IEICE TRANS. FUNDAMENTALS, VOL. E80-A, NO. 4 APRIL 1997

PAPER Special Section on Discrete Mathematics and Its Application

Counting the Number of Paths in a Graph via BDDs

Kyoko SEKINE†, *Nonmember and* Hiroshi IMAI†, *Member*

SUMMARY This paper proposes a unified approach by means of the binary decision diagram, BDD in short, to solve #P-hard problems of counting the number of paths between two terminals in undirected and directed graphs. Our approach provides algorithms running in $O(2^{O(\sqrt{n})})$ time for typical planar graphs such as grid graphs. In fact, for any class of graphs having a good elimination ordering, this paradigm provides efficient solutions.

key words: *graphs, paths, #P-complete, BDD*

1. Introduction

Developing efficient algorithms for hard problems are very important. #P-hard problems form a class of counting problems hard to solve. The pioneering paper concerning #P-hard problems by Valiant [19] shows that counting the number of paths between two terminals of a graph in undirected and directed cases is #P-hard.

This paper proposes a unified approach by means of the binary decision diagram, BDD in short, to solve #P-hard problems for the path counting problem. Specifically, our approach provides the following solutions to the above-mentioned #P-complete problems:

- $O(n^{O(n^{\alpha})})$-time algorithm for a class of $O(n^{\alpha})$-separable graph with n vertices $(0 < \alpha < 1)$,
- $O(2^{O(\sqrt{n})})$-time algorithm for typical planar graphs such as grid graphs.

In fact, for a class of graphs having a good elimination ordering, say a constant bandwidth, this paradigm provides efficient solutions. Of course these bounds are still exponential, but since very related problems are #P-hard [15] even when restricted to planar cases and this would hold also for the cases here, this would be inevitable. A crucial point is that the exponents above are sublinear in n, so that we can solve moderate-size counting problems exactly by this paradigm.

This framework is an extension of constructing the BDD representing all spanning trees of a graph in an output-size sensitive manner developed by Sekine, Imai and Tani [16]. In that paper, an algorithm is presented which constructs the BDD of spanning trees for graphs with at most 14 vertices and $\binom{14}{2} = 91$ edges and planar graphs such as 12×12 lattice graph with 144 vertices and 264 edges in reasonable time. However, in our previous paper, there was a restriction on the edge ordering for BDDs. We here generalize it by using isomorphism induced by contractions so that edges can be ordered arbitrarily.

For the problem of counting the number of paths, we combine the BDD representing forests with another BDD representing the flow conservation constraints so that meaningless cycles can be removed.

As for related research, first, we should note that the path counting problem in a acyclic graph is linear-time solvable. In fact, in the case of the ordinary BDD representing a Boolean function, the number of paths from the root to the 1-node is the number of truth assignments which make the function true, and this number can be computed in time linear to the size of the BDD, as is practically implemented.

Next, we may utilize existing path enumeration algorithms, since counting problems can naturally be solved by their corresponding enumeration algorithms (e.g., see Eppstein [8] and its references). However, such algorithms inevitably requires time at least proportional to the number of paths, which is purely exponential in most cases. Our approach may compute the number in time sublinear to itself by compactly representing paths via a BDD.

There are many papers discussing the number of specific paths, especially Hamiltonian paths, since they are connected with other discrete problems. There are a few papers which directly discuss the problem of counting the number of paths [2], [3], but as far as the authors know algorithms proposed so far have time complexity of $\Omega(2^n)$ for a n-vertex graph, and our bounds are better in general.

Finally, the path counting problem itself is considered to be a type of network reliability problem. For example, one may easily modify the path counting algorithms to count the number of paths passing through a specified edge; this number compared with other path numbers represents a certain degree of 'importance' of the edge. Also, in connection with the general network reliability problem, much has been done. The network reliability itself forms a wide field of research. See books by Colbourn [6], Harms et al. [9], and also that by Welsh [20] for its relations to other fundamental

Manuscript received September 11, 1996.
Manuscript revised November 12, 1996.

†The authors are with the Department of Information Science, University of Tokyo, Tokyo, 113 Japan.

combinatorial structures and problems.

2. BDD of Spanning Trees by Sharing Some Isomorphic Minors

We first describe necessary definitions. Let $G = (V, E)$ be a simple connected undirected graph with a vertex set V and an edge set E. A graph obtained from G by deletions and contractions of edges is called a *minor* of G. A loop is an edge connecting the same vertex, and a coloop (or isthmus) is an edge whose removal decreases the rank of the graph by 1.

