分布推定に基づく確率的関数進化アルゴリズム

Probabilistic Function Evolution based on Estimation of Distribution

長谷川禎彦
Yoshihiko Hasegawa

指導教員：伊庭斉志教授

2007年12月14日

目 次

図目次

表目次

アルゴリズム一覧

概要

本論文は分布推定に基づく確率的関数進化アルゴリズムと題し，五章からなり，木構造表現を扱う最適化アルゴリズムを提案し，計算機実験結果に基づき提案手法の有用性を検証している．

第一章では，本論文の研究目的について説明し，分布推定アルゴリズムが提案された背景と，木構造分布推定アルゴリズムの研究を本論文で行う理由について述べる．また，本論文の目的についても記述する．

第二章では，分布推定アルゴリズムの基本事項について述べる．本論文では，進化アルゴリズムは未知の優良解分布からのサンプリングを繰り返す手法であると考えており，遺伝的アルゴリズムと分布推定アルゴリズムの違いはサンプリング方法にあると捉えている．この章では，従来の遺伝的アルゴリズムや遺伝的プログラミングと，分布推定アルゴリズムの相違点及び共通点について述べ，一般的な分布推定アルゴリズムの枠組みについて説明する．さらに，固定長一次元配列を用いる分布推定アルゴリズム，及び木構造を用いる分布推定アルゴリズムについての説明を中心に行う．木構造分布推定アルゴリズムは大きく分類すると，プロトタイプ木に基づく手法と，確率文脈自由文法に基づく手法に大分されるが，これらについても説明する．また，分布推定アルゴリズムの従来手法についても簡単に紹介する．

第三章では，提案手法 POLE（Program Optimization with Linkage Estimation）について述べる．POLE はプロトタイプ木に基づく手法であるが，従来手法においては確率モデルとして，単変数モデルや親子関係モデルなど，あらかじめ固定された依存関係のみを考慮している．また，従来のプロトタイプ木に基づく木構造分布推定アルゴリズムは，木構造を単純に固定長配列に変換しているが，単純な変換ではシンボル数の多さの問題やノイズの問題を避けて通ることが出来ない．POLE は拡張構文木と呼ばれる表現手法と，ベイジアンネットワーク学習を組み合わせた手法であり，上記の問題点を解決している．POLE は三つのベンチマークテストに適用されており，有効性が示されている．特に，騙し最大値問題や Royal Tree 問題では，従来手法を大きく上回る探索性能を示している．この章では，POLE と関連するベイジアンネットワークについても，パラメータ学習や構造学習について簡単に説明している．

第四章では二つ目の提案手法である PAGE（Programming with Annotated Grammar Estimation）の詳細が記されている．確率文脈自由文法に基づく木構造分布推定アルゴリズムは，今までに数多く提案されているが，確率文脈自由文法は文脈自由を仮定しているため，ノード間の依存関係を一切考慮することが出来ない．従来手法では，アノテーションを付加することで，文脈自由仮定を緩和しているが，親ノードや深さなどのヒュー

リスティクスに基づくアノテーションが用いられてきた．このようなアノテーションが有効である問題も存在するが，どのようなアノテーションモデルが有効であるかは事前には分らないことが多い．PAGE では，自然言語処理で提案された PCFG-LA（PCFG with Latent Annotations）と呼ばれる確率文脈自由文法モデルを用いている．PCFG-LA ではアノテーションを隠れ変数として扱っており，ヒューリスティクスに基づくアノテーションモデルと比較して柔軟であると考えられる．PAGE においては，PCFG-LA のモデル学習に EM アルゴリズムと変分ベイズ法を用いている．この章では，これら関連手法についても説明を行っている．PAGE は大きく分けると二つの手法 PAGE-EM（EM アルゴリズム），PAGE-VB（変分ベイズ法）に分けられるが，それぞれのモデル学習アルゴリズムについて詳細に説明している．二つの PAGE の有効性を，いくつかのベンチマークテストに適用することで示している．

第五章では，本論文で得られた結果についてまとめ，将来研究の方向性について議論する．

第1章　序論

1.1　研究の目的

最適化問題は，制約条件のもとで目的とする関数の最大値及び最小値を探索する問題である．物理学などにおいても，エネルギー最小化問題として同様の定式化がなされている．最適化問題の解法は，数学的なアプローチや，物理学的アプローチ，人工知能的アプローチなど多種多様である．本論文が対象とする最適化問題は主に情報学の観点から見たものであるが，情報学としての最適化アルゴリズムは，情報科学において盛んに研究されている分野の一つである．最適化アルゴリズムとしてもっとも用いられるのは，決定的最適化である．決定的最適化手法は，山登り法に代表される乱数を用いない最適化手法である．しかし，適用対象となる問題に関する情報が乏しい場合や，多くの局所解を有する場合，決定的である山登り法などでは解くことが出来ない（満足の行く解が得られない）場合も多い．このような問題の解決法として考えられるのが乱数を用いる確率的（stochastic）最適化手法である．この種のアルゴリズムとしては，シミューレーティッドアニーリング（SA：Simulated Annealing），マルコフ連鎖モンテカルロ（MCMC：Markov Chain Monte Carlo），進化アルゴリズム（EA：Evolutionary Algorithm）が挙げられる．SA や MCMC は確率分布（ボルツマン分布）に従って改悪をある程度許しつつ探索を行うアルゴリズムである．一方，EA は突然変異や交叉と言った，生物進化の考え方に基づいた手法であり，その根底にある思想は異なる．しかし，一方で，EA は優良解の分布を推定し，推定分布からのサンプリングを繰り返し行うという側面から見ることが出来る．すなわち，突然変異や交叉はあくまで未知の分布からサンプリングするための道具であり，EA は根本的には確率分布からのサンプリング手法であるとする考え方である．このように，優良解の分布からのサンプリングという視点から EA を構築し直した手法が分布推定アルゴリズム（EDA：Estimation of Distribution Algorithm）[1]である．GA では，未知の優良解分布からのサンプリングを突然変異などの遺伝的オペレータで行っていたのに対して，EDA では未知の分布をパラメトリックな分布と仮定し，モデル学習（モデル選択とパラメータ推定）と，推定分布からのサンプリングを繰り返す手法と捉えている．このように考えると，EDA は SA や MCMC などの手法に非常に近い．実際に，ボルツマン分布を用いる EDA（BEDA：Boltzmann EDA）[Mühlenbein 99b] は，平均場近似を用いる SA と非常に近いアルゴリズムである．しかし，EA が他の確率的探索手法と異なるのは，EA は探索能力としての進化のみならず，表現能力に関しても大きく進化してきた点にある．EA

[1]PMBGA（Probabilistic Model Building GA）[Pelikan 02] や IDEA（Iterated Density Estimation Evolutionary Algorithm）[Bosman 99] と呼ばれることもある．

はもともと，GA などのバイナリ固定長一次元配列を用いた組み合わせ最適化手法であったが，次第に木構造などのグラフ表現を扱うことが出来るように発展している．木構造を用いる EA は遺伝的プログラミング（GP：Genetic Programming）と呼ばれるが，その表現能力は非常に高い．固定長一次元配列が対象とする問題は，パラメータ調整などのタスクが主であるが，木構造を用いることで，関数，プログラムなどの構造的な表現を行うことが出来る．特に，木構造にメモリや再帰構造を付加することで，その表現能力はチューリング完全となることが知られている [Teller 94a, Teller 94b, Angeline 98, 矢吹 04]．このように，GP は EA に特徴的な手法であるが，通常の GP も GA のように，突然変異などの遺伝的オペレータに基づく探索手法をとっている．GA の固定長一次元配列における突然変異や交叉は，生物の遺伝子における突然変異や交叉のアナロジーであるが，木構造を用いる GP にとっての交叉や突然変異は生物のそれとはかけ離れており，生物の進化をもとにしているという考え方には GA の場合ほどの正当性はない．近年，この GP の扱う木構造に対しても，EDA の考え方が取り入れられつつある．すなわち，木構造に対してモデルを仮定し，モデル選択とパラメータ学習を行い，学習した分布より新しい木構造を生成する手法である．このような分野は，GP-EDA（Genetic Programming - EDA）や PMBGP（Probabilistic Model Building GP）[2]などと呼ばれる．前述のように，GP のオペレータは「遺伝的」と呼ぶには少し無理があり，寧ろ優良解の未知の分布からのサンプリングと考えた EDA 的な見方の方が自然であるとも考えられる．このような背景がありながらも，木構造に対する EDA の研究が本格化してきたのは比較的最近の話である．EDA ではどのようなモデルで確率分布を捉えるかを仮定する必要があるが，GA のように固定長であり位置依存でよいシンボルに対して確率分布を推定する場合，ベイジアンネットワークやマルコフランダム場などのモデルが非常によくマッチする．一方で，GP のように木構造の場合には，モデルの選定が難しいことが，最近まであまり研究が行われなかった理由に挙げられる．GP-EDA の研究は，現在までに大きく二つの種類に分類される（2.4.2 節）．

- プロトタイプ木モデル（2.4.2.1 節）
- 確率文脈自由文法モデル（2.4.2.6 節）

前者の考え方は，木構造を固定長配列に変換し，変換した配列に対して，GA-EDA で用いられる手法を用いたものである．一方，後者は，GP の木構造を導出木で表現し，確率文脈自由文法（PCFG：Probabilistic Context Free Grammar）の手法を用いたものである．これらの二つの手法は，木構造に対しての確率分布の適用を可能にする．

プロトタイプ木モデルの問題点は，文法的正しさの要請とシンボル数増加からくる依存関係推定の難しさである．バイナリ GA の場合，長さ n の 0 と 1 からなる配列の集合 $\{0,1\}^n$ に属する個体は全て評価可能である（$00\cdots000$ から $11\cdots111$ まで）．一方 GP の場合，木構造が関数として意味を持たなければならないという制約がある．例えば図 1.1 に文法的に正しい関数（a）と正しくない関数（b）の例を示す．図 1.1（b）の木構造で用

[2]本論文では，配列型の EDA を GA-EDA と記述する（一般的な用法ではない）．単に EDA と記述している場合は，GA-EDA と GP-EDA の両方を含む．

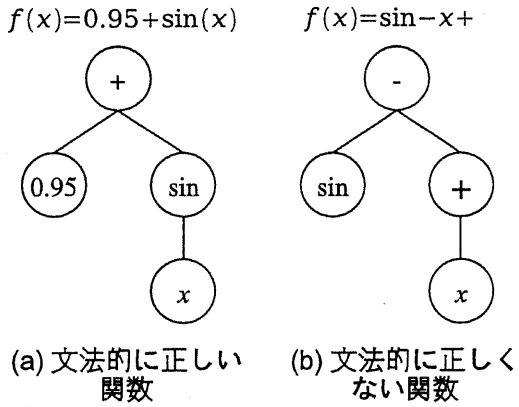


図 1.1: 文法的に正しい木構造（a）と正しくない木構造（b）.

いられるノードは正しいが，それらの組み合わせである木構造として見た場合は評価できない個体である．確率分布に従って個体を生成する場合，必ず文法的に正しくなるようにする工夫が必要である．また，GP ではシンボル数が GA の場合より多い．バイナリ GA では 2 種類しかシンボルを用いないが，GP の場合は $\mathfrak{F} = \{+, \times, \div, -\}, \mathfrak{T} = \{x, y, \Re\}$ など多くのシンボルを用いる．シンボル数の多いモデルは複雑度も増すため，パラメータ学習により多くの学習例を必要とする．必要学習例の増加は，EA においては個体数の増加につながり，適合度評価回数の増加をもたらす．本論文で提案する手法：POLE（Program Optimization with Linkage Estimation）は，これらの問題を解決した手法である（3 章）．POLE はベイジアンネットワークによってノード間の依存関係を推定する．POLE では，従来のプロトタイプ木の欠点を克服するために，拡張構文木（Expanded Parse Tree，3.4 節）と呼ばれる表現手法を用いる．拡張構文木を用いることで，シンボル数や文法的正しさの保証を行うことが可能である．

PCFG を用いる手法に関しては，PCFG モデルの選択の難しさがある．基本的な PCFG では，その名の通り文脈自由を仮定しており，文脈依存性を一切考慮することが出来ない．これは，GP-EDA の観点から見れば，ノード間の依存関係を一切考慮出来ないことと等しく，原理的に部分構造の推定が出来ないことを意味する．自然言語処理（NLP：Natural Language Processing）の分野では，アノテーションなどの付加的な情報が非終端記号に付加される PCFG が提案されている．アノテーションは，PCFG の文脈自由という制約の中で，限定した形で文脈依存性を付加するアイディアである．GP-EDA においても，深さなどの「文脈」がアノテーションとして用いられているが，このようなアノテーションでは，非常に限定した問題に対してしか適用することが出来ない．提案手法：PAGE では，NLP で提案された PCFG-LA（PCFG with Latent Annotations）を基本文法として用いている．PCFG-LA ではアノテーションは潜在変数としており，潜在変数を含む生成規則は EM アルゴリズムで学習される．PAGE では，PCFG-LA と同様に EM アルゴリズムによるパラメータ学習手法を用いるとともに，変分ベイズ法による学習アルゴリズムも用いた．PCFG-LA を基本文法として用いることで，PAGE は位置に依存しない部分

構造の推定が可能である.

1.2 本研究の目標

EDA の最大の利点は，モデルの仮定によって様々な EA を構築出来ることにある．そのため，モデル如何では，GA や GP に非常に近いアルゴリズムを構築することが出来る一方で，GA や GP では解くことの出来ない問題を解くことが出来る．進化アルゴリズムの利点は，問題に対する前知識をほとんど用いずに最適化を行うことが出来るという点にある．GA-EDA の研究では，GA が解くことの難しい問題として，騙し問題の例が挙げられている．この論文で提案する POLE 及び PAGE においても，GP で解きにくい問題：GP-Hard な問題を解くことを焦点とする．EDA では，仮定したモデル集合の下で最大事後確率を与えるモデルと，パラメータを推定する．そのため，どのようなモデルを仮定するかは非常に重要である．本論文では，有効性を示すベンチマークテストとして，GP-Hard である状況を理想化した問題を用いる．ここの理想化は以下の意味である．

- イントロン[3]を全く含まない
- 最適構造は一つであり，最適構造から少しでも構造が異なると適合度は最適解ではない

このような条件を満たす問題を考案し，有効性の検証に用いた．このような理想化した問題を用いる利点は，以下の点である．

- 手法の有効性が検証しやすい
- 手法の振る舞いの解析が行いやすい

イントロンの影響を考慮する場合，イントロン部分が分布推定に影響を及ぼす一方で，適合度には影響を及ぼさない．そのため，優良解の近似としての分布という仮定が成り立たない．新しいアルゴリズムを提案する際，最も重要なのは，なぜそのアルゴリズムが有効なのかを示すことにある．このような理想化した問題においては，有効性の検証が行いやすい．イントロンがある場合に置いても，ある操作を加えることで，イントロンを含まない構造への変換が可能である（4.11 章参照）．そのため，このような仮定は将来の実問題への適用を考えた場合に，強すぎる仮定ではないと考えられる．

[3]適合度計算に影響を及ぼさない部分構造である.

第2章　分布推定アルゴリズム

2.1　進化アルゴリズム

進化アルゴリズム（EA：Evolutionary Algorithm）は生物の進化の考え方を工学的に応用した手法である．EAは，生物の環境への適用プロセスを学習と捉えている．生物は長い年月をかけて進化してきたが，進化の過程で環境へ適用していない種は絶滅し，適用している種は繁栄するということを繰り返してきた．これを工学の問題に置き換えると，「環境」は「問題」及び「学習例」に相当し，「適応度」は「解の良さ」に相当する．EAは，進化のメカニズムの本質を工学的に応用出来れば，最適化問題を進化的に解くことが出来るのではないか，という考え方に基づく．EAと一口にいっても，EAには多種多様なアルゴリズムが存在している．本論文では，戦略としての違いではなく，表現の違いに重点をおき，EAを大きく二つの手法に分類する：遺伝的アルゴリズム（GA：Genetic Algorithm）と遺伝的プログラミング（GP：Genetic Programming）である．

GAはHolland[Holland 75]によってはじめに提案された手法である．GAはEAの最も重要な手法であり，最も多くの応用がなされている．GAの最大の特徴は，固定長一次元配列を持つことにある．GAが一次元配列を用いるのは，工学的な要請というよりは，生物の遺伝子のアナロジーである面が大きい．GAは生物の進化のメカニズムの中で，選択，突然変異，交叉を主要な要素と捉えている．GAは固定長一次元配列に対して，選択，突然変異，交叉を行うことで解の進化を行う．

選択は，集団から優良な個体を選び出す操作である．自然界では，環境により適応したものが生き延びやすい．選択の操作はこれを人工的に作り出すものである．突然変異は，生物における遺伝子の突然変異を模倣している．遺伝子は化学的，及び放射線などによって，ヌクレオチドが変異する．GAでは，遺伝子座にある値が確率的に書き換わることでこれを模倣している．交叉は有性生殖の模倣であり，GAでは二つの遺伝子が組変わることで行われる．突然変異と交叉を図2.1に図示する．

GPは，GAの染色体構造を木構造に拡張したものである．GPをはじめに提案したのは，Cramer[Cramer 85]やKoza[Koza 89]による．GAのような固定長一次元配列を用いた場合，それぞれの値が位置に特有の意味をもつため，部分構造そのものに意味を持たせることが出来ない（構造的な表現が行えない）．そのため，GAの適用される問題は主にパラメータ探索が中心である．関数やプログラムなどの高い表現を行うためには，固定長一次元配列では限界がある．GPでは木構造を表現として用いるが，LISP言語などに代表される関数型言語のプログラム（S式）が木構造で表現されることに由来する．GPはLISP言語のS式を表現することが可能であるが，関数型言語のプログラムを表現することの出来るGPは

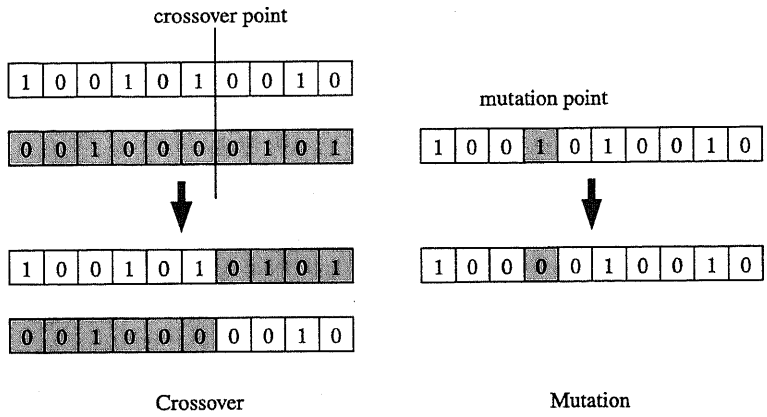


図 2.1: GA の交叉と突然変異.

Algorithm 1 General GA and GP.

1. $g \leftarrow 0$
   $\mathcal{P}_g \leftarrow$Generate M random individuals.
2. $\mathcal{D}_g \leftarrow$Select $N \leq M$ promising individuals with the selection method
3. $\mathcal{P}_{g+1} \leftarrow$Apply mutation and crossover to $\mathcal{D}_g$ to generate new population
   $g \leftarrow g+1$

表現能力が非常に高い．特にメモリの読み書きを加えたり，再帰を付加することで，GP はチューリング完全となることが示されている [Teller 94a, Teller 94b, Angeline 98, 矢吹 04]. さらに，木構造より一般のグラフ構造を用いる GP も提案されている [Teller 95, Teller 96]. GP はその表現能力の高さから，適用可能範囲は GA より広い．近年，GP の工学的な応用が，金融工学，ロボティクス，バイオインフォマティクス，など多様な分野で適用されている．特許の再発見や，特許より優れたものが GP で獲得されたという報告がなされるほどである [伊庭 01].

高いレベルから見た GP のアルゴリズムは GA と同じである（アルゴリズム 1）．GP と GA の違いはあくまで染色体の違いのみである．GP のオペレータは，GA のオペレータと同様に突然変異と交叉が用いられる．しかし，木構造を用いることから，それぞれのオペレータの働きは大きく異なる．木構造における突然変異は，部分木を入れ替える手法や部分ノードを変化させる方法などがある．木構造における交叉は部分木の入れ替えである．GA の場合，各遺伝子座の値に意味付けがなされており，交叉による解の組み合わせは行いやすい．しかし，GP の部分木は，それが出現する位置によって，その木構造の意味も変わってしまうため，交叉の効果は GA の場合より複雑である．

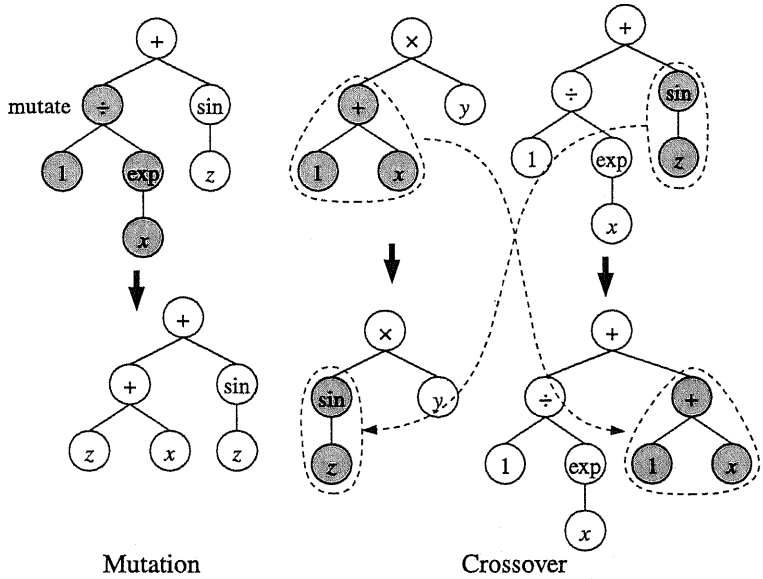


図 2.2: GP における交叉（crossover）と突然変異（mutation）.

Algorithm 2 General EA.

1. $g \leftarrow 0$
   $\mathcal{P}_g \leftarrow$Generate M random individuals.
2. $\mathcal{D}_g \leftarrow$Select $N \leq M$ promising individuals with the selection method
3. $\mathcal{P}_{g+1} \leftarrow$Generate new individuals from a distribution of $\mathcal{D}_g$
   $g \leftarrow g+1$

2.2 分布推定アルゴリズム

交叉や突然変異に基づく EA は，生物の進化を工学的にモデル化した手法であると同時に，確率的な探索手法でもある．GA や GP などの手法が突然変異や交叉と言った表現を用いているが，この操作自体，未知の分布からのサンプリングの近似手法であると考えれば，完全に SA や MCMC などの「分布からのサンプリングアルゴリズム」と捉えられる．

世代 g における集団を $\mathcal{P}_g$ で表し，優良個体の集合を $\mathcal{D}_g$ で表す（$\mathcal{D}$ は本論文では主にデータを表すが，EDA（Estimation of Distribution Algorithm）では優良個体の集合がデータとなることに由来する）．GA や GP では，世代 g における優良解集合 $\mathcal{D}_g$ の分布（分布は未知）から，世代 $g+1$ の集団 $\mathcal{P}_{g+1}$ サンプリングする手法と考えることが出来る．$\mathcal{Y}_g, \mathcal{X}_g$ を個体（$\mathcal{Y}_g \in \mathcal{P}_g$，$\mathcal{X}_g \in \mathcal{D}_g$），$X$ を個体とすると，優良解が与えられた時の分布，

$$\mathcal{Y}_{g+1} \sim P(X|\mathcal{D}_g)$$

に従って，サンプルを生成することに相当する．しかし，一般的に分布 $P(X|\mathcal{D}_g)$ の形は未知であるため，何らかの近似手法をとることで，分布からのサンプリングを行う必要がある．揺らぎ（ランダム性）を持ちつつサンプリングすることで分布を近似する方法を取るのがGAやGPであり，モデルを仮定し，学習したパラメトリックなモデルからサンプリングするのがEDAである[1]．そのため，GAやGP，EDAもそれぞれ優良解からのサンプリングアルゴリズムであるが，分布の近似の仕方が異なるアルゴリズムであると考えられる．まとめると，以下のように表すことが出来る．

- GA，GPの場合（図2.3（A））
  突然変異などのランダム性によって近似的にサンプリングを行う．

$$\mathcal{Y}_{g+1} = \mathcal{X}_g + randomness$$

- EDAの場合（図2.3（B））
  モデル $\mathcal{M}$ を仮定し，$\mathcal{D}_g$ が与えられた時の予測事後分布（predictive posterior distribution）に従ってサンプリングする（2.3.4節参照）．

$$\mathcal{Y}_{g+1} \sim \sum_{m \in \mathcal{M}} \int d\Theta P(X|m,\Theta) P(\Theta|\mathcal{D}_g, m) P(m|\mathcal{D}_g)$$

図2.3は，GA，GPとEDAの違いを図で表現したものである．両方のアルゴリズムでは，選択（Selection）を行い，優良解 $\mathcal{D}_g$ を選び出す．その優良解 $\mathcal{D}_g$ の分布に従う新しい個体群 $\mathcal{P}_{g+1}$ を生成する際に，GAやGPでは，その染色体構造の一部分を少し変えることで近似的にサンプリングを行う（図2.3（A））．この場合，一部分が変化することで，適合度が少し変わり，それによって分布から近似的にサンプリングすることが可能となる．一方，EDAでは，$\mathcal{D}_g$ をパラメトリックな分布として表現する．その上で，推定された分布からGibbs SamplingやPLS（Probabilistic Logic Sampling）などのサンプリング方法を用いて，$\mathcal{P}_{g+1}$ を生成する（図2.3（B））．

GA，GPの場合とEDAの場合のどちらの方が優れているかは場合による．しかし，ランダム性に基づくサンプリングの場合，正しくサンプリングされるには仮定が必要である．ここで，二つの個体 X, X' の距離を $distance(X, X') \in \mathbb{R}^+$ で表す．この距離は，突然変異などを施した場合，少ない適用回数で $X \to X'$ となる場合ほど0に近い関数で，GAの場合はハミング距離などに相当する．ランダム性に基づくサンプリングの場合，$distance(X, X') \simeq 0$ である二つの個体の適合度が $|f(X) - f(X')| \simeq 0$ という仮定が成り立たなければならない．図2.3（A）にあるように，ランダムネスを付加することで，それぞれの個体は，もとの状態の近くに遷移している．しかし，上の仮定が成り立たない場合，ランダムネスを付加した個体が離れた場所に遷移してしまい，$\mathcal{D}_g$ の分布からサンプリングを行うことが出来ない．GAやGPが良い探索性能を示す問題では，この条件がある程度満たされていると考えられる．例えば，GAで良く用いられるOneMax問題を考

[1] ここではパラメトリックと限定したが，ノンパラメトリックなEDAも存在する [筒井 03].

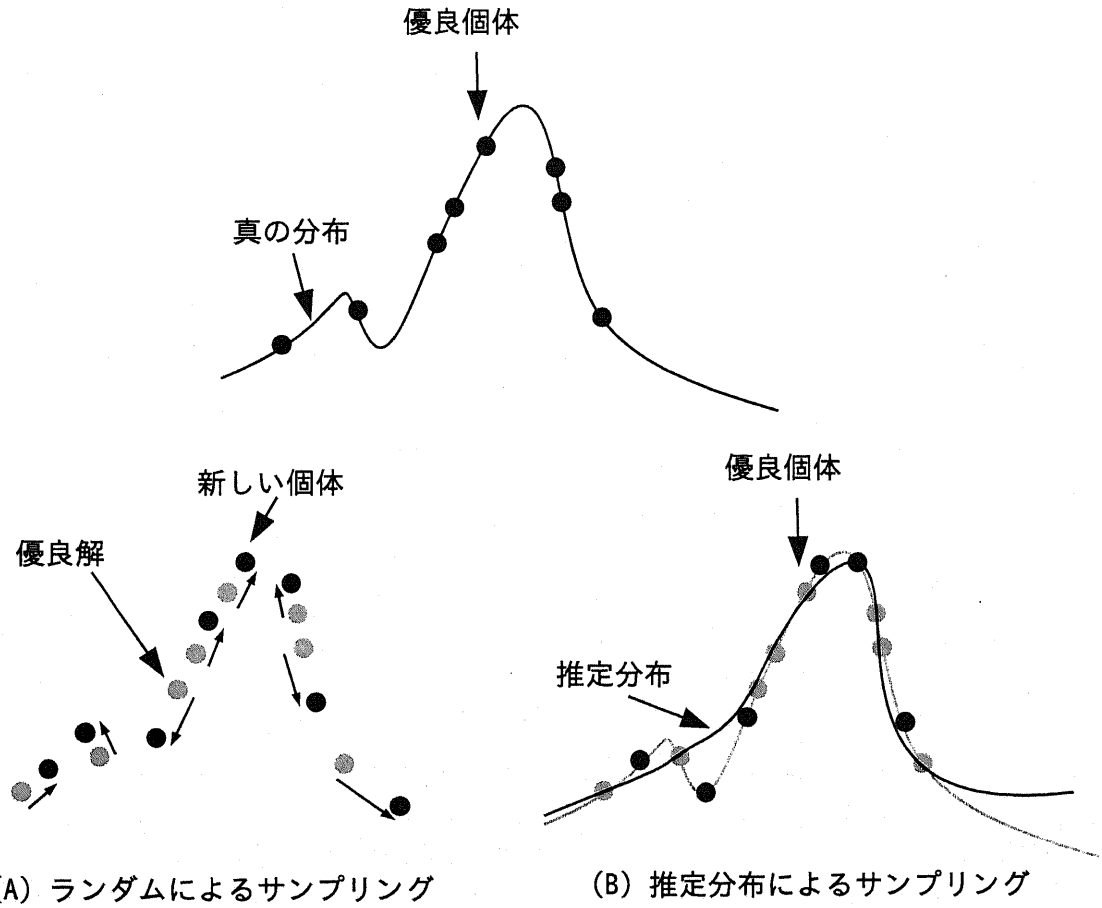


図 2.3: ランダム性によるサンプリングと，推定分布によるサンプリングのイメージ.

える．この問題の適合度は式2.1で表される問題であり，ビット列の中の1の数が適合度となる．

$$fitness(X) = \sum_{i=1}^{n} x_i \tag{2.1}$$

この問題では，ハミング距離を $distant(X, X')$ とすると，適合度の違い $|f(X) - f(X')|$ の間には線形関係が成り立つ．実際に，GAはOneMax問題を非常に早く解くことが出来る．しかし，OneMax問題のように単純な関係がある場合は山登り法においても解くことが可能である上，一般的な問題を考えた場合，このような仮定が常に成り立つとは限らない．このような状況が成り立たない例が，騙し問題である．Goldbergが提案した騙し問題（deceptive function of order k）の適合度関数は式2.2で表される[Goldberg 93]．この式で，s_j は X の k 個の要素からなる部分集合であり（$s_j \subset X$），$s_i \cap s_j = \emptyset (i \neq j)$ を満たす．s_j の個数を s としている．

$$fitness(X) = \sum_{j=1}^{s} f_{dec}^{k}(s_j) \tag{2.2}$$

f_{dec}^{k} は部分適合度関数であり，$k = 4$ の場合の適合度は以下のような値である（図2.4）．

$$f_{dec}^{4}(s_j) = \begin{cases} 5 & u = 0 \\ 1 & u = 1 \\ 2 & u = 2 \\ 3 & u = 3 \\ 4 & u = 4 \end{cases}$$

この問題の最適解は0000の組み合わせである．しかし，0010と0000のハミング距離は1なのに対して，適合度の差は5と，適合度の差は非常に大きい．そのため，この問題はGAにとっては難しい問題であると想像される．実際にこの問題は，問題サイズが大きくなると（s が大きい問題），通常のGAで解くことは難しくなる．一方で，依存関係を推定するEDAでは，容易に解くことが出来ると報告されている[Pelikan 99a]．

EDAはモデルを用意し，そのモデルの下で推定を行う．どのようなモデルでも用いることが出来る（GAのようなモデルも用いることが可能）ため，EDAはGA的な探索を行うことも可能である．そのため，モデルを適切に選びさえすれば，EDAは少なくともGA以上の性能が得られることになる．このような理由から，EDAにおける研究の多くは，適切な確率モデルの探索に当てられている．

EDAで用いられるモデルはどのようなモデルでも構わないが，サンプリングのコストが小さいモデルの方が好まれる．$X = \{x_1, x_2, \cdots, x_n\}$ とし，モデルを $P(x_1, x_2, \cdots, x_n)$ のようにすべてが完全に依存しあった同時確率で表される場合を考える．単純にすべての場合の確率を考慮に入れてサンプリングする場合，場合の数が最も少ないバイナリの場合でも 2^n の計算量が必要となる．通常，$n \sim 100$ のオーダーであるので，実用上不可能である．また，Gibbs Samplingを用いた場合も，定常分布となるまで多くの繰り

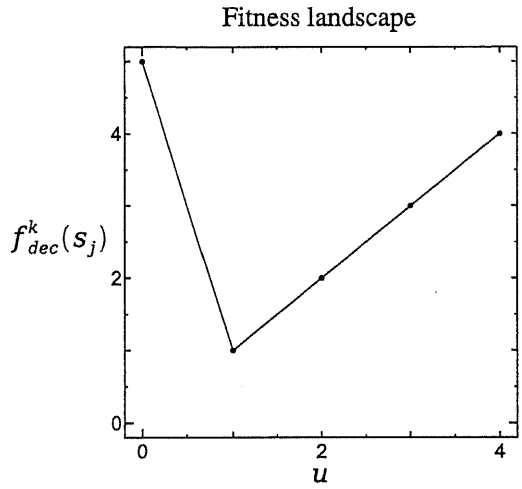


図 2.4: 騙し問題の適合度地形.

Algorithm 3 General EDA.

1. $g \leftarrow 0$
   $\mathcal{P}_g \leftarrow$Generate M random individuals by $P(X|\Theta_0, m_0)$
2. $\mathcal{D}_g \leftarrow$Select $N \leq M$ promising individuals with the selection method
3. $\mathcal{P}_{g+1} \leftarrow$Generate new individuals by sampling from a predictive posterior distribution
   $$P(X|\mathcal{D}_g) = \sum_{m \in \mathcal{M}} \int d\Theta P(X|\Theta, m) P(\Theta, m|\mathcal{D}_g)$$
   $g \leftarrow g + 1$

返しが必要であり，間隔を開けてサンプリングする必要があるため，非常に計算量が多い．一方，BOA（Bayesian Optimization Algorithm）やEBNA（Estimation of Bayesian Network Algorithm）などで用いられるベイジアンネットワークはPLS（Probabilistic Logic Sampling，3.2.4節参照）を用いることが出来るので，サンプリングのコストは非常に小さく，都合の良いモデルと言える．

2.3 アルゴリズム

ここで，EDAのアルゴリズムについて説明する．EDAのハイレベルのPseudoコードはアルゴリズム3に示している．一般的なEAのPseudoコードはアルゴリズム1に示す．EDAでは，サンプリング過程がEDAの場合モデルを用いていることが分かる．以下では，EDAの各操作について説明する．

2.3.1 初期集団の生成

EDAでは，最初に初期個体をM個生成する．生成方法はGAの場合と同様である．初期集団は分布

$$P(X|\Theta_0, m_0)$$

に従って生成される．ここで，Θ_0は初期パラメータセットであり，$\theta_0^i(k)$をi番目の変数がkの場合の確率を表すとすると，バイナリEDAの場合$\theta_0^i(0) = \theta_0^i(1) = 0.5$のように，一様に設定される．$m_0$は初期モデルであり，通常はもっとも基本的なモデルが割り当てられる（ベイジアンネットワークの場合，エッジが一つもないグラフ構造となる）．

2.3.2 優良個体の選択

優良個体が選択される．基本的には，EDAとしてはどのような選択方法も用いることができるが，通常は以下のような選択方法が用いられる．

- **Roulette Selection**
  ルーレット選択．適合度に比例する確率で，個体が選択される．
- **Truncate Selection**
  切り取り選択とも言う．個体を適合度でソートし，上位$P_s \times M$個の個体を選択する（P_sは選択率）．EDAではもっとも頻繁に用いられる選択である．
- **Tournament Selection**
  GA及びGPで広く用いられる選択である．ルーレット選択の場合，選択率は適合度の絶対的な値に左右される．しかし，適合度の上下関係には意味があるものの，絶対的な値にあまり意味のない問題では，ルーレット選択では淘汰圧がかかりすぎたり，かからなかったりする．トーナメント選択は，適合度の上下関係のみを用いるため，このような問題点が起きない．
- **Boltzmann Selection**
  ボルツマン選択は，ボルツマン分布に従って選択を行う．BEDA（Boltzmann EDA）などの，ボルツマン分布を用いるEDAで用いられる．

多くのEDAでは，Truncate Selectionが用いられる場合が多い．Truncate Selectionは一般的に淘汰圧が強いため，収束が早い．しかし，グラフ構造の学習を伴うEDAでは，ある程度の淘汰圧があった方が，グラフ構造を推定しやすい．

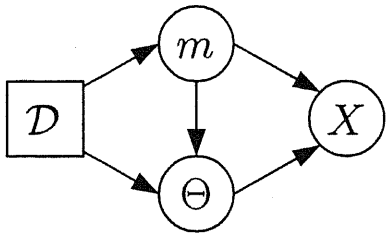


図 2.5: 予測事後分布のグラフィカルモデル.

