Chem.*, 31(7):2219–2227, 1966.

[69] Q. Ge and D. Štefankovič. The complexity of counting Eulerian tours in 4-regular graphs. *Algorithmica*, 63(3):588–601, 2012.

[70] P. M. Girard, S. Francesconi, M. Pozzebon, D. Graindorge, P. Rochette, R. Drouin, and E. Sage. UVA-induced damage to dna and proteins: direct versus indirect photochemical processes. *J. Phys.: Conf. Ser.*, 261(1):1–10, 2011.

[71] C. Goffin, V. Bailly, and W. G. Verly. Nicks 3' to 5' to AP sites or to mispaired bases, and one-nucleotide gaps can be sealed by T4 DNA ligase. *Nucleic Acids Res.*, 15(21):8755–8771, 1987.

[72] L. M. Goldschlager and I. Parberry. On the construction of parallel computers from various bases of boolean functions. *Theor. Comput. Sci.*, 43(1):43–58, 1986.

[73] P. R. Goodey. Hamiltonian circuits in polytopes with even sided faces. *Israel J. Math.*, 22(1):52–56, 1975.

[74] O. Gotoh, Y. Murakami, and A. Suyama. Multiplex cDNA quantification method that facilitates the standardization of gene expression data. *Nucleic Acids Res.*, 39(10):1–11, 2011.

[75] D. Gouyou-Beauchamps. The Hamiltonian circuit problem is polynomial for 4-connected planar graphs. *SIAM J. Comput.*, 11(3):529–539, 1982.

[76] M. Green and S. S. Cohen. Studies on the biosynthesis of bacterial and viral pyrimidines III. Derivatives of dihydrocytosine. *J. Biol. Chem.*, 228(2):601–609, 1958.

[77] B. Grunbaum. Polytopes, graphs, and complexes. *Bull. Amer. Math. Soc.*, 76:1131–1201, 1970.

[78] A. Hertel. A survey & strengthening of Barnette's conjecture. *Department of Computer Science, University of Toronto, Technical Report*, 2005.

[79] J. F. Hervagault and S. Canu. Bistability and irreversible transitions in a simple substrate cycle. *J. Theor. Biol.*, 127(4):439–449, 1987.

[80] D. A. Holton, B. Manvel, and B. D. McKay. Hamiltonian cycles in cubic 3-connected bipartite planar graphs. *J. Combin. Theory Ser. B*, 38(3):279–297, 1985.

[81] J. Hopcroft and R. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[82] H. B. Hunt, M. V. Marathe, V. Radhakrishnan, and R. E. Stearns. The complexity of planar counting problems. *SIAM J. Comput.*, 27(4):1142–1167, 1998.

[83] OEIS Foundation Inc. The on-line encyclopedia of integer sequences, http://oeis.org/a070968. 2017.

[84] S. Istrail. Statistical mechanics, three-dimensionality and NP-completeness. I. Universality of intractability for the partition function of the Ising model across non-planar lattices. *Proceedings of the 32nd annual ACM Symposium on Theory of Computing (STOC)*, pages 87–96, 2000.

[85] K. Iwama and T. Nakashima. An improved exact algorithm for cubic graph TSP. *Proceedings of the 13th international Computing and Combinatorics Conference (COCOON)*, pages 108–117, 2007.

[86] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabasi. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, 2000.

[87] M. Jerrum. Counting, sampling, and integrating: algorithms and complexity. *Lectures in Mathematics ETH Zürich, Birkhäuser Verlag: Basel, Switzerland*, 2003.

[88] M. Jiang and B. Zhu. Protein folding on the hexagonal lattice in the HP model. *J. Bioinform. Comput. Biol.*, 3(1):19–34, 2005.

[89] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Inform. Process. Lett.*, 27(3):119–123, 1988.

[90] M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res.*, 28(1):27–30, 2000.

[91] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996.

[92] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds.*, pages 85–103, 1972.

[93] R. M. Karp and M. Luby. Monte-Carlo algorithms for enumeration and reliability problems. *Proceedings of the 24th annual symposium on Foundations Of Computer Science (FOCS)*, pages 56–64, 1983.