Suppose we apply contraction or deletion operation for edges in the order of $e_1, e_2, \ldots, e_m$ $(m = |E|)$. Here, if e_i is a loop in a minor obtained by operations on $e_1 \ldots, e_{i-1}$, we only consider deletion operation for it, and, if e_i is a coloop in the minor, we only consider contraction operation. This process can be represented as a binary tree such that the root corresponds to the original graph G, each node† in the tree corresponds to a minor obtained by the operations specified by the path from the root to the node. For the minor with e_i as a loop or coloop as above, the corresponding node have only one child.

By the operations above, each leaf of the expansion tree corresponds to a spanning tree of G one-to-one, that is, to a spanning tree formed by the contracted edges by the operations along the path from the root to the leaf. In this expansion tree, nodes can be divided by their levels in the tree, and nodes in the same level correspond to minors on the same edge subset. The root is considered to be in the 0-th level of the expansion tree. For $i = 1, \ldots, m$, define the *i-th elimination front* $\widetilde{V}_i$ of the i-th level to be a vertex subset consisting of vertices v such that v is incident to some edges e_j with $j \leqq i$ and some edges e_k with $k > i$.

Each node in the i-th level of the expansion tree corresponds to a distinct subset of contracted edges for $\{e_1, \ldots, e_i\}$. For a node in the level, by the contracted edges for this node, we can define an equivalence relation on $\widetilde{V}_i$ such that two vertices are in the same equivalence class iff, in the process of obtaining the minor, they are unified into one vertex by the contractions. Then consider a partition of $\widetilde{V}_i$ into the equivalence classes by this relation. We call this partition the *i-th elimination partition* of this node.

For example, for K_4 at the top of Fig. 1 with vertices $v_1, \ldots, v_4$ and edges numbered by their indices from 1 to 6, $\widetilde{V}_1 = \{v_1, v_4\}$ and $\widetilde{V}_2 = \{v_1, v_2, v_4\}$; that is, from $\widetilde{V}_1$ to $\widetilde{V}_2$, v_2 comes into the elimination front. $\widetilde{V}_3 = \{v_1, v_2, v_3, v_4\}$ and $\widetilde{V}_4 = \{v_1, v_2, v_3\}$; that is, from $\widetilde{V}_3$ to $\widetilde{V}_4$, v_4 goes out of the elimination front. As for the elimination partition, for nodes in the level 1, $\{\{v_1, v_4\}\}$ is the 1st elimination partition of a node obtained by contracting e_1, while $\{\{v_1\}, \{v_4\}\}$ is that of a node obtained by deleting e_1.

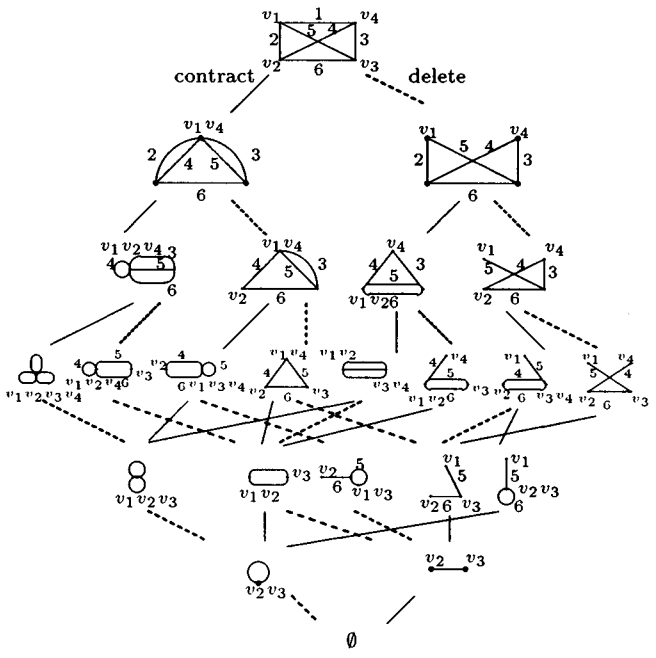


Fig. 1 BDD of spanning trees for K_4.

Then, the following holds.