2.3.3 パラメータ及びモデル学習

EDA では，優良個体からモデルの事後確率 $P(m|\mathcal{D}_g)$ を計算し，アンサンブルを用いて予測事後分布が計算される．モデルの事後分布は，式 2.3 で計算される．

$$P(m|\mathcal{D}_g) \propto P(m)P(\mathcal{D}_g|m) \tag{2.3}$$

パラメータも同様に，パラメータの事後分布が計算される（式 2.4）.

$$P(\Theta|\mathcal{D}_g, m) \propto P(\Theta)P(\mathcal{D}_g|\Theta, m) \tag{2.4}$$

ここで，$P(m)$ 及び $P(\Theta)$ は m と Θ の事前分布である．通常，極端な事前分布は用いない場合が多く，$P(m) \propto 1$，$P(\Theta) \propto 1$ のように一様分布が用いられる場合が多い．また，後述のように，モデルやパラメータに関しては MAP（Maximum a Posteriori）推定値や最尤推定値などの点推定値で近似される場合がほとんどである．

2.3.4 推定分布からの個体生成

アルゴリズム 3 にあるように，EDA は優良個体から次の世代の個体を生成する．個体の生成は予測事後分布（predictive posterior distribution）が用いられる．すなわち，データ $\mathcal{D}_g$ が与えられた時の，個体 X の確率，

$$\begin{aligned} P(X|\mathcal{D}_g) &= \sum_{m\in\mathcal{M}} \int d\Theta P(X|\Theta, m)P(\Theta, m|\mathcal{D}_g) \\ &= \sum_{m\in\mathcal{M}} \int d\Theta P(X|\Theta, m)P(\Theta|\mathcal{D}_g, m)P(m|\mathcal{D}_g) \end{aligned} \tag{2.5}$$

に従って，新しい個体がサンプリングされる．ここで，$\mathcal{M}$ はモデルの集合である（ベイジアンネットワークモデルを仮定した場合，$\mathcal{M}$ は全ての可能なグラフ構造に相当する）．しかし，ベイジアンネットワークの場合など，全ての可能なモデル（グラフ構造）に対して，和をとるのは現実的に不可能である．また，モデルが少ない場合でも，パラメータ空間で積分を行った予測事後分布は複雑な形をしており，PLS などの高速なサンプリング

方法を用いることが出来ない．そのため，一部の手法 [Gallagher 07] をのぞいて，MAP 推定値や最尤推定値などの点推定値による近似が行われることが多い．これを，式で表すと式2.6のように表される．

$$P(X|\mathcal{D}_g) = P(x|\widehat{\Theta}, \widehat{m}) \tag{2.6}$$

但し，ここでは$\widehat{\ }$はそれぞれ，MAP 値であり，以下のように定義される．

$$\widehat{m} = \underset{m}{\operatorname{argmax}}\, P(m|\mathcal{D}_g)$$

$$\widehat{\Theta} = \underset{\Theta}{\operatorname{argmax}}\, P(\Theta|\mathcal{D}_g, \widehat{m})$$

これは，式2.5において

$$P(\Theta|\mathcal{D}_g, m) = \delta(\Theta - \widehat{\Theta})$$

$$P(m|\mathcal{D}_g) = \delta_{m,\widehat{m}}$$

と，点推定で近似していることに等しい．

点推定値を用いた場合，一部のグラフィカルモデルの場合を除いて，PLS（3.2.4節参照）によるサンプリングが可能である．ただし，マルコフランダム場を用いる MN-EDA (Markov Network EDA) においては，PLS を用いることができないため，Gibbs Sampling によって新しい個体が生成される．

2.4 EDA の種々のモデル

本節では，現在までに提案されている EDA の中から，提案手法 POLE（Program Optimization with Linkage Estimation：3章参照）と PAGE（Programming with Annotated Grammar Estimation：4章参照）に関係の深い手法について紹介する．前述のように，EDA ではモデル $\mathcal{M}$ を適切に設定することが非常に重要となる．そのため，本節においても，その点に重点をおいて説明する．EDA は，GA のような固定長一次元配列を用いる手法と，GP のような木構造を用いる手法に大別される．本論文では前者を GA-EDA と記述し，後者を GP-EDA と記述する．単に EDA と記述した場合は，両者を指す．

2.4.1 固定長一次元配列型 EDA（GA モデル：GA-EDA）

EDA における研究は，GA のような固定長一次元配列を用いた手法から行われた．特に，バイナリ配列を用いた手法の研究が広く行われている．連続値に対する手法も存在するが，バイナリ配列の手法が多い理由は以下のような点による．

- バイナリは最も単純であり，モデル化が行いやすい．

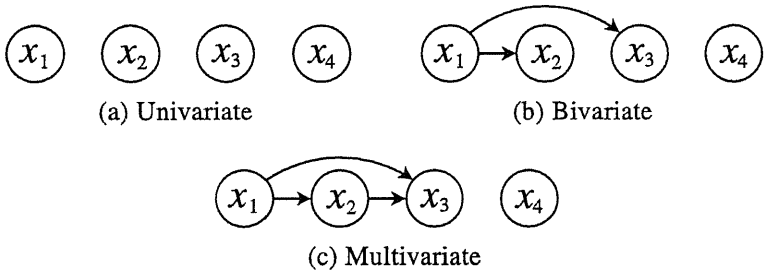


図 2.6: EDA のグラフィカルモデル.

- 多値のモデルは，一般性を失うことなくバイナリになる.
- 連続値の場合よりモデルが当てはまりやすい.

本節では，GA-EDA に関してはバイナリ EDA のみの説明を行う．バイナリ EDA はさらにいくつかのクラスに大別される：単変数モデル，二変数モデル，多変数モデルである．以下にその代表アルゴリズムを示す.

- 単変数モデル（Univariate EDA）
  UMDA（Univariate Marginal Distribution Algorithm）[Mühlenbein 96]，PBIL（Population Based Incremental Learning）[Baluja 94] など
- 二変数モデル（Bivariate EDA）
  MIMIC（Mutual Information Maximizing Input Clustering）[Bonet 96]，COMIT（Combining Optimizers with Mutual Information Trees）[Baluja 97]，BMDA（Bivariate Marginal Distribution Algorithm）[Pelikan 99b] など
- 多変数モデル（Multivariate EDA）
  EBNA（Estimation of Bayesian Network Algorithm）[Etxeberria 99]，BOA（Bayesian Optimization Algorithm）[Pelikan 99a]，LFDA（Learning Factorized Distribution Algorithm）[Mühlenbein 99a]，MN-EDA（Markov Network EDA）[Santana 05] など

ここでは単変数，二変数，多変数 EDA（Bayesian Network EDA）の説明を行う．特に Bayesian Network EDA は，本論文で提案している POLE（3 章参照）と関係がある.

2.4.1.1 単変数 EDA

単変数モデルとして，もっとも有名な手法は UMDA（Univariate Marginal Distribution Algorithm）である．UMDA は EDA としては非常に初期の手法である．UMDA では変数間の依存関係を一切考慮しない（図 2.6）．統計力学で用いられる Weiss の平均場近似と同じ考え方である．UMDA は単変数モデルであるため，最良モデル学習の必要がなく，

学習はパラメータ推定のみである．UMDA は実装がシンプルであり，計算コストが非常に小さいことから，非常に幅広く用いられている．UMDA では，単一のモデルのみを用いるため，事後分布は式 2.7 によって表される．

$$\begin{aligned} P(X|\mathcal{D}_g) &= P(X;\Theta_g) \\ &= \prod_{i=1}^{n} P(x_i;\theta_g^i) \end{aligned} \tag{2.7}$$

ここで θ_g^i は i 番目の変数のパラメータである（世代 g）．i 番目に置ける 1 の確率と 0 の確率を $\theta_g^i(1)$ と $\theta_g^i(0)$ によって表す．i 番目の遺伝子座における 1 の出現回数を $n_g^i(1)$，0 の出現回数を $n_g^i(0)$ とすると，パラメータは

$$\theta^i(1) = \frac{n_g^i(1)+\alpha}{N+2\alpha}, \theta^i(0) = \frac{n_g^i(0)+\alpha}{N+2\alpha}$$

のように推定される．α は事前分布のハイパーパラメータに由来する項であり，UMDA では突然変異のように機能する．

2.4.1.2 二変数 EDA

単変数の次に複雑なネットワークは，二変数モデルである．これは，ベイジアンネットワークにおいて，Singlely Connected な構造に相当する．一般的に，Singlely Conntected な構造は木構造で表される（図 2.6 (b)）．二変数モデルでは，直鎖構造（MIMIC），単一の木構造（COMIT），複数の木構造（BMDA）が提案されている．MIMIC と COMIT は基本的なアイディアは同じであり，KL ダイバージェンス（Kullback Leibler Divergence）を最小化する構造（直鎖：MIMIC，木構造：COMIT）を探す．これは，最尤構造を探すのと同じである．

MIMIC は以下のようにして，最尤直鎖構造を獲得する．1 から n までの順列を $\pi = \bigcup_{j=1}^{n}\{i_j\}$ で表す．順列 π で並んだ直鎖の確率分布を

$$P_\pi(X) = P(x_{i_1}|x_{i_2})P(x_{i_2}|x_{i_3})\cdots P(x_{i_{n-1}}|x_{i_n})P(x_{i_n}) \tag{2.8}$$

のように表す．MIMIC の目標は，$P(X)$ と $P_\pi(X)$ の KL ダイバージェンスを最小化するような，順列 π を求めることにある．二つの分布間の KL ダイバージェンスを計算すると，

$$\mathrm{KL}(P|P_\pi) = -H(P) + H(x_{i_1}|x_{i_2}) + H(x_{i_2}|x_{i_3}) + \cdots + H(x_{i_n})$$

となる．ここで，$H(\cdot)$ はエントロピー，$H(\cdot|\cdot)$ は条件付きエントロピーを表す．すなわち，KL ダイバージェンスの最小化は，

$$J_\pi(X) = H(x_{i_1}|x_{i_2}) + H(x_{i_2}|x_{i_3}) + \cdots + H(x_{i_n})$$

の最小化に等しい．真の最尤直鎖構造を求めるには $n!$ の場合すべてについて $J_\pi(X)$ を計算する必要がある．しかし，EA で扱う問題サイズを考えると（$n \sim 100$）これは計算不可能であるから，貪欲的に最尤直鎖構造を探索する．つまり，$H(x_{i_n})$ をまず最小化し，次に $H(x_{i_{n-1}}|x_{i_n})$ を最小化する．これを繰り返すことで，$O(n^2)$ で近似的に最尤直鎖構造が求まる．

2.4.1.3 ベイジアンネットワーク EDA

EBNA[Etxeberria 99]，BOA[Pelikan 99b]，LFDA[Mühlenbein 99a] は確率モデルとして，ベイジアンネットワーク（3.2 節参照）を用いた手法である．BMDA や MIMIC などが用いる，ツリー構造及び直鎖構造と比較して，リッチなモデルであるため，モデルとしては，より複雑な関係を捉えることが可能であると考えられる．また，後述するようにベイジアンネットワークからのサンプリングは容易であるため，高速に個体生成が行えることも大きな利点である．EBNA では，

$$P(X|\mathcal{D}_g) = \int d\Theta P(X|\Theta, \hat{m}) P(\Theta|\hat{m})$$

によって新しい個体が生成される．ここで，$\hat{m}$ は最良のベイジアンネットワークに相当する．

$$\hat{m} = \underset{m}{\operatorname{argmax}}\, P(m|\mathcal{D}_g)$$

ベイジアンネットワークの予測事後分布は，以下のように計算される（記号の意味は 3.2 節参照）．ここで，r_i と 1 からくる項は，事前分布のハイパーパラメータに由来する（一様分布）．

$$P(X|\mathcal{D}_g) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{N_{ijk} + 1}{N_{ij} + r_i} \tag{2.9}$$

$\hat{m}$ の推定方法は，アルゴリズムによって異なる．BOA においては，ネットワークの事後確率を直接用いる（3.2.2.1 節参照）．EBNA 及び LFDA では BIC（Bayesian Information Criteria，3.2.2.2）近似を用いる．

2.4.2 木構造 EDA（GP モデル：GP-EDA）

GA-EDA の場合，固定長一次元配列を用いることからモデルの構築が非常に容易である．しかし，GA の表現能力が GP のそれより低いことからわかるように，固定長一次元配列ではプログラムや関数などの構造表現を行うことができない．そのため，EDA へ木構造を導入する試みが近年行われている．本論文では，前述のように木構造 EDA をまとめて GP-EDA と記述する[2]．

[2]PMBGP（Probabilistic Model Building GP）などとも呼ばれるが，一般的に普及している略語は今のところない．

GP-EDAの研究は，GA-EDAのそれと比較してあまり行われいない．その理由として考えられるのは以下の点である．

1. 木構造であること．
2. 位置非依存の部分構造が存在する．
3. イントロンが存在する．

まず，第一の点であるが，木構造は可変長である．GA-EDAの場合のように$X = \{x_1, x_2, \cdots, x_n\}$のように単純に番号付けした場合，可変長であるが故に一対一対応をとることが出来ない．そのため，なんらかの工夫が必要となる．第二点として，GAの部分構造は位置依存であるのに対して，GPのそれは位置非依存である場合が存在する．GAでは，遺伝子座それぞれに意味があるが，GPでは部分木構造それ自身に意味がある．そのため，GPにおける部分構造には明確な依存関係が存在する．さらに，GPではイントロンの問題もある．イントロンは適合度に影響を及ぼさない冗長構造である．現在まで提案されている手法は1と2については考慮しているが，問題点3については考慮していない．すなわち，あくまでBuilding Blockの推定に焦点を当て，イントロンの影響については一切考慮しない．本論文で提案するPOLE及びPAGEにおいても同様であり，イントロンについては考慮しない．

上に挙げたような問題点を踏まえ，GPの確率モデルとしては，現在までに以下の二つのアプローチが考案されている．

- プロトタイプ木モデル
  これはPIPE（Probabilistic Incremental Program Evolution）ではじめに提案されたモデルである．木構造を固定長の構造に変換し，その上でGA-EDAのアプローチを適用する手法である．直感的であり，わかりやすい反面，位置に依存しない部分構造の推定は難しい．
- 確率文脈自由文法モデル
  GGGP（Grammar Guided GP，4.2節参照）は，初めて文脈自由文法（CFG:Context Free Grammar）をGPに導入した手法である．SG-GP（Stochastic Grammar based GP）はそれをさらに確率文脈自由文法にした手法である．この手法では，すべてのGPの関数とプログラムはCFGで表現される．

以下，この二つのモデルを詳しく説明する．PCFG（Probabilistic Context Free Grammar）に基づくGP-EDAに関しては，文献[Shan 06]を参考にしている．

2.4.2.1 プロトタイプ木モデル

木構造に対して確率モデルを提要する場合，最も単純なアイディアは，木構造を線形固定長配列に変換し，その配列に対してEDAで用いられるような確率モデルを適用するこ

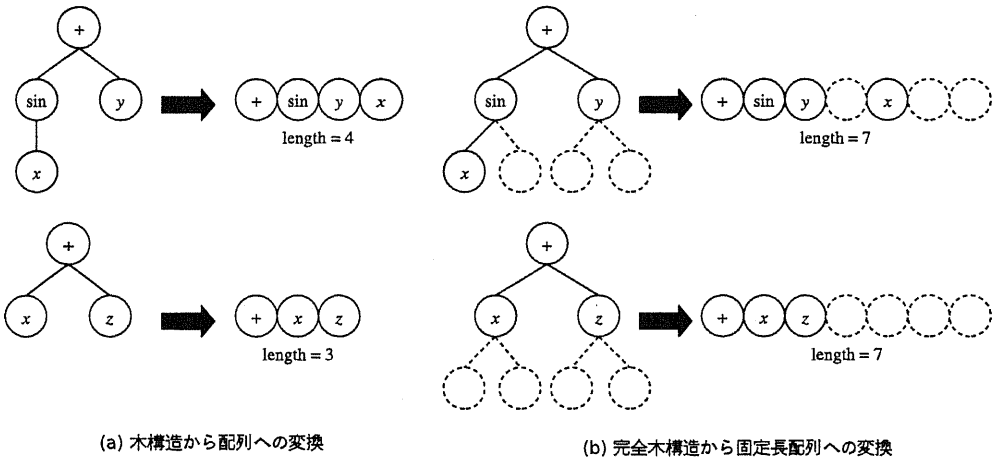


図 2.7: 木構造から配列への変換.

とである．このような手法ははじめに PIPE（2.4.2.2 節）によって導入された．例えば，図 2.7（a）に幅優先を用いた場合の，木構造 → 配列の変換例を示す．このような操作によって，任意の木構造は一般性を失うことなく線形配列に変換される．しかし，この操作によって生成される線形配列のサイズは，元の木構造の大きさに左右されるため，個体によって長さが異なる．異なる長さの配列では，GA-EDA の確率モデルの適用は行えない．これの解決策として，現在までの多くの手法では，図 2.7（b）に示すように，完全木を考えることで，すべての木構造を同じサイズに変換している．図 2.7（b）の構造はそのままでは，完全木にならないため，ダミーノードを付加することで完全木にしている．

プロトタイプ木モデルを用いる利点は，GA-EDA の手法を容易に導入出きるという点にある．GA-EDA は，非常に多くの手法が提案されている．そのため，GA-EDA の確立した手法を導入することで，実装などの面で非常に有利である．また，プログラムと一対一対応なため，依存関係を扱いやすいというメリットもある．半面，プロトタイプ木モデルでは，位置非依存の部分構造を扱いにくいなどの問題点も存在する．

以下では，プロトタイプ木の代表的な手法について解説する．

2.4.2.2 PIPE

PIPE（Probabilistic Incremental Program Evolution）[Sałustowicz 97a, Sałustowicz 97b] は，GP-EDA の分野において初めてプロトタイプ木を導入した手法である．PIPE ではプロトタイプ木を用いることで，木構造に対して確率モデルを適用している．PIPE は確率分布として，各ノードに依存関係のない式 2.10 で表されるような分布を仮定する．

$$P(x_1, x_2, \cdots, x_n) = \prod_i P(x_i) \tag{2.10}$$

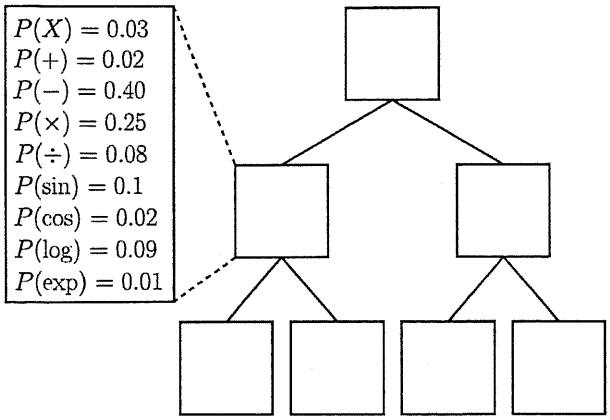


図 2.8: PIPE のプロトタイプ木．それぞれのノードには，シンボルの確率情報が記されている．

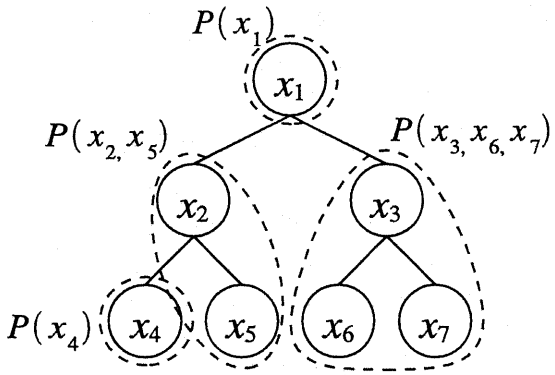


図 2.9: ECGP の確率モデル．

各ノードには，それぞれのシンボル（関数，終端ノード）の確率が記述されている（図 2.8）．PIPE はこの確率分布に従い，各位置でのノードが生成され，個体が生成される．パラメータ学習は最尤推定ではなく，逐次学習を用いている．集団中の最良個体が生成される確率が高くなるように分布は更新される．GA-EDA では，PBIL が同様の学習法を用いている．そのため，PIPE は PBIL を GP に拡張した手法と考えられる．

PIPE は非常に単純なモデルであるため，計算コストが少ない半面，ノード間の依存関係を一切考慮しないモデルであるため，基本的に部分構造の推定には適していない．PIPE の論文では，関数同定問題などに適用されているが，これらの問題は突然変異のみによっても解の獲得が可能な問題である．そのため，PIPE のモデルは，ノード間の依存関係の強い問題には適していないと考えられる．

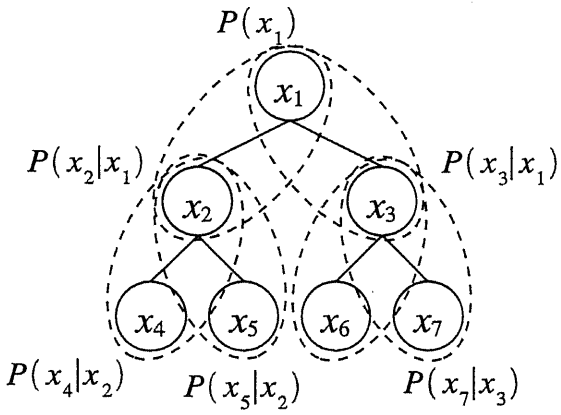


図 2.10: EDP の依存関係.

2.4.2.3 ECGP

GA-EDA である ECGA は，同時確率として表現された確率モデルを用いている．同時確率分布を用いることで，ノード間の依存関係を考慮したモデルである．ECGA が確率分布を同時確率として扱うことは，UMDA の場合とは異なり，式 2.11 を仮定していることに相当する．

$$P(x_1, x_2, \cdots, x_n|\mathcal{D}_g) = \prod_{s\in\Omega} P(s|\mathcal{D}_g) \tag{2.11}$$

但し，Ω は $s \subset X = \bigcup_{i=1}^{n}\{x_i\}$ によって構成される集合であり，$s_1, s_2 \in \Omega(s_1 \neq s_2)$ に対して $s_1 \cap s_2 = \emptyset$ を満たす．つまり，全体の分布が，重複の無い部分同時分布の積によって表されると仮定している．Ω がスキーマの集合であれば，$s \in \Omega$ はスキーマを表す．同時分布で近似することで，UMDA の場合とは異なり，スキーマが保存される．ECGA では，Ω を推定する必要があるが，最小記述長（MDL：Minimum Description Length）[Rissanen 78] を用いて学習される．具体的には式 2.12 によって表される．

$$MDL = f(N)\sum_{i=1}^{m_{prt}}(r^{k_i}-1) + N\sum_{i=1}^{m_{prt}}\sum_{j=1}^{r^{k_i}}(-p_{ij}\log p_{ij}) \tag{2.12}$$

ここで，m_{prt} はパーティション数を表し，k_i は i 番目のパーティションに含まれる変数の数を表す．r は各 x_i の取りうる値の数 $r = ||x_i||$ である．前の項の和で表される部分はパラメータ数に相当し，後ろの項は集団の尤度に相当する．$f(N)$ は，情報量基準によって異なるが（AIC（Akaike Information Criteria）の場合 $f(N) = 1$，MDL の場合 $f(N) = \log N$），ECGA の元の論文では MDL が用いられるている．

ECGP（Extended Compact GP）はこのアイディアをプロトタイプ木に拡張したものである．図 2.9 に ECGP のモデルの例を示す．この例では，全体のプロトタイプ木を

$$P(x_1, x_2, \cdots, x_7|\mathcal{D}_g) = P(x_1|\mathcal{D}_g)P(x_2, x_5|\mathcal{D}_g)P(x_4|\mathcal{D}_g)P(x_3, x_6, x_7|\mathcal{D}_g)$$

で分解していることを表す．もし，x_2 と x_4 の依存関係が深ければ，PIPE のモデルと比較して，正確な分布を推定できることになる．

この手法の欠点は，部分同時分布のサイズが，ある程度小さいものに限られるということである．部分同時分布 $s \subset \Omega$ の確率分布表は

$$K = r^{|s|} - 1$$

となる．GP の場合，平均 10 種類ほどのシンボルを用いるため，$r = 10$ とすると，せいぜい $|s| \sim 3$ 程度の部分構造しか推定できないこととなる．これは，部分構造を十分に把握出来る大きさではない．この問題は以下の二つの原因から起こる．

- シンボル数が多い．
- 同時確率を用いている．

本論文で提案する POLE（3 章参照）では，ベイジアンネットワークと拡張構文木（3.4 節参照）の二つを組合わせることで，上記の二つの問題を克服している．

2.4.2.4 EDP

GP の染色体構造は木構造であるため，PIPE のような独立性を仮定することは，GA-EDA の場合より近似精度が悪くなる．EDP（Estimation of Distribution Programming）[Yanai 03] は木構造の親と子の条件付確率を計算することで，独立の仮定より良い近似精度が得られると考えている．特に，親子というようにグラフ構造が固定であれば，グラフ学習に一切の計算を要しない．通常，ベイジアンネットワークの構造学習には多大な計算コストが必要であるため，固定構造によって効率的に探索出来るとしたら，固定構造を用いるメリットは大きいと考えられる．

x_i を木構造上で i 番目のノードとし，$ch(x_j)$ を x_j の木構造上での子ノードの集合を返す関数とする．その場合，EDP では，以下の分解の仮定を置いていることに等しい．

$$P(x_1, x_2, \cdots, x_n|\mathcal{D}_g) = P(x_1|\mathcal{D}_g) \prod_{x_{ch} \in ch(x_1)} P(x_{ch}|x_1, \mathcal{D}_g) \prod_{x'_{ch} \in ch(x_{ch})} P(x'_{ch}|x_{ch}, \mathcal{D}_g) \cdots$$

EDP のパラメータ学習は，基本的には最尤推定である．しかし，EDP では，ある程度の揺らぎを持ちつつ探索を行うためにラプラス修正によって，頻度 0 のシンボルの確率が 0 にならないように修正している．

さらに，XEDP（Extended EDP）と呼ばれる拡張手法が提案されている [柳井 05]．XEDP では，位置非依存の構造を導入することで，部分構造の抽出を試みている．XEDP

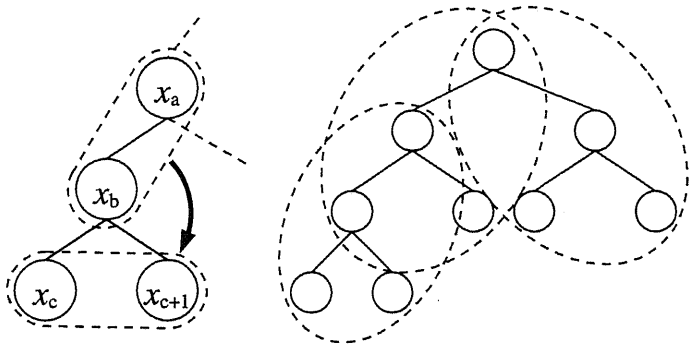


図 2.11: XEDP で推定される，再帰分布.

で推定される，位置非依存の構造は図 2.11 で表される．これは，親とその親の条件付確率によって，子ノードが決定されることを表している．

EDP 及び XEDP はモデル学習を伴わないので，計算コストは非常に少ない．しかし，モデルが固定されているため，汎用性の面では構造学習を伴う手法より弱い．

2.4.2.5 その他のアプローチ

GP で用いられる構文木ではなく，ラムダ関数を木構造に用いた GP-EDA も提案されている．BOAP（BOA Programming）[Looks 05a, Looks 05b] はラムダ関数に基づく *zigzag* 木と構造を用いている．この木のもとでは，すべての関数は 1 引数関数に変換される．さらに，特殊なオペレータを用いることで，一般性を失うことなくジグザグな木を作り，それに対してベイジアンネットワーク学習を適用している．

2.4.2.6 確率文脈自由文法モデル

プロトタイプ木を用いた手法には，GA-EDA の手法の適用が可能であるという利点がある．しかし，プロトタイプ木は固定長であり，完全 a_{max} 分木（a_{max} は関数の最大引数）の構造に限定されるため，完全木では a_{max} が大きくなるにつれノード数が指数的に増加する．ノード間の依存関係の推定（構造学習）に多くの計算量が必要となる上，適合度に影響を及ぼさないイントロン部分が多くなり，探索に不利である．また，位置依存の手法であるため，GP が想定する位置に依存しない部分構造の推定には不適である．

このような点から，PCFG（4.2 節参照）を確率モデルとして用いた手法が提案されている．PCFG を GP-EDA で用いる場合の利点は以下である．

- 木構造のトポロジーが制限されない．プロトタイプ木の場合は完全木に限定される．
- 位置に依存しない部分構造を推定可能である．

プロトタイプ木モデルの場合，ノード間の依存関係の考慮は，EDP や ECGP で分かるように容易に行われる．逆に PCFG は，PCFG の文脈非依存という仮定のため，ノード間

の依存関係を考慮することが難しい．そのため，以下で説明する手法の多くは，ノード間の依存関係を考慮しないモデルを用いている．

2.4.2.7 SG-GP

SG-GP（Stochastic Grammar based GP）[Ratle 01] は，PCFG を元にした GP-EDA としては，もっとも基本的な手法である．SG-GP は基本的な PCFG を確率モデルとして採用している．自然言語処理（NLP：Natural Language Processing）の分野では，導出木が観測されない状況を考えるため，パラメータ推定は EM アルゴリズム（4.3.1 節参照）などの，隠れ変数を含むモデル学習に用いられるアルゴリズムが適用されるが，GP では導出木が観測されるため，パラメータ学習は非常に簡単である．SG-GP の重み（パラメータ）は優良解及び劣等解を用いて，逐次的に更新される．新しい個体は，重みを用いて生成される．

SG-GP の用いる文法は文脈独立性を強く仮定しており，この仮定のもとでは，ノード間の依存関係の推定は原理的に不可能である．また，大きな導出木の確率が必然的に小さくなるため，自由度の大きい問題への適用には不適である．SG-GP の拡張手法として，深さを考慮した Vertical SG-GP と呼ばれる手法も提案されている．Vertical SG-GP では，生成規則のルールがある深さを考慮する．しかし，深さを考慮した場合，PCFG モデルの利点である，場所に依存しない部分構造の推定という特徴は失われる．

2.4.2.8 PEEL

PEEL（Program Evolution with Explicit Learning）[Shan 03] は L-System（Lindenmayer System）と呼ばれる方法に基づいている．L-System は，通常の PCFG に対して，二つの変数を加えたモデルと同等である．PEEL では，式 2.13 で表される生成規則を用いる．

$$X(d,l) \rightarrow \zeta \tag{2.13}$$

この式において，d は生成規則の現れる深さを表し，l は相対的な位置を表す．PEEL は，探索の序盤では，位置をあまり特定しない生成規則（式 2.13 で，$d = \#$，$l = \#$ などとする．ここで $\#$ は Don't care 記号）を用いる．探索が進むにつれ，徐々に d や l に具体的な値を代入することで，生成規則を特殊化していく．

2.4.2.9 Grammar Transformation in an EDA

Grammar Transformation in an EDA (以下 GT-EDA)[Bosman 04] は部分木抽出に焦点を当てたアルゴリズムである．GT-EDA における部分木は，生成規則の展開である．仮に，元の生成規則が以下で表される場合を考える．

$$
\begin{aligned}
Rule\,0 : \mathcal{S} &\rightarrow x \\
Rule\,1 : \mathcal{S} &\rightarrow +\mathcal{S}\mathcal{S} \\
Rule\,2 : \mathcal{S} &\rightarrow -\mathcal{S}\mathcal{S}
\end{aligned}
$$

GT-EDA では，右辺の $\mathcal{S}$ を一つを特定の生成規則と入れ替えることで，生成規則の特殊化を行う．$Rule\,2$ の $\mathcal{S}$ が特殊化して $+\mathcal{S}\mathcal{S}$ となった場合，生成規則は以下のようになる．

$$
\begin{aligned}
Rule\,0 : \mathcal{S} &\rightarrow x \\
Rule\,1 : \mathcal{S} &\rightarrow +\mathcal{S}^{0,1,2,3}\,\mathcal{S}^{0,1,2,3} \\
Rule\,2 : \mathcal{S} &\rightarrow -\mathcal{S}^{0,2,3}\,\mathcal{S}^{0,1,2,3} \\
Rule\,3 : \mathcal{S} &\rightarrow -\left(+\mathcal{S}^{0,1,2,3}\,\mathcal{S}^{0,1,2,3}\right)\mathcal{S}^{0,1,2,3}
\end{aligned}
$$

この生成規則で，右肩についている Index は特定に生成規則しか生成しないことを表す（$\mathcal{S}^{0,2,3}$ には $Rule\,1$ を適用できない）．GT-EDA では，このような生成規則の特殊化を MDL で行い，探索範囲を狭めて行く．GT-EDA は基本文法として，深さを考慮したモデルを用いている．

2.4.2.10 GMPE

GMPE（Grammar Model based Program Evolution）[Shan 04] はモデルマージング法 [Stolcke 94] を元にしたアルゴリズムである．GMPE は非常に自由度の高いモデルであり，部分構造を推定することが可能である．モデルマージング法は，学習データから，生成規則を推定する．

GMPE の最初の生成規則は学習データ（優良解の集合）のみを生成する．GMPE のマージオペレータを用いることで，非終端記号を統合して行き，徐々に生成規則を一般化させていく．$V, W \in \mathcal{N}$，$\zeta_i \in (\mathcal{N} \cup \mathcal{T})^*$ とする（記号の意味は 4.2 節を参照）．

$$
\begin{aligned}
V_1 &\rightarrow \zeta_1 \\
V_2 &\rightarrow \zeta_2 \\
&\Downarrow\ merge(V_1, V_2) = W \\
W &\rightarrow \zeta_1 \\
W &\rightarrow \zeta_2
\end{aligned}
$$

GMPE は非常に自由度が高いが，非常に計算コストの高い手法でもある．GMPE の最初の生成規則は学習データが用いる生成規則の分だけ存在する．その生成規則に対して $merge$ オペレータを適用するが，貪欲法によって探索を行ったとしても非常に計算量が多

い．そのため，GMPEの提案文献における計算機シミュレーションでは個体数が～100程度であり，通常のEDAと比較すると非常に小さい．個体数が少ない場合，初期世代でカバーされる探索空間は小さくなり，局所解にとらわれる確率も高くなる．

第3章　ベイジアンネットワーク推定に基づく手法

3.1　はじめに

本章ではPOLE（Program Optimization with Linkage Estimation）を提案する．提案手法POLE[長谷川 07, Hasegawa 06]では確率モデルとしてベイジアンネットワークを用いる．GPに対してベイジアンネットワークを用いる際の問題点を，拡張構文木（expanded parse tree）と呼ばれる表現手法を用いることで解決する．

GP-EDAの先行研究では，独立モデル，親子関係モデルなど様々な確率モデルが適用されているが，より一般的な依存関係を推定するためには，ベイジアンネットワークを用いることが望ましいと考えられる．しかし，GP-EDAに対してベイジアンネットワークを適用するには，ひとつの問題点が存在する．バイナリGAにおいては用いられるシンボルの数は2種類（0，1）であるが，GPにおいては非常に多くのシンボルを用いる．そのためGPの場合，ノードのシンボル数が多くなり，結果として定量的関係を表す条件付確率表（CPT：Conditional Probability Table）も大きくなる．このような状況では，ノード間の依存関係の推定に多くのサンプル数を必要とし，適切な推定が難しくなるという問題点が存在する（3.3章参照）．また，ベイジアンネットワークによって生成されたプログラムが常に文法的に正しいものである必要がある．以上の問題点を解決すべく，本提案手法では拡張構文木と呼ばれる表現手法を用いている．

本提案手法は以下のような特徴を有する．

- 確率モデルとしてベイジアンネットワークを用いる
- 染色体表現に拡張構文木を用いる

本章は以下のように構成される．まず，POLEで用いられるベイジアンネットワークについて3.2節で説明する．3.3節では，従来のプロトタイプ木GP-EDAの問題点について述べる．POLEでは染色体表現として，拡張構文木が用いられる．3.4節では，拡張構文木について説明する．3.5節でPOLEの詳細について述べる．POLEの有効性を示すために，三つのベンチマークテストに適用した．3.6節では，その実験結果について詳細に述べる．さらに，3.7節で，実験結果の考察を行い，3.8節で本章のまとめを行う．

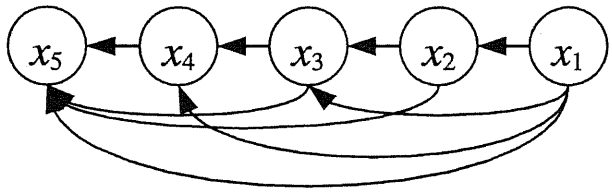


図 3.1: 式3.1のグラフィカルモデル.

3.2 ベイジアンネットワーク

ここでは，ベイジアンネットワーク（Bayesian Network）について説明を行う．なお，本節は文献 [Heckerman 95] と [Neapolitan 04] を参考にしている．

大量のデータが与えられたとき，そのデータをどのようにモデル化するかは非常に重要な問題である．通常，データには不確定要素が含まれるため，決定木などの決定的な表現よりも，確率などを用いた非決定的な表現が好ましい．ベイジアンネットワークは，有向グラフと CPT を用いて，変数間の依存関係や不確定性の度合いなどを図と確率表によって表現するグラフィカルモデルである．ベイジアンネットワークを用いることで，非決定的な事象間の因果関係を表現することが可能である．ベイジアンネットワークは GA-EDA においても非常によく用いられるグラフィカルモデルである．表現能力が比較的高いモデルである上，ループなし有向グラフであるため，サンプリングが容易であるという大きな利点を持つためである．

n 個の変数 $X = \bigcup_{i=1}^{n}\{x_i\}$ が与えられたとする．ベイジアンネットワークは X の同時確率分布を表現する．X の同時確率分布はベイズのチェインルールから，

$$P(x_1, x_2, \cdots, x_n) = \prod_{i=1}^{n} P(x_i|x_1, \cdots, x_{i-1}) \tag{3.1}$$

と分解される．式3.1をグラフとして表すと，図3.1のようになる．この状態だと，すべてのノードが繋がっており，データをわかりやすく表現しているとは言えない．ここで，X の部分集合 $\Pi_i \subset X$ を考える．Π_i が与えられたとき，x_i は X と条件付独立であるような Π_i が存在するはずである．すなわち，

$$P(x_i|x_1, x_2, \cdots x_n) = P(x_i|\Pi_i)$$

である．この条件付独立性は，ベイジアンネットワークのグラフ構造を決定する．ベイジアンネットワークの構造では，Π_i は x_i の親ノードの集合に相当する．条件付独立性を用いると，同時確率分布は

$$P(x_1, x_2, \cdots, x_n) = \prod_{i=1}^{n} P(x_i|\Pi_i) \tag{3.2}$$

のように表現される．

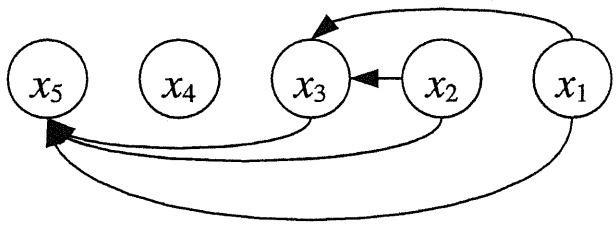


図 3.2: ベイジアンネットワークの例.

図 3.1 の場合において，$\Pi_5 = \{1,2,3\}, \Pi_4 = \emptyset, \Pi_3 = \{1,2\}, \Pi_2 = \emptyset$ である場合のベイジアンネットワークは図 3.2 のようになる.