[94] L. Kazak, E. T. Chouchani, M. P. Jedrychowski, B. K. Erickson, K. Shinoda, P. Cohen, R. Vetrivelan, G. Z. Lu, D. Laznik-Bogoslavski, S. C. Hasenfuss, S. Kajimura, S. P. Gygi, and B. M. Spiegelman. A creatine-driven substrate cycle enhances energy expenditure and thermogenesis in beige fat. *Cell*, 163(3):643–655, 2015.

[95] A. K. Kelmans. Constructions of cubic bipartite 3-connected graphs without Hamiltonian cycles. *Amer. Math. Soc. Transl. (Ser. 2)*, 158:127–140, 1994.

[96] J. Kim and M. Mrksich. Profiling the selectivity of DNA ligases in an array format with mass spectrometry. *Nucleic Acids Res.*, 38(1):1–10, 2010.

[97] M. Kimoto, R. Kawai, T. Mitsui, S. Yokoyama, and I. Hirao. An unnatural base pair system for efficient PCR amplification and functionalization of DNA molecules. *Nucleic Acids Res.*, 37(2):1–9, 2008.

[98] G. Kirchhoff. Über die auflösung der gleichungen, auf welche man bei der untersuchung der linearen verteilung galvanischer ströme geführt wird. *Annalen der Physik und Chemie*, 72:497–508, 1847.

[99] T. P. Kirkman. On the representation of polyedra. *Philosophical Transactions of the Royal Society London*, 146:413–418, 1856.

[100] W. Kladwang, J. Hum, and R. Das. Ultraviolet shadowing of RNA can cause significant chemical damage in seconds. *Scientific Reports*, 2(article 517):1–7, 2012.

[101] S. Klamt, J. Gagneur, and A. von Kamp. Algorithmic approaches for computing elementary modes in large biochemical reaction networks. *IEE Proc. Syst. Biol.*, 152(4),:249–255, 2005.

[102] F. Klepper, K. Polborn, and T. Carell. Robust synthesis and crystal-structure analysis of 7-cyano-7-deazaguanine (preQ0 base) and 7-(aminomethyl)-7-deazaguanine ($preQ_1$ base). *Helv. Chim. Acta*, 88(10):2610–2616, 2005.

[103] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Discrete Math.*, 5(3):422–427, 1992.

[104] T. M. G. Koning, R. J. H. Davies, and R. Kaptein. The solution structure of the intramolecular photoproduct of d(TpA) derived with the use of NMR and a combination of distance geometry and molecular dynamics. *Nucleic Acids Res.*, 18(2):277–284, 1990.

[105] K. Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15(1):271–283, 1930.

[106] V. Labet, C. Morell, T. Douki, J. Cadet, L. A. Eriksson, and A. Grand. Hydrolytic deamination of 5,6-dihydrocytosine in a protic medium: a theoretical study. *J. Phys. Chem. A*, 114(4):1826–1834, 2010.

[107] D. J. R. Laurence. Chain breakage of deoxyribonucleic acid following treatment with low doses of sulphur mustard. *Proc. R. Soc. Lond. A*, 271(1347):520–530, 1963.

[108] P. D. Lawley, J. H. Lethbridge, P. A. Edwards, and K. V. Shooter. Inactivation of bacteriophage T7 by mono- and difunctional sulphur mustards in relation to cross-linking and depurination of bacteriophage DNA. *J. Mol. Biol.*, 39(1):181–198, 1969.

[109] A. Levskaya, O. D. Weiner, W. A. Lim, and C. A. Voigt. Spatiotemporal control of cell signalling using a light–switchable protein interaction. *Nature*, 461(7266):997–1001, 2009.

[110] Y. Li and R. R. Breaker. Kinetics of RNA degradation by specific base catalysis of transesterification involving the 2'-hydroxyl group. *J. Am. Chem. Soc.*, 121(23):5364–5372, 1999.

[111] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):1–16, 1982.

[112] T. Lindahl. Instability and decay of the primary structure of DNA. *Nature*, 362(6422):709–715, 1993.

[113] T. Lindahl and A. Andersson. Rate of chain breakage at apurinic sites in double-stranded deoxyribonucleic acid. *Biochemistry*, 11(19):3618–3623, 1972.

[114] T. Lindahl and O. Karlstrom. Heat-induced depyrimidination of deoxyribonucleic acid in neutral solution. *Biochemistry*, 12(25):5151–5154, 1973.