Lemma 1: If two minors in the same level of the expansion tree have the same i-th elimination partition, they are isomorphic.

proof: By considering the map induced by the identical elimination partition for vertices in the elimination front, and considering the identity map for the other vertices, an isomorphism is constructed. □

If there are isomorphic minors in the same level of the expansion tree, the subtree rooted at the nodes corresponding to these minors are also isomorphic. Hence, we can completely share these isomorphic subtrees without losing the information concerning spanning trees of G (we also combine all leaves into one), which makes the expansion tree into a rooted acyclic graph with the single source (root) and single sink (edges are oriented from the root to the sink). In this graph, each path from the root to the sink corresponds to a spanning tree of G one-to-one. We call this acyclic graph the BDD of spanning trees by the partition isomorphism with respect to the edge ordering $e_1, e_2, \ldots, e_m$ (The definition of BDDs of Boolean functions is given in Sect. 3.1). See an example of the BDD for K_4 in Fig. 1. The size of this BDD is defined to be the number of its nodes. The width of a level is defined to be the number of nodes at the level. The width of this BDD is defined to be the maximum among the widths of all levels.

Lemma 1 can be used not only to transform the expansion tree to the BDD but also to analyze the size of

† We use the term 'node' for a vertex of this binary tree, called an expansion tree, and a vertex of BDDs to distinguish them from vertices of a given graph.

the BDD. Concerning the latter, we have the following.

Theorem 1: Let l be the maximum size of the elimination front for a given edge ordering. Then, the width of the BDD of all spanning trees of G by the partition isomorphism is bounded by B_l, where B_l, called the Bell number, is the number of partitions of a set of l elements.

proof: From Lemma 1, the number of nodes of this BDD in the i-th level is bounded by the number of distinct partitions of the i-th elimination front. Then, from the definitions of Bell number and width of the BDD, the theorem follows. □

A known formula of B_l is complicated. Trivially, $B_l \leqq l^l$ and B_l is much smaller than l^l in practice. Asymptotically, B_l is slightly less than $\Theta(l^l)$ (see [14]).

The size of the BDD depends on the ordering of edges. The existence of a good ordering of edges to make the size of BDD smaller has strong connection with the existence of a small separators. There are useful classes of graph satisfying this property, especially the class of planar graphs (see Lipton and Tarjan [13] and Alon, Seymour, and Thomas [1]).

We now recall the definition of separators of graphs. For a connected undirected graph $G = (V, E)$ with $|V| = n$ vertices, G is $f(n)$-separable iff V can be partitioned into three subsets A, B and C, such that no edge of G joins a vertex in A with a vertex in B, neither A nor B contains more than βn ($\frac{1}{2} \leqq \beta < 1$) vertices, and C (called a separator) contains no more than $f(n)$ vertices. Let $\mathcal{G}_\alpha$ be a class of graphs having $O(n^\alpha)$-separability closed under subgraph relation for $0 < \alpha < 1$. The planar separator theorem [13] states that any planar graph with n vertices is in $\mathcal{G}_{1/2}$.

The separator partition can recursively be applied to graphs induced by A, B belonging to $\mathcal{G}_\alpha$. Then, from this recursive decomposition, we can naturally define the following edge ordering: Suppose A, B, C are as in the definition of separability. We first rename vertices from 1 to n in the order of A, B, C, where vertices in A, and those in B are recursively arranged among then by applying the separator partition, and vertices in C are arranged arbitrarily. This vertex ordering is used in the generalized nested dissection [12]. Then, edges e connecting vertices u, v are ordered in the increasing lexicographic order of (u, v) with $u < v$. The following lemma bounds the size of elimination front for this ordering.

Lemma 2: For an $O(n^\alpha)$-separable graph in $\mathcal{G}_\alpha$ with $0 < \alpha < 1$, the size of elimination front is $O(n^\alpha)$ at any level under the edge ordering by separator partitions.

proof: It is easy to see that, for the edge ordering, the size of the elimination front at any level is at most the sum of the separator size of each level of the separator decomposition. This sum is bounded for some constant c by

$$c\left(n^\alpha + \left(\frac{2}{3}n\right)^\alpha + \left(\left(\frac{2}{3}\right)^2 n\right)^\alpha + \cdots + \left(\left(\frac{2}{3}\right)^{O(\log n)} n\right)^\alpha\right) = O(n^\alpha)$$

□

If we use an edge ordering induced from a vertex ordering as in this lemma, the existence of a good elimination ordering implies the existence of a good separator; simply consider the elimination front at the stage that the first half of vertices in the vertex ordering have just been deleted from the front, and then the elimination front at this stage is a separator.

Combining Theorem 1 and this lemma, for graphs in $\mathcal{G}_\alpha$, and further with the top-down breadth-first construction algorithm in [18], we have the following.