3.2.1 パラメータ学習

ベイジアンネットワークは，前述のようにグラフ構造と CPT によって表現される．CPT のパラメータはベイズ推定によって推定される．一変数の場合について考える．データの要素を $y \in \bigcup_{i=1}^{r}\{i\}$ とし，$\mathcal{D} = \bigcup_{i=1}^{N}\{y_i\}$ とする．データ $\mathcal{D}$ を観測したあとの，次の入力の確率（予測事後分布）は式 3.3 で表される.

$$P(y_{N+1} = k|\mathcal{D}) = \int d\Theta P(y_{N+1} = k|\Theta)P(\Theta|\mathcal{D}) \tag{3.3}$$

ここで，θ_k を k 番目の値をとる時の確率（パラメータ）とし，$\Theta \equiv \bigcup_{k=1}^{r}\{\theta_k\}$，$\sum_{k=1}^{r}\theta_k = 1$ とする．$P(y_{N+1} = k|\Theta) = \theta_k$ であるから，式 3.3 は

$$P(y_{N+1} = k|\mathcal{D}) = \int d\Theta \cdot \theta_k \cdot P(\Theta|\mathcal{D}) \tag{3.4}$$

となる．$\mathcal{D}$ を観測したあとでは $P(\mathcal{D})$ が一定であることを考えると，パラメータの事後分布はベイズの定理より

$$P(\Theta|\mathcal{D}) \propto P(\Theta)P(\mathcal{D}|\Theta) \tag{3.5}$$

となる．多項分布を考えると，$P(\mathcal{D}|\Theta) \propto \prod_{k=1}^{r}\theta_k^{N_k}$ である（N_k は $\mathcal{D}$ の中で $y = k$ のサンプルの数．$\sum_{k=1}^{r}N_k = N$）．パラメータの事前分布 $P(\Theta)$ は，極端な分布でなければ基本的にはどのような分布でも構わないが，計算上の理由から自然共役な Dirichlet 分布が用いられる．すなわち

$$P(\Theta) \propto \prod_{k=1}^{r}\theta_k^{N'_k-1}$$

正規化条件は，積分値が 1 である．この式で N'_k は事前分布のパラメータである．$P(\Theta)$ が自然共役な分布であるため，式 3.5 は簡単に求められる.

$$P(\Theta|\mathcal{D}) = \frac{\Gamma\left(\sum_{k=1}^{r}(N_k+N'_k)\right)}{\prod_{k=1}^{r}\Gamma(N_k+N'_k)}\prod_{k=1}^{r}\theta_k^{N_k+N'_k-1} \propto \prod_{k=1}^{r}\theta_k^{N_k+N'_k-1}$$

ここまでは，一変数のパラメータ学習であるが，パラメータの独立性を仮定することでベイジアンネットワークの場合に適用できる．パラメータの独立性とは，どのパラメータも他のパラメータに影響されないという仮定である．パラメータ独立の仮定のもとでは，各パラメータを独立に推定することが可能である．

n 変数の完全データ（隠れデータや欠損データがない）の要素を $X=\bigcup_{i=1}^{n}\{x_i\}$ とし，データを $\mathcal{D}=\bigcup_{i=1}^{N}\{X_i\}$ とする．x_i のとりうる場合の数を r_i（$\|x_i\|=r_i$），Π_i のとりうる場合の数を q_i とする（$\|\Pi_i\|=q_i$）．θ_{ijk} は x_i が k 番目の状態にあり，その親 Π_i が j 番目にある確率を表す（パラメータ）．

$\Theta_{ij}\equiv\bigcup_{k=1}^{r_i}\{\theta_{ijk}\}$，$\Theta\equiv\bigcup_{i=1}^{n}\bigcup_{j=1}^{q_i}\Theta_{ij}$ とする．Θ を Dirichlet 分布に従うとすると，Θ_{ij} の事前分布は式 3.6 によって表される．ここで N_{ijk} は x_i が k，Π_i が j 番目にあるサンプルの数を表し，N'_{ijk} は事前分布のパラメータである．

$$P(\Theta_{ij}|G)\propto\prod_{k}\theta_{ijk}^{N'_{ijk}-1} \tag{3.6}$$

Dirichlet 分布は自然共役な分布であるので，一変数の議論と同様の計算によってデータ $\mathcal{D}$ が与えられた時の事後分布は

$$P(\Theta_{ij}|\mathcal{D},G)\propto\prod_{k}\theta_{ijk}^{N'_{ijk}+N_{ijk}-1}$$

となる．構造 G とデータ $\mathcal{D}$ が与えられた時の予測事後分布 $P(X|\mathcal{D},G)$ は，Θ について積分することで計算される．これから，予測事後分布は式 3.7 で表される．$N_{ij}=\sum_k N_{ijk}$，$N'_{ij}=\sum_k N'_{ijk}$ である．

$$P(X|\mathcal{D},G)=\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{N_{ijk}+N'_{ijk}}{N_{ij}+N'_{ij}} \tag{3.7}$$

EBNA（Estimation of Bayesian Network Algorithm）などでは，事前分布を $N'_{ijk}=1$ としている．そのため，事後分布は式 2.9 のように計算される．

3.2.2 ネットワークの事後確率

パラメータ学習は，構造が与えられたときの推定であるが，データ $\mathcal{D}$ のみが与えられた場合は，構造に関しては未知である．そのような場合，構造の探索とパラメータの推定

の二つのタスクを行う必要がある．EDA においても，変数間の依存関係が未知である場合は，依存関係のグラフを推定する必要がある．

EDA においては，予測事後分布 $P(X|\mathcal{D})$ に従って，新しい個体が生成される．構造を G，$\mathcal{G}$ を可能なネットワーク構造とする．ベイジアンネットワークを用いた場合，予測事後分布は式 3.8 で表される．

$$\begin{aligned} P(X|\mathcal{D}) &= \sum_{G\in\mathcal{G}} \int d\Theta P(X|\Theta,G)P(\Theta,G|\mathcal{D}) \\ &= \sum_{G\in\mathcal{G}} \int d\Theta P(X|\Theta,G)P(\Theta|\mathcal{D},G)P(G|\mathcal{D}) \end{aligned} \tag{3.8}$$

通常，全ての構造に対して和を計算することは不可能である．そのため，最良の一つの構造によって近似（MAP（Maximum a Posteriori）近似）される．

$$P(X|\mathcal{D}) \simeq \int d\Theta P(X|\Theta,\widehat{G})P(\Theta|\mathcal{D},\widehat{G})$$

ここで，$\widehat{G} = \underset{G\in\mathcal{G}}{\operatorname{argmax}} P(G|\mathcal{D})$．右辺の計算は，構造が与えられた時の通常のパラメータ学習である．データが与えられた時の，グラフ G の事後確率 $P(G|\mathcal{D})$ を計算し，最大の値を与える G を選択することで MAP 近似は行われる．ベイズの定理から

$$P(G|\mathcal{D}) \propto P(\mathcal{D}|G)P(G)$$

である．ここで，$P(G)$ はネットワークの事前確率である．これから，$P(\mathcal{D}|G)$ の計算を行えばよいことが分かる．$P(G)$ としては，一様分布なども用いられるが，エッジの多い構造にペナルティーをかける事前分布も用いられる．

3.2.2.1 Bayesian Dirichlet

$P(\mathcal{D}|G)$ をベイズのチェインルールを用いて計算したのが，Bayesian Dirichlet スコアである [Heckerman 94]．チェインルールを用いることで，式 3.9 のように変形される．

$$P(\mathcal{D}|G) = \prod_{l=1}^{N} P(X_l|X_1,\cdots,X_{l-1},G) \tag{3.9}$$

積の各項に対して，式 3.7 を適用することで，式 3.10 となる．

$$P(\mathcal{D}|G) \propto \prod_{i=1}^{n}\prod_{j=1}^{q_i} \cdot \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij}+N_{ij})} \cdot \prod_{k=1}^{r_i} \cdot \frac{\Gamma(N'_{ijk}+N_{ijk})}{\Gamma(N'_{ijk})} \tag{3.10}$$

3.2.2.2 BIC Approximation

BIC（Bayesian Information Criteria）近似は，$P(\mathcal{D}|G)$ の $N \to \infty$ の極限においての近似である [Schwarz 78].

$$BIC(G,\mathcal{D}) = \sum_{i,j,k} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - \frac{1}{2} \log N \sum_{i=1}^{n} q_i(r_i - 1) \tag{3.11}$$

最尤推定によって，最尤構造を探索する（KL ダイバージェンスで，データ $P(\mathcal{D};\widehat{\Theta})$ と構造 $P(\mathcal{D}|G;\widehat{\Theta})$ の分布を最小化する）場合，構造が複雑になればなるほど単純に尤度が向上するため，モデル選択を行うことが出来ない．MDL（Minimum Description Length：最小記述長）や AIC（Akaike Information Criteria）[坂元 83] は，最大対数尤度から，パラメータ数に定数をかけた値をペナルティとして引いたものである．BIC も同様であり，定数が異なる手法と捉えることが出来る．

3.2.3 グラフ構造学習アルゴリズム

原理的には，全ての $G \in \mathcal{G}$ について事後確率 $P(G|\mathcal{D})$ を計算し，最大の事後確率を与える G を選択すればよい．しかし，現実的には，全ての構造について計算することは不可能である．そこで，近似的に $P(G|\mathcal{D})$ が最大となる構造を探索するアルゴリズムが提案されている．POLE では K2 アルゴリズムをもとにしたアルゴリズムを用いている．ここでは，K2 アルゴリズムについて説明する．

3.2.3.1 K2 アルゴリズム

ベイジアンネットワークは DAG（Directed Acyclic Graph）である．すなわち，ネットワーク上において，矢印の向きにたどった場合ループが生じない必要がある．貪欲的に $P(G|\mathcal{D})$ を最大化する G を探す場合，エッジを付加する度にループが生じていないことを確かめる必要がある．この計算コストを排除したアルゴリズムが K2 アルゴリズムである．K2 アルゴリズムでは，変数 $X = \bigcup_{i=1}^{n}\{x_i\}$ に対して順番を付ける．仮に，$x_1, x_2, \cdots, x_n$ を順番であるとすると，x_i の親ノード候補は $Parent(x_i) = \{x_j | j < i\}$ である．この制限下ではループは生じない．また，このような制限下では，探索空間が小さいため計算量自体も少なくなる．

K2 アルゴリズムで用いられるスコアは，Bayesian Dirichlet スコアにおいて，事前分布のハイパーパラメータを $N'_{ijk} = 1$，$P(G) \propto 1$ とした場合である．本論文で提案している POLE においても，K2 アルゴリズムが基本として用いられている（3.5.2 節）．

3.2.4 サンプリング

通常，任意の確率分布 $\rho(X)$，$X = \bigcup_{i=1}^{n}\{x_i\}$ からサンプリングする場合，Gibbs Sampling が用いられる．しかし，ベイジアンネットワークの場合，Probabilistic Logic Sampling

Algorithm 4 Probabilistic Logic Sampling (PLS)

1. Find ancestral ordering $o = \{o_1, o_2, \cdots, o_n\}$.
2. From $i = 1$ to n
   $x_{o_i} \leftarrow$select value from $P(x_{o_i}|\Pi_{o_i})$

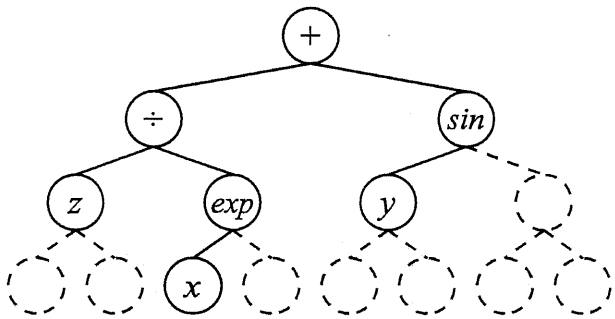


図 3.3: GP や既存の PMBGP で用いられる表現手法.

(PLS) と呼ばれる，非常に効率的なアルゴリズムによって，サンプリングが行える．GA-EDA において，ランダムマルコフ場などのグラフィカルモデルと比較して，ベイジアンネットワークが頻繁に用いられる大きな理由が PLS の高速さにあると考えられる．PLS では，Ancestral Ordering にそって，各インスタンスが決定される（アルゴリズム 4）.

3.3 既存のプロトタイプ木手法の問題点

PIPE（Probabilistic Incremental Program Evolution），EDP（Estimation of Distribution Programming），ECGP（Extended Compact GP : 2.4.2 節参照）などの既存の手法では，GP で用いられる通常の構文木によって個体を表現している．通常の構文木を用いた場合，図 3.3 のように，完全 a_{max} 分木のトポロジーを考える（a_{max} は関数ノードの中で最も多い引数を表す．図は完全 2 分木.）．その上で，それぞれの個体のルートを重ね合わせることで，シンボルの頻度を計算し確率を求める．通常の構文木を用いた場合，それぞれのノードの位置にくるシンボルは関数ノード，終端ノード両方の場合が考えられる．例えば，図 3.3 では z と exp は同じ深さにあるものの，それぞれ終端ノードと関数ノードであり，異なる種類のノードである．

ベイジアンネットワークにおいては，依存関係をグラフと CPT によって表現する．一般的な CPT のサイズ（パラメータ数）K は式 3.12 を用いて表される．ここで，r_i はノード x_i の可能なシンボルの数，Π_i は x_i のベイジアンネットワーク上での親ノードの集合，q_i は Π_i の可能な状態数を表す.

$$K = \sum_{i=1}^{n} q_i (r_i - 1) \tag{3.12}$$

通常の構文木を用いた場合，$r_i = |\mathfrak{F} \cup \mathfrak{T}|$，$q_i = |\mathfrak{F} \cup \mathfrak{T}|^{|\Pi_i|}$ となる（$\mathfrak{F}, \mathfrak{T}$ は GP の関数，終端ノードの集合）．$|\Pi_i|$ は Π_i の要素数（ノード数），n はノード数を表す．通常，GP では関数ノードと終端ノードとの合計で 10 個程度のシンボルを用いるため，通常の構文木のもとでは $r_i = |\mathfrak{F} \cup \mathfrak{T}| = 10$ となる．バイナリ GA の場合は 0 と 1 二つのシンボルしか用いず，$r_i = 2$ となるため，通常の構文木の場合の CPT のサイズは GA の場合と比較すると非常に大きくなることが分かる．

サンプル数と，学習されたネットワークの関係については文献 [Friedman 96] で議論されている．この文献では，ϵ 漸近近似，信頼度 δ でネットワークを学習する場合の必要なサンプル数が議論されている（式 3.13）．必要なサンプル数のオーダーの上限は式 3.14 で表される．Friedman らの議論では，MDL を元にしたネットワークスコアを用いた場合で議論している．これらの式で，B^* は学習データの由来する分布を表し，$Lrn()$ は構造学習ルーチンを表す．KL$(\cdot)$ は KL ダイバージェンス（Kullback Leibler Divergence）である．

$$P(\mathrm{KL}(P_{B^*}|P_{Lrn()}) > \epsilon) < \delta \tag{3.13}$$

$$O\left(\left(\frac{1}{\epsilon}\right)^{\frac{3}{4}} \log\frac{1}{\epsilon} \log\frac{1}{\delta} \log\log\frac{1}{\delta}\right) \tag{3.14}$$

この上限の式は，サンプルにノイズが含まれていてサンプルが充分に与えられた場合であり，サンプル数の上限のオーダーを決定する要因は CPT サイズ（パラメータ数）ではなく，サンプルのノイズであることを示している．

進化アルゴリズムでは，集団数は通常 $M \sim 1000$ 程度である．ここで，選択率 $P_s \sim 0.1$ 程度を考えた場合，学習サンプル数は 100 程度である．このように，充分でないサンプル数の場合，CPT サイズの影響を無視することは出来ないと考えられる．ベイジアンネットワークでは，各 CPT の値はサンプルによって推定される．もし，CPT サイズが大きく，サンプル数が少ない場合は，各パラメータは正確に推定されない．例えば，各ノードのとりうる場合の数が $r_i = 10$ である場合を考え，このノードが二つの親ノードを有するとすると，この CPT サイズは 900 である．これは，サンプル数より大きいため，正確にパラメータが推定されない．

本論文では，CPT サイズの影響を，信頼区間の視点から議論する．CPT サイズの推定を，多項分布を使って近似的に取り扱う．サンプル数が N であり，各サンプルは K 通りの値をとるとする．$\mathbf{n} = \{n_1, 2_2, \cdots, n_K\}$ は，多項分布において，それぞれの値であるサンプル数を表し（$N = \sum_i n_i$），その確率を $\mathbf{p} = \{p_1, p_2, \cdots, p_K\}$ で表す（$\sum_i p_i = 1$）．

多項分布における $100(1-\alpha)\%$ の信頼区間（α は定数）は式 3.15 で与えられる [Quesenberry 64].

$$p_j^{\pm} = \frac{\chi^2_{\frac{\alpha}{2}}(1) + 2n_j \pm \sqrt{\Delta_j}}{2(N + \chi^2_{\frac{\alpha}{2}}(1))} \tag{3.15}$$

$$\Delta_j = \chi^2_{\frac{\alpha}{2}}(1)\left\{\chi^2_{\frac{\alpha}{2}}(1) + \frac{4n_j(N-n_j)}{N}\right\} \tag{3.16}$$

この式で，$p_j^{\pm}$ は区間の上限（+）と下限（−）を表し，Δ_j は式3.16で表される．$\chi^2_{\frac{\alpha}{K}}(1)$ は自由度1のカイ二乗分布の上側確率 $100\alpha/K$ パーセント点を表す．

信頼区間がある定数 $\eta \ll 1$ で表される $\frac{n_j}{N}\eta$ より小さくなるためのサンプル数の下限を求めたい．そのための条件は式3.17で表される．

$$p_j^+ - p_j^- \leq \frac{n_j}{N}\eta \tag{3.17}$$

$n_j = \frac{N}{K}$ の場合に，式3.17はサンプル数の下限を与える．これを解くと，

$$N \geq -\frac{(2+\eta^2-2K)\chi^2_{\frac{\alpha}{K}}(1)}{\eta^2} + \chi^2_{\frac{\alpha}{K}}(1)\frac{\sqrt{(\eta^2(-2+K)^2+4(-1+K)^2)}}{\eta^2}$$

となる．$\eta \ll 1$ という条件を用いると，

$$\begin{aligned} N &\geq \frac{4\chi^2_{\frac{\alpha}{K}}(1)}{\eta^2}(K-1) \\ &= \frac{4Z^2_{\frac{\alpha}{2K}}}{\eta^2}(K-1) \end{aligned} \tag{3.18}$$

となる．式3.18から，必要なサンプル数は，ほぼ K に比例していることが分かる（正確にはリニアより若干速く増える）．これから，CPTサイズが大きくなると，必要なサンプル数が増加する．CPTサイズはとりうる値の数に大きく依存するため，とりうる値の数が増えることで，必要なサンプル数も増加する．サンプル数の増加はEDAでは集団数の増加に相当するが，集団数が多い場合，適合度評価回数も多くなる．結果として解の獲得により多くの計算量が必要となる．

このような問題を解決するために，本提案手法では拡張構文木と呼ばれる表現手法を用いる．拡張構文木においては，シンボル数を削減することが可能であり，それゆえCPTを小さく表現することができる．次節において，拡張構文木の詳細について説明を行う．

3.4 拡張構文木

拡張構文木（expanded parse tree）は，Winebergによって考案されたEP-I（Evolutionary Programming with Introns）と呼ばれるアルゴリズムで用いられている表現手法である[Wineberg 94]．GPはGAに対して木構造を扱えるように拡張したアルゴリズムであるが，GAの交叉オペレータなどを直接GPの木構造に用いることは出来ない．GP

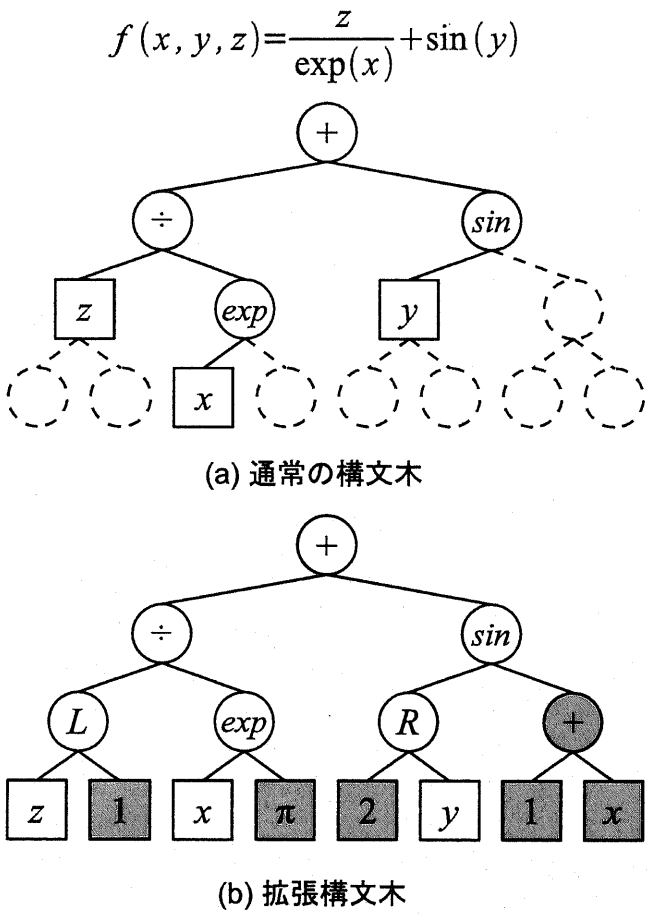


図 3.4: 拡張構文木（b）と通常の構文木（a）．灰色のノードはイントロンを表し，四角ノードは終端，円ノードは関数を表す．(a)，(b) とも文法的には同一である．

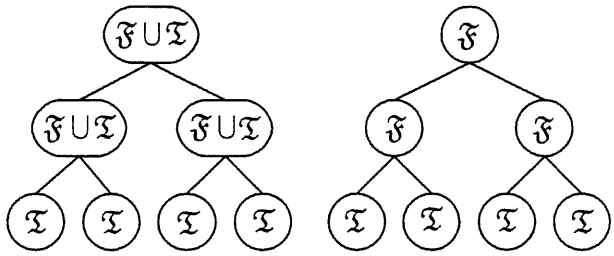


図 3.5: それぞれの構文木における，シンボルの種類と位置．$\mathfrak{F}$ は関数ノード，$\mathfrak{T}$ は終端ノードを表す．

の木構造を 1 次元の配列にすることは出来るものの，個体毎にそれぞれのインデックスにあるノードの種類（関数，終端）が異なる．そのため，GA のような交叉を行った場合に文法的に破壊された個体が出来てしまう．

EP-I では，選択ノードとイントロンを用いることで上記の問題点を解決し，GP に対して GA のオペレータを用いることに成功している．EP-I で用いられるこの表現手法は拡張構文木（expanded parse tree）と呼ばれる．図 3.4（b）は拡張構文木の例である．図 3.4（a）は通常の構文木であり，(b) と文法的に完全に同一の関数を表している．この図において L 及び R は選択ノードであり，$L(x,y)=x$，$R(x,y)=y$ と評価される．拡張構文木では，選択ノード L と R を用いることで，終端ノードを完全木の葉の部分に移動させる．また，この図において sin や exp は 1 引数関数であり，完全木にはならないので，イントロン（図の灰色のノード）と呼ばれる余分な部分構造を付け加えることで，完全木にしている．このような操作によって，すべての個体の配列の長さは等しくなり，関数ノードは必ず完全木の幹の部分に，そして終端ノードは必ず完全木の葉の部分にくるようになる（図 3.5）．すべての個体で，関数及び終端ノードの位置が固定されるため，GA のような交叉を行った場合でも文法的に構造が破壊されなくなる．

3.3 節では，GP-EDA における大きな問題として，シンボル数の多さ，すなわち CPT のサイズの大きさを挙げた．シンボル数が多くなってしまうのは，各ノード毎に関数ノードと終端ノードの両方の場合を考える必要があるためである．しかし，ここで説明した拡張構文木を用いることで，関数ノードと終端ノードの出現位置が固定される．そのため，拡張構文木の葉のノードでは終端ノードの場合のみ，幹では関数ノードの場合のみを扱えばよい（図 3.5）．これにより，各ノードの可能なシンボル数は小さくなる．

本提案手法では，拡張構文木を個体の表現に用いる．EP-I においては選択ノードとして，L 及び R が使われているが，L のみでも同様に表現することが出来る．また，L のみの場合の方がシンボル数が小さくなることから，本提案手法では L のみを採用する．

次に，CPT のサイズの縮小，及び必要となるサンプル数の違いを見るために，簡単な例を用いて説明する．分かりやすい例としてパリティ問題を考える．用いるシンボルは $\mathfrak{F}=\{And, Or, Nand, Nor\}$，$\mathfrak{T}=\{d_0, d_1, d_2, d_3, d_4, d_5\}$ とする．その上で図 3.6 にあるように，完全木上で幹に属するノードから，幹に属するノードへの依存関係がある場合を考

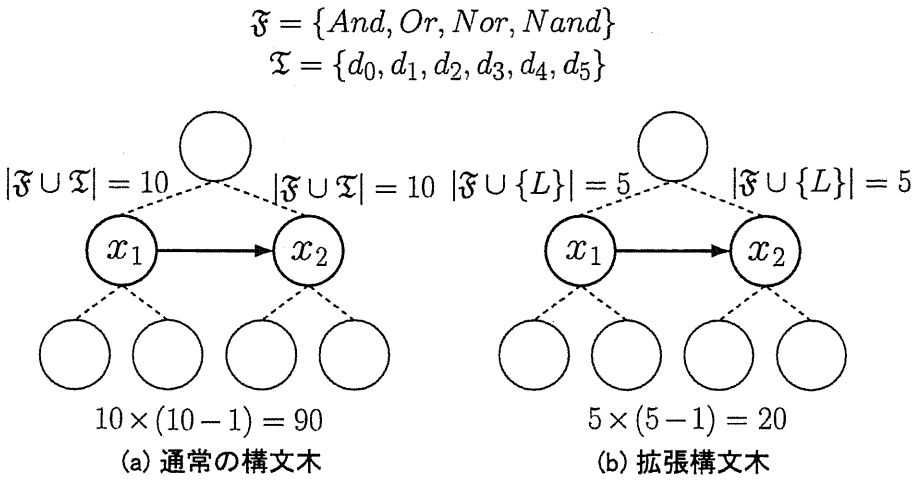


図 3.6: CPT サイズの比較.

える．通常の構文木の場合，幹に来るノードは $\mathfrak{F} \cup \mathfrak{T}$ であるので 10 通り，葉に来るノードは $\mathfrak{T}$ であるので 4 通りである．そのため，CPT の $P(x_2|x_1)$ のサイズは 90 と計算される．一方，提案手法で用いる拡張構文木の場合，幹に来るノードは $\mathfrak{F} \cup \{L\}$ (L は選択ノード) のみであるので，CPT のサイズは 20 と計算される．この例の場合，拡張構文木を用いた場合の CPT のサイズは 2/9 のサイズとなる．

提案手法はバイナリ GA に対してベイジアンネットワークを適用した BOA (Bayesian Optimization Algorithm)，EBNA (2.4.1 節参照) のプログラム進化への適用である．提案手法においては拡張構文木の導入により GA の線形表現と木構造を結びつけることに成功している．また前述のように，拡張構文木を用いることにより CPT のサイズの縮小が可能となっており，プログラム進化におけるシンボル数の問題点を改善している．さらに GP においては，プログラムは全て文法的に正しい必要がある．拡張構文木においては，幹の部分に任意の $\mathfrak{F} \cup \{L\}$，葉の部分に任意の $\mathfrak{T}$ のシンボルがあった場合においても，必ず文法的に正しいプログラムとなる．そのため，ベイジアンネットワークから新しい個体を生成する場合，特別な処理を必要としないというメリットが存在する．

3.5 アルゴリズム

この節では，提案手法である POLE (Program Optimization with Linkage Estimation) の詳細を説明する．POLE の探索の流れは他の確率モデル手法と同じであり，以下のように表される．

Step 1 初期個体の生成

初期集団が生成される．集団数 (M) 個ランダムに生成される．

Step 2 適合度の計算

各個体に対して適合度の計算を行う．

Algorithm 5 POLE

1. $\Theta_0 \leftarrow$Initialize parameters
   $G_0 \leftarrow$Empty network (no edges)
   $g \leftarrow 0$
2. $\mathcal{P}_0 \leftarrow$Generate M individuals from $P(X|\Theta_0, G_0)$
3. $\mathcal{D}_g \leftarrow$Select $N \leq M$ promising solutions from $\mathcal{P}_g$ using a truncate selection
4. $G_g \leftarrow$Find the optimum network with K2 algorithm
   $\Theta_g \leftarrow$Estimate parameters
5. $\mathcal{P}_g \leftarrow$Generate M individuals from $P(X|\Theta_g, G_g)$
   $g \leftarrow g + 1$

Step 3 個体の選択
各個体の適合度に従い，ネットワーク構築及びパラメータ推定などに用いる個体を選択する．選択する数は，選択率（P_s）を用いて，$N = M \times P_s$ 個と表される．選択方法としては Truncate Selection を用いている．

Step 4 確率モデルの構築
Step 3 で選択したサンプルを用い，ベイジアンネットワークの構築およびパラメータ推定を行う．BIC において最もスコアが高いネットワークを，K2 アルゴリズムによって探索する．

Step 5 新しい個体の生成
Step 4 で構築した一つの最良のネットワークを用い，新しい個体を生成する．エリート選択を用いる場合は，優良個体 $M \times P_e$ 個をそのままコピーして用いる．P_e はエリート率とする．

探索終了条件を満たすまで，Step 2 から Step 5 まで繰り返し行われる．POLE の Pseudo コードをアルゴリズム 5 に示す．ここで，変数についている下付文字は，その変数が g 世代目の変数であることを表す．以下の節では各 Step について詳しく説明する．

3.5.1 木構造の初期化

集団の初期化においてはランダムに木構造を生成する必要がある．GP においては，木構造の初期化手法として Grow（ランダムな構造の木を生成する）と Full（最大深さまで必ず生成する）という方法があるが，POLE においても同様の手法を用いることが出来る．POLE における Grow の場合，ルートノードより順に確率 P_F で任意の関数ノード，$1 - P_F$ で任意の終端ノードを選択する．もし，木の幹部分で終端ノードが選択された場

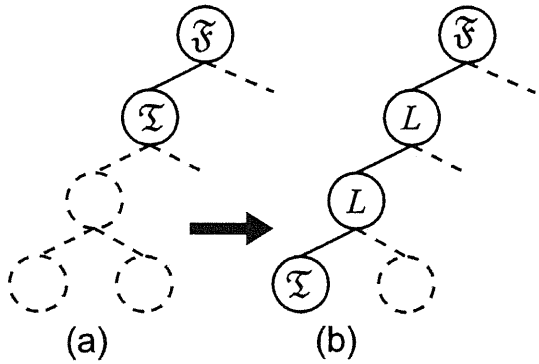


図 3.7: 拡張構文木における木構造の初期化手法.

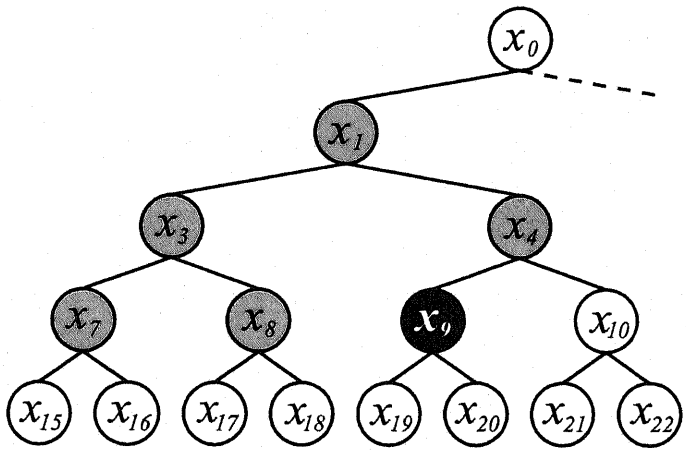
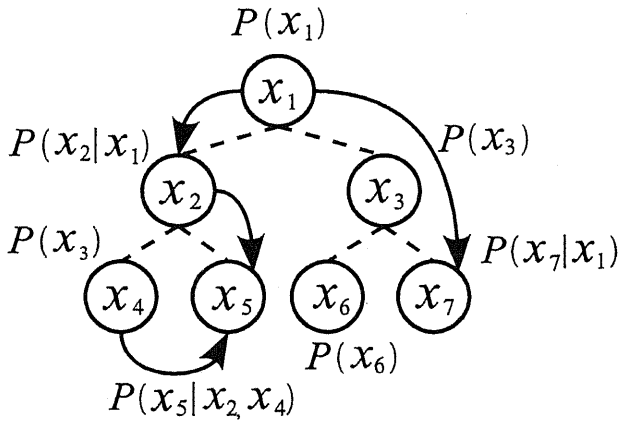
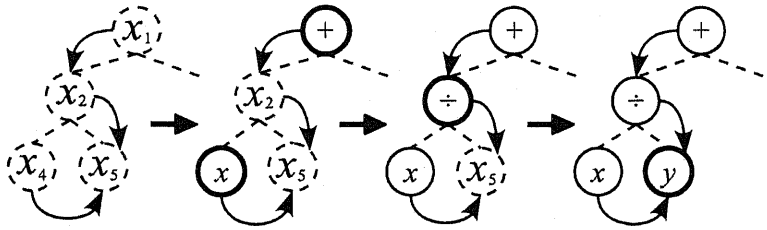


図 3.8: $R_P = 2$ の場合における，ノード X_9 に対するベイジアンネットワークの親ノード候補を灰色ノードで表している.



(a) ベイジアンネットワークの例



(b) 個体の生成

図 3.9: ノード間のベイジアンネットワークの例と，そのネットワークにおける個体の生成．図の中の矢印は依存関係を表す．

合は，そこでは終端記号にせず，それより下の第一引数の部分に選択ノード L を挿入し，最大深さ部分の木の葉の部分まで移動させる（図 3.7）．最大深さに到達した場合は自動的に終端ノードを選択する．このような手順によって拡張構文木において Grow を再現することが出来る．Full は $P_F = 1$ とすることで容易に再現できる．本章での計算機実験では，POLE の初期化に Grow を用いている．

3.5.2 ベイジアンネットワークの構築

本提案手法では，確率モデルとしてベイジアンネットワークを用いる．ベイジアンネットワークはプログラムのノード間の依存関係を推定するために用いられる．

本論文ではネットワークのスコアとして BIC[Schwarz 78] を用いている（3.2.2 節参照）．BIC はベイジアンネットワークのスコア計算手法として，EBNA [Etxeberria 99] などで用いられている方法である．BIC スコアは式 3.19 で表される．

$$BIC(G,\mathcal{D}) = \sum_{i,j,k} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - \frac{K}{2} \log N \tag{3.19}$$

これらの式において，$\mathcal{D}$ はデータ，G はグラフを表す．式3.19において，N_{ijk} は x_i が k，Π_i が j 番目にあるサンプルの数を表す．$N_{ij} = \sum_k N_{ijk}$ である．K は式3.12で表され，CPTのサイズを表す．式3.19から分かるように，BICにおいては最大対数尤度から，パラメータ数，すなわちCPTのサイズを引いた値がネットワークのスコアとなっている．

ベイジアンネットワークの構築は非常に計算量が多いことで知られている．様々な手法が提案されているが，本提案手法で用いる構築アルゴリズムはK2アルゴリズム [Cooper 92] に基づいており（3.2.3節参照），スコア計算方法は前述のBICを用いている．K2アルゴリズムは比較的高速にベイジアンネットワークを構築することが出来る．K2アルゴリズムにおいてはノードに順番をつけ，ノード x_i の親ノードをそれより小さい番号を有するノードに限定する．ノード x_i に対して，x_i の親候補が親であった場合のスコアを全て計算し，ネットワークのスコアが最も高い場合の候補を実際に親とする．スコアが上昇しなくなるまで繰り返し，上昇しなくなった場合 $i = i+1$ として再び探索を行う．ただし，最大入来エッジ数 k は超えてはならない．

K2アルゴリズムでは，各ノードに番号を1から順につける必要がある．本提案手法では木構造に対して幅優先[1]によって番号をつける．これによって，任意のノードの親候補は，そのノードと同じか，上の階層に位置することになり，自然な依存関係となると考えられる．

K2アルゴリズムでは，ベイジアンネットワークでの親ノードを限定することで高速にネットワークを構築することが出来るが，それでもなおGPのように非常に多くのノードを扱う場合には非常に重い処理となる．木構造においては，近くにあるノード同士の方が遠くにあるノード同士より関係が深いと考えられる．そのため，提案手法ではこの仮定にもとづき，さらに親ノードの候補を絞っている．POLEにおいては以下の二つの条件を満たすノードが，ノード x_i の親候補となる．

- ノード x_i に対して R_P 階層上のノードを根とするような部分木に属するノード
- ノードの番号が i より小さいノード

図3.8に例を示す．この図では $R_P = 2$ の場合のノード x_9（黒ノード）の親ノードとなり得るノードを示している．親ノード候補は灰色で示すノードであり，x_1，x_3，x_4，x_7，x_8 の5つである．

図3.9 (a) はノード間のベイジアンネットワークの構築例である．この図において，矢印はノード間の依存関係を表している．POLEで用いられているK2アルゴリズムのPseudoコードをアルゴリズム6に示す．このPseudoコードにおいて，$Cand(x_i)$ はノード x_i の親候補の集合を表し，$addedge(G, A \rightarrow B)$ は構造 G にエッジ $A \rightarrow B$ が付加された構造を返す．

[1]ルートノードを1とし，同じ深さのノードについては左から右へと数える．右端まで到達したら，次の深さへ移動し同様の操作を行う．

Algorithm 6 Modified K2 algorithm used in POLE.

Inputs :

The number of maximum incoming edges k, Data $\mathcal{D}$, Parent candidacies $Cand(x_i)$

Outputs :

The network which approximately maximize $BIC(G, \mathcal{D})$

```
for (i = 1;i ≤ n;i++) {
  G = empty network;
  P_old = BIC(G, D);
  findmore = true;
  while (findmore && |Π_i| < k) {
    Z =node in Cand(x_i) − Π_i, that maximizes
        BIC(addedge(G, Z → x_i), D);
    P_new = BIC(addedge(G, Z → x_i), D);
    if (P_new > P_old) {
      P_old = P_new;
      G = addedge(G, Z → x_i);
    }
    else {
      findmore = false;
    }
  }
}
```

表 3.1: POLE の主要パラメータ.