[115] T. Lindahl and B. Nyberg. Rate of depurination of native deoxyribonucleic acid. *Biochemistry*, 11(19):3610–3618, 1972.

[116] T. Lindahl and B. Nyberg. Heat-induced deamination of cytosine residues in deoxyribonucleic acid. *Biochemistry*, 13(16):3405–3410, 1974.

[117] N. Linial. Hard enumeration problems in geometry and combinatorics. *SIAM J. Alg. Disc. Meth.*, 7(2):331–335, 1986.

[118] M. Liskiewicz, M. Ogihara, and S. Toda. The complexity of counting self-avoiding walks in subgraphs of two-dimensional grids and hypercubes. *Theoret. Comput. Sci.*, 304(1-3):129–156, 2003.

[119] Y. Liu, P. Marchioro, and R. Petreschi. At most single-bend embeddings of cubic graphs. *Appl. Math. J. Chinese Univ. Ser. B*, 9(2):127–142, 1994.

[120] N. Livne. A note on #P-completeness of NP-witnessing relations. *Inf. Process. Lett.*, 109(5):259–261, 2009.

[121] V. T. Luyen, Y. Ooka, S. Alam, H. Suzuki, K. Fujimoto, and T. Tsukahara. Chemical RNA editing as a possibility novel therapy for genetic disorders. *Int. J. Adv. Comp. Sci.*, 2(6):237–241, 2012.

[122] B. D. McKay and A. Piperno. Practical graph isomorphism, II. *J. Symbolic. Comput.*, 60:94–112, 2014.

[123] B. M. E. Moret. Planar NAE3SAT is in P. *ACM SIGACT News*, 19(2):51–54, 1988.

[124] S. Mouret, C. Baudouin, M. Charveron, A. Favier, J. Cadet, and T. Douki. Cyclobutane pyrimidine dimers are predominant DNA lesions in whole human skin exposed to UVA radiation. *PNAS*, 103(37):13765–13770, 2006.

[125] A. Munaro. On line graphs of subcubic triangle-free graphs. *Discrete Math.*, 340(6):1210–1226, 2017.

[126] S. Nakamura, S. Ogasawara, S. Matuda, I. Saito, and K. Fujimoto. Template directed reversible photochemical ligation of oligodeoxynucleotides. *Molecules*, 17(1):163–178, 2012.

[127] E. A. Newsholme and B. Crabtree. Substrate cycles in metabolic regulation and in heat generation. *Biochem. Soc. Symp.*, 41:61–109, 1976.

[128] N. Nishida, T. Tanabe, K. Hashido, K. Hirayasu, M. Takasu, A. Suyama, and K. Tokunaga. DigiTag assay for multiplex single nucleotide polymorphism typing with high success rate. *Anal. Biochem.*, 346(2):281–288, 2005.

[129] N. Nishida, T. Tanabe, M. Takasu, A. Suyama, and K. Tokunaga. Further development of multiplex single nucleotide polymorphism typing method, the digiTag2 assay. *Anal. Biochem.*, 364(1):78–85, 2007.

[130] J. M. Nitsche, H. C. Chang, P. A. Weber, and B. J. Nicholson. A transient diffusion model yields unitary gap junctional permeabilities from images of cell-to-cell fluorescent dye transfer between Xenopus oocytes. *Biophys. J.*, 86(4):2058–2077, 2004.

[131] S. Ogasawara and K. Fujimoto. A novel method to synthesize versatile multiple-branched DNA (MB-DNA) by reversible photochemical ligation. *ChemBioChem*, 6(10):1756–1760, 2005.

[132] M. Ogino and K. Fujimoto. Photochemical synthesis of R-shaped DNA toward DNA recombination and processing in vitro. *Angew. Chem. Int. Ed.*, 45(43):7223–7226, 2006.

[133] M. Ogino, D. Okamura, and K. Fujimoto. Replication of cyclobutane pyrimidine dimer analogue by Ex Taq DNA polymerase. *Sci. Technol. Adv. Mater.*, 8(4):318–322, 2007.

[134] M. Ogino, Y. Taya, and K. Fujimoto. Highly selective detection of 5-methylcytosine using photochemical ligation. *Chem. Commun.*, 45:5996–5998, 2008.

[135] M. Ogino, Y. Taya, and K. Fujimoto. Detection of methylcytosine by DNA photoligation via hydrophobic interaction of the alkyl group. *Org. Biomol. Chem.*, 7(15):3163–3167, 2009.