Theorem 2: Under the edge ordering based on separator decomposition, for an n-vertex graph in $\mathcal{G}_\alpha$ ($0 < \alpha < 1$). The width of the BDD representing all spanning trees is $2^{O(n^\alpha \log n)}$, and this BDD can be computed in $2^{O(n^\alpha \log n)}$ time.

proof: From Theorem 1 and Lemma 2. □

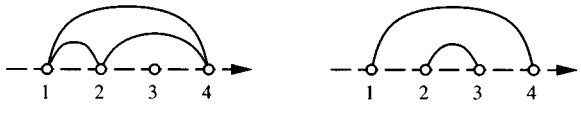
In [16], by imposing a condition on the edge ordering, this algorithm is shown to produce a canonical BDD from the viewpoint of Boolean function. Here, we do not impose such a condition, and hence lose the canonical property. However, for reliability computation, this is sufficient and enables us to use the ordering induced from the separators which does not satisfy the condition in general.

For planar graphs, a better bound holds by virtue of the planarity. This follows from a fact that possible patterns of elimination partitions under planar constraints are limited, and their number is expressed in terms of the Catalan number, which is well-known in combinatorics (e.g., see [17], [18]). Although the following holds for a general planar graph, we here only prove this theorem in the case of bounded-degree planar graphs to give a short proof.

The following lemma is used in the proof.

Lemma 3: Consider N points on a horizontal line in the plane and suppose that, for some pairs among the points, each pair is connected by a continuous curve lying in the upper half plane. Supposing that, when curves intersect, their endpoints are all in the same connected component, the number of possible distinct partitions of the points induced by the connected components composed of arbitrary curves is given by the Catalan number $C_{N+1} = \frac{1}{N+1}\binom{2N}{N} \leqq 4^N$.

An illustrative example is given in Fig. 2 with two induced partitions for $N = 4$ points $\{1, 2, 3, 4\}$. Since we take intersections among curves in the upper half plane into account, if there is a curve connecting points 1 and 3 and another curve 2 and 4, then these four points are contained in the same connected component. With



(a) $\{\{1,2,4\},\{3\}\}$ (b) $\{\{1,4\},\{2,3\}\}$

Fig. 2 Two ways of partitioning points on a line by curves in the upper half plane.

this observation, the number of partitions obtained in this way is seen to be C_{N+1} by using standard techniques in combinatorics, and the proof is omitted.

Theorem 3: Under the edge ordering in Lemma 2 for an n-vertex planar graph such that the degree of each vertex is at most some constant c, the width of the BDD representing all spanning trees is $O(2^{O(\sqrt{n})})$, and it can be computed in $O(2^{O(\sqrt{n})})$ time.

proof: The edge ordering in Lemma 2 is defined based on the vertex ordering. To prove the theorem, we have only to bound the width of the i-th level such that there is a vertex numbered u and $e_1, \ldots, e_i$ coincide with edges incident to some vertex numbered at most u. This is because the degree of a vertex numbered $u+1$ is at most c and in the subsequent levels corresponding to edges incident to this vertex the width may become large by at most a factor of 2^c.

Now, let us consider the i-th level satisfying the above condition. Vertices are partitioned into a subset V_u of vertices numbered at most u and its complement. Consider the connected components of the subgraph induced by vertex set V_u. For each component, consider a subset V' of the i-th elimination front which are adjacent to some vertex in the component. By the contractions and deletions of edges incident to vertices in the component, the number of partitions induced by the connectedness of contracted edges is bounded by $O(2^{O(|V'|)})$ from Lemma 3.

The multiplication of these numbers for all the connected components is an upper bound on the size of possible elimination partitions. Since the degree is bounded by c, this multiplication is bounded by $O(2^{O(\sqrt{n})})$. □

Figure 3 illustrates the proof by a small example.

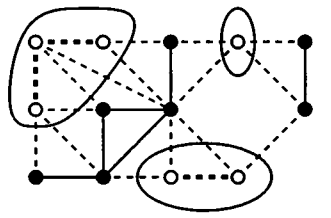


Fig. 3 Explanation of the proof: ○ vertices are numbered at most u and • vertices are numbered more than u; dotted lines are numbered at most i and real lines are numbered more than i; there are three connected components of V_u; the connected components of sizes 3, 2, 1 are adjacent to subsets, of the elimination front, of sizes 5, 3, 4, respectively, and in this case the proof here bounds the number of elimination partitions by $C_{5+1} \times C_{3+1} \times C_{4+1}$.

3. Computing the Number of Paths

We now consider the path counting problem. To solve this problem, we need more properties of BDDs, and before going into detail about our results we first describe basics of BDDs.

3.1 Ordered Binary Decision Diagrams

An ordered binary decision diagram, OBDD in short, is a binary branching program representing a Boolean function with some conditions. In this paper we only consider the subgraph of OBDD reachable to the 1-node, and call this OBDD.