変数	意味	値
M	集団数	–
G	世代数	–
D_P	木構造の深さ	–
P_s	選択率	0.1 (MAX, DMAX) 0.2 (Royal Tree)
R_P	親候補の範囲	2 (MAX, DMAX) 4 (Royal Tree)
k	ノード当りの入来エッジ数制限	∞
P_e	エリート率	0.005
P_F	初期化時の関数選択率	0.8

3.5.3 個体の生成

構築したベイジアンネットワークと，推定をした条件付確率分布を用いて新しい個体を生成する．POLE では PLS によって個体が生成される（3.2.4 節参照）．まずベイジアンネットワーク上で親ノードの無いノードのシンボルが決定される．その次に，すべての親ノードのシンボルが既に決まっているノードのシンボルが条件付確率分布を用いて決められる．ベイジアンネットワークは DAG であり循環構造を持たないので，上記の作業を繰り返すことですべてのノードのシンボルを決めることが出来る．図 3.9 (b) は (a) のベイジアンネットワークの場合の，ノードのシンボル決定の様子を図示したものである．

3.6 計算機実験

本提案手法の有効性を示すために，3 つの問題に適用した．本提案手法との比較は，以下の 4 つの手法と行った．

- **POLE**
  提案手法である．ノード間の依存関係を推定するために，ベイジアンネットワークが構築される．ノードへの最大入来エッジ数 k は制限しない．個体の初期化は 2 つの問題とも Grow を用いる．

- **Univariate Model (≈PIPE)**
  POLE において最大入来エッジ数 k を 0 とした場合であり，ベイジアンネットワークは構築されずノード間の依存関係は推定されない．PIPE と同様，確率モデルは独立となる．Univariate Model では，拡張構文木を個体表現として用いる．個体の初期化は 2 つの問題とも Grow を用いる．

- **Adjacent Model (≈EDP)**
  確率モデルとして，EDP で用いられる親子の関係を考慮したモデルである．EDP とは異なり，拡張構文木を用いている．個体の初期化は 2 つの問題とも Grow を用いる．

- **Simple GP (SGP)**
  Simple GP は，一般に用いられる GP を表す．この実験では，$P_e = 0.005$（エリート率），$P_c = 0.995$（交叉率），$P_m = 0$（突然変異率），$P_r = 0$（Reproduction 率）というパラメータを用いている．交叉に関しては，文献 [Koza 92] に従い，一つ目の交叉点は，0.9 の確率で関数ノードから，0.1 の確率で終端ノードから選択する．また，二つ目の交叉点は，交叉対象の二つの個体が深さ制限を越えないように，ランダムに選択される．トーナメントサイズは $T_s = 2$ としている．また，個体の初期化には Grow を用いている．

上記のすべてのアルゴリズムにおいて，集団数 M は以下のように決定される．まず，集団数 $M = 100$ より開始し，$\sqrt[5]{10}$ 倍づつ徐々に増やす．

$$M = 100, 160, 250, 400, 630, 1000, 1600, ..., 25000$$

各集団数のもとで，20 回アルゴリズムを走らせ，そのアルゴリズムが 20 回中 20 回の割合で最適解を獲得した場合，集団数の増加を止める．その上で，平均適合度評価回数 F_{avr} を計算する．このような，集団数決定方法は [Pelikan 99a, Etxeberria 99] などで用いられている．F_{avr} は非常に大きな分散を持つため，上記の操作を 10 回行い，F_{avr} の平均を計算し，t 検定を行った．

各アルゴリズムの停止条件は以下の通りである．まず，最適解を獲得した場合はどのアルゴリズムにおいても探索を終了する．また，POLE，Univariate Model，Adjacent Model は，比較的収束が速いため，10 世代に渡って最良適合度の改善がなかった場合に停止する．SGP では，世代 g から世代 $2g$（$g > 10$）まで最良適合度の改善がなかった場合に停止する．

各問題で用いられるパラメータの具体的な値は表 3.1 の通りである．Univariate Model 及び Adjacent Model で用いるパラメータは，該当するパラメータについては POLE と同様の値を用いている．

3.6.1 最大値問題（MAX Problem）

3.6.1.1 問題設定

最大値問題（MAX problem）[Gathercole 96, Langdon 97] は GP の交叉メカニズムを調べるために考案されたベンチマーク問題であり，GP-EDA の分野において幅広く用いられている [Shan 04, 柳井 05]．最大値問題の目的は，決められたノードを用い，最大深さ制限のもとで，最大の値を返す関数を生成することが目的である．通常，以下の 3 つのノードが用いられる（式 3.20）．

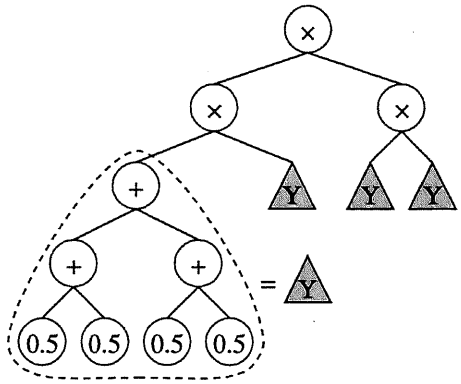


図 3.10: 最大値問題の最適解の構造．Y は値が 2 である部分構造である（Y = ((0.5 + 0.5) + (0.5 + 0.5))）．

$$\begin{aligned} \mathfrak{F} &= \{+, \times\} \\ \mathfrak{T} &= \{0.5\} \end{aligned} \tag{3.20}$$

この問題の最適解は以下のように求まる．まず，0.5 を 4 つ足すことで 2 を作る．その上で，この 2 を × を用いて掛け合わせることで，最大値が生成される（図 3.10）．この場合，問題の最大深さを D_P とすると，最大値は $2^{2^{D_P}}$ となる．

3.6.1.2 結果

表 3.2 は，最適解を獲得するのに必要な平均適合度評価回数と，その標準偏差を示した表である．図 3.11 は表 3.2 を図示したものであり，横軸に問題サイズ（木のサイズ），縦軸に評価回数をプロットしている．ここで，木のサイズは最適構造に含まれるノード数を表す．最大値問題では，木のサイズは $2^{D_P} - 1$ となる．図 3.11 から分かるように，SGP はその他の手法と比較して，多くの適合度評価が必要である．POLE は最も少ない評価回数で最適値を獲得しているが，Univariate Model との差は非常に小さい．表 3.3 は，t 検定の結果を示した表である．表 3.3 から分かるように，POLE と Univariate Model のパーセント点は 1%より小さい．そのため，POLE と Univariate Model の平均の差は優位であると考えられる（優位水準 1%）．POLE と Univariate Model の差は統計的に優位であるものの，差は非常に小さい．これは，最大値問題には，騙し要素がなく，山登り的手法によって解けるためである．Adjacent Model は SGP よりは優れているものの，POLE や Univariate Model と比較すると探索性能が低い．最大値問題のノード間にはほとんど依存関係がないため，余分な依存関係の考慮はかえって探索能力を低下させることが分かる．提案手法 POLE では，優良解より依存関係を推定するため，Adjacent Model のように余分な関係を推定することはなく，効率よく探索することが可能である．

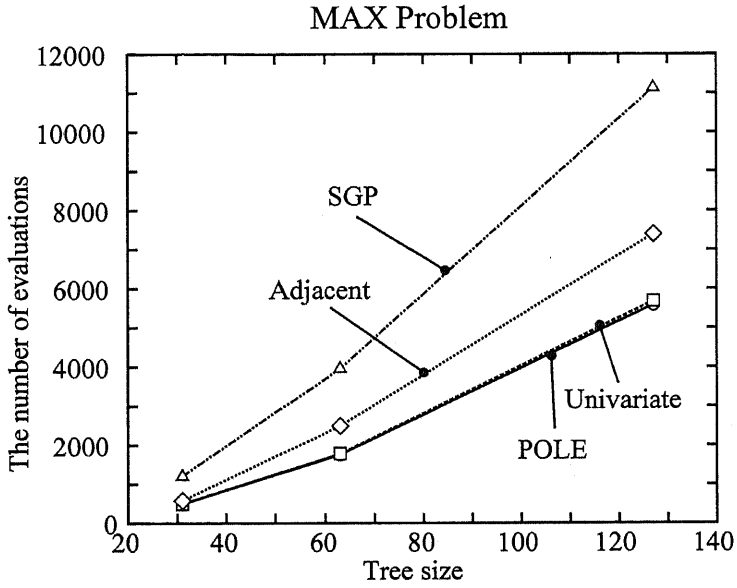


図 3.11: MAX 問題における適合度評価回数.

表 3.2: MAX 問題における適合度評価回数. 括弧内の数字は標準偏差を表す.

		$D_P = 5$	$D_P = 6$	$D_P = 7$
POLE	Average	487	1758	5588
	Stdev	(21)	(59)	(75)
Univariate	Average	497	1786	5688
	Stdev	(21)	(108)	(46)
Adjacent	Average	581	2496	7401
	Stdev	(28)	(186)	(213)
SGP	Average	1209	3962	10540
	Stdev	(268)	(825)	(2797)

表 3.3: t 検定の結果. 各値は P 値を表す.

t-value	$D_P = 5$	$D_P = 6$	$D_P = 7$
POLE vs Univariate	3.30×10^{-1}	4.86×10^{-1}	2.69×10^{-3}
POLE vs Adjacent	1.85×10^{-7}	1.47×10^{-7}	3.06×10^{-11}
POLE vs SGP	1.27×10^{-5}	1.37×10^{-5}	3.34×10^{-4}

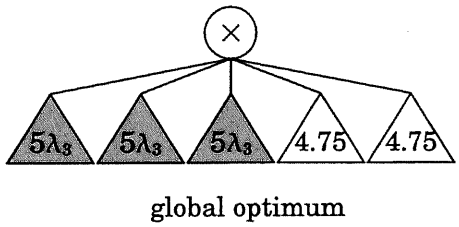


図 3.12: DMAX 問題の最適値（$m = 5, r = 3, D_P = 3$）.

3.6.2 騙し最大値問題（DMAX problem）

3.6.2.1 問題設定

POLE を騙し最大値（DMAX：Deceptive MAX）問題にも適用した．DMAX 問題は最大値問題の拡張問題である．最大値問題は前節の実験結果から分かるように，騙し要素が無い問題なため，ノード間の依存関係を考慮しない GP-EDA であっても，容易に解くことが可能である．そこで，最大値問題に騙し要素を加えた騙し最大値問題（DMAX Problem）を考案した．DMAX 問題の目的は最大値問題と同じであり，深さ制限のもとで最大の実数を返す関数を求めることである．DMAX 問題では以下のノードを用いる．

$$\begin{aligned} \mathfrak{F} &= \{add_m, multiply_m\} \\ \mathfrak{T} &= \{\lambda, 0.95\} \end{aligned} \tag{3.21}$$

$$(\lambda)^r = 1, \lambda \in \mathbb{C}, r \in \mathbb{R} \tag{3.22}$$

$$\begin{aligned} add_m(a_0, a_1, ..., a_{m-1}) &= \sum_{i=0}^{m-1} a_i \\ multiply_m(a_0, a_1, ..., a_{m-1}) &= \prod_{i=0}^{m-1} a_i \end{aligned} \tag{3.23}$$

式 3.21 で，add_m 及び $multiply_m$ は m 引数関数であり，式 3.23 によって定義される関数である．終端 λ は複素数であり，r 乗すると 1 になる（式 3.22）．λ が複素数であるため，関数の値は一般的に複素数となる．個体の適合度は，関数の値の実数部である．関数の値が $v + wi$ $(v, w \in \mathbb{R},\ i = \sqrt{-1})$ である場合，その個体の適合度は v となる．DMAX 問題は 3 つのパラメータ m（引数の数），r（乗数），D_P（最大深さ）によって与えられる．この問題の難しさはこの 3 つのパラメータに依存する．この論文では，$m = 5$，$r = 3$，$D_P = 3, 4$ で行っている．この設定では，DMAX 問題は非常に強い騙し要素を持つ．

$m = 5$，$r = 3$，$D_P = 3$ の場合の最大値を考える．この場合，まず λ を 5 つ足し合わせ 5λ を作る．この部分構造 5 つを $multiply_5$ にかけることで $(5\lambda)^5 = 5^5\lambda^2$ となる．しかし，$\mathrm{Re}(5^5\lambda^2)$ は負の値であるため，最適解ではない．そこで，掛け合わされる 5 つの部分構造の内 2 つを 5×0.95 の部分構造で置き換える．これにより，最大値は

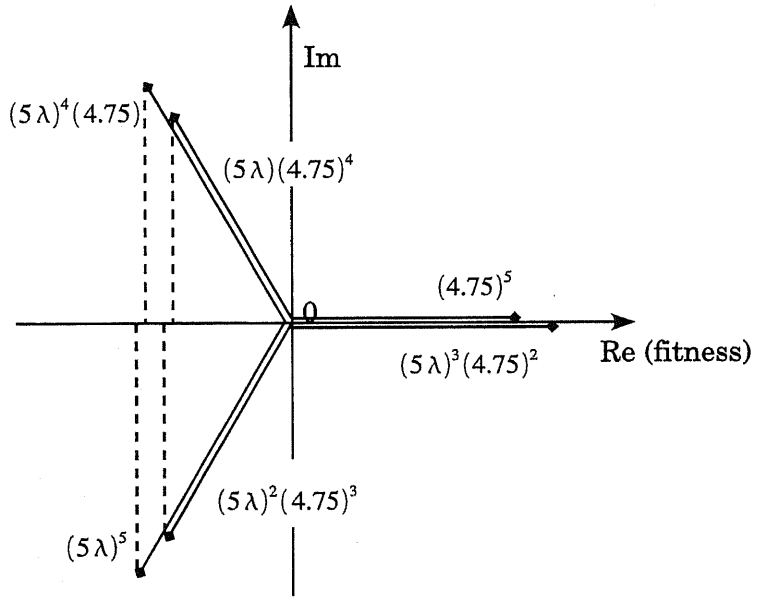


図 3.13: DMAX 問題における最適解と局所解の，複素平面上での表現.

$(5\lambda)^3(0.95\times5)^2 = 2820.3125$ となる．$D_P = 4$ の場合には，$(5\lambda)^{24}(0.95\times5) = 2.83\times10^{17}$ となる．

図 3.13 は，DMAX 問題において，大きな絶対値を持つ解を複素平面上に示した図である．この図から明らかなように，一回 5×0.95 と 5λ を入れ替えると，偏角が 120° 変化する．そのため，3 回入れ替えることで，元の偏角に戻る．

通常の最大値問題では，$(0.5+0.5+0.5+0.5)$ という部分構造を単純にかけることで容易に最適解を獲得することが可能である．一方，DMAX 問題では複素数と実数の部分構造を組み合わせる必要があるため，最大値問題と比較して難しい問題である．

3.6.2.2 結果と考察

表 3.4 は，適合度評価回数を示したものである．$D_P = 3, 4$ の場合についての評価回数を示している．括弧内の数字は標準偏差である．図 3.16 は表 3.4 をグラフ化した図である．Univariate Model と Adjacent Model は，$M = 25000$ においても，最適解を獲得することが出来なかったため，結果は載せていない．

表 3.4 から分かるように，SGP は POLE の約 10 倍の適合度評価回数を要している．SGP の交叉を用いた場合，DMAX 問題は適合度地形は騙し要素を含む．$m = 5$，$r = 3$，$D_P = 3$ の場合における局所解と最適解は図 3.14 (a) 及び (d) で表される．構造 (a) から (d) に，交叉を用いて最短で変化するためには，構造 (b) と (c) を経由する必要がある．しかし，(b) と (c) の適合度は負の値であるため，次世代の選択の段階で死滅する確率が非常に高い．

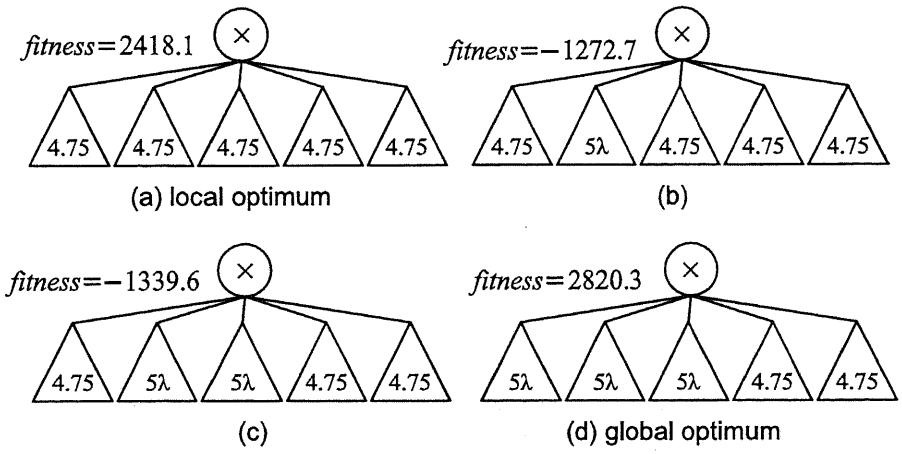


図 3.14: $m = 5$, $r = 3$, $D_P = 3$ における局所解 (a) と最適解 (d). (b) と (c) は中間構造である.

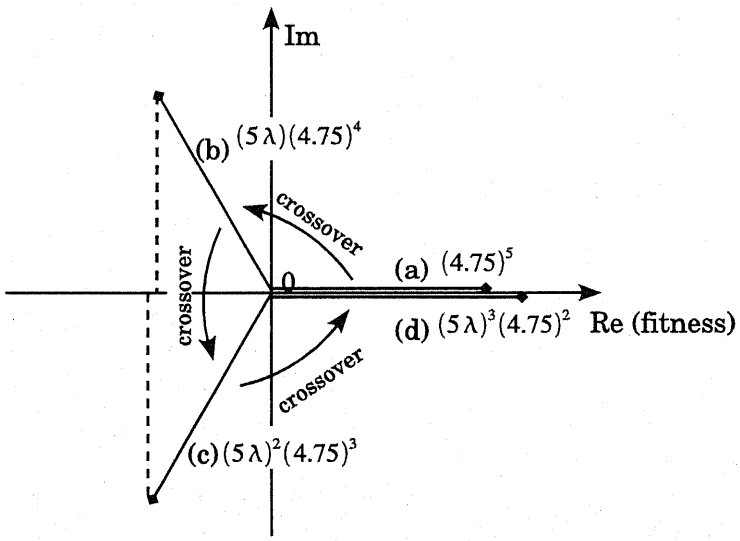


図 3.15: DMAX 問題において, GP の交叉を用いた場合の適合度変化を表した図. 図は複素平面である.

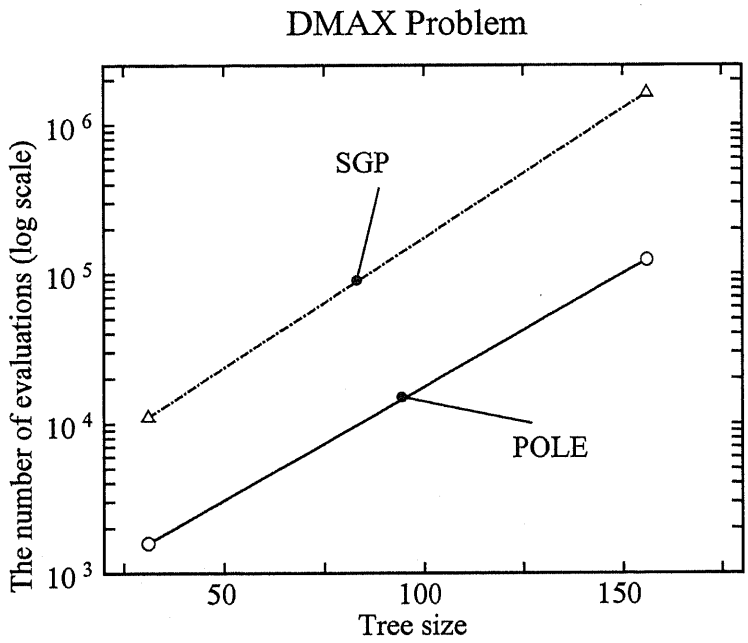


図 3.16: DMAX 問題における，各アルゴリズムの木のサイズに対する平均適合度評価回数.

図 3.15 は，交叉を用いた場合の局所解 (a) から (d) への変化の様子を，複素平面で表した図である．一度交叉を適用すると，偏角が 120° 増加する．そのため，局所解 (a) に対して 3 回交叉を適用することで最適解 (d) を獲得することが可能である．この図から，一度局所解にとらわれると，最適解を獲得することが非常に難しくなることが分かる．

Univariate Model と Adjacent Model は DMAX 問題の依存関係を推定していない．そのため，部分構造が破壊され，$D_P = 4$ において最適解を獲得できていない．POLE はベイジアンネットワーク推定によって，DMAX 問題の依存関係を推定しているため，最も少ない適合度評価回数で最適解を獲得している．

POLE によって推定された依存関係を図 3.17 に示す．図 3.17 は，ある任意の試行の 17 世代目のベイジアンネットワークを示したものである ($D_P = 4, M = 4000$)．この図から，POLE は終端記号間の依存関係を推定していることが分かる．この問題では，主な依存関係は終端記号間にある．DMAX 問題には 5λ と 5×0.95 という 2 つのビルディングブロックがあり，λ と 0.95 が混じり合った構造は許容されない．また，DMAX 問題にはもう一つの依存関係がある．図 3.13 で説明したように，単にランダムに 5λ と 5×0.95 を組み合わせるだけでは最適解を獲得することが出来ない．図 3.17 で分かるように，部分構造間の依存関係も推定されている．このため，POLE は DMAX 問題に非常に有効であることが分かる．

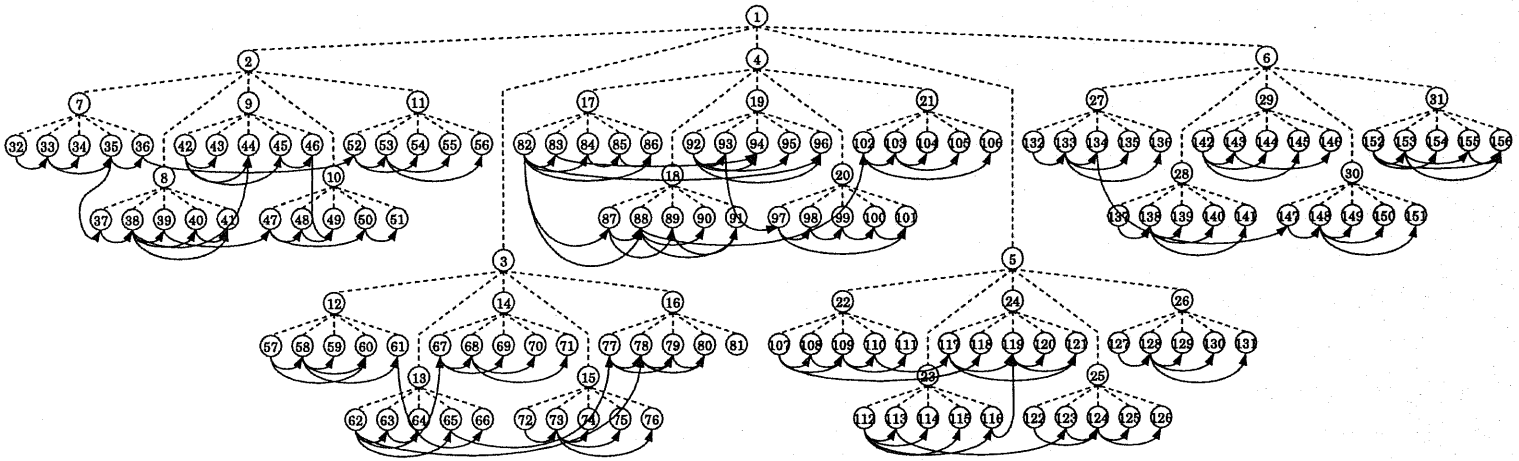


図 3.17: DMAX 問題（深さ 4）におけるベイジアンネットワークの様子．最適解が獲得された試行における，エッジの多い世代のネットワークを表したものである．

表 3.4: DMAX 問題における適合度評価回数．括弧内の数字は標準偏差を表す．

		$D_P = 3$	$D_P = 4$
POLE	Average	1587	124531
	Stdev	(139)	(23403)
Univariate	Average	1409	–
	Stdev	(107)	–
Adjacent	Average	1440	–
	Stdev	(119)	–
SGP	Average	11105	1632159
	Stdev	(6958)	(645696)

表 3.5: t 検定の結果．各値は P 値を表す．

t-value	$D_P = 3$	$D_P = 4$
POLE vs Univariate	5.10×10^{-3}	–
POLE vs Adjacent	2.05×10^{-2}	–
POLE vs SGP	1.92×10^{-3}	4.14×10^{-5}

3.6.2.3 シンボルに対する冗長性

POLE の冗長性を調べるために DMAX 問題において，冗長シンボルを加えた状態で実験を行った．ここでは，式 3.24 と 3.25 で表される二種類の設定で行った．

$$\begin{aligned} \mathfrak{F} &= \{add_m, add_m^*, multiply_m, multiply_m^*\} \\ \mathfrak{T} &= \{\lambda, 0.95\} \end{aligned} \tag{3.24}$$

$$\begin{aligned} \mathfrak{F} &= \{add_m, multiply_m\} \\ \mathfrak{T} &= \{\lambda, \lambda^*, 0.95, 0.95^*\} \end{aligned} \tag{3.25}$$

式 3.24 では冗長な add_m^* と $multiply_m^*$ が加えられている．この二つの関数の機能は add_m と $multiply_m$ と完全に同一であるものの，パラメータ推定の段階では別のシンボルとして扱われる．二つめの式 3.25 の設定では，同様に，冗長な二つの終端記号が加えられている．

表 3.6 は冗長シンボルを用いた場合の，適合度評価回数である．この表で"No addition"の項は表 3.4 の POLE の値と同一である．"Addition of extra functions" は，冗長な関数ノードを加えた場合であるが，値から分かるように，評価回数は少ししか増加していない．一方，"Addition of extra terminals" は冗長な終端ノードを加えた場合である．この

表 3.6: DMAX 問題において，冗長シンボルを用いた場合の適合度評価回数.

		$D_P = 3$	$D_P = 4$
No addition	Average	1587	124531
(Table 3.4 POLE)	Stdev	(139)	(23403)
Addition of	Average	1806	130916
extra functions	Stdev	(162)	(19527)
Addition of	Average	2033	–
extra terminals	Stdev	(230)	–

場合，$D_P = 4$ において，最適解を獲得できていない（$M = 25000$ においても）．この設定において，推定されたネットワーク構造を観測したが，ほとんどエッジが生成されていなかった．前節で述べたように，DMAX 問題は，終端記号間に強い相関関係が存在する．冗長な終端記号を追加した場合，二つの終端記号 λ と λ^* は，その意味は同じであるものの，分布の上では異なるシンボルとして扱われる．そのため，$\lambda + \lambda + \lambda^* + \lambda^* + \lambda$ のような，二つのシンボルが混合された部分構造が生じるため，依存関係の推定が難しくなっている．

3.6.2.4 選択率の効果

本論文では，選択率を $P_s = 0.1$ として実験を行った．POLE は Truncate Selection を用いているため，上位 $M \times 0.1$ 個体が，モデル推定に用いられることになる．本節では，P_s が探索性能にどのような影響を与えるかを調べるために，他の全てのパラメータをそのままにし，P_s のみを $P_s = 0.05, 0.1, 0.2, 0.5$ と変化させて実験を行った．問題の深さ $D_P = 4$，集団数は $M = 6300$ とし，各世代における探索成功確率を観測した．

図 3.18 は各 P_s における，探索成功確率の遷移である．この図からわかるように，小さい P_s の場合の方が，探索が速いことが分かる．しかし，$P_s = 0.05$ では，探索成功確率が 1 となっておらず，探索途中で局所解に陥っている．これから，淘汰圧が強すぎる場合，探索の収束が速すぎ，最適解が獲得されない場合があることを示している．一方で，$P_s = 0.5$ の場合においては，一度も最適解を獲得していない．これは，弱すぎる淘汰圧の場合，選択された優良解は多くのノイズを含む．このノイズのため，POLE は依存関係を推定することが出来なかったと推定される．

3.6.3 Royal Tree 問題

3.6.3.1 問題設定

提案手法の有効性を示すため，Royal Tree 問題 [Punch 98] へも適用した．Royal Tree 問題は前述の最大値問題のように，GP の探索メカニズムを調べるために考案された問題である．Royal Tree 問題は，GA で有名な Royal Road 関数 [Mitchell 92] を木構造に拡張

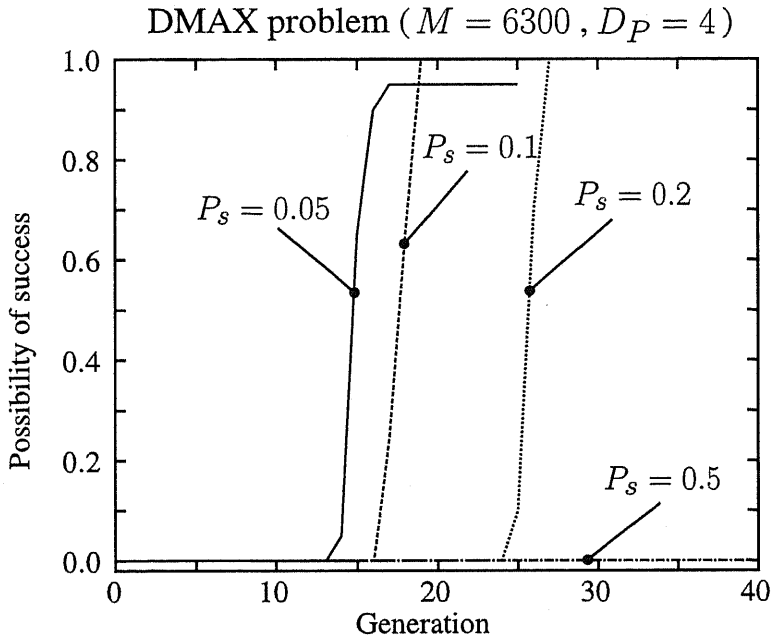


図 3.18: $P_s = 0.05, 0.1, 0.2, 0.5$ の各値における，DMAX 問題（$D_P = 4$）の探索成功確率．P_s 以外のパラメータは同一である．

した問題である．Royal Tree 問題では，小さな部分構造を次々に組み合わせることで，より大きな部分構造を獲得する．

Royal Tree 問題では関数ノードとして $\mathfrak{F} = \{A, B, C, ...\}$ のように幾つかのアルファベットで定義される関数ノード，及び終端ノード $\mathfrak{T} = \{x, y, ...\}$ を定義する．その上で Perfect Tree と呼ばれる状態を定義する．Perfect Tree とは，自分の子ノードが常に自分のアルファベットの一つ前のアルファベットの関数ノードである状態をさす．但し，関数 A の子ノードは終端ノード x である状態である（図 3.19 の（a）（b）（g））．

Royal Tree における木（部分木）のスコアは，ルートのスコアである．それぞれの関数ノードは自分の子ノードのスコアに重みをかけ，それらを足す．もし子ノードをルートとする部分木が Perfect Tree であるならば，子ノードのスコアに *Full Bonus* をかけた値を足す．もし子ノードが正しいノードであるものの，その部分木が Perfect Tree で無い場合 *Partial Bonus* をかけた値を足す．子ノードが間違ったノードである場合には *Penalty* を掛けた値を足す．さらに自分をルートとする部分木が Perfect Tree であるならば，スコアに *Complete Bonus* をかける．通常 *Full Bonus* = 2, *Partial Bonus* = 1, *Penalty* = 1/3, *Complete Bonus* = 2 という値が用いられる．

もとの Royal Tree 問題は変数 x の場合のみにスコアを与えている．この問題の作者らは，他の変数（y など）の場合にもある程度のスコアを与えることで，ある種の騙し要素を加えることが出来ると述べている．本実験では，$x = 1$, $y = 0.95$ とすることで，騙し要素を加えた Royal Tree 問題を用いる．y の場合にも同様に Perfect Tree を定義している（図 3.19（c））．騙し要素を含む場合のスコアの例を図 3.19（a）～（g）に示す（レベ

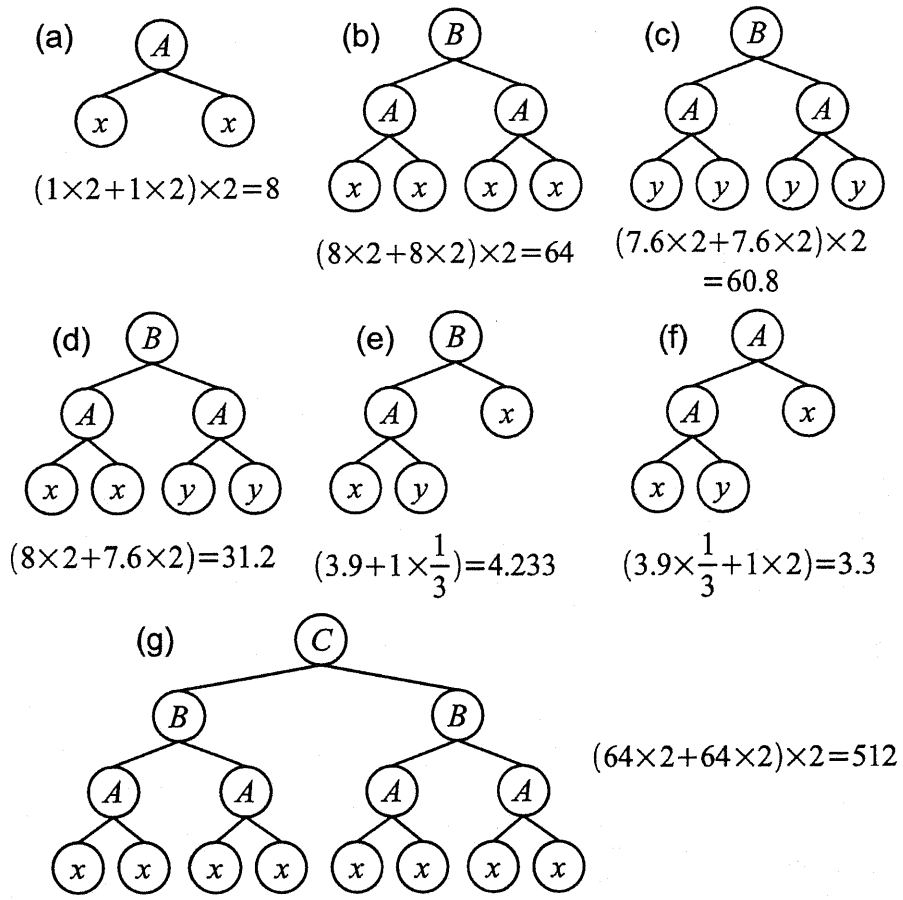


図 3.19: 本章で用いる Royal Tree における木構造のスコアの例.

ル C までの木).

もとの Royal Tree 問題では，問題をより難しくするために，関数 A は 1 引数，関数 B は 2 引数，関数 C は 3 引数のように，設定されているが，本実験で用いる Royal Tree は騙し要素を有し，十分に難しいのでこの要素は省いている．Royal Tree 問題の本質は，小さな部分構造を組み合わせることで，より大きな部分構造が出来るという点にある．そのため，この変更は Royal Tree 問題の本質を失うものではない．

Royal Tree 問題のレベル F の問題に対して適用した．レベル F の場合の最適解は $2^{18} = 262144$ である．

実験で用いたシンボルは以下のようなシンボルである．関数ノードとしてさらに，選択ノード L を用いている．

$$\begin{aligned} \mathfrak{F} &= \{A, B, C, D, E, F\} \\ \mathfrak{T} &= \{x, y\} \end{aligned} \tag{3.26}$$

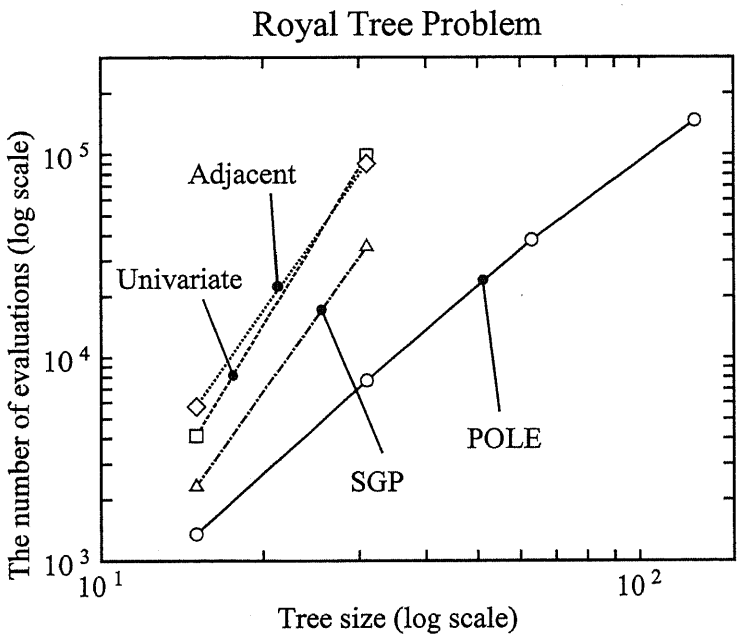


図 3.20: Royal Tree 問題における問題サイズと適合度評価回数の関係.

3.6.3.2 結果と考察

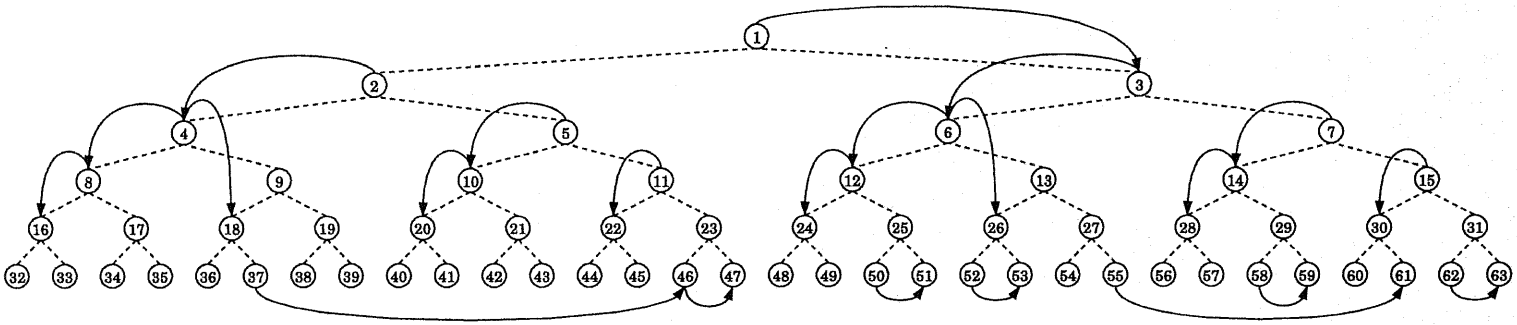
表 3.7 は，20 試行での，平均適合度評価回数とその標準偏差を示したものであり，図 3.20 はそれをグラフ化した図である．この図では，$D_P = 6, 7$ では $M = 25000$ でも，確率 1 で最適解を獲得することが出来ていないため，SGP，Univariate Model，Adjacent Model の結果は表示していない．図 3.20 から分かるように，POLE は最も少ない適合度評価回数で最適解を獲得している．$D_P = 6, 7$ では POLE のみが最適解を確実に獲得している．

この問題には 2 種類の依存関係が存在する．一つは親子間の関係であり（関数 E の子ノードは関数 D など），もう一つは終端記号間の関係である（終端 x の隣は終端 x など）．Univariate Model は上記のどちらの依存関係も考慮していないため，図 3.19 (d) に代表されるような局所解に捕われることが非常に多い．Adjacent Model は親子間の関係のみを考慮しているため，同様な局所解に捕われ，最適解を獲得できていない．

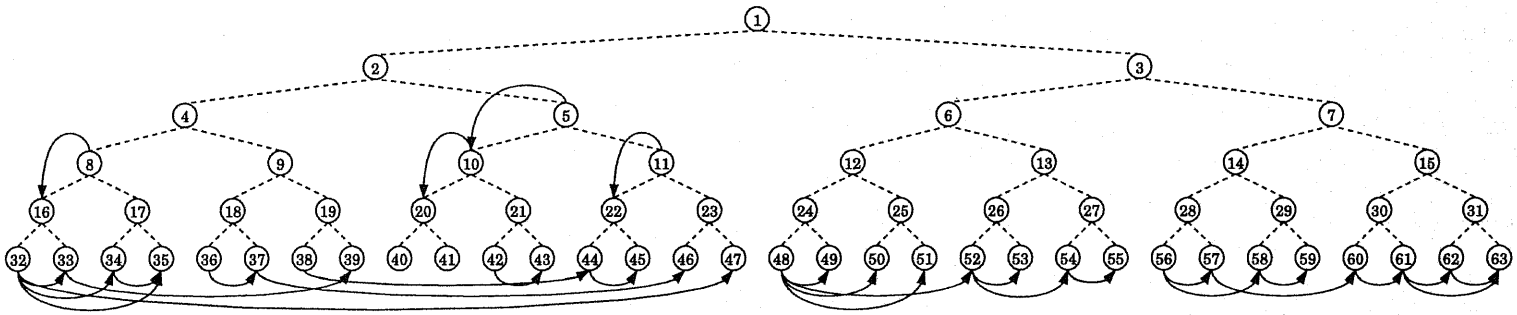
表 3.8 は t 検定の結果である．この表は，POLE とそのほかの手法の評価回数の平均に対しての t 検定の P 値を示している．この表から分かるように，優位水準 1%において，POLE とその他の手法の平均の差は統計的に優位であると言える．

SGP では，最適解を獲得する試行は多くあるものの，SGP はしばしば，図 3.19 (c) のような局所解に捕われている．DMAX 問題の場合と同様に，中間構造の適合度が低いため，SGP では一度局所解を獲得すると，その局所解から抜け出すのは非常に難しい．

POLE は優良解より柔軟にネットワークを構築するため，Royal Tree 問題における依存関係の推定に成功している．POLE では，遥かに少ない評価回数で最適解を獲得して



(a) generation = 2



(b) generation = 10

図 3.21: Royal Tree 問題におけるベイジアンネットワークの様子.