[136] J. G. Oxley. *Matroid theory*. Oxford University Press: New York, NY, 1st edition, 1992.

[137] A. Pagourtzis and S. Zachos. The complexity of counting functions with easy decision version. *Proceedings of the 31st international symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 741–752, 2006.

[138] V. S. Pande, A. U. Grosberg, C. Joerg, and T. Tanaka. Enumerations of the Hamiltonian walks on a cubic sublattice. *J. Phys. A: Math. Gen.*, 27(18):6231–6236, 1994.

[139] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley: Reading, MA, 1st edition, 1994.

[140] C. H. Papadimitriou and S. Zachos. Two remarks on the power of counting. *Proceedings of the 6th GI-Conference in Theoretical Computer Science (GITCS)*, pages 269–276, 1983.

[141] B. H. Park and M. Levitt. The complexity and accuracy of discrete state models of protein structure. *J. Mol. Biol.*, 249(2):493–507, 1995.

[142] H. J. Park, K. Zhang, Y. Ren, S. Nadji, N. Sinha, J. S. Taylor, and C. H. Kang. Crystal structure of a DNA decamer containing a cis-syn thymine dimer. *PNAS*, 99(25):15965–15970, 2002.

[143] S. V. Pemmaraju and S. S. Skiena. *Computational discrete mathematics: combinatorics and graph theory with Mathematica*. Cambridge University Press: New York, NY, 2003.

[144] D. Perdiz, P. Grof, M. Mezzina, O. Nikaido, E. Moustacchi, and E. Sage. Distribution and repair of bipyrimidine photoproducts in solar UV-irradiated mammalian cells. Possible role of Dewar photoproducts in solar mutagenesis. *J. Biol. Chem.*, 275(35):26732–26742, 2000.

[145] G. P. Pfeifer. Formation and processing of UV photoproducts: effects of DNA sequence and chromatin environment. *Photochem. Photobiol.*, 65(2):270–283, 1997.

[146] J. Plesnik. The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two. *Inform. Process. Lett.*, 8(4):199–201, 1979.

[147] S. Podell, W. Maske, E. Ibanez, and E. Jablonski. Comparison of solution hybridization efficiencies using alkaline phosphatase-labelled and 32P-labelled oligodeoxynucleotide probes. *Mol. Cell Probes*, 5(2):117–124, 1991.

[148] C. E. Pritchard and E. M. Southern. Effects of base mismatches on joining of short oligodeoxynucleotides by DNA ligases. *Nucleic Acids Res.*, 25(17):3403–3407, 2007.

[149] A. Radzicka and R. Wolfenden. A proficient enzyme. *Science*, 267(5194):90–93, 1995.

[150] E. Sage. Distribution and repair of photolesions in DNA: genetic consequences and the role of sequence context. *Photochem. Photobiol.*, 57(1):163–174, 1993.

[151] I. Saito, Y. Miyauchi, Y. Saito, and K. Fujimoto. Template-directed photoreversible ligation of DNA via 7-carboxyvinyl-7-deaza-2'-deoxyadenosine. *Tet. Lett.*, 46(1):97–99, 2005.

[152] J. Jr. SantaLucia. A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *PNAS*, 95(4):1460–1465, 1998.

[153] T. J. Schaefer. The complexity of satisfiability problems. *Proceedings of the 23rd annual ACM Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.

[154] C. H. Schilling, D. Letscher, and B. O. Palsson. Theory for the systemic definition of metabolic pathways and their use in interpreting metabolic function from a pathway-oriented perspective. *J. Theor. Biol.*, 203(3):229–248, 2000.

[155] S. Schuster and C. Hilgetag. On elementary flux modes in biochemical reaction systems at steady state. *J. Biol. Syst.*, 2(2):165–182, 1994.

[156] E. Shakhnovich and A. Gutin. Enumeration of all compact conformations of copolymers with random sequence of links. *J. Chem. Phys.*, 93(8):5967–5971, 1990.

[157] R. Shapiro and M. Danzig. Acidic hydrolysis of deoxycytidine and deoxyuridine derivatives. The general mechanism of deoxyribonucleoside hydrolysis. *Biochemistry*, 11(1):23–29, 1972.