An OBDD represents a Boolean function f of m variables $x_1, \ldots, x_m$ by a labeled acyclic graph with a single source (root) and a single sink (1-node). Each node besides the sink has at most two edges emanating from it, one is labeled as 0-edge and the other 1-edge. The sink node is labeled as 1, and is called the 1-node. All the directed paths from the source to the 1-node have the same number of edges, and the level of a node is defined to be the number of edges of directed paths from the source to the node. The level of the source is 0, and that of the sink is m. Nodes in the $(i-1)$-th level $(i = 0, \ldots, m-1)$ correspond to a variable x_i. Each directed path from the root to the 1-node corresponds one-to-one to an assignment of x_i to the label of the edge, emanating from the node of x_i on this path $(i = 1, \ldots, m)$, with which the function value is 1.

The width w_i of the i-th level of OBDD of f is the number of nodes in the i-th level. We also define the proper width $\tilde{w}_i$ of the i-th level as follows. We can consider the OBDD of the negation of f, and denote the width of the i-th level of this OBDD by w'_i. Then, the proper width of the OBDD of f is defined to be $\tilde{w}_i = \max\{w_i, w'_i\}$. The width of OBDD is the maximum among the widths over all levels. The size of OBDD is the total number of nodes, and varies by the ordering of variables.

In the sequel, we regard the logical values of 0 and 1 as integers, and consider addition among them as integers. This type of Boolean formula is called an arithmetic Boolean formula. We denote the Boolean AND, OR, NOT by $\bigwedge$, $\bigvee$, $\overline{x_i}$, respectively.

Bryant [5] proposes algorithms for taking Boolean operations among OBDDs. Concerning the time to perform such operations and the size of computed OBDDs for nontrivial functions, the following holds which can be obtained by careful analysis, with using some ideas in [18]. We here omit the proof due to the space limitation.

Lemma 4: Suppose two OBDDs whose proper width at the i-th level are $\tilde{w}_i^-$ and $\tilde{w}_i^+$. Then, the OBDD of a

function obtained by taking the Boolean AND or OR for two functions of given OBDDs can be computed in $O(\sum_{i=0}^{m} \widetilde{w}_i^- \widetilde{w}_i^+)$. Furthermore, its proper width at the i-th level is $O(\widetilde{w}_i^- \widetilde{w}_i^+)$.

proof: First, note that, when we consider OBDDs explicitly having the 0-node, the corresponding property to this lemma is well known (the conventional algorithm by Bryant [5] proves this).

For a function f obtained by taking the Boolean AND of two given functions, it becomes true iff the two original functions are true. Hence, at the i-th level, the width of the BDD of f is at most $\widetilde{w}_i^- \widetilde{w}_i^+$.

In the case of the Boolean OR, the corresponding function becomes true iff one of the two functions are true. Then, the width is trivially bounded by $(w_i^- + 1)(w_i^+ + 1)$ by using the width w_i^- and w_i^+. By carefully analyzing relations between the width and the proper width, we can obtain the lemma. □

3.2 OBDDs Representing 1-Flows

As is well known, problems related to paths such as the shortest path problem can be formulated as a flow problem. This subsection shows that flows of value 1 can be represented by a compact OBDD.

First, consider the undirected case. Denote by δv the set of edges incident to v in G. Define a Boolean function $flow(\boldsymbol{x})$ of $\boldsymbol{x} = (x_1, \ldots, x_m)$ for graph $G = (V, E)$ with two terminal vertices s and t by

$$flow_{s-t}(\boldsymbol{x}) = \bigwedge_{v \in V-\{s,t\}} \left\{ \left(\sum_{i \in \delta v} x_i = 2\right) \vee \left(\sum_{i \in \delta v} x_i = 0\right) \right\} \wedge \left(\sum_{k \in \delta s} x_k = 1\right) \wedge \left(\sum_{l \in \delta t} x_l = 1\right)$$

This simply represents the flow conservation condition, and hence we have the following.

Lemma 5: $flow_{s-t}(\boldsymbol{x})$ becomes 1 iff edges with $x_i = 1$ correspond to a flow of value 1 between s and t in G.