表 3.7: Royal Tree 問題における適合度評価回数．括弧内の数字は標準偏差を表す．

		$D_P = 4$	$D_P = 5$	$D_P = 6$	$D_P = 7$
POLE	Average	1357	7711	38039	147030
	Stdev	(339)	(1291)	(5847)	(13191)
Univariate	Average	4135	98837	–	–
	Stdev	(864)	(29131)	–	–
Adjacent	Average	5752	90080	–	–
	Stdev	(1791)	(20016)	–	–
SGP	Average	2331	35238	–	–
	Stdev	(989)	(16234)	–	–

表 3.8: t 検定の結果．各値は P 値を表す．

t-value	$D_P = 4$	$D_P = 5$
POLE vs Univariate	7.89×10^{-7}	3.83×10^{-6}
POLE vs Adjacent	2.20×10^{-5}	3.62×10^{-7}
POLE vs SGP	1.32×10^{-2}	4.46×10^{-4}

いる．POLE は Royal Tree 問題に非常に有効であると考えられる．

Royal Tree 問題において，POLE が推定したネットワーク構造を見てみる．図 3.21 (a) と (b) は，POLE の探索中の，世代 2 と世代 10 におけるネットワークの様子を図示したものである．これは任意に選択された試行である（$M = 4000$，$D_P = 6$．紙面のスペースの都合より深さは 6 である）．図 3.21 (a) から，探索の序盤には，POLE は親子関係の依存関係を中心に推定していることが分かる．Royal Tree 問題においては，関数 E の子が関数 D のように，親子間に強い関係がある．POLE は，この依存関係を推定していると考えられる．図 3.21 (b) は探索終盤の図であるが，POLE は終端ノード間の依存関係を中心に推定していることが分かる．Royal Tree 問題には，終端 y で構成された局所解と，終端 x で構成された最適解がある．そのため，関数ノードが決定された後は，終端ノード間に非常に強い依存関係が存在することとなる．

3.6.3.3 選択率の効果

Royal Tree 問題では，$P_s = 0.2$ を用いた．選択率 P_s が POLE の Royal Tree 問題の探索効率に与える影響を調べるために，P_s 以外のパラメータはそのままで，$P_s = 0.05, 0.1, 0.2, 0.5$ として実験を行った．Royal Tree 問題の深さは $D_P = 7$ とし，集団数は $M = 4000$ とした．20 回試行した上で，探索成功確率の遷移を各 P_s で観測した．

図 3.22 は各 P_s での探索成功確率の遷移を図示したものである．Royal Tree 問題では，P_s が探索性能に及ぼす影響は，DMAX 問題の場合より小さいことが分かる．前述した様

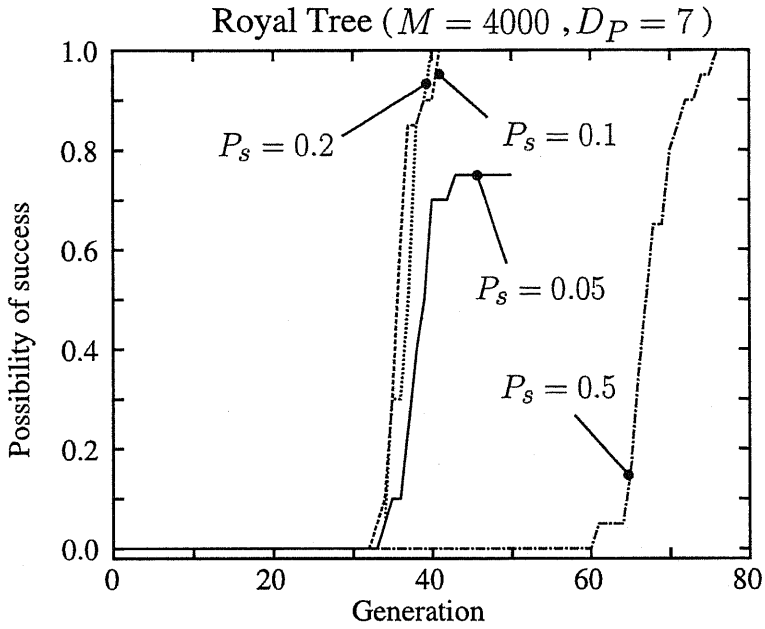


図 3.22: $P_s = 0.05, 0.1, 0.2, 0.5$ の各値における，Royal Tree 問題（$D_P = 7$）の探索成功確率．P_s 以外のパラメータは同一である．

に，Royal Tree 問題のノード間の依存関係は，DMAX 問題の場合より複雑である．POLE が Royal Tree 問題の最適解を獲得するためには，探索の序盤で関数ノードを推定し，探索の終盤で終端ノードを推定する必要がある．このため，Royal Tree 問題ではより多くの適合度評価が必要であり，これが，DMAX 問題における探索との違いになっていると考えられる．

3.6.3.4 通常の構文木 VS 拡張構文木

3.4 節で，拡張構文木を用いることで，確率モデルが適用しやすいと述べた．この仮説を裏付けるために，確率モデルはそのままにして，拡張構文木を通常の構文木に変えたアルゴリズムを実装し，拡張構文木の有効性を調べた．確率モデルとしては，ベイジアンネットワーク，Univariate，Adjacent の三つを用いた（通常の構文木とこれらのモデルを組み合わせたアルゴリズムは，表 3.9 では Bayesnet + SPT，Univariate + SPT，Adjacent + SPT と表記している）．前述した様に，通常の構文木を用いた場合，生成されたプログラムが文法的に正しい必要がある．ここでは，空欄ノードの部分（図 3.4 (a) の点線の円）を，イントロンで満たすことで，全てのプログラムが文法的に正しいことを保証した．

表 3.9 は，六つのアルゴリズムにおける平均適合度評価回数と，標準偏差を示したものである．図 3.23 はこの表をグラフ化したものである．この表と図で Bayesnet + EPT，Univariate + EPT，Adjacent + EPT と表記されているアルゴリズムは，3 章を通して POLE，Univariate，Adjacent と記述されているものと同一である．そのため，これらの

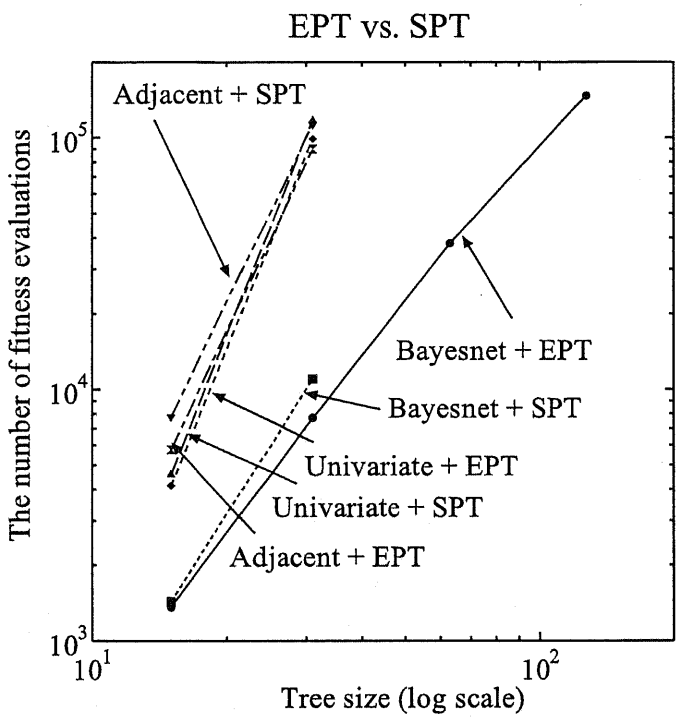


図 3.23: 二つの染色体表現（通常の構文木と拡張構文木）での問題サイズと適合度評価回数の関係．EPT は拡張構文木（Expanded Parse Tree）を表し，SPT は通常の構文木（Standard Parse Tree）を表す．

値は表 3.7 からコピーしたものである．表 3.10 は，t 検定を施した結果である．

ベイジアンネットワークの場合，$D_P = 5, 6, 7$ の場合において，通常の構文木を用いるより拡張構文木を用いた方がよい結果を得られている．$D_P = 5$ では，拡張構文木を用いたアルゴリズムの方が適合度評価回数が少ない（両者の違いは統計的に優位である）．$D_P = 6, 7$ では，拡張構文木を用いた場合は最適解が獲得されているのに対して，通常の構文木を用いた場合は，最適解が獲得されていない．木構造の深さが深くなるにつれ，木構造はより多くのノイズを含む．そのため，通常の構文木を用いた場合，ネットワークが正しく推定されず，最適解の獲得に失敗している．Univariate モデルの場合，ベイジアンネットワークの場合とは異なり，二つの染色体を用いた場合で，大きな差は見られない．Univariate モデルの場合，ネットワーク学習の必要がないため，ノイズの影響を受けにくいためと考えられる．Adjacent モデルでは，拡張構文木が通常の構文木よりよい結果を示している．

これらの結果から，拡張構文木は，特にネットワーク学習を行う場合に有効であることが分かる．拡張構文木を用いた場合，ノードの出現位置が固定される（3.4 節の最後を参照）．そのため，CPT サイズが小さくなるため，ネットワーク構築の際に有利に働く．また，出現位置が固定されるため，ノイズが軽減される．この効果も，ネットワーク学習では効果があると考えられる．

表 3.9: 二つの染色体表現（通常の構文木と拡張構文木）での，適合度評価回数．EPT と SPT はここでは，拡張構文木（Expanded Parse Tree）と通常の構文木（Standard Parse Tree）を表す．

		$D_P = 4$	$D_P = 5$	$D_P = 6$	$D_P = 7$
Bayesnet + EPT	Average	1357	7711	38039	147030
(Table 3.7 POLE)	Stdev	(339)	(1291)	(5847)	(13191)
Bayesnet + SPT	Average	1429	11003	–	–
	Stdev	(362)	(2554)	–	–
Univariate + EPT	Average	4135	98837	–	–
(Table 3.7 Univariate)	Stdev	(864)	(29131)	–	–
Univariate + SPT	Average	4584	117432	–	–
	Stdev	(1338)	(38959)	–	–
Adjacent + EPT	Average	5752	90080	–	–
(Table 3.7 Adjacent)	Stdev	(1791)	(20016)	–	–
Adjacent + SPT	Average	7765	112494	–	–
	Stdev	(2220)	(34258)	–	–

表 3.10: t 検定を行った結果．各値は，通常の構文木と拡張構文木の平均の間の P 値表す．

	$D_P = 4$	$D_P = 5$
Bayesnet EPT vs SPT	6.51×10^{-1}	2.90×10^{-3}
Univariate EPT vs SPT	3.86×10^{-1}	2.44×10^{-1}
Adjacent EPT vs SPT	3.92×10^{-2}	8.28×10^{-2}

3.7 考察

本論文では，ベイジアンネットワーク推定によってノード間の依存関係を推定するアルゴリズムPOLEを提案し，探索性能を他の手法と比較した．Univariateモデルと Adjacentモデルは固定構造を用い，優良解から構造を学習しない．そのため，これらの手法では部分構造が破壊される．本論文では，POLE，Adjacentモデル，Univariateモデルを最大値（MAX）問題（依存関係なし），DMAX問題（終端ノード間の依存関係あり），Royal Tree問題（関数ノード間，終端ノード間の依存関係あり）の三つのベンチマーク問題に適用した．Univariateモデルは，各ノードを独立と仮定しているため，依存関係のあるDMAX問題とRoyal Tree問題に関してはほとんど解くことが出来ていない．Adjacentモデルは，Royal Tree問題に関しては，Univariateモデルより若干良い性能を示したものの，ベイジアンネットワークを用いたPOLEと比較すると非常に劣る探索性能となっている．POLEは優良解よりノード間の依存関係を推定するため，他のモデルを用いた場合より遥に少ない適合度評価回数で最適解を獲得している．

４つの手法で，必要となる計算量について考察する．POLEはベイジアンネットワークの構造学習を伴うため，UnivariateやAdjacentモデルと比較すると多くの計算量が必要である．しかし，POLEでは遠いノード間の依存関係を考慮しないなどの工夫をすることで，構造学習に必要な計算量を大幅に減らしている．また，通常のEAで扱う問題では，適合度評価に最も多くの計算量が必要である．この点から考えても，POLEは他の手法より大幅に少ない適合度評価回数で解を獲得しているため，有効な手法であると考えられる．

本論文では，グラフモデルとしてベイジアンネットワークを用いた．ベイジアンネットワークを用いた理由としては，サンプリングが速い点や，構造学習アルゴリズムが確立している点が大きな理由である．しかし，純粋に探索性能のみを追求した場合，より進んだモデルを用いることも検討の余地がある．例えば，GA-EDAで用いられている混合ベイジアンネットワーク [Peña 05] やマルコフネットワーク [Santana 05] などである．これらのネットワークモデルは，ほとんどそのままPOLEに導入することが可能である．GA-EDAでは，これらのネットワークモデルを用いた手法は，単純なベイジアンネットワークを用いた場合より高い探索性能を示すことが，上記の論文で報告されている．

本論文では，POLEの有効性を主に探索性能の観点から見た．しかし，適用する問題の中には，推定されたネットワーク構造及びCPT自体が重要となる問題が存在する可能性がある．このような問題に置いては，POLEは単なる探索アルゴリズムではなく，データマイニングを行うアルゴリズムになる可能性がある．

3.8 おわりに

本章では，ノード間の依存関係を推定するためにベイジアンネットワークを用いたプログラム進化手法POLEを提案した．GPにおいては，可能なシンボルの数が増大し，依存関係の推定が困難になるという問題点があるが，提案手法では拡張構文木を用いその問

題点を解決している．提案手法の有効性を示すために，既存の確率モデルとの比較実験を行った．ベイジアンネットワークを確率モデルとして用いたPOLEは，DMAX問題及びRoyal Tree問題において，従来手法と比較して，大きく上回る探索性能を示した．このように，ベイジアンネットワークを用いた提案手法は騙し要素の強い問題に対して，その他のモデルを用いたアルゴリズムと比較して非常に有効であることが分かった．GPは非常に多くの分野に対して適用されているが，POLEの実問題への適用などはこれからの課題である．

第4章　潜在アノテーション確率文法に基づく手法

4.1　はじめに

本章では，隠れアノテーション確率文脈自由文法（PCFG-LA:Probabilistic Context Free Grammar with Latent Annotations）[Matsuzaki 05] を用いた新しい進化計算アルゴリズム PAGE（Programming with Annotated Grammar Estimation）を提案する [長谷川 08]. PAGE は二つの手法に分かれている．EM アルゴリズムをベースとした PAGE-EM と変分ベイズ法をベースとした PAGE-VB である．

木構造を用いる EDA（以下 GP-EDA）は，大きく分けて二つの手法に分類できる（2.4.2 節）：プロトタイプ木に基づく手法と確率文脈自由文法（PCFG：Probabilistic Context Free Grammar）に基づく手法である．プロトタイプ木に基づく手法では，可変長である木構造を固定長の構造（通常は配列）に変換した上で，ベイジアンネットワークなどの確率モデルを適用する．一方，PCFG に基づく手法では，関数やプログラムを文脈自由文法（CFG：Context Free Grammar）で表し，その適用確率を学習する．プロトタイプ木を用いた手法には，GA-EDA の手法の適用が可能であるという利点がある．しかし，プロトタイプ木は固定長であり，完全 a_{max} 分木（a_{max} は関数の最大引数）の構造に限定されるため，完全木では a_{max} が大きくなるにつれノード数が指数的に増加する．ノード間の依存関係の推定（構造学習）に多くの計算量が必要となる上，適合度に影響を及ぼさないイントロン部分が多くなり，探索に不利である．また，位置依存の手法であるため，GP が想定する位置に依存しない部分構造の推定には不適である．一方，PCFG を用いた場合，構造上の制限がなく，位置に依存しない PCFG を考えた場合は位置非依存の部分構造の推定が可能となる．このような利点から，GP-EDA においては PCFG を確率モデルとして採用した手法（ii）が数多く提案されている．

PCFG においては文脈依存性を考慮しない．基本的な PCFG に基づく GP-EDA では，単純に各生成規則の適用回数を数え，推定したパラメータに基づき，新しい個体を生成する．このような手法はパラメータ推定が容易であるという利点があるものの，GP においてはノード間の依存関係が存在するため，文脈非依存という仮定は強すぎる場合が多い．PCFG の研究では，種々のアノテーションを用いることで，文脈独立性という仮定を緩和する手法が考案されている．アノテーションとして用いられる要素としては，親ノードや兄弟ノードが挙げられる．既存の GP-EDA においても，PEEL（Program Evolution with Explicit Learning）や Grammar Transformation in an EDA（2.4.2.9 節参照）などでは，深さを考慮した PCFG が用いられている．しかし，深さをアノテーションとして

用いた場合，同じ深さでは同じ確率分布が適用されるため，位置依存である上に，解が基本的に左右対称な場合にしか用いることが出来ないという問題点がある．

自然言語処理（NLP：Natural Language Processing）においてはPCFG-LA（PCFG with Latent Annotations）[Matsuzaki 05]と呼ばれる手法が提案されている．この手法では，アノテーションは潜在変数であり観測されない．PCFG-LAでは，アノテーション付きの生成規則の確率はEMアルゴリズムによって推定される．PCFG-LAは，深さなどのヒューリスティクスに基づくアノテーションと比較して柔軟なモデルであり，進化計算のように様々な問題への適用を前提としたアルゴリズムにおいては，非常に適していると考えられる．そこで本論文では，PCFG-LAを基本モデルとして用い，深さや親ノードなどのヒューリスティクスによって決められたアノテーションモデルにおける問題点を解決したアルゴリズムPAGEを提案する．

本章は以下のように構成される．4.2節では，PCFGについて説明し，PCFGとGPの関係について述べる．PAGEはモデル学習として，EMアルゴリズムと変分ベイズ法を用いる．4.3節において，両推定手法について簡単に述べる．また，PAGEは基本文法としてPCFG-LAを用いているため，4.4節ではPCFG-LAの詳細の説明などを行う．

さらにPAGEにおけるPCFG-LAの学習については，4.5節で述べる．ここでは，EMアルゴリズムによる学習と変分ベイズ法による学習について述べる．4.6節では，推定された分布からのサンプリング方法について述べ，4.7節においてPAGE-VBのアノテーション推定方法について説明する．4.8節では提案手法PAGE-EM及びPAGE-VBの説明をする．提案手法の有効性を示すために，計算機シミュレーションを4.9節で行う．提案手法の有効性を示すために2つの問題（Royal Tree問題4.9.1節，騙し最大値（DMAX：Deceptive MAX）問題4.9.2節）に適用した．得られた結果に対する考察を4.10節で行う．4.11節ではこの章のまとめを行う．

4.2 確率文脈自由文法とGP

本章で提案するアルゴリズムPAGEは，PCFGに基づいており，個体表現として導出木を用いている．本節では，PCFGの説明を行い，GPとPCFGの関連について記述する．本節は文献[北 99, 伊庭 01]を参考にしている．

CFGは，$G = \{\mathcal{N}, \mathcal{T}, \mathcal{R}, \mathcal{B}\}$の4変数によって表現される．ここで，それぞれの記号の意味は以下の通りである．

- $\mathcal{N}$：非終端記号（Non terminal symbol）の有限集合
- $\mathcal{T}$：終端記号（Terminal symbol）の有限集合
- $\mathcal{R}$：生成規則（Production rule）の有限集合
- $\mathcal{B}$：開始記号（Start symbol）[1]

[1]NLP（Natural Language Processing）では通常開式号としてSを用いるが，PAGEで用いた非終端記号のSと区別をつけやすくするため，$\mathcal{B}$を持ちている．

ただし，GP における終端，非終端記号とは意味が異なる点に注意が必要である．生成規則は

$$A \to \alpha \ (A \in \mathcal{N}, \alpha \in (\mathcal{N} \cup \mathcal{T})^*)$$

で表現される．ここで，$(\mathcal{N} \cup \mathcal{T})^*$ は $(\mathcal{N} \cup \mathcal{T})$ を要素とした全ての語を表す．生成規則 $\mathcal{R}$ を開始記号 $\mathcal{B}$ に適用することで，文法 G は文を生成する．文法 G の文の集合は $L(G)$ で表されるが，$l \in L(G)$ とすると $l \in \mathcal{T}^*$ である．

生成規則を適用することで，文字列中の A が書き換えられる．例えば，$\alpha_1 A \alpha_2 (\alpha_1, \alpha_2 \in (\mathcal{N} \cup \mathcal{T})^*, A \in \mathcal{N})$ に対して上の生成規則を適用した場合，$\alpha_1 \alpha \alpha_2$ となる．$\alpha_1 A \alpha_2$ は $\alpha_1 \alpha \alpha_2$ を導出（derive）するといい，以下のように表現される．

$$\alpha_1 A \alpha_2 \underset{G}{\Rightarrow} \alpha_1 \alpha \alpha_2$$

また，

$$\alpha_1 \underset{G}{\Rightarrow} \alpha_2 \cdots \underset{G}{\Rightarrow} \alpha_n (\alpha_i \in (\mathcal{N} \cup \mathcal{T})^*)$$

であるとき，α_n は α_1 から導出されるいい，$\alpha_1 \overset{*}{\underset{G}{\Rightarrow}} \alpha_n$ と表現される．この導出過程は木構造によって表現され，この木構造は導出木（derivation tree）と呼ばれる．文法 G の導出木は次のように定義される．

1. 導出木のノードは $(\mathcal{N} \cup \mathcal{T})$ の要素である
2. 導出木のルートは $\mathcal{B}$ である
3. 導出木の幹ノードは $\mathcal{N}$ の要素である
4. ラベル $A \in \mathcal{N}$ の子ノードが左から $\alpha_1 \alpha_2 \cdots \alpha_k$ $(\alpha_i \in (\mathcal{N} \cup \mathcal{T}))$ であるならば，生成規則 $A \to \alpha_1 \alpha_2 \cdots \alpha_k$ は $\mathcal{R}$ の要素である

ここで簡単な例を用いる．四則演算と $\sin, \cos, \exp, \log$ で表現される一変数関数を文とする．この文を生成する文法 $G_{regression}$ を考える．この場合，開始記号は $\langle expr \rangle$ であり，非終端記号は

$$\mathcal{N} = \{\langle expr \rangle, \langle op2 \rangle, \langle op1 \rangle, \langle var \rangle, \langle const \rangle\}$$

終端記号は

$$\mathcal{T} = \{+, -, \times, \div, \sin, \cos, \exp, \log, x, C\}$$

である．上記の文を生成する規則は以下で表される．

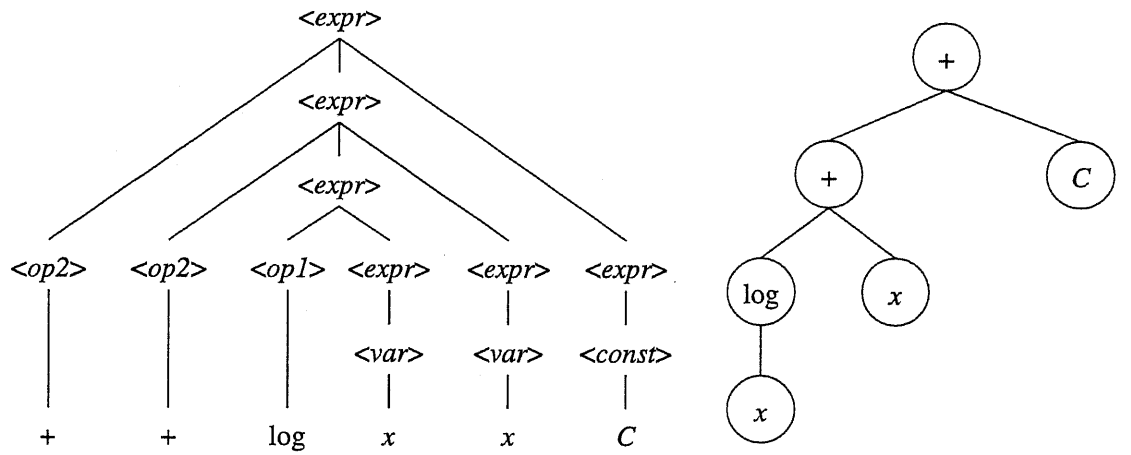
番号	生成規則
0	$\langle expr \rangle \rightarrow \langle op2 \rangle \langle expr \rangle \langle expr \rangle$
1	$\langle expr \rangle \rightarrow \langle op1 \rangle \langle expr \rangle$
2	$\langle expr \rangle \rightarrow \langle var \rangle$
3	$\langle expr \rangle \rightarrow \langle const \rangle$
4	$\langle op2 \rangle \rightarrow +$
5	$\langle op2 \rangle \rightarrow -$
6	$\langle op2 \rangle \rightarrow \times$
7	$\langle op2 \rangle \rightarrow \div$
8	$\langle op1 \rangle \rightarrow \sin$
9	$\langle op1 \rangle \rightarrow \cos$
10	$\langle op1 \rangle \rightarrow \exp$
11	$\langle op1 \rangle \rightarrow \log$
12	$\langle var \rangle \rightarrow x$
13	$\langle const \rangle \rightarrow C$ (定数)

上で定義される文法は，一変数関数生成を行うための言語の定義となる．$G_{regression}$ によって生成される文の集合 $L(G_{regression})$ は，四則演算と $\sin, \cos, \exp, \log$ の一般の一変数関数の集合である．例として，文法 $G_{regression}$ によって以下の文を生成させる．

$$
\begin{aligned}
\langle expr \rangle &\rightarrow \langle op2 \rangle \langle expr \rangle \langle expr \rangle \\
&\rightarrow + \langle expr \rangle \langle expr \rangle \\
&\rightarrow + \langle op2 \rangle \langle expr \rangle \langle expr \rangle \langle expr \rangle \\
&\rightarrow + + \langle expr \rangle \langle expr \rangle \langle expr \rangle \\
&\rightarrow + + \langle op1 \rangle \langle expr \rangle \langle expr \rangle \langle expr \rangle \\
&\rightarrow + + \log \langle expr \rangle \langle expr \rangle \langle expr \rangle \\
&\rightarrow + + \log \langle var \rangle \langle expr \rangle \langle expr \rangle \\
&\rightarrow + + \log x \ \langle expr \rangle \langle expr \rangle \\
&\rightarrow + + \log x \ \langle var \rangle \langle expr \rangle \\
&\rightarrow + + \log x \, x \ \langle expr \rangle \\
&\rightarrow + + \log x \, x \ \langle const \rangle \\
&\rightarrow + + \log x \, x \, C
\end{aligned}
$$

この場合，最終的に導出される文は

$$\log x + x + C$$



(a) 導出木で表した場合　　(b) 木構造 (S式) で表した場合

図 4.1: 関数 $\log y + x + y$ に対する導出木 (a) とそれに対応する木構造 (b).

となっており，一変数関数となっていることが分かる．上記の導出過程は，図4.1 (a) で表現されるような導出木によって表される[2].

GP では，このような関数は通常の木構造によって表現されてきた (図4.1 (b)) ．しかし，この例で分かるように，導出木によっても，同じ関数を表現することが出来る．Whigham によって提案された GP である GGGP (Grammar Guided Genetic Programming) [Whigham 95, Whigham 96] は，この考えに基づき，GP の個体表現として導出木を用いたアルゴリズムである．GGGP においては，導出木から導出された関数やプログラムは P-TYPE として機能する．GGGP における進化は，通常の GP と同様に行われる．すなわち，優良解を選択した後に，その個体群に対して交叉及び突然変異が適用される．通常の GP の場合，任意の部分木の交叉が許されるが，GGGP では，同じ非終端記号の間でしか交叉は行われない．突然変異も同様に，非終端記号の制限の下で行われる．GGGP のこのような制限付きの交叉や突然変異は，型付き GP と同様の制限である．すなわち，GGGP は Bool 型や実数型などが混在した関数の生成などに用いることが出来る．

PCFG[3] は CFG の生成規則に確率 $\beta(A \to \alpha)$ を割り当てることで，確率として扱えるようにしたモデルである．ここで，$\beta(A \to \alpha)$ は生成規則 $A \to \alpha$ の出現確率である．前述の $G_{regression}$ の生成規則を例にすると，PCFG ではそれぞれの生成規則に確率が付加されている (下表) .

[2]本論文を通して，導出木の場合は縁を囲まないノードを用い，S 式などの木構造の場合は縁を囲ったノードを用いている．

[3]SCFG : Stochastic Context Free Grammar とも呼ばれる

番号	生成規則	確率
0	$\langle expr\rangle \rightarrow \langle op2\rangle\,\langle expr\rangle\,\langle expr\rangle$	0.3
1	$\langle expr\rangle \rightarrow \langle op1\rangle\,\langle expr\rangle$	0.1
2	$\langle expr\rangle \rightarrow \langle var\rangle$	0.25
3	$\langle expr\rangle \rightarrow \langle const\rangle$	0.35
4	$\langle op2\rangle \rightarrow +$	0.05
5	$\langle op2\rangle \rightarrow -$	0.15
6	$\langle op2\rangle \rightarrow \times$	0.4
7	$\langle op2\rangle \rightarrow \div$	0.4
8	$\langle op1\rangle \rightarrow \sin$	0.03
9	$\langle op1\rangle \rightarrow \cos$	0.37
10	$\langle op1\rangle \rightarrow \exp$	0.41
11	$\langle op1\rangle \rightarrow \log$	0.19
12	$\langle var\rangle \rightarrow x$	1.0
13	$\langle const\rangle \rightarrow C$ (定数)	1.0

導出木の確率は，用いられている生成規則の確率の積で表される．例えば，図4.1の導出木の尤度（確率）は，以下のように計算される．ここで，$\pi(\langle expr\rangle)$ はルートにおける非終端記号 $\langle expr\rangle$ の確率を表す．

$$\begin{aligned} P(W,T) &= \pi(\langle expr\rangle)\cdot\beta(\langle expr\rangle \rightarrow \langle op2\rangle\,\langle expr\rangle\,\langle expr\rangle)^2\cdot\beta(\langle op2\rangle \rightarrow +)^2 \\ &\times \beta(\langle expr\rangle \rightarrow \langle op1\rangle\,\langle expr\rangle)\cdot\beta(\langle op1\rangle \rightarrow \log) \\ &\times \beta(\langle expr\rangle \rightarrow \langle const\rangle)\cdot\beta(\langle expr\rangle \rightarrow \langle var\rangle)^2\cdot\beta(\langle const\rangle \rightarrow C)\cdot\beta(\langle var\rangle \rightarrow x)^2 \end{aligned}$$

さらに，文 W の確率 $P(W)$ は $P(W,T)$ を周辺化することで，計算される．この式で，$\Phi(W)$ は文 W を導出する全ての導出木の集合を表す．

$$P(W) = \sum_{T\in\Phi(W)} P(W,T)$$

NLPでは，通常適用確率 $\beta(A \rightarrow \alpha)$ の推定は，文からなる学習データセットによって行われる（$\mathbf{W} = \{W_1, W_2, \cdots\}$）．この学習データは文の集合であり，導出過程が明示されていない．全ての導出 $\Phi(W)$ を用いた確率 $P(W)$ の計算は，サイズの大きい文では多くの導出が存在するため，計算量の面から難しい．そのため，EMアルゴリズムである内側・外側アルゴリズム（inside outside algorithm）[Baker 79] と呼ばれる手法によってパラメータを推定する必要がある．内側・外側アルゴリズムでは，動的計画法（DP：Dynamic Programming）を用いることで，現実的な計算量によってパラメータを推定することが可能である．一方，GP-EDAの場合，導出木が既に与えられているため，パラメータ推定は非常に容易であり，EMアルゴリズムなどの推定手法を用いる必要はない．

但し，これは単純なPCFGを用いた場合に言えることで，複雑なPCFGを用いた場合のパラメータ推定には，複雑な推定手法が必要である．

4.3 隠れ変数を含むモデルの推定

提案手法PAGEは隠れ変数を有するPCFG-LAを基本文法モデルとして用いている．そのため，PCFG-LAの推定は，隠れ変数を含むモデルに対して適用可能なEMアルゴリズムや変分ベイズ法を用いている．本節では，これらの推定手法について簡単に述べる．本節の記述は文献[樺島 05, 北 99, 上田 01, Attias 99]を参考にしている．

単純な推定問題では，データがすべて観測される場合を考えるが，より一般の問題を考えた場合，変数は大きく分けて二種類に分類される．一つは観測される顕在変数であり，もう一つは観測されない潜在変数である．例えば，混合正規分布などの混合モデルでは，各データがどの正規分布から生成されたかを示す隠れた変数を考える必要がある．また，欠損したデータを含むようなデータを用いた場合の推定においても，隠れ変数を用いる必要がある．このような場合とは別に，複雑なモデルに対して潜在変数を導入することで，より単純な見通しのよいモデルとなる場合もある．

今，観測データを$\mathcal{D}$とし，潜在変数に対応するデータをZとする．この場合$(\mathcal{D}, Z)$は完全データとなる．完全データの尤度は，パラメータをΘとすると$P(\mathcal{D}, Z; \Theta)$であり，観測データの尤度は$Z$について周辺分布を取ることで，以下の式で表される．

$$P(\mathcal{D}; \Theta) = \sum_{Z} P(\mathcal{D}, Z; \Theta) \tag{4.1}$$

最尤推定（MLE）は，尤度関数を最大化するパラメータを求めるため，

$$\widehat{\Theta} = \underset{\Theta}{\operatorname{argmax}}\, P(\mathcal{D}; \Theta) \tag{4.2}$$

を満たすΘが最尤推定値$\widehat{\Theta}$となる．この種の隠れ変数を含む問題に対して，式4.2をそのまま解くよりも効率的に解く手法がEMアルゴリズムである．

4.3.1 EMアルゴリズム

EMアルゴリズム[Dempster 77]では，Jensenの不等式を用いることで，対数尤度の差の下限を最大化するようにパラメータを推定していく．EMアルゴリズムは繰り返しにより，各ステップでより高い尤度を与えるパラメータを計算する手法である．現在のパラメータセットをΘとし，更新後のパラメータを$\overline{\Theta}$とする．その場合，二つのパラメータにおける対数尤度の差は以下のように計算される．

$$\begin{aligned}
\log P(\mathcal{D};\overline{\Theta}) - \log P(\mathcal{D};\Theta) &= \log \frac{P(\mathcal{D};\overline{\Theta})}{P(\mathcal{D};\Theta)} \\
&= \sum_{Z} P(Z|\mathcal{D};\Theta) \log \frac{P(\mathcal{D};\overline{\Theta})}{P(\mathcal{D};\Theta)} \\
&= \sum_{Z} P(Z|\mathcal{D};\Theta) \log \frac{P(\mathcal{D},Z;\overline{\Theta})}{P(\mathcal{D},Z;\Theta)} \frac{P(Z|\mathcal{D};\Theta)}{P(Z|\mathcal{D};\overline{\Theta})} \\
&\geq \sum_{Z} P(Z|\mathcal{D};\Theta) \log \frac{P(\mathcal{D},Z;\overline{\Theta})}{P(\mathcal{D},Z;\Theta)} \qquad (4.3)
\end{aligned}$$

最後の行の式変形は，Jensen の不等式を用いている．これから，

$$L_B(\overline{\Theta}|\Theta) = \sum_{Z} P(Z|\mathcal{D};\Theta) \log \frac{P(\mathcal{D},Z;\overline{\Theta})}{P(\mathcal{D},Z;\Theta)} \qquad (4.4)$$

が正であるようなパラメータ $\overline{\Theta}$ を計算すれば，

$$\begin{aligned}
\log P(\mathcal{D};\overline{\Theta}) &\geq \log P(\mathcal{D};\Theta) + L_B(\overline{\Theta}|\Theta) \\
&\geq \log P(\mathcal{D};\Theta)
\end{aligned}$$

から，必ず尤度は上昇する．最も尤度を上昇させるためには式 4.4 を最大化させる $\overline{\Theta}$ を求めれば良い．これは式 4.5 で表される $\mathcal{Q}(\overline{\Theta}|\Theta)$ を最大化させることと同値である．

$$\mathcal{Q}(\overline{\Theta}|\Theta) = \sum_{Z} P(Z|\mathcal{D};\Theta) \log P(\mathcal{D},Z;\overline{\Theta}) \qquad (4.5)$$

対数尤度関数が上限値を持つ限り，EM アルゴリズムによって必ず局所最適解に収束する．

更新後のパラメータは，

$$\overline{\Theta} = \underset{\overline{\Theta}}{\operatorname{argmax}} \mathcal{Q}(\overline{\Theta}|\Theta)$$