[158] A. Shigeno, T. Sakamoto, Y. Yoshimura, and K. Fujimoto. Quick regulation of mRNA functions by a few seconds of photoirradiation. *Org. Biomol. Chem.*, 10(38):7820–7825, 2012.

[159] J. Simon. On the difference between one and many. *Proceedings of the 4th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 480–491, 1977.

[160] M. Sipser. *Introduction to the theory of computation*. Thomson Course Technology: Boston, MA, 2nd edition, 2005.

[161] C. A. B. Smith and W. T. Tutte. On unicursal paths in a network of degree 4. *Amer. Math. Monthly*, 48(4):233–237, 1941.

[162] C. Solnon. Alldifferent-based filtering for subgraph isomorphism. *Artificial Intelligence*, 174(12-13):850–864, 2010.

[163] R. P. Stanley. Acyclic orientations of graphs. *Discrete Math.*, 5(2):171–178, 1973.

[164] W. A. Stein et al. *Sage Mathematics Software (Version 7.2.0)*. The Sage Development Team, (2016). `http://www.sagemath.org`.

[165] E. Steinitz. Polyeder und raumeinteilungen. *Encyklopadie der mathematischen Wissenschaften. Bd. III-1B, Hft. 9*, pages 1–139, 1922.

[166] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoret. Comput. Sci.*, 3(1):1–22, 1976.

[167] H. Sugiyama, T. Fujiwara, A. Ura, T. Tashiro, K. Yamamoto, S. Kawanishi, and I. Saito. Chemistry of thermal degradation of abasic sites in DNA. Mechanistic investigation on thermal DNA strand cleavage of alkylated DNA. *Chem. Res. Toxicol.*, 7(5):673–683, 1994.

[168] C. Switzer, S. E. Moroney, and S. A. Benner. Enzymatic incorporation of a new base pair into DNA and RNA. *J. Am. Chem. Soc.*, 111(21):8322–8323, 1989.

[169] P. G. Tait. Listing's topologie. *Philosophical Magazine (5th ser.)*, pages 30–46, 1884.

[170] C. Thomassen. The graph genus problem is NP-complete. *J. Algorithms*, 10(4):568–576, 1989.

[171] Texas Instruments (TI). "dlp9500: Dlp® 0.95 1080p 2x lvds type a dmd." (posted: Aug. 2012, revised march 2017). *http://www.tij.co.jp/jp/lit/ds/symlink/dlp9500.pdf*.

[172] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[173] J. Tong, W. Cao, and F. Barany. Biochemical properties of a high fidelity DNA ligase from thermus species AK16D. *Nucleic Acids Res.*, 27(3):788–794, 1999.

[174] J. Torán. Structural properties of the counting hierarchies. *Ph.D Thesis (Facultat d'Informatica de Barcelona, Barcelona)*, 1988.

[175] M. Tsuruoka, K. Yano, K. Ikebukuro, H. Nakayama, Y. Masuda, and I. Karube. Optimization of the rate of DNA hybridization and rapid detection of methicillin resistant Staphylococcus aureus DNA using fluorescence polarization. *J. Biotechnol.*, 48(3):201–208, 1996.

[176] W. T. Tutte. On Hamiltonian circuits. *J. London Math. Soc.*, s1-21(2):98–101, 1946.

[177] W. T. Tutte. A theorem on planar graphs. *Trans. Amer. Math. Soc.*, 82:99–116, 1956.

[178] W. T. Tutte. On the 2-factors of bicubic graphs. *Discrete Math.*, 1(2):203–208, 1971.

[179] R. M. Tyrrell. Induction of pyrimidine dimers in bacterial DNA by 365 nm radiation. *Photochem. Photobiol.*, 17(1):69–73, 1973.

[180] L. G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2):189–201, 1979.

[181] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

[182] L. G. Valiant. Completeness for parity problems. *Proceedings of the 11th international Computing and Combinatorics Conference (COCOON)*, pages 1–8, 2005.

[183] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoret. Comput. Sci.*, 47:85–93, 1986.

[184] T. van Aardenne-Ehrenfest and N. G. de Bruijn. Circuits and trees in oriented linear graphs. *Simon Stevin: wis- en natuurkundig tijdschrift*, 28:203–217, 1951.

[185] M. Las Vergnas. Convexity in oriented matroids. *J. Combin. Theory Ser. B*, 29(2):231–243, 1980.