In the directed case, denoting by $\delta^+ v$ and $\delta^- v$ the sets of edges emanating and entering the vertex v, respectively, we similarly define a function $flow_{s \to t}(\boldsymbol{x})$ by

$$flow_{s \to t}(\boldsymbol{x}) = \bigwedge_{v \in V-\{s,t\}} \left\{ \left(\sum_{i \in \delta^+ v} x_i = 1 \wedge \sum_{j \in \delta^- v} x_j = 1\right) \vee \left(\sum_{i \in \delta^+ v} x_i = 0 \wedge \sum_{j \in \delta^- v} x_j = 0\right) \right\} \wedge \left(\sum_{k \in \delta^+ s} x_k = 1\right) \wedge \left(\sum_{k \in \delta^- s} x_k = 0\right) \wedge \left(\sum_{l \in \delta^+ t} x_l = 0\right) \wedge \left(\sum_{l \in \delta^- t} x_l = 1\right)$$

Lemma 6: $flow_{s \to t}(\boldsymbol{x})$ is a Boolean function representing all flows of value 1 from s to t in G.

We now analyze the size of BDD representing these flows when the graph has an edge ordering with small elimination front. The discussion for the undirected case using δv and that for the directed case using δ^+ and δ^- are almost similar, and in the sequel we just consider the directed case. $\delta^{\pm}$ means one of δ^+ and δ^-. Suppose edges are ordered from e_1 to e_m.

Lemma 7: For an OBDD of $(\sum_{i \in \delta^+ v} x_i = 1 \wedge \sum_{j \in \delta^- v} x_j = 1) \vee (\sum_{i \in \delta^+ v} x_i = 0 \wedge \sum_{j \in \delta^- v} x_j = 0)$ for a vertex v, the proper width at the i-th level is made to be at most 4 when v is in the i-th elimination front, and 1 otherwise.

proof: It is easy to find an OBDD of the function $\sum_{i \in \delta^{\pm} v} x_i = 1$ such that its proper width is at most 2 when v is in the elimination front, and 1 otherwise. For $(\sum_{i \in \delta^+ v} x_i = 0 \wedge \sum_{j \in \delta^- v} x_j = 0)$, an OBDD of proper width 1 exists. Then, using Lemma 4 with some detailed analysis, the proper width is bounded by $2 \times 2 = 4$ when v is in the i-th elimination front, and 1 otherwise. □

For $\sum_{k \in \delta^- s} x_k = 1$ and $\sum_{l \in \delta^+ t} x_l = 1$, similar results holds by replacing 4 by 2 above. Then, by the definition of the elimination front, and again by Lemma 4, we obtain the following (we here assume the elimination front size is $\Omega(\log n)$).

Lemma 8: For the graph G with n vertices and an edge ordering whose maximum elimination front consists of at most l vertices, there is an OBDD of $flow_{s \to t}(\boldsymbol{x})$ whose width is at most 4^l. Such an OBDD can be constructed in $O(2^{O(l)})$ time.

In the undirected case, a similar lemma holds for $flow_{s-t}$ with replacing 4 in this lemma by 3.

3.3 Counting the Number of Paths

The OBDDs representing the flow condition can thus be computed as above. However, from this OBDD, the number of paths cannot be counted directly, since a flow of value 1 does not necessarily correspond to a simple path. In fact, a simple path between two terminal vertices s and t in an undirected graph is an undirected flow of value 1 without any cycles. Same for the directed case. Hence, we have to remove flows having circular flows of value 1.

Let $tree(\boldsymbol{x})$, $forest(\boldsymbol{x})$ be Boolean functions representing all the spanning trees and all the forests, respectively. That is,

$$tree(\boldsymbol{x}) = \begin{cases} 1 & \text{edges } e_i \text{ with } x_i = 1 \\ & \quad \text{form a spanning tree} \\ 0 & \text{othereise} \end{cases}$$

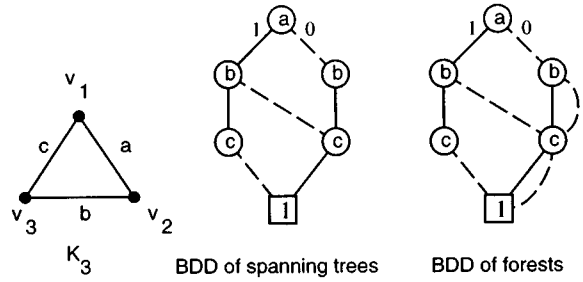


Fig. 4 OBDDs of forests and trees of K_3.

$tree(\boldsymbol{x})$ has already been treated in Sect. 2, and an algorithm constructing a certain type of BDD of $tree(\boldsymbol{x})$ has been given. $forest(\boldsymbol{x})$ becomes 1 when edges with $x_i = 1$ does not contain any cycle. Then, from the above observations, the following hold.

Lemma 9: (a) $path_{s-t}(\boldsymbol{x}) \equiv flow_{s-t}(\boldsymbol{x}) \wedge forest(\boldsymbol{x})$ becomes 1 when edges with $x_i = 1$ form a simple path between s and t in the undirected case.