より求めることが出来る．

4.3.2 変分ベイズ法

変分ベイズ法（variational Bayes）[Attias 99, 上田 01] は，EM アルゴリズムをベイズ推定へと拡張した手法である．変分ベイズ法では，平均場近似で用いられるような分布の分解の仮定を置くことで，解析的に計算できるようにした手法である．最尤推定が点推定値を用いるのに対し，ベイズ推定は分布を用いるため，一般的に汎化誤差が小さい．また，最尤推定では，モデルが複雑になるにつれ尤度が単純に上昇するのに対して，ベイズ推定では最適なモデルにおいて最大の事後確率を与える．

EDA では個体の生成は，個体を X とすると，$P(X|\mathcal{D})$ に基づいて行われる（2.3.4 節）．最尤推定の場合，未知のデータ X の分布は式 4.6 で表される．

$$P(X|\mathcal{D}) = P(X;\widehat{\Theta},\widehat{m}) \tag{4.6}$$

ここで，$\widehat{\Theta}$ は最大対数尤度を与えるパラメータであり，$\widehat{m}$ は最尤モデルである．このように，最尤推定では単一の点推定値を用いることで与えられる．ベイズ推定では，データが観測された後の事後分布を用いて，未知データの分布を計算する．ベイズ推定では，予測事後分布は式4.7で表される．最尤推定が点推定であるのに対し，ベイズ推定はこのようなアンサンブル学習であるため，汎化誤差が小さい．

$$P(X|\mathcal{D}) = \sum_{m\in\mathcal{M}} \int d\Theta P(X|\Theta,m)P(\Theta,m|\mathcal{D}) \tag{4.7}$$

一方，本節で考える隠れ変数を含むモデルにおいては，事後分布 $P(\Theta,m|\mathcal{D})$ は，Z について周辺化することで以下の式で表される．

$$P(\Theta,m|\mathcal{D}) = \sum_{Z} P(\Theta,Z,m|\mathcal{D})$$

これから，隠れ変数を有するモデルにおいて求めるべきなのは，事後分布 $P(\Theta,Z,m|\mathcal{D})$ であることが分かる．しかし，事後分布 $P(\Theta,Z,m|\mathcal{D})$ は下の式で表されるように，積分計算を含む．

$$P(\Theta,Z,m|\mathcal{D}) = \frac{P(\mathcal{D},Z|\Theta,m)P(\Theta|m)P(m)}{\sum_{m\in\mathcal{M}}\sum_{Z}\int d\Theta P(\Theta,Z,m,\mathcal{D})} \tag{4.8}$$

この積分計算は計算することが難しい．変分ベイズ法では，$P(Z|\mathcal{D}), P(\Theta|\mathcal{D}), P(m|\mathcal{D})$ とそれぞれに因数分解された事後分布を考えることで，近似的に事後分布を求める．

ベイズ推定における対数周辺化尤度は，パラメータ空間で積分することにより

$$\begin{aligned}
\log P(\mathcal{D}) &= \log\left\{\sum_{m\in\mathcal{M}}\sum_{Z}\int d\Theta P(\mathcal{D},Z,\Theta,m)\right\} \\
&= \log\left\{\sum_{m\in\mathcal{M}}\sum_{Z}\int d\Theta Q(Z,\Theta,m)\frac{P(\mathcal{D},Z,\Theta,m)}{Q(Z,\Theta,m)}\right\} \\
&\geq \sum_{m\in\mathcal{M}}\sum_{Z}\int d\Theta Q(Z,\Theta,m)\log\frac{P(\mathcal{D},Z,\Theta,m)}{Q(Z,\Theta,m)}
\end{aligned}$$

と計算される（データ $\mathcal{D}$ のみに依存する）．最後の式変形はEMアルゴリズムの場合のように，Jensenの不等式を用いている．この式で，$Q(Z,\Theta,m)$ はテスト分布と呼ばれる関数（分布）である．最後の式の

$$\mathcal{F} \equiv \sum_{m\in\mathcal{M}}\sum_{Z}\int d\Theta Q(Z,\Theta,m)\log\frac{P(\mathcal{D},Z,\Theta,m)}{Q(Z,\Theta,m)} \tag{4.9}$$

は対数尤度 $\log P(\mathcal{D})$ の下限を与える．$\log P(\mathcal{D})$ はデータのみに依存するため，$\mathcal{D}$ が与えられれば定数である．$\mathcal{F}$ を最大化するような $Q(\cdot)$ を推定すると，それは真の事後分布 $P(\cdot|\mathcal{D})$ のもっとも近い近似となる（KL ダイバージェンスが最小である）．そのため，変分ベイズ法の目的は，事後分布の近似 $Q(\cdot)$ を求めることにある．但し，このままでは，解析的に $Q(\cdot)$ を求めることは出来ない．そこで，変分ベイズ法では，以下のような分解の仮定を置いている（式 4.10）．

$$Q(Z,\Theta,m) = Q(Z|m)Q(\Theta|m)Q(m) \tag{4.10}$$

パラメータの数を I 個とし，まとめて $\Theta = \bigcup_{i=1}^{I}\{\theta_i\}$ と記述すると，分解の仮定から

$$Q(\Theta|m) = \prod_{i=1}^{I} Q(\theta_i|m)$$

と表すことが出来る．なお，$Q(\cdot)$ は事後分布であるので，正規化条件を満たす（$\int d\theta_i Q(\theta_i|m) = 1$，$\sum_Z Q(Z|m) = 1$，$\sum_{m\in\mathcal{M}} Q(m) = 1$）以上の仮定を，式 4.4 に代入する．

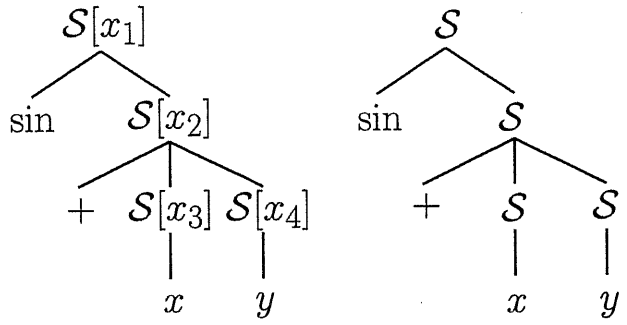
$$\begin{aligned}
\mathcal{F} &= \sum_{m\in\mathcal{M}}\sum_{Z}\int d\Theta Q(Z,\Theta,m)\log\frac{P(\mathcal{D},Z,\Theta,m)}{Q(Z,\Theta,m)} \\
&= \sum_{m\in\mathcal{M}}\sum_{Z}\int d\Theta Q(Z|m)Q(\Theta|m)Q(m)\log\frac{P(\mathcal{D},Z|\Theta,m)P(\Theta|m)P(m)}{Q(Z|m)Q(\Theta|m)Q(m)} \\
&= \sum_{m\in\mathcal{M}} Q(m)\log\frac{P(m)}{Q(m)} + \sum_{m\in\mathcal{M}}\int d\Theta Q(m)Q(\Theta|m)\log\frac{P(\Theta|m)}{Q(\Theta|m)} \\
&\quad + \sum_{m\in\mathcal{M}}\sum_{Z}\int d\Theta Q(Z|m)Q(\Theta|m)Q(m)\log\frac{P(\mathcal{D},Z|\Theta,m)}{Q(Z|m)}
\end{aligned} \tag{4.11}$$

これから，$\mathcal{F}$ は Θ, Z をそれぞれ独立に求めることが出来る．まず，隠れデータ Z の近似事後分布 $Q(Z|m)$ を求める．まず，制約条件付最大化であるため，ラグランジュの未定乗数法から，以下の関数の微分 $\frac{\partial\mathcal{L}(Q(Z|m))}{\partial Q(Z|m)} = 0$ を計算すれば良い（μ はラグランジュ定数）．

$$\mathcal{L}(Q(Z|m)) = \sum_{m\in\mathcal{M}}\sum_{Z}\int d\Theta Q(Z|m)Q(\Theta|m)Q(m)\log\frac{P(\mathcal{D},Z|\Theta,m)}{Q(Z|m)} + \mu\left(1-\sum_{Z}Q(Z|m)\right)$$

一方，$Q(\theta_i|m)$ も同様であるが，式 4.13 に対して，汎関数の微分 $\frac{\delta\mathcal{J}[Q(\theta_i|m)]}{\delta Q(\theta_i|m)} = 0$[4]を行う

[4]
$$\begin{aligned}
\frac{\delta}{\delta\psi(y)}\int dx f(\psi(x)) &= \lim_{\epsilon\to 0}\frac{\int[f(\psi(x)+\epsilon\delta(x-y))-f(\psi(x))]\,dx}{\epsilon} \\
&= \lim_{\epsilon\to 0}\frac{\int[f(\psi(x))+\epsilon\delta(x-y)f'(\psi(x))+O(\epsilon^2)-f(\psi(x))]\,dx}{\epsilon} \\
&= f'(\psi(y))
\end{aligned} \tag{4.12}$$



(a) 完全データの木　　(b) 観測される木

図 4.2: PCFG-LA において，完全データの木（a）と観測される木（b）.

必要がある.

$$\begin{aligned}\mathcal{J}[Q(\theta_i|m)] &= \sum_{m\in\mathcal{M}}\int d\Theta Q(m)Q(\Theta|m)\log\frac{P(\Theta|m)}{Q(\Theta|m)} \\ &\quad+\sum_{m\in\mathcal{M}}\sum_{Z}\int d\Theta Q(Z|m)Q(\Theta|m)Q(m)\log\frac{P(\mathcal{D},Z|\Theta,m)}{Q(Z|m)} \\ &\quad+\nu\left(1-\int d\theta_i Q(\theta_i|m)\right) \end{aligned} \tag{4.13}$$

これらから，事後分布の近似は最終的に以下のように計算される.

$$Q(Z|m)\propto\exp\left\{\int d\Theta Q(\Theta|m)\log P(\mathcal{D},Z|\Theta,m)\right\} \tag{4.14}$$

$$Q(\theta_i|m)\propto P(\theta_i|m)\exp\left\{\sum_{Z}\int d\Theta_{-i}\prod_{j=1,j\neq i}^{I}Q(\theta_j|m)Q(Z|m)\log P(\mathcal{D},Z|\Theta,m)\right\} \tag{4.15}$$

ここで，$\int d\Theta_{-i}$ は $\int\prod_{j=1,j\neq i}^{I}d\theta_j$ の積分を表す．この二つの式は互いに依存しているため，交互にパラメータを求めることで近似的に計算される.

EM アルゴリズムではパラメータを更新したが，変分ベイズ法では，事後分布 $P(\cdot|\mathcal{D})$ の近似を求めている．実際には，事前分布として Dirichlet 分布などのパラメトリックな分布を仮定するため，ハイパーパラメータを更新することに相当する.

4.4 PCFG with Latent Annotations

本節では，PAGE の基本 PCFG モデルとして用いられている PCFG-LA について説明する.

基本的なPCFG[北 99]においては，文脈独立性を仮定している．しかし，GP-EDAの視点からみると，文脈独立性は，ノード間の依存関係について一切考慮しないモデルと捉えられる．GPにおいては部分構造は木構造であるため，ノード間の依存関係は非常に強い．NLPでは，文脈独立性を緩和させる手法として，非終端記号へアノテーションを付加する手法が提案されている．アノテーションは付加的な情報であり，通常親ノード情報などが用いられる．PCFGに基づくGP-EDAにおいてもアノテーションが用いられるが，良く用いられるアノテーションは深さ情報である．深さアノテーションを用いた場合，関数ノードの確率分布は，そのノードが出現する深さに依存する．しかし，深さアノテーションでは，位置依存の構造しか推定できない上，同じ深さでは同じ分布が適用されるため，基本的には左右対称な構造以外の推定には不適である．また，仮に親ノードをアノテーションとして用いたとしても，推定される部分構造は親ノード依存であるため，非常に限定的な構造しか推定出来ない．このような，あらかじめ与えられたアノテーションではなく，アノテーション自体も学習データから推定する手法が，NLPの分野で提案されている[Matsuzaki 05]．文献[Matsuzaki 05]では，PCFG-LAと呼ばれるPCFGが提案されているが，PCFG-LAではアノテーションは潜在変数であり，観測されないと仮定している．PCFG-LAでは，アノテーション数を十分にとることで，深さなどの固定アノテーションの場合より柔軟に文脈を考慮出来る．このPCFG-LAの特徴は，GP-EDAで扱うような複雑な部分構造の推定に非常に適していると考えられる．本論文では，このPCFG-LAを，GP-EDAに適用したアルゴリズムを提案している．PCFG-LAの提案文献[Matsuzaki 05]では，隠れ変数を含むPCFG-LAのパラメータ学習として，EMアルゴリズムが用いられている．提案手法で用いられるPCFG-LAはGP-EDAに特殊化したモデルであるが，基本部分は元のPCFG-LAと同じである．以下では，PCFG-LAの説明を行う．なおPCFG-LAの詳細については，元の文献[Matsuzaki 05]を参照されたい．

PCFG-LAにおいては，全ての非終端記号にはアノテーションがラベル付されている．非終端記号は，アノテーション付表現では$A[x]$のように表現される．Aは非終端記号であり，xはアノテーションを表す（$x \in H$，Hはアノテーションの集合）．図4.2は完全データの木構造（a）と，観測される木構造（b）を表している．

完全データT，Xの尤度は式4.16で表される．

$$P(T_i, X_i|\boldsymbol{\beta}, \boldsymbol{\pi}, h) = \prod_{x \in H} \pi(\mathcal{S}[x])^{\delta(x;T_i,X_i)} \prod_{\mathcal{S}[x] \to \alpha \in \mathcal{R}[H]} \beta(\mathcal{S}[x] \to \alpha)^{c(\mathcal{S}[x] \to \alpha;T_i,X_i)} \quad (4.16)$$

ここで，T_iは導出木，x_i^jは木T_iのj番目の非終端記号のアノテーション，Xは$X_i = \{x_i^1, x_i^2, ...\}$，$\pi(\mathcal{S}[x])$はルートでの$\mathcal{S}[x]$の確率，$\beta(r)$は生成規則$r$の確率，$c(\mathcal{S}[x] \to \alpha; T_i, X_i)$は完全木$T_i, X_i$における生成規則$\mathcal{S}[x] \to \alpha$の出現回数，$\delta(x; T_i, X_i)$は完全木$T_i, X_i$においてルートのアノテーションが$x$であれば1を返し，それ以外では0を返す関数，$h$はアノテーション数を表す．$\mathcal{R}[H]$はアノテーション付生成規則$\mathcal{S}[x] \to \alpha$の集合を表す（式4.21）．パラメータはまとめて，$\boldsymbol{\beta} = \{\beta(\mathcal{S} \to \alpha)|\mathcal{S} \to \alpha \in \mathcal{R}[H]\}$，$\boldsymbol{\pi} = \{\pi(\mathcal{S}[x])|x \in H\}$と表記する．

不完全データの尤度は，式4.16をXについて和をとることで計算される（式4.17）．

$$P(T_i;\beta,\pi,h)=\sum_{X_i}P(T_i,X_i;\beta,\pi,h). \tag{4.17}$$

以上がPCFG-LAモデルの枠組みである．PCFG-LAのモデル学習はパラメータβ,πの推定である．モデル学習の前に，生成規則の形式について考える．NLPにおいては，チョムスキー標準型（CNF：Chomsky Normal Form）を仮定する場合が多い（元のPCFG-LAにおいてもCNFを仮定している）．しかし，GP-EDAにおいては，チョムスキー標準型にする利点があまりなく，また用いないほうが関数を分かりやすく表現可能である．一般的なGPによって表せる全ての関数は以下の生成規則（式4.18）によって表現できる．この表記はグライバッハ標準型（GNF：Greibach Normal Form）の特別な場合である．

$$\mathcal{S}\rightarrow g\mathcal{S}\mathcal{S}\cdots\mathcal{S}. \tag{4.18}$$

ここで，$\mathcal{S}\in\mathcal{N}$，$g\in\mathcal{T}$，$\mathcal{S}\rightarrow g\mathcal{S}...\mathcal{S}\in\mathcal{R}$である（$\mathcal{N}$，$\mathcal{T}$はCFGにおける非終端，終端ノードの集合を表し，$\mathcal{R}$はアノテーションなし生成規則の集合を表す．なお，GPにおける関数，終端ノードの集合は$\mathfrak{F}$及び$\mathfrak{T}$と表記する）．gは$g\in\mathfrak{F}\cup\mathfrak{T}$であり，GPにおいては関数ノード（$+,-,\sin,\cos$など）又は終端ノード（$x,y$など）を表す．PCFGの非終端記号は$\mathcal{S}$一つであっても，生成される文の一般性は失われない（$\mathcal{N}=\{\mathcal{S}\}$）．式4.18の右辺にある$\mathcal{S}$の個数は，$g$の引数の数に等しい．例えば，$g=+$の場合$\mathcal{S}\rightarrow+\mathcal{S}\mathcal{S}$であり，$g=x$の場合$\mathcal{S}\rightarrow x$となる．

アノテーション付の生成規則は式4.19によって表現出来る．

$$\mathcal{S}[x]\rightarrow g\,\mathcal{S}[z_1]\,\mathcal{S}[z_2]\cdots\mathcal{S}[z_\alpha] \tag{4.19}$$

ここで$x,z_m\in H$であり，αはgの引数の数を表す．hをアノテーションの数（Hの要素数）とし，gの引数の数をαとする．この場合，式4.19を表現するのに必要なパラメータ数は$h^{\alpha+1}$となる．アノテーションの数が多くなるにつれ，パラメータ数が指数的に増加し，パラメータ推定の計算量の点や，精度の面で大きな問題となる．そこで，本提案手法では式4.20で表されるような生成規則に限定する．

$$\mathcal{S}[x]\rightarrow g\,\mathcal{S}[y]\,\mathcal{S}[y]\cdots\mathcal{S}[y] \tag{4.20}$$

ここで，$x,y\in H$である．この生成規則では，右辺のアノテーションは全て同じであると仮定している．このような仮定を置くことで，パラメータの数はh^2となる．以下では，この生成規則の集合を$\mathcal{R}[H]$と表す（式4.21）．

$$\mathcal{R}[H]=\{\mathcal{S}[x]\rightarrow g\,\mathcal{S}[y]\,\mathcal{S}[y]...\mathcal{S}[y]|x,y,\in H,g\in\mathcal{T}\} \tag{4.21}$$

また，生成規則の右辺の集合を$\mathcal{R}_r[H]$とする（式4.22）．

$$\mathcal{R}_r[H]=\{\alpha|\mathcal{S}\rightarrow\alpha\in\mathcal{R}[H],x\in H\} \tag{4.22}$$

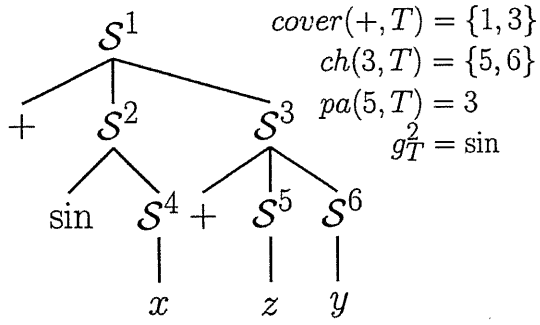


図 4.3: 木構造の例 T と各関数の実際の値．$\mathcal{S}^j$ は j 番目の非終端記号であることを表す．

これ以降の説明及び数式は，生成規則を式 4.20 の形に仮定して記述する．PAGE では上記のような仮定をおいているものの，PCFG-LA としての基本部分は文献 [Matsuzaki 05] と同じである．

4.4.1 前向き・後向き確率

モデル学習の説明の前に，前向き・後ろ向き確率について説明する．PCFG-LA の元の文献では，EM アルゴリズムを計算するために，前向き・後向き確率が用いられている[5]．後向き確率 $b_T^i(x;\boldsymbol{\beta},\boldsymbol{\pi},h)$ は i 番目の非終端記号 $\mathcal{S}[x]$ より下の部分が生成される確率であり，前向き確率 $f_T^i(y;\boldsymbol{\beta},\boldsymbol{\pi},h)$ は i 番目の非終端記号 $\mathcal{S}[y]$ より上の部分が生成される確率である．前向き・後向き確率は再帰的に計算可能である（式 4.23，4.24，4.25）．

$$b_T^i(x;\boldsymbol{\beta},\boldsymbol{\pi},h) = \sum_{y\in H} \beta(\mathcal{S}[x] \to g_T^i\, \mathcal{S}[y]...\mathcal{S}[y]) \prod_{j\in ch(i,T)} b_T^j(y;\boldsymbol{\beta},\boldsymbol{\pi},h) \tag{4.23}$$

$$\begin{aligned} &f_T^i(y;\boldsymbol{\beta},\boldsymbol{\pi},h) \\ &= \sum_{x\in H} f_T^{pa(i,T)}(x;\boldsymbol{\beta},\boldsymbol{\pi},h)\beta(\mathcal{S}[x] \to g_T^{pa(i,T)}\, \mathcal{S}[y]...\mathcal{S}[y]) \\ &\times \prod_{j\in ch(pa(i,T),T),j\neq i} b_T^j(y;\boldsymbol{\beta},\boldsymbol{\pi},h) \quad (i\neq 1) \end{aligned} \tag{4.24}$$

$$f_T^i(y;\boldsymbol{\beta},\boldsymbol{\pi},h) = \pi(\mathcal{S}[y]) \quad (i=1) \tag{4.25}$$

ここで，$ch(i,T)$ は T における i 番目の非終端記号の子ノードインデックスの集合を返す関数であり，$pa(i,T)$ は i 番目の非終端記号の親ノードインデックスを返す関数である．g_T^i は i 番目の非終端記号に接続された終端記号を表す（図 4.3）．

[5]通常，PCFG で用いられる動的計画法の確率は，外側・内側確率と呼ばれるが，本論文では文献 [Matsuzaki 05] の記述に従い，前向き・後向き確率と記述している．

	推定	最適化対象	最良モデル選択
EM アルゴリズム	最尤推定	パラメータ	不可
変分ベイズ法	ベイズ推定	分布	可

表 4.1: EM アルゴリズム対変分ベイズ法.

アルゴリズム	推定手法	最適化対象
PAGE-EM	EM アルゴリズム	パラメータ
PAGE-VB	変分ベイズ法	ハイパーパラメータ

表 4.2: PAGE-EM と PAGE-VB の比較.

前向き・後向き確率を用いることで，不完全データの尤度 $P(T;\boldsymbol{\beta},\boldsymbol{\pi},h)$ は，以下のように表現出来る.

$$P(T;\boldsymbol{\beta},\boldsymbol{\pi},h)=\sum_{x\in H}\pi(\mathcal{S}[x])b_T^1(x;\boldsymbol{\beta},\boldsymbol{\pi},h) \tag{4.26}$$

$$\begin{aligned}&P(T;\boldsymbol{\beta},\boldsymbol{\pi},h)\\&=\sum_{x,y\in H}\beta(\mathcal{S}[x]\rightarrow g\,\mathcal{S}[y]...\mathcal{S}[y])f_T^i(x;\boldsymbol{\beta},\boldsymbol{\pi},h)\prod_{j\in ch(i,T)}b_T^j(y;\boldsymbol{\beta},\boldsymbol{\pi},h)\\&(i\in cover(g,T))\end{aligned} \tag{4.27}$$

ここで，$cover(g,T)$ は，$g\in\mathcal{T}$ が T において接続されている非終端ノードの集合を表す (図 4.3) .

4.5 モデル学習

PCFG-LA は隠れ変数を有するモデルであるため，学習 ($\boldsymbol{\beta},\boldsymbol{\pi}$ の推定) には EM アルゴリズム (4.3.1 節参照) や変分ベイズ法 (4.3.2 節参照) を用いる必要がある (表 4.1) . PCFG-LA の提案論文 [Matsuzaki 05] では EM アルゴリズムが用いられている．しかし，EM アルゴリズムで推定する場合，アノテーション数 h が既知である必要がある．通常，最適化アルゴリズムを適用する場合，問題に対する知識は非常に少なく，h がどれくらいであれば良いかという判断は行いづらい．そのため，優良解 (学習データ) から，アノテーション数自体も推測出来ることが望ましい．EM アルゴリズムは最尤推定であり，点推定であるため，最良モデル選択は出来ない．4.3.2 節で述べたように，このような隠れ変数を含むモデルのモデル選択は，変分ベイズ法によって行われる．本論文では，EM アルゴリズムに基づく PCFG-LA を用いた関数進化アルゴリズムを提案すると同時に，変分ベイズ法を PCFG-LA に適用し，その変分ベイズ法 PCFG-LA を関数進化アルゴリズムに適用した手法の二つを提案する．本論文で提案するアルゴリズムの名称は PAGE (Programming

with Annotated Grammar Estimation）であるが，EMアルゴリズムを用いたPAGEはPAGE-EM，変分ベイズ法を用いたPAGEをPAGE-VBと表記する．単にPAGEと記述した場合は双方のアルゴリズムを指す．

以下，EMアルゴリズム（4.5.1節）と変分ベイズ法によるPCFG-LA（4.5.2節）の学習を示す．

4.5.1 EMアルゴリズムによる学習

ここではEMアルゴリズムによるPCFG-LAの学習を説明する．PCFG-LAの提案論文[Matsuzaki 05]においてもEMアルゴリズムが用いられている．式4.3にあるように，EMアルゴリズムでは，現在のパラメータ$\boldsymbol{\beta},\boldsymbol{\pi}$と更新後のパラメータ$\overline{\boldsymbol{\beta}},\overline{\boldsymbol{\pi}}$における，木$T$の対数尤度の差を考える．対数尤度の差の下限は，Jensenの不等式から以下（式4.29）のように計算される．

$$
\begin{aligned}
&\log P(T;\overline{\boldsymbol{\beta}},\overline{\boldsymbol{\pi}},h) - \log P(T;\boldsymbol{\beta},\boldsymbol{\pi},h) \\
&= \log \frac{P(T;\overline{\boldsymbol{\beta}},\overline{\boldsymbol{\pi}},h)}{P(T;\boldsymbol{\beta},\boldsymbol{\pi},h)} \\
&= \sum_X P(X|T;\boldsymbol{\beta},\boldsymbol{\pi},h) \\
&\times \log \frac{P(T,X;\overline{\boldsymbol{\beta}},\overline{\boldsymbol{\pi}},h)}{P(T,X;\boldsymbol{\beta},\boldsymbol{\pi},h)} \frac{P(X|T;\boldsymbol{\beta},\boldsymbol{\pi},h)}{P(X|T;\overline{\boldsymbol{\beta}},\overline{\boldsymbol{\pi}},h)} \qquad (4.28) \\
&\geq \sum_X P(X|T;\boldsymbol{\beta},\boldsymbol{\pi},h) \log \frac{P(T,X;\overline{\boldsymbol{\beta}},\overline{\boldsymbol{\pi}},h)}{P(T,X;\boldsymbol{\beta},\boldsymbol{\pi},h)} \qquad (4.29)
\end{aligned}
$$

これから，EMアルゴリズムによって最大化するべき$\mathcal{Q}(\overline{\boldsymbol{\beta}},\overline{\boldsymbol{\pi}}|\boldsymbol{\beta},\boldsymbol{\pi})$は式4.30で表される．式4.30を最大化することで，パラメータ更新規則が求められる．この式で$\mathcal{D} = \bigcup_{i=1}^{N}\{T_i\}$は学習データを表す（EDAにおいては優良解の集合に相当する）．

$$
\mathcal{Q}(\overline{\boldsymbol{\beta}},\overline{\boldsymbol{\pi}}|\boldsymbol{\beta},\boldsymbol{\pi}) = \sum_{T_i\in\mathcal{D}} \sum_{X_i} P(X_i|T_i;\boldsymbol{\beta},\boldsymbol{\pi},h) \log P(T_i,X_i;\overline{\boldsymbol{\beta}},\overline{\boldsymbol{\pi}},h) \qquad (4.30)
$$

4.4.1節の前向き・後向き確率を用い，$\mathcal{Q}(\overline{\boldsymbol{\beta}},\overline{\boldsymbol{\pi}}|\boldsymbol{\beta},\boldsymbol{\pi})$を最大化することでパラメータ更新規則が求まる．導出法は文献[Matsuzaki 05]と同様であるが，本論文で用いる生成規則はCNFではなく，又右辺のアノテーションが同じであるという仮定を置いているため，更新規則は異なる（4.12節参照）．

以下に，パラメータ更新手順を示す．ここで，$\boldsymbol{\beta}^{(0)},\boldsymbol{\pi}^{(0)}$はEMアルゴリズムの初期値を表す．

1. $\boldsymbol{\beta}^{(0)},\boldsymbol{\pi}^{(0)}$の初期化．$t \leftarrow 0$.

2. パラメータを更新する.

$$\pi^{(t+1)}(\mathcal{S}[x]) \propto \pi^{(t)}(\mathcal{S}[x]) \sum_{T_i \in \mathcal{D}} \frac{b^1_{T_i}(x; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)}{P(T_i; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)} \tag{4.31}$$

$$\begin{aligned}
&\beta^{(t+1)}(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y]) \\
&\propto \beta^{(t)}(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y]) \\
&\times \sum_{T_i \in \mathcal{D}} \frac{1}{P(T_i; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)} \sum_{j \in cover(g, T_i)} f^j_{T_i}(x; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h) \\
&\times \prod_{k \in ch(j, T_i)} b^k_{T_i}(y; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)
\end{aligned} \tag{4.32}$$

3. 終了条件を満たした場合，更新終了．満たさない場合，$t \leftarrow t+1$ として，2 に戻る.

更新終了条件は以下の二つである（両方満たした場合）.

- $t \geq 10$
- $|L^{(t+1)}(\Theta; \mathcal{D}) - L^{(t)}(\Theta; \mathcal{D})| < 0.005 \times |L^{(t)}(\Theta; \mathcal{D})|$

但し，$L^{(t)}(\Theta; \mathcal{D})$ は第 t 反復における対数尤度であり，$L^{(t)}(\Theta; \mathcal{D}) = \sum_{T_i \in \mathcal{D}} \log P(T_i; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)$ である．第二の条件に用いられる 0.005 という値は，シミュレーション中に得られた経験に基づく値である.

パラメータの初期値 $\boldsymbol{\beta}^{(0)}, \boldsymbol{\pi}^{(0)}$ は，以下のように決定する．まず，GP の扱う構造は有限サイズであり，再帰的な生成規則を必要としない．そのため，$\mathcal{S}[x] \to g\,\mathcal{S}[x] \cdots \mathcal{S}[x]$ のような，左辺と右辺のアノテーションが等しい規則については $\beta^{(0)}(\mathcal{S}[x] \to g\,\mathcal{S}[x] \cdots \mathcal{S}[x]) = 0$ とする．その他のルールについては，アノテーションを考えることで増えるルールの分を考慮して，式 4.33 及び 4.34 のように初期化する（式 4.33 における分母 $h-1$ の -1 は，$\beta^{(0)}(\mathcal{S}[x] \to g\,\mathcal{S}[x] \cdots \mathcal{S}[x]) = 0$ に由来する）.

$$\beta^{(0)}(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y]) \propto \frac{e^{\kappa}}{h-1} \beta(\mathcal{S} \to g\,\mathcal{S} \cdots \mathcal{S}) \tag{4.33}$$

$$\beta^{(0)}(\mathcal{S}[x] \to g) \quad \propto \quad e^{\kappa} \beta(\mathcal{S} \to g) \tag{4.34}$$

式 4.33 及び 4.34 において，κ は $[-\log 3, \log 3]$ の一様乱数であり，$\beta(\mathcal{S} \to g\,\mathcal{S} \cdots \mathcal{S})$ はアノテーション無し生成規則の適用確率を表す（観測された導出木における適用確率）．なお，パラメータは以下の正規化条件を満たす.

$$\sum_{\zeta : \mathcal{S}[x] \to \zeta \in \mathcal{R}[H]} \beta(\mathcal{S}[x] \to \zeta) = 1 \tag{4.35}$$

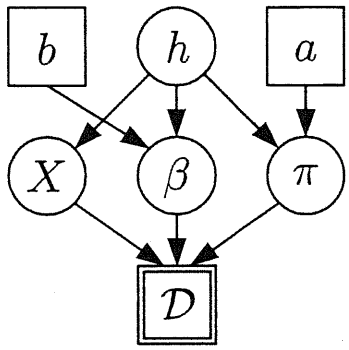


図 4.4: 変分ベイズ法の場合における，PCFG-LA の変数の関係.

4.5.2 変分ベイズ法による学習

本節では，変分ベイズ法の PCFG-LA への適用の詳細ついて説明する．変分ベイズ法は，隠れマルコフモデル [MacKay 97, Beal 03] や PCFG[Kurihara 04] などに適用されている．PCFG-LA への適用に関しても同様に行うことが可能である．以下，PCFG-LA の変分ベイズ法の導出を行う．

$\boldsymbol{\beta}$ の事前分布は Dirichlet 分布の積で表されるとする（自然共役な事前分布である）.

$$P(\boldsymbol{\beta}|h) = \prod_{x \in H} \mathrm{Dir}(\boldsymbol{\beta}_{\mathcal{S}[x]}; \mathbf{b}) \tag{4.36}$$

$$\mathrm{Dir}(\boldsymbol{\beta}_{\mathcal{S}[x]}; \mathbf{b}_{\mathcal{S}[x]}) = \frac{\Gamma\left(\sum_{\alpha \in \mathcal{R}_r[H]} b_{\mathcal{S}[x] \to \alpha}\right)}{\prod_{\alpha \in \mathcal{R}_r[H]} \Gamma(b_{\mathcal{S}[x] \to \alpha})} \prod_{\alpha \in \mathcal{R}_r[H]} \beta(\mathcal{S}[x] \to \alpha)^{b_{\mathcal{S}[x] \to \alpha} - 1} \tag{4.37}$$

ここで，$\Gamma(\cdot)$ は Gamma 関数，$\mathrm{Dir}(\cdot)$ は Dirichlet 分布を表す．$\beta_{\mathcal{S}[x]} = \{\beta(\mathcal{S}[x] \to \alpha) | \alpha \in \mathcal{R}_r[H]\}$ である．$\mathbf{b}_{\mathcal{S}[x]}$ は $\beta_{\mathcal{S}[x]}$ のハイパーパラメータであり，以下で定義される.

$$\mathbf{b}_{\mathcal{S}[x]} = \{b_{\mathcal{S}[x] \to \alpha} | \alpha \in \mathcal{R}_r[H]\} \tag{4.38}$$

$$\mathbf{b} = \{\mathbf{b}_{\mathcal{S}[x]} | x \in H\}$$

同様に，$\boldsymbol{\pi}$ の事前分布に関しても自然共役な Dirichlet 分布を仮定する.

$$
\begin{aligned}
P(\boldsymbol{\pi}|h) &= \mathrm{Dir}(\boldsymbol{\pi};\mathbf{a}) \\
&= \frac{\Gamma\left(\sum_{x\in H} a_{\mathcal{S}[x]}\right)}{\prod_{x\in H}\Gamma(a_{\mathcal{S}[x]})}\prod_{x\in H}\pi(\mathcal{S}[x])^{a_{\mathcal{S}[x]}-1}
\end{aligned}
\tag{4.39}
$$

ここで，$\boldsymbol{\pi} = \{\pi(\mathcal{S}[x])|x \in H\}$，$\mathbf{a}$ は π のハイパーパラメータであり，式 4.40 で定義される．

$$
\mathbf{a} = \{a_{\mathcal{S}[x]}|x \in H\} \tag{4.40}
$$

4.3.2 節で述べたように，隠れ変数に対するベイズ推定は，式の上では単純であっても，積分計算が困難なため行うことが難しい．変分ベイズ法では，分布の分解を仮定することで，解析的に行う．ここで，以下のような分布の分解を仮定する（4.41）．

$$
\begin{aligned}
Q(\boldsymbol{\beta},\boldsymbol{\pi},X|h) &= Q(\mathbf{X}|h)Q(\boldsymbol{\beta}|h)Q(\boldsymbol{\pi}|h) \\
&= \prod_{i=1}^{N} Q(X_i|h)\prod_{x\in H} Q(\boldsymbol{\beta}_{\mathcal{S}[x]}|h)Q(\boldsymbol{\pi}|h)
\end{aligned}
\tag{4.41}
$$

式 4.11 と同様に，尤度 $\log P(\mathcal{D}|h)$ の下限を，Jensen の不等式で評価する．

$$
\begin{aligned}
\log P(\mathcal{D}) &= \log\sum_{h\in\mathcal{H}}\sum_{\mathbf{X}}\int d\boldsymbol{\beta}d\boldsymbol{\pi}P(\mathcal{D},\mathbf{X},\boldsymbol{\beta},\boldsymbol{\pi},h) \\
&= \log\sum_{h\in\mathcal{H}}\sum_{\mathbf{X}}\int d\boldsymbol{\beta}d\boldsymbol{\pi}Q(\mathbf{X},\boldsymbol{\beta},\boldsymbol{\pi},h)\frac{P(\mathcal{D},\mathbf{X},\boldsymbol{\beta},\boldsymbol{\pi},h)}{Q(\mathbf{X},\boldsymbol{\beta},\boldsymbol{\pi},h)} \\
&\geq \sum_{h\in\mathcal{H}}\sum_{\mathbf{X}}\int d\boldsymbol{\beta}d\boldsymbol{\pi}Q(\mathbf{X},\boldsymbol{\beta},\boldsymbol{\pi},h)\log\frac{P(\mathcal{D},\mathbf{X},\boldsymbol{\beta},\boldsymbol{\pi},h)}{Q(\mathbf{X},\boldsymbol{\beta},\boldsymbol{\pi},h)} \\
&= \sum_{h\in\mathcal{H}}\sum_{\mathbf{X}}\int d\boldsymbol{\beta}d\boldsymbol{\pi}\Bigg\{Q(\mathbf{X}|h)Q(\boldsymbol{\beta}|h)Q(\boldsymbol{\pi}|h)Q(h) \\
&\qquad \times\log\frac{P(\mathcal{D},\mathbf{X}|\boldsymbol{\beta},\boldsymbol{\pi},h)P(\boldsymbol{\pi}|h)P(\boldsymbol{\beta}|h)P(h)}{Q(\mathbf{X}|h)Q(\boldsymbol{\beta}|h)Q(\boldsymbol{\pi}|h)Q(h)}\Bigg\} \qquad (4.42) \\
&\equiv \mathcal{F}[Q] \qquad (4.43)
\end{aligned}
$$