[186] K. Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114:570–590, 1937.

[187] P. A. Weber, H. C. Chang, K. E. Spaeth, J. M. Nitsche, and B. J. Nicholson. The permeability of gap junction channels to probes of different size is dependent on connexin composition and permeant-pore affinities. *Biophys. J.*, 87(2):958–973, 2004.

[188] D. J. A. Welsh. *Complexity: knots, colourings and counting*. Cambridge University Press: Cambridge, UK, 1st edition, 1993.

[189] H. Whitney. Congruent graphs and the connectivity of graphs. *Amer. J. Math.*, 54(1):150–168, 1932.

[190] Wolfram Research, Inc. *Mathematica 10.4.1.0 (64-bit)*, (2016). `http://www.wolfram.com`.

[191] M. Yamamoto. Approximately counting paths and cycles in a graph. *Discrete Appl. Math.*, 217(2):381–387, 2017.

[192] C. Yang, Y. Yu, K. Liu, D. Song, L. Wu, and H. Su. [2+2] photocycloaddition reaction dynamics of triplet pyrimidines. *J. Phys. Chem. A*, 115(21):5335–5345, 2011.

[193] M. Yokomori. *Personal communication*.

[194] M. Yokomori, O. Gotoh, Y. Murakami, K. Fujimoto, and A. Suyama. A multiplex RNA quantification method to determine the absolute amounts of mRNA without reverse transcription. *Anal. Biochem.*, 539:96–103, 2017.

[195] M. Yokomori, O. Gotoh, and A. Suyama. A multiplex and sensitive rna quantification method for determining the absolute amounts of mRNAs without reverse transcription processes. *Conference abstract:* 定量生物学の会第五回年会, 2012.

[196] M. Yokomori, O. Gotoh, and A. Suyama. Highly parallel and sensitive method for analyzing gene expression kinetics. *Conference abstract: CBI*学会*2013*年大会-生命医薬情報学連合大会-テーマ「オミックス　計算　そして創薬」, 2013.

[197] Y. Yoshimura, Y. Ito, and K. Fujimoto. Interstrand DNA photocrosslinking by photoresponsive artificial nucleic acid. *Nucleic Acids Symp. Ser. (Oxf)*, 48(1):81–82, 2004.

[198] Y. Yoshimura, Y. Ito, and K. Fujimoto. Interstrand photocrosslinking of DNA via p-carbamoylvinyl phenol nucleoside. *Bioorg. Med. Chem. Lett.*, 15(5):1299–1301, 2005.

[199] Y. Yoshimura, Y. Noguchi, H. Sato, and K. Fujimoto. Template-directed DNA photoligation in rapid and selective detection of RNA point mutations. *ChemBioChem*, 7(4):598–601, 2006.

[200] Y. Yoshimura, T. Ohtake, H. Okada, T. Ami, T. Tsukaguchi, and K. Fujimoto. SNP genotyping by DNA photoligation: application to SNP detection of genes from food crops. *Sci. Technol. Adv. Mater.*, 10(3):1–4, 2009.

[201] Y. Yoshimura, T. Ohtake, H. Okada, and K. Fujimoto. A new approach for reversible RNA photocrosslinking reaction: application to sequence-specific RNA selection. *ChemBioChem*, 10(9):1473–1476, 2009.

[202] Y. Yoshimura, H. Okada, and K. Fujimoto. Photoreversible DNA end capping for the formation of hairpin structures. *Org. Biomol. Chem.*, 8(7):1523–1526, 2010.

[203] V. Zanko. #P-completeness via many-one reductions. *Int. J. Found. Comput. Sci.*, 2(1):77–82, 1991.

[204] R. B. Zhang and L. A. Eriksson. A triplet mechanism for the formation of cyclobutane pyrimidine dimers in UV-irradiated DNA. *J. Phys. Chem. B*, 110(14):7556–7562, 2006.

[205] X. Zhao, S. Nadji, J. L. F. Kao, and J. S. Taylor. The structure of d(TpA)*, the major photoproduct of thymidylyl-(3'-5')-deoxyadenosine. *Nucleic Acids Res.*, 24(8):1554–1560, 1996.

[206] C. Zimmer, G. Luck, H. Venner, and J. Fric. Studies on the conformation of protonated DNA. *Biopolymers*, 6(4):563–574, 1968.