(b) $path_{s\to t}(\boldsymbol{x}) \equiv flow_{s\to t}(\boldsymbol{x}) \wedge forest(\boldsymbol{x})$ becomes 1 when edges with $x_i = 1$ form a simple path from s to t in the directed case.

Concerning the OBDD of forests, from the OBDD of trees, it can be easily obtained.

Lemma 10: The OBDD of $forest(\boldsymbol{x})$ can be obtained from that of $tree(\boldsymbol{x})$ simply by replacing each 1-edge in the BDD corresponding to a coloop by both 1-edge and 0-edge.

proof: Before going into discussions of the proof, we first show an example. The OBDDs of forests and trees of a complete graph K_3 of three vertices are drawn in Fig. 4. In this example, in a minor obtained by deleting edge a, edge b is a coloop, and, in a minor obtained by deleting edge a and contracting edge b, edge c is a coloop. Correspondingly, the OBDD of forests can be obtained from that of trees by the operation described in this lemma.

This lemma basically holds for the BDD representing all bases and all independent sets of a matroid. As is well known, for a matroid, the greedy algorithm works. We here show this lemma based on this property.

For each forest, consider a lexicographically maximum spanning tree, with respect to the edge ordering, of the minor obtained by contracting edges in the forests. Then, edges in the forests and edges of the computed tree of the minor form a spanning tree of the whole graph. Since the computed tree of the minor is uniquely determined, this spanning tree is uniquely associated with the forest.

Then, it is easy to see that, along the path in the OBDD representing this spanning tree, the forest is uniquely represented by the operations in this lemma. □

It should be noted that, without the matroidal structure, this lemma does not necessarily hold. Then, applying Theorem 2 and 3, we obtain the following theorem.

Theorem 4: (a) For the $O(n^\alpha)$-separable graph G with n vertices, there is an OBDD of $path_{s-t}(\boldsymbol{x})$ and that of $path_{s\to t}(\boldsymbol{x})$ whose width is at most $O(n^{O(n^\alpha)})$. Such OBDDs can be constructed in $O(n^{O(n^\alpha)})$ time.

(b) For planar graphs with bounded degree, $O(n^{O(n^\alpha)})$ above can be replaced with $O(2^{O(\sqrt{n})})$.

Having an OBDD representing all simple paths, it is easy to count the number of paths, say by a similar algorithm for the reliability, in time proportional to the OBDD size. Hence, we obtain the following.

Theorem 5: (a) The number of paths between two terminals for the class of $O(n^\alpha)$-separable graphs can be computed in $O(n^{O(n^\alpha)})$ time in both undirected and directed cases.

(b) For a simple planar graph with n vertices and bounded degree, the number of paths between designated two terminals can be computed in $O(2^{O(\sqrt{n})})$ time.

4. Concluding Remarks

By further applying the conventional BDD algorithms, including one computing the BDD representing prime implicants from the BDD of a given Boolean function [7], to the BDDs considered in this paper, we can compute the following.

- the directed network reliability (from the root to other vertices, or from the source to the sink)
- generalized directed reliability guaranteeing high connectivity (from the root to other vertices there are k directed paths, etc.)
- counting the number of circuits and that of cutsets of a graph, or implicitly representing circuits and cutsets by BDDs (this may be done in several ways; for example, by first constructing the BDD of circulation flows and then applying the prime implicant algorithm, or by first constructing the BDD of the negation of $forest(\boldsymbol{x})$ and then applying the prime implicant algorithm (cf. [7], [10])).

However, in these cases, the number of Boolean operations performed among BDDs becomes proportional to the number of variables, and then the current theory just tells us a trivial exponential worst-case bound. This kind of situations are almost always the case with BDD, but even with such theoretical background it worked well. Our results in this paper show that as far as the numbers of paths are concerned this theoretical difficulty is overcome by applying the output-size sensitive top-down algorithm for the BDD of spanning trees. Performing a theoretically meaningful analysis for such generalized cases would be a challenge.

From the viewpoint of combinatorial structures, for

the above highly-connected version, matroid theory provides us a good framework such as matroid union and intersection. Furthermore, the directed network reliability from the root to other vertices has connection with greedoids and their Tutte polynomial (e.g., see [4], [11]). Since the Boolean functions can represent a set system as used in this paper, it would be very interesting to bridge the theory of BDD for Boolean functions and such combinatorial structures.

Acknowledgment

The authors would like to express our sincere thanks to the referees for giving many useful comments to the original version of this paper. Part of this work of the second author was supported by the Grant-in-Aid of the Ministry of Education, Science and Culture of Japan.