モデル（アノテーション数）に関する事後分布を分離すると，式 4.44 のように表される．

$$\begin{aligned}\log P(\mathcal{D}) &= \sum_{h\in\mathcal{H}} Q(h)\log\frac{P(h)}{Q(h)} \\ &\quad + \sum_{h\in\mathcal{H}}\sum_{\mathbf{X}}\int d\boldsymbol{\beta}d\boldsymbol{\pi}\left\{Q(\mathbf{X}|h)Q(\boldsymbol{\beta}|h)Q(\boldsymbol{\pi}|h)Q(h)\log\frac{P(\mathcal{D},\mathbf{X}|\boldsymbol{\beta},\boldsymbol{\pi},h)P(\boldsymbol{\pi}|h)P(\boldsymbol{\beta}|h)}{Q(\mathbf{X}|h)Q(\boldsymbol{\beta}|h)Q(\boldsymbol{\pi}|h)}\right\} \\ &= -\mathrm{KL}(Q(h)|P(h)) + \underline{\langle\mathcal{F}_h[Q]\rangle_{Q(h)}} \qquad (4.44)\end{aligned}$$

ここで，

$$\mathcal{F}_h[Q] \equiv \sum_{\mathbf{X}}\int d\boldsymbol{\beta}d\boldsymbol{\pi}\left\{Q(\mathbf{X}|h)Q(\boldsymbol{\beta}|h)Q(\boldsymbol{\pi}|h)\log\frac{P(\mathcal{D},\mathbf{X}|\boldsymbol{\beta},\boldsymbol{\pi},h)P(\boldsymbol{\pi}|h)P(\boldsymbol{\beta}|h)}{Q(\mathbf{X}|h)Q(\boldsymbol{\beta}|h)Q(\boldsymbol{\pi}|h)}\right\}$$

とおいた．

4.3.2節で述べたように，下限$\mathcal{F}_h[Q]$を最大化することはすなわち真の事後分布$P(\cdot|\mathcal{D},h)$とテスト分布（近似事後分布）$Q(\cdot|h)$のKLダイバージェンス（Kullback Leibler Divergence）を最小化することに等しい．$\mathcal{F}_h[Q]$の最小化は，汎関数の微分を行い，それぞれの近似事後分布を独立に最大化することで得られる）．変分ベイズに関する公式（式4.14, 4.15）から，

$$Q(\boldsymbol{\beta}_{\mathcal{S}[x]}|h) \propto P(\boldsymbol{\beta}_{\mathcal{S}[x]}|h)\exp\langle\log P(\mathcal{D},\mathbf{X}|\boldsymbol{\beta},\boldsymbol{\pi},h)\rangle_{Q(\mathbf{X}|h),Q(\boldsymbol{\beta}_{-\mathcal{S}[x]}|h),Q(\boldsymbol{\pi}|h)} \qquad (4.45)$$

$$Q(\boldsymbol{\pi}|h) \propto P(\boldsymbol{\pi}|h)\exp\langle\log P(\mathcal{D},\mathbf{X}|\boldsymbol{\beta},\boldsymbol{\pi},h)\rangle_{Q(\mathbf{X}|h),Q(\boldsymbol{\beta}|h)} \qquad (4.46)$$

と計算される．これらの式で，$\langle\cdot\rangle_{Q(\cdot)}$は$Q(\cdot)$に関して期待値を計算することを表す．これらを計算すると，$\boldsymbol{\beta}$の近似事後分布は以下のように表される．

$$Q(\boldsymbol{\beta}_{\mathcal{S}[x]}|h) = \mathrm{Dir}(\boldsymbol{\beta}_{\mathcal{S}[x]};\breve{\mathbf{b}}_{\mathcal{S}[x]}) \qquad (4.47)$$

$$\breve{\mathbf{b}}_{\mathcal{S}[x]} = \{\breve{b}_{\mathcal{S}[x]\to\alpha}|\alpha\in\mathcal{R}_T[H]\} \qquad (4.48)$$

$$\breve{b}_{\mathcal{S}[x]\to\alpha} = \sum_{i=1}^{N}\sum_{X_i} Q(X_i|h)\cdot c(\mathcal{S}[x]\to\alpha;X_i,T_i) + b_{\mathcal{S}[x]\to\alpha} \qquad (4.49)$$

また，$\boldsymbol{\pi}$の事後分布も同様に式4.50で表される．

$$Q(\boldsymbol{\pi}|h) = \mathrm{Dir}(\boldsymbol{\pi};\breve{\mathbf{a}}) \qquad (4.50)$$

$$\breve{\mathbf{a}} = \{\breve{a}_{\mathcal{S}[x]}|x\in H\} \qquad (4.51)$$

$$\breve{a}_{\mathcal{S}[x]} = \sum_{i=1}^{N} \sum_{X_i} Q(X_i|h) \cdot \delta(x; T_i, X_i) + a_{\mathcal{S}[x]} \tag{4.52}$$

隠れ変数に関する近似事後分布 $Q(\mathbf{X}|h) = \prod_{i=1}^{N} Q(X_i|h)$ は，式 4.15 から

$$Q(X_i|h) \propto \exp \langle \log P(T_i, X_i|\boldsymbol{\beta}, \boldsymbol{\pi}, h) \rangle_{Q(\boldsymbol{\pi}|h), Q(\boldsymbol{\beta}|h)} \tag{4.53}$$

であり，これを解くと式 4.54 が得られる．

$$Q(X_i|h) = \frac{1}{\mathcal{Z}(T_i)} \left\{ \prod_{x \in H} \widetilde{\pi}(\mathcal{S}[x])^{\delta(x;T_i,X_i)} \right\} \left\{ \prod_{\mathcal{S}[x] \to \alpha \in \mathcal{R}[H]} \widetilde{\beta}(\mathcal{S}[x] \to \alpha)^{c(\mathcal{S}[x] \to \alpha; T_i, X_i)} \right\} \tag{4.54}$$

$$\widetilde{\beta}(\mathcal{S}[x] \to \alpha) = \exp \left[\psi(\breve{b}_{\mathcal{S}[x] \to \alpha}) - \psi \left(\sum_{\alpha \in \mathcal{R}_r[H]} \breve{b}_{\mathcal{S}[x] \to \alpha} \right) \right]$$

$$\widetilde{\pi}(\mathcal{S}[x]) = \exp \left[\psi(\breve{a}_{\mathcal{S}[x]}) - \psi \left(\sum_{x \in H} \breve{a}_{\mathcal{S}[x]} \right) \right]$$

これらの式で，$\psi(\cdot)$ は Digamma 関数[6]を表し，$\mathcal{Z}(T_i)$ は正規化項を表す．式 4.54 は式 4.16 と非常に似た形をしていることが分かる．式 4.54 は式 4.16 のパラメータ $\boldsymbol{\beta}, \boldsymbol{\pi}$ を $\widetilde{\boldsymbol{\beta}}, \widetilde{\boldsymbol{\pi}}$ で置き換えたに過ぎない．それゆえ，正規化項 $\mathcal{Z}(T_i)$ は $P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, h)$ に相当する項であり，前向き・後ろ向き確率を用いて式 4.55 で表される．

$$\mathcal{Z}(T_i) = \sum_{x \in H} \widetilde{\pi}(\mathcal{S}[x]) b^1_{T_i}(x; \widetilde{\boldsymbol{\beta}}, \widetilde{\boldsymbol{\pi}}, h) \tag{4.55}$$

式 4.49 と 4.52 の $\sum_{X_i} Q(X_i|h) \cdot c(\mathcal{S}[x] \to \alpha; T_i, X_i)$ と $\sum_{X_i} Q(X_i|h) \cdot \delta(x; T_i, X_i)$ は，隠れ変数集合 X_i に関する和が含まれているが，ノード数が多くなってくると，直接計算することが出来ない．これらの項は，前向き・後ろ向き確率を使って効率的に計算可能である（付録 4.13 参照）．前向き・後ろ向き確率によって，これらの項は式 4.56 と 4.57 のように表される．

$$\widetilde{c}(\mathcal{S}[x] \to \alpha; T_i) = \sum_{X_i} Q(X_i|h) \cdot c(\mathcal{S}[x] \to \alpha; T_i, X_i)$$

$$\widetilde{\delta}(x; T_i) = \sum_{X_i} Q(X_i|h) \cdot \delta(x; T_i, X_i)$$

とおく．その場合，

[6] $\psi(x) = \frac{\partial}{\partial x} \log \Gamma(x)$

$$\begin{aligned}
&\widetilde{c}(\mathcal{S}[x] \to g\,\mathcal{S}[y]\,\mathcal{S}[y]\cdots\mathcal{S}[y]; T_i) \\
&= \frac{\widetilde{\beta}(\mathcal{S}[x] \to g\,\mathcal{S}[y]\,\mathcal{S}[y]\cdots\mathcal{S}[y])}{\mathcal{Z}(T_i)} \\
&\times \sum_{k \in cover(g, T_i)} f^k_{T_i}(x; \widetilde{\boldsymbol{\beta}}, \widetilde{\boldsymbol{\pi}}, h) \prod_{j \in ch(k, T_i)} b^j_{T_i}(y; \widetilde{\boldsymbol{\beta}}, \widetilde{\boldsymbol{\pi}}, h)
\end{aligned} \tag{4.56}$$

$$\widetilde{\delta}(x; T_i) = \frac{\widetilde{\pi}(\mathcal{S}[x])}{\mathcal{Z}(T_i)} b^1_{T_i}(x; \widetilde{\boldsymbol{\beta}}, \widetilde{\boldsymbol{\pi}}, h) \tag{4.57}$$

と表される.

更新式は互いに依存しあっており，閉形式で記述することが出来ない．そのため，交互に最適化される．これらをまとめると，以下のようになる.

$$Q^{(t+1)}(\boldsymbol{\beta}_{\mathcal{S}[x]}|h) = \mathrm{Dir}(\boldsymbol{\beta}_{\mathcal{S}[x]}; \mathbf{b}^{(t+1)}_{\mathcal{S}[x]})$$

$$\mathbf{b}^{(t+1)}_{\mathcal{S}[x]} = \{b^{(t+1)}_{\mathcal{S}[x]\to\alpha} | \mathcal{S}[x] \to \alpha \in \mathcal{R}[H]\}$$

$$b^{(t+1)}_{\mathcal{S}[x]\to\alpha} = \sum_{i=1}^{N} \sum_{X_i} Q^{(t)}(X_i|h) \cdot c(\mathcal{S}[x] \to \alpha; T_i, X_i) + b_{\mathcal{S}[x]\to\alpha}$$

$$Q^{(t+1)}(\boldsymbol{\pi}|h) = \mathrm{Dir}(\boldsymbol{\pi}; \mathbf{a}^{(t+1)})$$

$$\mathbf{a}^{(t+1)} = \{a^{(t+1)}_{\mathcal{S}[x]} | x \in H\}$$

$$a^{(t+1)}_{\mathcal{S}[x]} = \sum_{i=1}^{N} \sum_{X_i} Q^{(t)}(X_i|h) \cdot \delta(x; T_i, X_i) + a_{\mathcal{S}[x]}$$

ここで，$\mathbf{b}^{(0)} = \{b^{(0)}_{\mathcal{S}[x]\to\alpha} | \mathcal{S}[x] \to \alpha \in \mathcal{R}[H]\}, \mathbf{a}^{(0)} = \{a^{(0)}_{\mathcal{S}[x]} | x \in H\}$ は事前分布のハイパーパラメータである（つまり，$b^{(0)}_{\mathcal{S}[x]\to\alpha} = b_{\mathcal{S}[x]\to\alpha}$，$a^{(0)}_{\mathcal{S}[x]} = a_{\mathcal{S}[x]}$）．本論文では，以下のようにランダムに決定している.

$$a_{\mathcal{S}[x]} \in [1, 2],\ b_{\mathcal{S}[x]\to\alpha} \in [1, 2]$$

但し，PAGE-EM の場合と同様に，$\mathcal{S}[x] \to g\,\mathcal{S}[x]\,\mathcal{S}[x]\cdots\mathcal{S}[x]$ のように，左辺と右辺が同じルールのハイパーパラメータに関しては $b_{\mathcal{S}[x]\to g\,\mathcal{S}[x]\,\mathcal{S}[x]\cdots\mathcal{S}[x]} = 1$ のように，一様分布に設定した.

更新終了条件は以下の二つである（両方満たした場合）.

- $t \geq 20$
- $|\mathcal{F}^{(t+1)}_h[Q] - \mathcal{F}^{(t)}_h[Q]| < 0.002 \times |\mathcal{F}^{(t)}_h[Q]|$

ここで，$\mathcal{F}^{(t)}_h[Q]$ は第 t 反復における値を表す.

4.6 新しい個体の生成

2.3.4節では，新しい個体は予測事後分布によって生成されると述べた．すなわち，世代$g+1$の集団$\mathcal{P}_{g+1}$は予測事後分布$P(T,X|\mathcal{D}_g)$に従って生成される．

EMアルゴリズムの場合，予測事後分布は点推定値によって評価される（式4.58）．ここで，$\overline{\boldsymbol{\beta}}^*, \overline{\boldsymbol{\pi}}^*$はEMアルゴリズムによって得られた収束値である．

$$P(T,X|\mathcal{D}_g) = P(T,X|\overline{\boldsymbol{\beta}}^*, \overline{\boldsymbol{\pi}}^*, h) \tag{4.58}$$

分布が式4.58の場合，新しい個体はPLS（Probabilistic Logic Sampling）のように高速に生成することが出来る．変分ベイズ法の場合，式4.59に沿ってアンサンブルによって計算される．$\mathcal{H}$を可能なアノテーション数の集合とする（$\mathcal{H}=\bigcup_{i=1}^{\infty}\{i\}$）．

$$\begin{aligned} P(T,X|\mathcal{D}_g) &= \sum_{h\in\mathcal{H}} \int d\boldsymbol{\beta} d\boldsymbol{\pi} P(T,X|\boldsymbol{\beta},\boldsymbol{\pi},h) P(\boldsymbol{\beta},\boldsymbol{\pi},h|\mathcal{D}_g) \\ &= \sum_{h\in\mathcal{H}} \int d\boldsymbol{\beta} d\boldsymbol{\pi} P(T,X|\boldsymbol{\beta},\boldsymbol{\pi},h) P(\boldsymbol{\beta},\boldsymbol{\pi}|\mathcal{D}_g,h) P(h|\mathcal{D}_g) \end{aligned} \tag{4.59}$$

式4.59において，可能なアノテーション数について和を取るのは不可能である．多くのEDAがそうするように，PAGE-VBにおいても，MAP近似によって近似的に扱う．すなわち，$h_g^{opt} = \underset{h}{\operatorname{argmax}} P(h|\mathcal{D}_g)$によって点推定近似する．真の事後分布$P(\cdot|\mathcal{D})$を近似事後分布$Q(\cdot)$で近似することで，式4.59は式4.60によって表される．

$$\begin{aligned} P(T,X|\mathcal{D}_g) &\simeq \int d\boldsymbol{\beta} d\boldsymbol{\pi} P(T,X|\boldsymbol{\beta},\boldsymbol{\pi},h_g^{opt}) P(\boldsymbol{\beta},\boldsymbol{\pi}|\mathcal{D}_g,h_g^{opt}) \\ &\simeq \int d\boldsymbol{\beta} d\boldsymbol{\pi} P(T,X|\boldsymbol{\beta},\boldsymbol{\pi}) Q^*(\boldsymbol{\beta}|h_g^{opt}) Q^*(\boldsymbol{\pi}|h_g^{opt}) \end{aligned} \tag{4.60}$$

ここで，$Q^*(\cdot)$は，変分ベイズ法で求められた収束分布である．この式を計算すると，最終的に式4.61が計算される．

$$P(T,X|\mathcal{D},h_g^{opt}) \propto \frac{\prod_{x\in H}\Gamma(\acute{a}_{\mathcal{S}[x]})}{\Gamma\left(\sum_{x\in H}\acute{a}_{\mathcal{S}[x]}\right)} \cdot \prod_{x\in H} \frac{\prod_{\alpha\in\mathcal{R}_r[H]}\Gamma(\acute{b}_{\mathcal{S}[x]\to\alpha})}{\Gamma\left(\sum_{\alpha\in\mathcal{R}_r[H]}\acute{b}_{\mathcal{S}[x]\to\alpha}\right)} \tag{4.61}$$

$$\acute{a}_{\mathcal{S}[x]} = \delta(x;T,X) + \breve{a}^*_{\mathcal{S}[x]}$$

$$\acute{b}_{\mathcal{S}[x]\to\alpha} = c(\mathcal{S}[x]\to\alpha;T,X) + \breve{b}^*_{\mathcal{S}[x]}$$

ここで，$\breve{a}^*_{\mathcal{S}[x]}$ 及び $\breve{b}^*_{\mathcal{S}[x]}$ は，変分ベイズ法によって得られた収束値である．原理的には式4.61からサンプリングすることで，PAGE-VBの個体は生成される．しかし，この式は生成規則ごとに分解されていないため，Gibbs Samplingによってサンプリングする必要がある．実際に，式4.61に基づくアルゴリズムを初期段階で実装したものの，計算量が非常に多く，実用的な速度で個体を生成することが出来なかった．そこで，さらに分布のMAP（Maximum a Posteriori）値のみを用いる近似を行い，以下の式を得る．

$$\int d\boldsymbol{\beta} d\boldsymbol{\pi} P(T, X|\boldsymbol{\beta}, \boldsymbol{\pi}, h_g^{opt}) Q^*(\boldsymbol{\beta}|h_g^{opt}) Q^*(\boldsymbol{\pi}|h_g^{opt}) \simeq P(T, X|\boldsymbol{\beta}^*_{\mathrm{MAP}}, \boldsymbol{\pi}^*_{\mathrm{MAP}}, h_g^{opt}) \quad (4.62)$$

ここで，$\boldsymbol{\beta}^*_{\mathrm{MAP}}, \boldsymbol{\pi}^*_{\mathrm{MAP}}$ は収束分布 $Q^*(\cdot)$ のMAP値である．Dirichlet分布のMAP値は，ラグランジュの未定乗数法より以下のように計算される．

$$\pi^*_{\mathrm{MAP}}(\mathcal{S}[x]) = \frac{\breve{a}^*_{\mathcal{S}[x]} - 1}{\sum_{x \in H} (\breve{a}^*_{\mathcal{S}[x]} - 1)} \quad (4.63)$$

$$\beta^*_{\mathrm{MAP}}(\mathcal{S}[x] \to \alpha) = \frac{\breve{b}^*_{\mathcal{S}[x] \to \alpha} - 1}{\sum_{\alpha \in \mathcal{R}_r[H]} (\breve{b}^*_{\mathcal{S}[x] \to \alpha} - 1)} \quad (4.64)$$

ここで，$\breve{a}^*_{\mathcal{S}[x]}$ は $\breve{a}_{\mathcal{S}[x]}$，$\breve{b}^*_{\mathcal{S}[x] \to \alpha}$ は $\breve{b}_{\mathcal{S}[x] \to \alpha}$ の収束値を表す．MAP近似を用いることで，高速にサンプリング可能であるが，ベイズ推定の分布としての特徴（汎化誤差の小ささなど）は失われる．

4.7 アノテーション数選択

4.6節では，最適アノテーションサイズ h_g^{opt} による点近似を行った．モデルに関する事前分布 $P(h)$ が一様分布の場合，h_g^{opt} は尤度の下限の収束値 $\mathcal{F}^*_h[Q]$ の最大値を与える h と同じである [上田 01]．すなわち，

$$h_g^{opt} = \operatorname*{argmax}_h \mathcal{F}^*_h[Q]$$

である．なぜなら，式4.44に関して，制約条件 $\sum_h Q(h) = 1$ で最適化すると，

$$Q^*(h) = \frac{P(h) \exp\{\mathcal{F}^*_h[Q]\}}{\sum_{h' \in \mathcal{H}} P(h') \exp\{\mathcal{F}^*_{h'}[Q]\}}$$

となり，$\mathcal{F}^*_h[Q]$ に対して $Q(h)$ が単調増加するためである．

$\mathcal{F}^*_h[Q]$ を計算すると，以下の式で表される．

$$\mathcal{F}_h^*[Q] = \sum_{i=1}^{N} \log \mathcal{Z}(T_i) - \mathrm{KL}(Q^*(\boldsymbol{\beta}|h)|P(\boldsymbol{\beta}|h)) - \mathrm{KL}(Q^*(\boldsymbol{\pi}|h)|P(\boldsymbol{\pi}|h))$$

ここで，後ろの二つの項は以下のように計算される（付録4.14参照）．

$$\begin{aligned}
\mathrm{KL}(Q^*(\boldsymbol{\beta}|h)|P(\boldsymbol{\beta}|h)) &= \sum_{x\in H}\left\{\sum_{\alpha\in\mathcal{R}_r[H]}\left(\log\Gamma(b_{S[x]\to\alpha}) - \log\Gamma(\breve{b}^*_{S[x]\to\alpha})\right)\right. \\
&+ \left.\log\Gamma\left(\sum_{\alpha\in\mathcal{R}_r[H]}\breve{b}^*_{S[x]\to\alpha}\right) - \log\Gamma\left(\sum_{\alpha\in\mathcal{R}_r[H]} b_{S[x]\to\alpha}\right)\right\} \\
&+ \sum_{x\in H}\sum_{\alpha\in\mathcal{R}_r[H]}\left(\breve{b}^*_{S[x]\to\alpha} - b_{S[x]\to\alpha}\right) \\
&\times \left\{\psi(\breve{b}^*_{S[x]\to\alpha}) - \psi\left(\sum_{\alpha'\in\mathcal{R}_r[H]}\breve{b}^*_{S[x]\to\alpha'}\right)\right\}
\end{aligned}$$

$$\begin{aligned}
\mathrm{KL}(Q^*(\boldsymbol{\pi}|h)|P(\boldsymbol{\pi}|h)) &= \sum_{x\in H}\left(\log\Gamma(a_{S[x]}) - \log\Gamma(\breve{a}^*_{S[x]})\right) \\
&+ \log\Gamma\left(\sum_{x\in H}\breve{a}^*_{S[x]}\right) - \log\Gamma\left(\sum_{x\in H} a_{S[x]}\right) \\
&+ \sum_{x\in H}(\breve{a}^*_{S[x]} - a_{S[x]})\left\{\psi(\breve{a}^*_{S[x]}) - \psi\left(\sum_{x'\in H}\breve{a}^*_{S[x']}\right)\right\}
\end{aligned}$$

最適なアノテーションサイズの探索は，各$\mathcal{F}_h^*[Q]$, $h\in\mathcal{H}$を計算し，最大値を与えるhを選択することで行われるが，各世代において$\mathcal{F}_h^*[Q]$をすべて計算することは，計算量が多い．PAGE-VBでは，以下の二つの仮定を置き，計算量を削減する．

- **Sparse Exploration**
  各世代で，スパースに$\mathcal{F}_h^*[Q]$の計算を行う．$\mathcal{H}_0$を初期世代において，探索するアノテーション数の集合とする．$\mathcal{H}_0=\{1,2,4,8\}$の場合，$\mathcal{F}_1^*[Q],\mathcal{F}_2^*[Q],\mathcal{F}_4^*[Q],\mathcal{F}_8^*[Q]$のみを計算し，最大の値を与える$h$を選択する．

- **Exploration Restriction**
  隣接する世代の最適アノテーション数は類似しているという仮定を置く．すなわち，世代$g+1$では，世代gの最適アノテーション数h_{opt}に対して，$\pm h_{range}$の範囲の$\mathcal{F}_h^*[Q]$のみを計算する（図4.5）．例えば，世代gで$h_{opt}=5$であり，$h_{range}=2$の場合，世代$g+1$では$\mathcal{F}_3^*[Q],\mathcal{F}_4^*[Q],\mathcal{F}_5^*[Q],\mathcal{F}_6^*[Q],\mathcal{F}_7^*[Q]$の計算を行う．

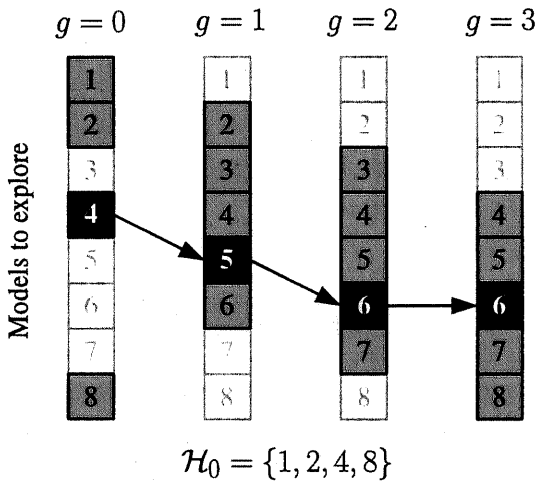


図 4.5: アノテーション選択の模式図. g は世代を表し，灰色のセルは $\mathcal{F}_h^*[Q]$ を計算したアノテーション，黒いセルは選択されたアノテーション h_{opt} を表す.

Algorithm 7 Parameter update formula for PAGE-EM.

1. Initialize parameters $\boldsymbol{\beta}^{(0)}, \boldsymbol{\pi}^{(0)}$.

2. Update parameters using equations below.

$$\pi^{(t+1)}(\mathcal{S}[x]) \propto \pi^{(t)}(\mathcal{S}[x]) \sum_{T_i \in \mathcal{D}} \frac{b^1_{T_i}(x; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)}{P(T_i; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)}$$

$$\begin{aligned}\beta^{(t+1)}(\mathcal{S}[x] \to g\,\mathcal{S}[y]...\mathcal{S}[y]) \quad &\propto \quad \beta^{(t)}(\mathcal{S}[x] \to g\,\mathcal{S}[y]...\mathcal{S}[y]) \sum_{T_i \in \mathcal{D}} \frac{1}{P(T_i; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)} \\ &\times \sum_{j \in cover(g, T_i)} f^j_{T_i}(x; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h) \prod_{k \in ch(j, T_i)} b^k_{T_i}(y; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)\end{aligned}$$

3. If the update terminate criteria are not met, goto 2. Otherwise stop updating.

Algorithm 8 Distribution update formula for PAGE-VB.

1. Initialize hyper parameters $\mathbf{a}^{(0)} = \mathbf{a}, \mathbf{b}^{(0)} = \mathbf{b}$.
2. Update distribution (hyper-parameters) using equations below.

$$Q^{(t+1)}(\boldsymbol{\beta}_{S[x]}|h) = \mathrm{Dir}(\boldsymbol{\beta}_{S[x]}; \mathbf{b}_{S[x]}^{(t+1)})$$

$$\mathbf{b}_{S[x]}^{(t+1)} = \{b_{S[x]\to\alpha}^{(t+1)} | S[x] \to \alpha \in \mathcal{R}[H]\}$$

$$b_{S[x]\to\alpha}^{(t+1)} = \sum_{i=1}^{N} \sum_{X_i} Q^{(t)}(X_i|h) \cdot c(S[x] \to \alpha; T_i, X_i) + b_{S[x]\to\alpha}$$

$$Q^{(t+1)}(\boldsymbol{\pi}|h) = \mathrm{Dir}(\boldsymbol{\pi}; \mathbf{a}^{(t+1)})$$

$$\mathbf{a}^{(t+1)} = \{a_{S[x]}^{(t+1)} | x \in H\}$$

$$a_{S[x]}^{(t+1)} = \sum_{i=1}^{N} \sum_{X_i} Q^{(t)}(X_i|h) \cdot \delta(x; T_i, X_i) + a_{S[x]}$$

3. If the update terminate criteria are not met, goto 2. Otherwise stop updating.

Algorithm 9 PAGE-EM

1. $\boldsymbol{\beta}_0, \boldsymbol{\pi}_0, h \leftarrow$Initialize parameters
   $g \leftarrow 0$
2. $\mathcal{P}_0 \leftarrow$Generate M initial individuals from $P(T, X|\boldsymbol{\beta}_0, \boldsymbol{\pi}_0, h)$
3. $\mathcal{D}_g \leftarrow$Select $N \leq M$ promising solutions using a truncate selection
4. $\overline{\boldsymbol{\beta}}^*, \overline{\boldsymbol{\pi}}^* \leftarrow$Estimate parameters with $\mathcal{D}_g$ using EM algorithm
5. $\mathcal{P}_{g+1} \leftarrow$Generate M new individuals from $P(T, X|\overline{\boldsymbol{\beta}}^*, \overline{\boldsymbol{\pi}}^*, h)$
   $g \leftarrow g + 1$

Steps from 3 to 5 are repeated until termination criteria are met.

Algorithm 10 PAGE-VB

1. $\boldsymbol{\beta}_0, \boldsymbol{\pi}_0 \leftarrow$Initialize parameters
   $g \leftarrow 0$
2. $\mathcal{P}_0 \leftarrow$Generate M initial individuals from $P(T, X|\boldsymbol{\beta}_0, \boldsymbol{\pi}_0, h = 1)$
3. $\mathcal{D}_g \leftarrow$Select $N \leq M$ promising solutions using a truncate selection
4. $\Omega_h = \{\breve{\mathbf{a}}^*, \breve{\mathbf{b}}^*\} \leftarrow$Calculate hyper-parameters in $h_{g-1}^{opt} - h_{range} \leq h \leq h_{g-1}^{opt} + h_{range}$ $(g \neq 0)$, $h \in \mathcal{H}_0$ $(g = 0)$ using $\mathcal{D}_g$
5. $\mathcal{F}_h^*[Q] \leftarrow$Calculate lower bound in each h using Ω_h
   $h_g^{opt} \leftarrow \underset{h_{g-1}^{opt} - h_{range} \leq h \leq h_{g-1}^{opt} + h_{range}}{\operatorname{argmax}} \mathcal{F}_h^*[Q]$ $(g \neq 0)$
   $h_g^{opt} \leftarrow \underset{h \in \mathcal{H}_0}{\operatorname{argmax}} \mathcal{F}_h^*[Q]$ $(g = 0)$
6. $\boldsymbol{\beta}_{\mathrm{MAP}}^*, \boldsymbol{\pi}_{\mathrm{MAP}}^* \leftarrow$Calculate MAP parameters with $\Omega_{h_g^{opt}}$
7. $\mathcal{P}_{g+1} \leftarrow$Generate M new individuals from $P(T, X|\boldsymbol{\beta}_{\mathrm{MAP}}^*, \boldsymbol{\pi}_{\mathrm{MAP}}^*, h_g^{opt})$
   $g \leftarrow g + 1$

Steps from 3 to 7 are repeated until termination criteria are met.

4.8 アルゴリズム：PAGE-EM，PAGE-VB

本節では提案手法PAGE-EM（EMアルゴリズムによる学習）及びPAGE-VB（変分ベイズ法による学習）のアルゴリズムの詳細について説明する．EMアルゴリズムを用いた場合のパラメータ更新規則をアルゴリズム7，変分ベイズ法を用いた場合のハイパーパラメータ更新規則をアルゴリズム8に示す．

PAGE-EMとPAGE-VBのPseudoコードをアルゴリズム9とアルゴリズム10に示す．本節では，PAGE-EMとPAGE-VBのアルゴリズムの詳細について説明を行う．ここで，Θ_gのように変数についている下付添字は，g世代目における変数を示す．また，括弧内にPAGE-EM及びPAGE-VBが記述されているが，その項目がPAGE-EM，PAGE-VBに関係しているか否かを表す．

1. 集団の初期化（PAGE-EM，PAGE-VB）
   初期個体$\mathcal{P}_0$を生成する．初期個体はパラメータ$\boldsymbol{\beta}_0, \boldsymbol{\pi}_0$に従って生成される．初期パラメータは任意であるが，小さすぎる個体や大きすぎる個体ばかり生成されないようにするため，本論文では最大深さに達するまでは，終端ノードと関数ノードが生成される割合を1:4としている．また，終端ノード間，関数ノード間の選択される割合は等しくする．

2. 個体の選択（PAGE-EM，PAGE-VB）
   パラメータ推定に用いる優良個体$\mathcal{D}_g$を，個体群$\mathcal{P}_g$より$M \times P_s$個選択する．様々な選択方法が考えられるが，本提案手法ではTruncate選択（2.3.2節）を用いる．

3. 学習
   PCFG-LAの学習を行う．PAGE-EMではパラメータを求め，PAGE-VBではアノテーションサイズとハイパーパラメータを求める．

   (a) パラメータ推定（PAGE-EM）
   選択された優良個体$\mathcal{D}_g$に基づき，パラメータ$\overline{\boldsymbol{\beta}}^*, \overline{\boldsymbol{\pi}}^*$を推定する．パラメータは初期値より始まり，アルゴリズム7に示す更新規則に従ってパラメータを更新する．

   (b) アノテーションサイズとハイパーパラメータ推定（PAGE-VB）
   世代$g=0$の場合，$h \in \mathcal{H}_0$に関して，それぞれアルゴリズム8に基づきハイパーパラメータと$\mathcal{F}_h^*[Q]$を計算する．世代$g>0$の場合，世代$g-1$のアノテーション数h_{g-1}^{opt}を用いて，範囲$h_{g-1}^{opt} - h_{range} \leq h \leq h_{g-1}^{opt} + h_{range}$の範囲で行う．その上で，世代$g$のアノテーション数を$h_g^{opt} = \underset{h}{\operatorname{argmax}} \mathcal{F}_h^*[Q]$とする．

4. 新しい個体の生成
   $P(T, X|\mathcal{D}_g)$に従って，$\mathcal{P}_g$が生成される．

   (a) 最尤推定パラメータによる生成（PAGE-EM）
   推定されたパラメータ$\overline{\boldsymbol{\beta}}^*, \overline{\boldsymbol{\pi}}^*$を用い，新しい個体群$\mathcal{D}_{g+1}$を生成する．エリー

表 4.3: PAGE-EM と PAGE-VB の主要パラメータ.

PAGE-EM		
変数	パラメータ	値
M	集団数	1000
h	アノテーション数	2, 4, 8, 16 (Royal Tree) 8 (DMAX)
D_P	最大深さ	5 (Royal Tree) 4 (DMAX)
P_s	選択率	0.1
P_e	エリート率	0.1

PAGE-VB		
変数	パラメータ	値
M	集団数	1000
$\mathcal{H}_0$	初期世代のアノテーション候補	$\{1, 2, 4, 8, 16\}$
h_{range}	アノテーション探索範囲	2
D_P	最大深さ	5 (Royal Tree) 4 (DMAX)
P_s	選択率	0.1
P_e	エリート率	0.1

ト選択を用いる場合は，前世代の上位 $M \times P_e$ 個体をそのままコピーして用い，$M(1 - P_e)$ 個体を生成する.

(b) **MAP 推定値による生成（PAGE-VB）**
h_g^{opt} における MAP 推定値によって（式 4.64, 4.63），新しい個体が生成される.

PAGE の主要パラメータを表 4.3 に示す.

4.9 計算機実験

提案手法 PAGE-EM と PAGE-VB の有効性を見るために，二つのベンチマークテストに適用し，探索能力を調べた．ベンチマークテストとしては，前述の Royal Tree 問題と DMAX 問題を選択した.

4.9.1 Royal Tree 問題

4.9.1.1 問題設定

Royal Tree 問題は GP の探索メカニズムを調べるために考案された問題であり，GA で有名な Royal Road 関数 [Mitchell 92] を木構造に拡張した問題である（3.6.3 節参照）．3.6.3 節で用いた Royal Tree 問題は終端として $\mathfrak{T} = \{x, y\}$ を用い，すべての関数ノードの引数を 2 で固定していたが，ここで用いる Royal Tree 問題は通常の設定である（$\mathfrak{F} = \{a, b, c, d\}$, $\mathfrak{T} = \{x\}$）．本節では，$Full\,Bonus = 2$, $Partial\,Bonus = 1$, $Penalty = 1/3$, $Complete\,Bonus = 2$ という値が用いられる．

本問題は，PAGE-EM におけるアノテーション数が探索性能に与える影響と，PAGE-VB の探索の振る舞いを調べる．Royal Tree 問題で用いるアルゴリズムは以下の三つである．

- **PAGE-EM**
  EM アルゴリズムを用いた PCFG-LA を基本文法としている．アノテーション数は $h = 2, 4, 8, 16$ で実験を行った．

- **PAGE-VB**
  変分ベイズ法を用い，アノテーション数自体も学習データから推定する．世代 0 での探索アノテーションは $\mathcal{H}_0 = \{1, 2, 4, 8, 16\}$ と設定した．その他のパラメータ設定は，PAGE-EM と同じである．

- **PCFG-GP**
  アノテーションを用いない，標準的な PCFG を用いた GP-EDA である．Scaler SG-GP（Stochastic Grammar based GP）[Ratle 01] に近いモデルである．基本的なパラメータ設定は PAGE-EM と同じである．

三つのモデルで用いられる生成規則は以下のような規則である．

$$\begin{array}{ll} \mathcal{S} \to a\mathcal{S} & \mathcal{S} \to c\mathcal{S}\mathcal{S}\mathcal{S} \\ \mathcal{S} \to b\mathcal{S}\mathcal{S} & \mathcal{S} \to d\mathcal{S}\mathcal{S}\mathcal{S}\mathcal{S} \\ \mathcal{S} \to x & \end{array}$$

この場合の Royal Tree 問題の最大深さは $D_P = 5$ となり，最適解は 6144 である．各アルゴリズムを 50 回実行し，世代ごとの探索成功確率を観測した．

4.9.1.2 結果

図 4.6 は，各アノテーションサイズでの PAGE-EM と PAGE-VB，PCFG-GP の，適合度評価回数に対する探索成功確率を表す．この図から分かるように，アノテーションを一切用いない PCFG-GP は最適解を一度も獲得していないことが分かる．PCFG-GP が最適解を獲得していない理由は以下のように考えられる．木構造においては，終端ノードの割合が最も高いため，生成規則 $\mathcal{S} \to x$ の確率が最も高い．通常の PCFG を用いた場合，

大きなサイズの導出木の確率が小さくなるため，大きい構造の最適解はほとんど生成されない．一方，隠れアノテーションを用いた場合，それぞれのシンボルの生成に，別のアノテーションが用いられるため，高い確率で最適構造を生成することが可能である．表4.4はPAGE-EM（$h=8,16$）が推定した生成規則の確率を示す．$h=16$の場合から分かるように，$\mathcal{S}[5]$がd，$\mathcal{S}[10]$がcというように，別のシンボルの生成には，別のアノテーションが用いられている．このように，アノテーションを用いることで場所に依存しないノード間の依存関係を表すことが可能である．表4.4で灰色の生成規則は，ルートから最も高い確率の生成規則を選択していったものであり，これらの生成規則を用いることで最適構造を獲得することが出来る．PAGE-EMでは，アノテーション数が探索性能に大きな影響を及ぼす．図4.6にあるように，$h=4$より小さい数では，最適構造が獲得されていない．$h=16$では，最も少ない適合度評価回数で最適解を獲得している．表4.4には$h=8$の場合の生成規則も示している（左側）．$h=8$の場合から，$\mathcal{S}[2]$はaを生成すると同時に，他の関数ノードの生成も行っていることが分かる．このように重複して用いられるアノテーションが多く存在すると，部分構造の確率が小さくなり，部分構造の推定に不利である．そのため，ある程度以上のアノテーション数が必要となる．一方で，アノテーション数が大きすぎる場合，計算量が非常に多くなる．通常の生成規則に関して言えば，推定するべきパラメータ数はh^2に比例する．そのため，無駄に大きなアノテーションサイズを指定しても効率的に探索を行うことが出来ない．また，大きなアノテーションサイズでは，過学習の問題も起こる．アノテーションサイズを非常に大きくとれば，学習データのみを生成するような，生成規則が出来る．しかし，最適化アルゴリズムの観点から見ればまったく意味のないモデルである．PAGE-EMでは，あらかじめアノテーションサイズを与える必要があるが，PAGE-VBでは，対数尤度の下限$\mathcal{F}_h^*[Q]$を最大化するhを選択することで，学習データから自動的にアノテーションサイズを推定する．PAGE-VBの探索効率はPAGE-EMより低いが，PAGE-EMの$h=8$程度の評価回数で最適解を獲得できており，PAGE-EMにおいてアノテーション数決定に必要な計算量を考えると，有効な手法であると考えられる．特に，PAGE-EMでは$h=4$と$h=8$の探索性能の差は非常に大きい．計算量の面とのトレードオフを考慮するとPAGE-VBはバランスの取れたアノテーション数を選択したと考えられる．図4.7に，尤度の下限が，アノテーション数毎にどのようになっているかを示すグラフである．このグラフは，Sparse ExplorationやExploration Restrictionなどを用いず，$h=1,2,3,\cdots 15,16$のすべての場合で下限$\mathcal{F}_h^*[Q]$を計算したものである．この図から分かるように，$h=5$のところにピークがあるのが分かる．この試行では，PAGE-VBはアノテーション数を$h=5$程度に見積もっていることが分かる．

図4.8は，PAGE-EM（$h=16$）のEMアルゴリズムが対数尤度を反復毎に改善している様子である．この図は，世代0と世代5の図である．この図から，約10反復程度でパラメータが収束していることが分かる．対数尤度の上昇分が，世代5の場合の方が大きい．これは，世代が進むにつれ，優良解の構造が収束しつつあることを示す．図4.9はPAGE-VBにおける，対数尤度の下限$\mathcal{F}_h[Q]$の反復毎の変化を示したものである．世代0と世代6における値を示している．どちらの世代においても，約20反復程度で下限$\mathcal{F}_h[Q]$

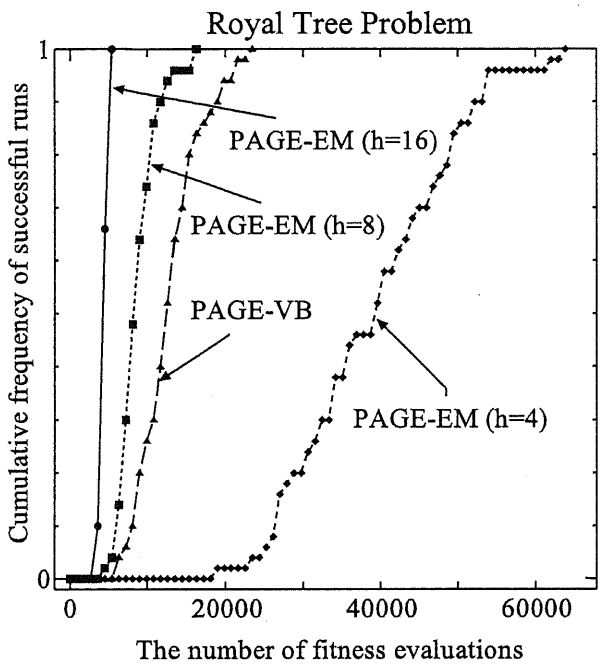


図 4.6: Royal Tree 問題における，探索成功確率.