References

[1] N. Alon, P. Seymour, and R. Thomas, "A separator theorem for graphs with an excluded minor and its applications," Proc. of the 22nd Annual ACM Symp. on Theory of Computing, pp.293–299, 1990.
[2] E.T. Bax, "Algorithms to count paths and cycles," Infor. Process. Lett., vol.52, pp.249–252, 1994.
[3] E. Biondi, L. Divieti, and G. Guardabassi, "Counting paths, circuits, chains, and cycles in graphs: a unified approach," Canad. J. Math., vol.22, pp.22–35, 1970.
[4] A. Björner and G.M. Ziegler, "Introduction to Greedoids," in Matroid Applications, ed. N. White, Encyclopedia of Mathematics and Its Applications, vol.26, pp.284–357, Cambridge University Press, 1992.
[5] R.E. Bryant, "Graph-based algorithms for Boolean function manipulation," IEEE Trans. Comput., vol.C-35, pp.677–691, 1986.
[6] C.J. Colbourn, "The Combinatorics of Network Reliability," Oxford University Press, 1987.
[7] O. Coudert and J. Madre, "Implicit and incremental computation of primes and essential primes of Boolean functions," Proc. 29th ACM/IEEE DAC, pp.36–39, 1992.
[8] D. Eppstein, "Finding the k shortest paths," Proc. of the 25th Annual IEEE Symp. on Foundations of Computer Science, pp.154–165, 1994.
[9] D.D. Harms, M. Kraetzl, C.J. Colbourn, and J.S. Devitt, "Network Reliability: Experiments with a Symbolic Algebra Environment," CRC Press, Inc., 1995.
[10] K. Hayase and H. Imai, "OBDDs of a monotone function and of its prime implicants," ISAAC'96, Lecture Notes in Computer Science, vol.1178, pp.136–145, Springer-Verlag, 1996.
[11] B. Korte, L. Lovász, and R. Schrader, "Greedoids," Algorithms and Combinatorics, vol.4, Springer-Verlag, 1991.
[12] R.J. Lipton, D.J. Rose, and R.E. Tarjan, "Generalized nested dissection," SIAM J. Numer. Anal., vol.16, no.2, pp.346–358, 1979.
[13] R.J. Lipton and R.E. Tarjan, "A separator theorem for planar graphs," SIAM J. on Appl. Math., vol.36, no.2, pp.177–189, 1979.
[14] L. Lovász, "Combinatorial Problems and Exercises," North-Holland, Amsterdam-New York, 1979; 2nd edition, 1993.
[15] J.S. Provan, "The complexity of reliability computations in planar and acyclic graphs," SIAM J. Comput., vol.15, no.3, pp.694–702, 1986.
[16] K. Sekine, H. Imai, and S. Tani, "Computing the Tutte polynomial of a graph of moderate size," ISAAC'95, Lecture Notes in Computer Science, vol.1004, pp.224–233, Springer-Verlag, 1995.
[17] S. Tani, "An Extended Framework of Ordered Binary Decision Diagrams for Combinatorial Graph Problems," Master's Thesis, University of Tokyo, 1995.
[18] S. Tani and H. Imai, "A reordering operation for an ordered binary decision diagram and an extended framework for combinatorics of graphs," ISAAC'94, Lecture Notes in Computer Science, vol.834, pp.575–583, Springer-Verlag, 1994.
[19] L.G. Valiant, "The complexity of enumeration and reliability problems," SIAM J. Comput., vol.8, no.3, pp.410–421, 1979.
[20] D.J.A. Welsh, "Complexity: Knots, Colourings and Counting," London Mathematical Society Lecture Note Series, vol.186, Cambridge University Press, 1993.

Kyoko Sekine obtained M.Sc. in Mathematical Sciences, University of Oxford in 1993. Since then, she has been a doctoral candidate at Department of Information Science, University of Tokyo. Her research interests include graph theory, especially flow polynomial, Tutte polynomial and their variant. She is a member of OR Soc. Japan.

Hiroshi Imai obtained B.Eng. in Mathematical Engineering, and M.Eng. and D.Eng. in Information Engineering, University of Tokyo in 1981, 1983 and 1986, respectively. In 1986~1990, He was an associate professor of Department of Computer Science and Communication Engineering, Kyushu University. Since 1990, he has been an associate professor at Department of Information Science, University of Tokyo. His research interests include algorithms, computational geometry, and optimization. He is a member of IPSJ, OR Soc. Japan, ACM and IEEE.