の上昇が止まっている.

4.9.2 騙し最大値（DMAX）問題

PAGE-EM，PAGE-VB と他の関数進化アルゴリズムの比較を行うために，DMAX 問題に適用した．比較に用いたアルゴリズムは以下の通りである.

- **PAGE-EM**
  アノテーション数 $h = 8$，集団数 $M = 1000$，選択率 $P_s = 0.1$ と設定した.

- **PAGE-VB**
  初期アノテーション数の集合 $\mathcal{H}_0 = \{1, 2, 4, 8, 16\}$ と設定した．その他のパラメータに関しては，PAGE-EM と同じパラメータ設定を用いた.

- **PCFG-GP**
  基本的なパラメータ設定は，PAGE-EM と同じ設定とした.

- **Simple GP**
  Simple GP は一般的に用いられる GP である．GP のパラメータは $P_e = 0.01$（エリート率），$P_c = 0.9$（交叉率），$P_m = 0$（突然変異率），$P_r = 0.09$（Reproduction 率）としている．交叉点は以下のように決定される．二つの個体を交叉するにあたり，一つ目の交叉点は 0.9 の確率で関数ノードから，0.1 の確率で終端ノードから選

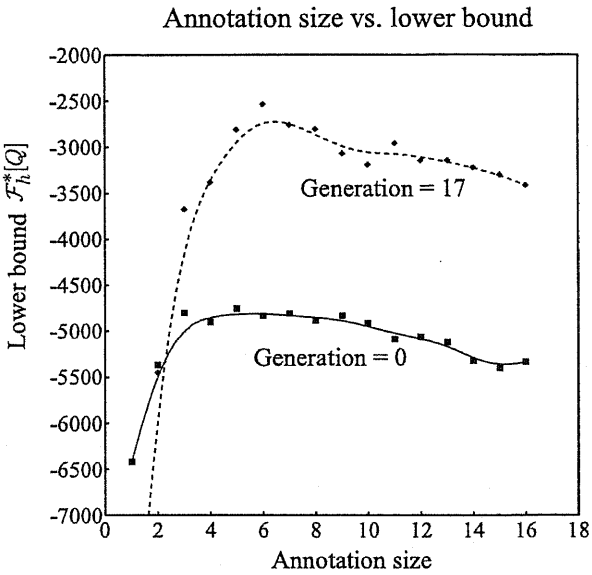


図 4.7: Royal Tree 問題での PAGE-VB の対数尤度の下限値の収束値 $\mathcal{F}_h^*[Q]$.

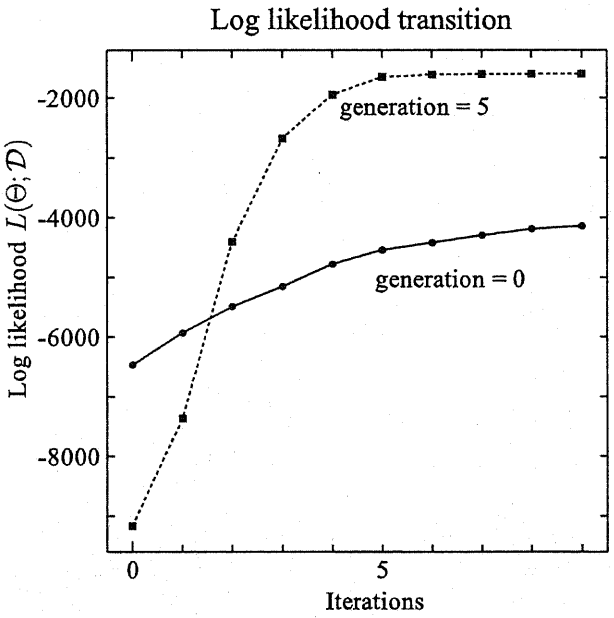


図 4.8: Royal Tree 問題における PAGE-EM の，EM アルゴリズム各ステップでの対数尤度の変化.

表 4.4: PAGE-EM（$h=8$, $h=16$）の推定した Royal Tree 問題の生成規則（左が $h=8$, 右が $h=16$）.

生成規則（PAGE-EM$h=8$）	確率
$\mathcal{S}[3]$	1.000
$\mathcal{S}[0] \to b\mathcal{S}[6]\mathcal{S}[6]$	0.018
$\mathcal{S}[0] \to c\mathcal{S}[6]\mathcal{S}[6]\mathcal{S}[6]$	0.972
$\mathcal{S}[1] \to x$	1.000
$\mathcal{S}[2] \to a\mathcal{S}[1]$	0.090
$\mathcal{S}[2] \to a\mathcal{S}[4]$	0.538
$\mathcal{S}[2] \to a\mathcal{S}[7]$	0.075
$\mathcal{S}[2] \to b\mathcal{S}[1]\mathcal{S}[1]$	0.014
$\mathcal{S}[2] \to b\mathcal{S}[4]\mathcal{S}[4]$	0.022
$\mathcal{S}[2] \to b\mathcal{S}[7]\mathcal{S}[7]$	0.025
$\mathcal{S}[2] \to c\mathcal{S}[4]\mathcal{S}[4]\mathcal{S}[4]$	0.059
$\mathcal{S}[2] \to c\mathcal{S}[7]\mathcal{S}[7]\mathcal{S}[7]$	0.032
$\mathcal{S}[2] \to d\mathcal{S}[4]\mathcal{S}[4]\mathcal{S}[4]\mathcal{S}[4]$	0.068
$\mathcal{S}[2] \to x$	0.061
$\mathcal{S}[3] \to d\mathcal{S}[0]\mathcal{S}[0]\mathcal{S}[0]\mathcal{S}[0]$	1.000
$\mathcal{S}[4] \to x$	1.000
$\mathcal{S}[5] \to a\mathcal{S}[1]$	0.161
$\mathcal{S}[5] \to a\mathcal{S}[4]$	0.468
$\mathcal{S}[5] \to a\mathcal{S}[7]$	0.356
$\mathcal{S}[6] \to a\mathcal{S}[2]$	0.012
$\mathcal{S}[6] \to b\mathcal{S}[2]\mathcal{S}[2]$	0.467
$\mathcal{S}[6] \to b\mathcal{S}[5]\mathcal{S}[5]$	0.399
$\mathcal{S}[6] \to c\mathcal{S}[2]\mathcal{S}[2]\mathcal{S}[2]$	0.030
$\mathcal{S}[6] \to d\mathcal{S}[2]\mathcal{S}[2]\mathcal{S}[2]\mathcal{S}[2]$	0.055
$\mathcal{S}[6] \to x$	0.029
$\mathcal{S}[7] \to x$	1.000

生成規則（PAGE-EM$h=16$）	確率
$\mathcal{S}[5]$	1.000
$\mathcal{S}[0] \to x$	1.000
$\mathcal{S}[1] \to x$	1.000
$\mathcal{S}[2] \to x$	1.000
$\mathcal{S}[3] \to c\mathcal{S}[13]\mathcal{S}[13]\mathcal{S}[13]$	0.063
$\mathcal{S}[3] \to c\mathcal{S}[14]\mathcal{S}[14]\mathcal{S}[14]$	0.131
$\mathcal{S}[3] \to c\mathcal{S}[9]\mathcal{S}[9]\mathcal{S}[9]$	0.093
$\mathcal{S}[3] \to d\mathcal{S}[14]\mathcal{S}[14]\mathcal{S}[14]\mathcal{S}[14]$	0.054
$\mathcal{S}[3] \to d\mathcal{S}[9]\mathcal{S}[9]\mathcal{S}[9]\mathcal{S}[9]$	0.060
$\mathcal{S}[3] \to x$	0.329
$\mathcal{S}[4] \to a\mathcal{S}[1]$	0.154
$\mathcal{S}[4] \to a\mathcal{S}[14]$	0.257
$\mathcal{S}[4] \to a\mathcal{S}[2]$	0.189
$\mathcal{S}[4] \to a\mathcal{S}[9]$	0.238
$\mathcal{S}[5] \to d\mathcal{S}[10]\mathcal{S}[10]\mathcal{S}[10]\mathcal{S}[10]$	1.000
$\mathcal{S}[6] \to a\mathcal{S}[14]$	0.144
$\mathcal{S}[6] \to a\mathcal{S}[9]$	0.092
$\mathcal{S}[6] \to b\mathcal{S}[1]\mathcal{S}[1]$	0.070
$\mathcal{S}[6] \to b\mathcal{S}[2]\mathcal{S}[2]$	0.079
$\mathcal{S}[6] \to d\mathcal{S}[14]\mathcal{S}[14]\mathcal{S}[14]\mathcal{S}[14]$	0.167
$\mathcal{S}[7] \to a\mathcal{S}[14]$	0.152
$\mathcal{S}[7] \to a\mathcal{S}[9]$	0.184
$\mathcal{S}[7] \to c\mathcal{S}[9]\mathcal{S}[9]\mathcal{S}[9]$	0.071
$\mathcal{S}[7] \to x$	0.319
$\mathcal{S}[8] \to b\mathcal{S}[11]\mathcal{S}[11]$	0.188
$\mathcal{S}[8] \to b\mathcal{S}[4]\mathcal{S}[4]$	0.397
$\mathcal{S}[8] \to b\mathcal{S}[6]\mathcal{S}[6]$	0.079
$\mathcal{S}[8] \to c\mathcal{S}[3]\mathcal{S}[3]\mathcal{S}[3]$	0.053
$\mathcal{S}[9] \to x$	1.000
$\mathcal{S}[10] \to c\mathcal{S}[15]\mathcal{S}[15]\mathcal{S}[15]$	0.900
$\mathcal{S}[10] \to c\mathcal{S}[8]\mathcal{S}[8]\mathcal{S}[8]$	0.093
$\mathcal{S}[11] \to a\mathcal{S}[1]$	0.167
$\mathcal{S}[11] \to a\mathcal{S}[12]$	0.080
$\mathcal{S}[11] \to a\mathcal{S}[13]$	0.181
$\mathcal{S}[11] \to a\mathcal{S}[14]$	0.209
$\mathcal{S}[11] \to a\mathcal{S}[2]$	0.143
$\mathcal{S}[11] \to a\mathcal{S}[9]$	0.208
$\mathcal{S}[12] \to x$	0.993
$\mathcal{S}[13] \to x$	1.000
$\mathcal{S}[14] \to x$	1.000
$\mathcal{S}[15] \to b\mathcal{S}[11]\mathcal{S}[11]$	0.648
$\mathcal{S}[15] \to b\mathcal{S}[4]\mathcal{S}[4]$	0.239

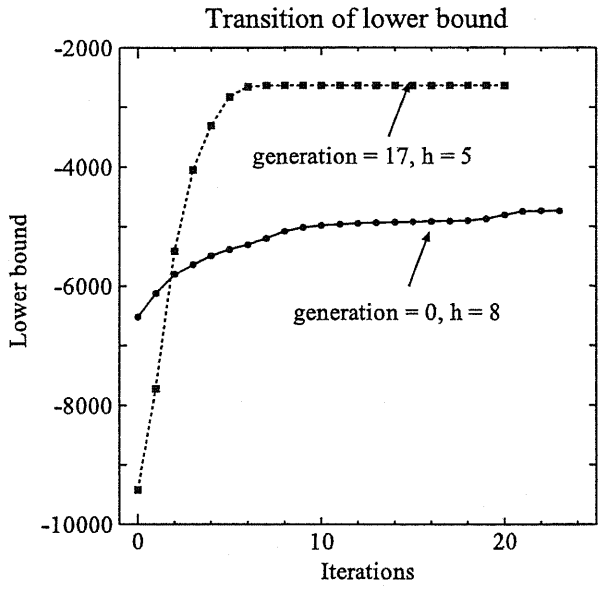


図 4.9: Royal Tree 問題における PAGE-VB の，変分ベイズ法各ステップでの下限 $\mathcal{F}_h[Q]$ の変化.

択される．二つめの交叉点は，二つの個体が深さ制限を越えないように，ランダムに選択される．週段数は $M = 16000$ であり，トーナメントサイズは $T_s = 2$ と設定した.

- **POLE**
  本論文の提案手法の一つである（3 章参照）．プロトタイプ木に基づく GP-EDA であり，拡張構文木によって個体を表現し，依存関係をベイジアンネットワークによって推定する．集団数 $M = 6300$，選択率 $P_s = 0.1$，エリート率 $P_e = 0.005$，親レンジ $R_P = 2$ と設定した.

各アルゴリズムを 50 回実行し，世代ごとの探索成功確率を観測した.

4.9.2.1 問題設定

騙し最大値（DMAX:Deceptive MAX）問題は，最大値問題（MAX problem）[Gathercole 96, Langdon 02] を拡張した問題である（3.6.2 節参照）．本節で用いる DMAX 問題は，3.6.2 節の設定と同一である．DMAX 問題では以下のノードを用いる.

$$\mathfrak{F} = \{+_m, \times_m\} \quad \mathfrak{T} = \{\lambda, 0.95\} \tag{4.65}$$

$$\lambda = \cos\frac{2\pi}{r} + i\sin\frac{2\pi}{r} \tag{4.66}$$

$+_m$ 及び $\times_m$ は m 引数関数であり，それぞれ m 引数の足し算とかけ算を定義する．$\lambda \in \mathbb{C}$ は絶対値 1 の複素数であり，$(\lambda)^r = 1$ を満たす．DMAX 問題の難しさは 3 つのパラメータ m，r，D_P によって決まる．本論文では，$m = 5$，$r = 3$，$D_P = 4$ の場合について実験を行う（3.6.2 節の設定と同一である）．

PAGE-EM，PAGE-VB，PCFG-GP で用いられる生成規則は DMAX 問題に必要なノード（式 4.65）から，以下で表される規則である．

$$\mathcal{S} \to +_5 \mathcal{S}\mathcal{S}\mathcal{S}\mathcal{S}\mathcal{S} \qquad \mathcal{S} \to \times_5 \mathcal{S}\mathcal{S}\mathcal{S}\mathcal{S}\mathcal{S}$$
$$\mathcal{S} \to 0.95 \qquad \mathcal{S} \to \lambda$$

4.9.2.2 結果

DMAX 問題における世代ごとの探索成功頻度を図 4.10 に示す．PCFG-GP は一度も最適解を獲得することが出来なかったため，図からは除外している．DMAX 問題では，二つの種類の部分構造を組み合わせる必要がある．一つは，0.95 が 5 つで組み合わされたものであり，もう一つは λ が 5 つで組み合わされたものである．生成規則の中で，0.95 と λ を生成する非終端記号が PCFG-GP の場合同じであるため，PCFG-GP は二つのシンボルを区別して生成することが出来ない．そのため，0.95 と λ の組合わさった構造が生成されてしまう（例えば，$\lambda + \lambda + 0.95 + \lambda + 0.95$）．これが原因で，アノテーションを用いない PCFG-GP は探索に成功していない．一方，PAGE-EM と PAGE-VB においては，0.95 と λ の生成に別のアノテーションが付くことで，別々に生成することが可能である．表 4.6 は PAGE-EM（$h = 8$）が推定した生成規則の確率である．この表から分かるように，λ と 0.95 は別の非終端記号から生成されている．$\mathcal{S}[0], \mathcal{S}[1], \mathcal{S}[3], \mathcal{S}[5]$ は λ を生成しており，$\mathcal{S}[7]$ は 0.95 を生成している．この生成規則のもとでは，λ と 0.95 の混合した構造はほとんど生成されない．

図 4.10 から見て取れるように，PAGE-EM と PAGE-VB は Simple GP や POLE と比較して，非常に少ない適合度評価回数で最適解を獲得している．3.6.2 節でも述べたように，DMAX 問題は，SGP を用いた場合，非常に強い騙し性を持つ．この騙し性のため，一度局所解にとらわれると，SGP が局所解から脱出するのは非常に困難である．表 4.5 は，探索成功確率が 90%以上になるのに必要な適合度評価回数を表している．

3 章で提案した POLE は SGP より少ない適合度評価回数で最適解を獲得している．POLE はベイジアンネットワークを推定することで，ノード間の依存関係を推定することが出来るため，DMAX 問題における依存関係の推定に成功している．しかし，POLE で推定することが可能な部分構造は，場所依存である．DMAX 問題には二つの部分構造 $\lambda + \lambda + \lambda + \lambda + \lambda$ と $0.95 + 0.95 + 0.95 + 0.95 + 0.95$ があるが，POLE では，各場所（正確には 25 箇所）で，二つの部分構造を推定する必要がある．この二つの部分構造は，深さ以外に絶対的場所に対する制限はない．そのため，POLE の部分構造の推定は非常に効率が悪くなっている．提案手法である，PAGE-EM と PAGE-VB は場所に依存しない部分構造の推定が可能である．この違いが，PAGE-EM，PAGE-VB と POLE の探索性能の違いを生んでいると思われる．

表 4.5: 累積探索成功確率が 90% となるのに必要な適合度評価回数.

	prob	#eval/gen	# of eval
PAGE-EM	90%	1000×0.9	17100
PAGE-VB	90%	1000×0.9	20700
PCFG-GP	–	–	–
POLE	90%	6300×0.995	119102
SGP	90%	16000×0.995	2228800

表 4.6: 推定された DMAX 問題の生成規則.

生成規則 (PAGE-EM ($h = 8$))	確率
$\mathcal{S}[4]$	1.000
$\mathcal{S}[0] \rightarrow \lambda$	0.983
$\mathcal{S}[0] \rightarrow 0.95$	0.017
$\mathcal{S}[1] \rightarrow \lambda$	0.941
$\mathcal{S}[1] \rightarrow 0.95$	0.059
$\mathcal{S}[2] \rightarrow \times_5\mathcal{S}[6]\mathcal{S}[6]\mathcal{S}[6]\mathcal{S}[6]\mathcal{S}[6]$	1.000
$\mathcal{S}[3] \rightarrow \lambda$	0.989
$\mathcal{S}[3] \rightarrow 0.95$	0.011
$\mathcal{S}[4] \rightarrow \times_5\mathcal{S}[2]\mathcal{S}[2]\mathcal{S}[2]\mathcal{S}[2]\mathcal{S}[2]$	1.000
$\mathcal{S}[5] \rightarrow \lambda$	0.995
$\mathcal{S}[6] \rightarrow +_5\mathcal{S}[0]\mathcal{S}[0]\mathcal{S}[0]\mathcal{S}[0]\mathcal{S}[0]$	0.399
$\mathcal{S}[6] \rightarrow +_5\mathcal{S}[1]\mathcal{S}[1]\mathcal{S}[1]\mathcal{S}[1]\mathcal{S}[1]$	0.010
$\mathcal{S}[6] \rightarrow +_5\mathcal{S}[3]\mathcal{S}[3]\mathcal{S}[3]\mathcal{S}[3]\mathcal{S}[3]$	0.234
$\mathcal{S}[6] \rightarrow +_5\mathcal{S}[5]\mathcal{S}[5]\mathcal{S}[5]\mathcal{S}[5]\mathcal{S}[5]$	0.096
$\mathcal{S}[6] \rightarrow +_5\mathcal{S}[7]\mathcal{S}[7]\mathcal{S}[7]\mathcal{S}[7]\mathcal{S}[7]$	0.261
$\mathcal{S}[7] \rightarrow \lambda$	0.025
$\mathcal{S}[7] \rightarrow 0.95$	0.975

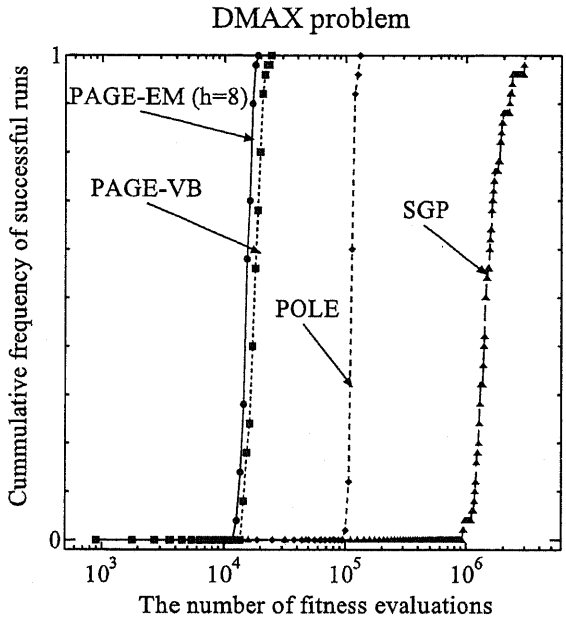


図 4.10: DMAX 問題での，世代ごとの探索成功確率．PCFG-GP は最適解を獲得していないため，図からは省いている．

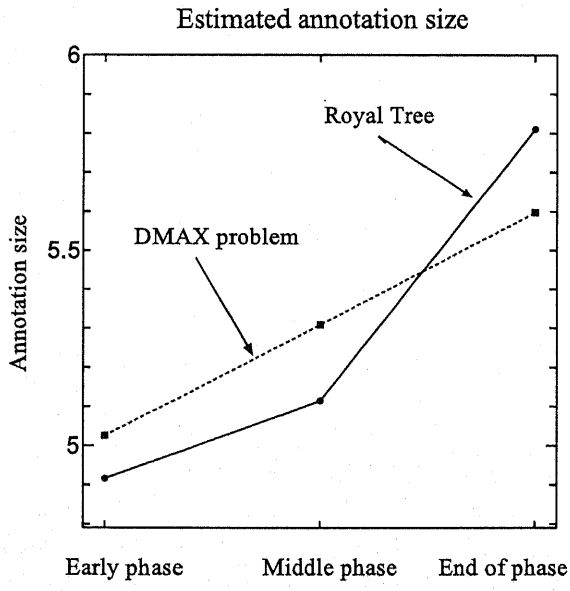


図 4.11: 推定されたアノテーション数の遷移．"Early phase"，"Middle phase"，"End of phase" は各試行の世代を三等分した領域を表す．

4.10 考察

本章で提案したPAGEは，PCFG-LAを基本文法として用いた．PCFG-LAは隠れ変数を有するモデルであるため，推定はEMアルゴリズム及び変分ベイズ法によって行われる．PCFG-LAの提案論文[Matsuzaki 05]では，EMアルゴリズムが用いられている．このPCFG-LAに基づくGP-EDAとしては，PAGE-EMを提案した．アノテーションサイズが適切に与えられた場合，PAGE-EMの探索性能は非常に高いことが，Royal Tree問題及びDMAX問題で示された．一方で，アノテーション数をどうように設定するかで，探索性能が大きく変わることも見て取れる．Royal Tree問題では，少ないアノテーション数（$h = 2, 4$）を用いた場合，非常に最適解の獲得が遅い．一方で，多すぎるアノテーション数を採用した場合，計算量が多くなるため探索には不利であると同時に，過学習の問題も生じる．そのため，PAGE-EMでアノテーション数を決定するためには，試行錯誤が必要である．PAGE-VBは，対数尤度の下限$\mathcal{F}_h^*[Q]$を計算することでアノテーション数を決定する．図4.11は，Royal Tree問題及びDMAX問題において，各試行を世代毎に三等分し，その区分毎に平均アノテーションサイズを計算したものである．この図がら分かるように，PAGE-VBは後の世代ほど，大きなアノテーションサイズを選択していることが分かる．これは，世代が進むにつれ，優良解の構造が大きくなると同時に，優良解の構造が収束し，自由度の高いモデル（大きいアノテーションサイズ）を用いた場合でも，学習に必要なサンプル数があることを示している．これらの結果から，PAGE-VBのアノテーションサイズ推定は正しく働いていることが分かる．DMAX問題においては，PAGE-EMとPAGE-VBの探索能力の差は小さいものの，Royal Tree問題ではPAGE-EM（$h = 16$）とPAGE-VBの間には，数倍の差がある．PAGE-VBでは，モデルの事前分布を$P(h) \propto 1$としている．より高い探索能力とするためには，この事前分布の最適化も必要である．ベイジアンネットワークの研究では，グラフ構造の事前分布として，エッジ数によるヒューリスティクスを考慮した事前分布などが用いられている．

本論文の3章で示したように，POLEはプロトタイプ木を用いた手法としては最新であり，DMAX問題においても非常に高い探索性能を示している（3.6.2節参照）．プロトタイプ木を用いたアプローチの場合，場所非依存の部分構造の推定が出来ない．POLEにおいても同様で，DMAXの部分構造に関しても，場所に依存した状態で学習している．

4.11 おわりに

本論文では，PCFG-LAを基本文法として用いた分布推定アルゴリズムPAGEを提案した．PAGEにおいては，アノテーションは優良解より推定されるため，親ノード，兄弟ノードや深さなどをアノテーションとして用いる場合より，柔軟なアルゴリズムである．PAGEをRoyal Tree問題及びDMAX問題に適用することで，親ノード，深さをアノテーションとして用いた場合より，高い探索性能があることが示された．また，GPと比較しても，少ない世代で，高い探索成功確率を示した．GPは非常に多くの分野に対して適用されているが，PAGEの実問題への適用などはこれからの課題である．

4.12 付録：パラメータ更新規則の導出

ここでは，パラメータ更新規則の導出の説明を行う（式 4.31 及び 4.32 の導出）．基本的な導出法は文献 [Matsuzaki 05] と同様である．

式 4.16 を用いることで，$\mathcal{Q}(\overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}|\boldsymbol{\beta}, \boldsymbol{\pi})$ は式 4.67 へと書き換え可能である．

$$\mathcal{Q}(\overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}|\boldsymbol{\beta}, \boldsymbol{\pi}) = \mathcal{Q}(\overline{\boldsymbol{\beta}}|\boldsymbol{\beta}, \boldsymbol{\pi}) + \mathcal{Q}(\overline{\boldsymbol{\pi}}|\boldsymbol{\beta}, \boldsymbol{\pi}) \tag{4.67}$$

$$\mathcal{Q}(\overline{\boldsymbol{\beta}}|\boldsymbol{\beta}, \boldsymbol{\pi}) = \sum_{T_i \in \mathcal{D}} \frac{1}{P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, h)} \sum_{X_i} P(T_i, X_i; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \sum_{r \in \mathcal{R}[H]} c(r; T_i, X_i) \log \overline{\beta}(r) \tag{4.68}$$

$$\mathcal{Q}(\overline{\boldsymbol{\pi}}|\boldsymbol{\beta}, \boldsymbol{\pi}) = \sum_{T_i \in \mathcal{D}} \frac{1}{P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, h)} \sum_{X_i} P(T_i, X_i; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \log \overline{\pi}(\mathcal{S}[x_i^1]) \tag{4.69}$$

式 4.67 は式 4.68 と式 4.69 で表される二つの項 $\mathcal{Q}(\overline{\boldsymbol{\beta}}|\boldsymbol{\beta}, \boldsymbol{\pi})$，$\mathcal{Q}(\overline{\boldsymbol{\pi}}|\boldsymbol{\beta}, \boldsymbol{\pi})$ から構成される．二つの項は，それぞれ独立して最適化することが出来る．付録では $\overline{\boldsymbol{\beta}}$ の導出のみ示す．$\overline{\boldsymbol{\pi}}$ の導出は $\overline{\boldsymbol{\beta}}$ の導出と同様に行うことが出来る（$\overline{\boldsymbol{\pi}}$ の導出の方が容易である）．

$\mathcal{Q}(\overline{\boldsymbol{\beta}}|\boldsymbol{\beta}, \boldsymbol{\pi})$ をさらに，$g \in \mathcal{T}$ ごとに分解すると式 4.68 は式 4.70 となる．

$$\mathcal{Q}(\overline{\boldsymbol{\beta}}|\boldsymbol{\beta}, \boldsymbol{\pi}) = \sum_{g \in \mathcal{T}} \mathcal{Q}_g(\overline{\boldsymbol{\beta}}|\boldsymbol{\beta}, \boldsymbol{\pi}) \tag{4.70}$$

x_i^j を木 T_i の j 番目の非終端記号のアノテーションとし，y_i^j をその右辺のアノテーションとする．

$$\begin{aligned}
&\mathcal{Q}_g(\overline{\boldsymbol{\beta}}|\boldsymbol{\beta}, \boldsymbol{\pi}) = \sum_{T_i \in \mathcal{D}} \frac{1}{P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, h)} \sum_{j \in cover(g, T_i)} \sum_{\mathbf{X}_i} P(T_i, X_i; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \\
&\times \log \overline{\beta}(\mathcal{S}[x_i^j] \to g\, \mathcal{S}[y_i^j] \cdots \mathcal{S}[y_i^j]) \\
&= \sum_{T_i \in \mathcal{D}} \frac{1}{P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, h)} \\
&\times \sum_{j \in cover(g, T_i)} \sum_{x_i^j, y_i^j \in H} \log \overline{\beta}(\mathcal{S}[x_i^j] \to g\, \mathcal{S}[y_i^j] \cdots \mathcal{S}[y_i^j]) \\
&\times \sum_{X_i \setminus x_i^j, y_i^j} P(T_i, X_i; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \\
&= \sum_{T_i \in \mathcal{D}} \frac{1}{P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, h)} \\
&\times \sum_{j \in cover(g, T_i)} \sum_{x, y \in H} \log \overline{\beta}(\mathcal{S}[x] \to g\, \mathcal{S}[y] \cdots \mathcal{S}[y]) \\
&\times \beta(\mathcal{S}[x] \to g\, \mathcal{S}[y] \cdots \mathcal{S}[y]) f_{T_i}^j(x; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \prod_{k \in ch(j, T_i)} b_{T_i}^k(y; \boldsymbol{\beta}, \boldsymbol{\pi}, h)
\end{aligned}$$

最後の式変形には式4.27を用いた．$\mathcal{Q}(\overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}|\boldsymbol{\beta}, \boldsymbol{\pi})$ を制約条件

$$\sum_{\zeta \in \mathcal{R}_r[H]} \overline{\beta}(\mathcal{S}[x] \to \zeta) = 1$$

で最大化するために，ラグランジュの未定乗数法が用いられる（式4.71及び4.72．μ はラグランジュ乗数）．

$$\mathcal{L} = \mathcal{Q}(\overline{\boldsymbol{\beta}}|\boldsymbol{\beta}, \boldsymbol{\pi}) + \mu \left\{ 1 - \sum_{\zeta \in \mathcal{R}_r[H]} \overline{\beta}(\mathcal{S}[x] \to \zeta) \right\} \tag{4.71}$$

$$\frac{\partial \mathcal{L}}{\partial \overline{\beta}(\mathcal{S}[x] \to g\, \mathcal{S}[y] \cdots \mathcal{S}[y])} = 0 \tag{4.72}$$

式4.72を解くと，

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial \overline{\beta}(\mathcal{S}[x] \to g\, \mathcal{S}[y] \cdots \mathcal{S}[y])} \\
&= \frac{\beta(\mathcal{S}[x] \to g\, \mathcal{S}[y] \cdots \mathcal{S}[y])}{\overline{\beta}(\mathcal{S}[x] \to g\, \mathcal{S}[y] \cdots \mathcal{S}[y])} \\
&\times \sum_{T_i \in \mathcal{D}} \frac{1}{P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi})} \sum_{j \in cover(g, T_i)} f^j_{T_i}(x; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \prod_{k \in ch(j, T_i)} b^k_{T_i}(y; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \\
&- \mu
\end{aligned}$$

であるから，パラメータ更新規則式4.31及び4.32が得られる．$\overline{\boldsymbol{\pi}}$ の更新規則も，$Q(\overline{\boldsymbol{\pi}}|\boldsymbol{\beta}, \boldsymbol{\pi})$ を最大化することで，同様に得ることが出来る．

4.13 付録：和に関する部分の計算

式4.52及び式4.49に含まれる，アノテーションに関する和は，ノード数が増えると計算量が指数関数的に増加するため，そのままでは計算することが出来ない．ここでは，動的計画法を用いた計算方法を示す．

4.13.1 β に関する項

式4.49の式4.73をまず考える．

$$\widetilde{c}(\mathcal{S}[x] \to \alpha; T_i) = \sum_{X_i} Q(X_i|h) \cdot c(\mathcal{S}[x] \to \alpha; T_i, X_i) \tag{4.73}$$

まず，最尤推定の場合を考えてみる．その場合，

$$
\begin{aligned}
c(\mathcal{S}[x] \to \alpha; T_i) &= \sum_{X_i} P(X_i|T_i;\boldsymbol{\beta},\boldsymbol{\pi},h) c(\mathcal{S}[x] \to \alpha; T_i, X_i) \\
&= \sum_{X_i} \frac{P(T_i, X_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}{P(T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)} c(\mathcal{S}[x] \to \alpha; T_i, X_i) \qquad (4.74)
\end{aligned}
$$

式4.16を$\beta(S[x] \to \alpha)$で偏微分する．

$$
\frac{\partial P(T_i, X_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}{\partial \beta(\mathcal{S}[x] \to \alpha)} = \frac{c(\mathcal{S}[x] \to \alpha; T_i, X_i)}{\beta(\mathcal{S}[x] \to \alpha)} P(T_i, X_i;\boldsymbol{\beta},\boldsymbol{\pi},h)
$$

式4.74を代入すると，

$$
\frac{\beta(\mathcal{S}[x] \to \alpha)}{P(T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)} \frac{\partial P(T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}{\partial \beta(\mathcal{S}[x] \to \alpha)} = c(\mathcal{S}[x] \to \alpha; T_i)
$$

となる．ここで，左辺の偏微分の項は以下のように計算される．

$$
\frac{\partial P(T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}{\partial \beta(\mathcal{S}[x] \to g\,\mathcal{S}[y]...\mathcal{S}[y])} = \sum_{i \in cover(g,T)} f_T^i(x;\boldsymbol{\beta},\boldsymbol{\pi},h) \prod_{j \in ch(i,T)} b_T^j(y;\boldsymbol{\beta},\boldsymbol{\pi},h)
$$

ここで$f_T^i(y;\boldsymbol{\beta},\boldsymbol{\pi},h)$，$b_T^i(j;\boldsymbol{\beta},\boldsymbol{\pi},h)$は前向き・後向き確率である．

$$
\begin{aligned}
f_T^i(y;\boldsymbol{\beta},\boldsymbol{\pi},h) &= \sum_{x \in H} f_T^{pa(i,T)}(x;\boldsymbol{\beta},\boldsymbol{\pi},h) \beta(\mathcal{S}[x] \to g_T^{pa(i,T)} \mathcal{S}[y]...\mathcal{S}[y]) \\
&\quad \times \prod_{j \in ch(pa(i,T),T), j \neq i} b_T^j(y;\boldsymbol{\beta},\boldsymbol{\pi},h)
\end{aligned}
$$

一方，変分ベイズの場合を見ると，$\boldsymbol{\beta},\boldsymbol{\pi}$と$\widetilde{\boldsymbol{\beta}},\widetilde{\boldsymbol{\pi}}$各パラメータが入れ替わった形である．これの類推から，

$$
\begin{aligned}
&\widetilde{c}(\mathcal{S}[x] \to g\,\mathcal{S}[y]\,\mathcal{S}[y]\cdots\mathcal{S}[y]; T_i) \\
&= \frac{\widetilde{\beta}(\mathcal{S}[x] \to g\,\mathcal{S}[y]\,\mathcal{S}[y]\cdots\mathcal{S}[y])}{\mathcal{Z}(T_i)} \\
&\times \sum_{k \in cover(g,T_i)} f_{T_i}^k(x;\widetilde{\boldsymbol{\beta}},\widetilde{\boldsymbol{\pi}},h) \prod_{j \in ch(k,T_i)} b_{T_i}^j(y;\widetilde{\boldsymbol{\beta}},\widetilde{\boldsymbol{\pi}},h) \qquad (4.75)
\end{aligned}
$$

と計算される．

4.13.2 πに関する項

式4.52に関する項についても同様である．まず，最尤推定の場合を考える．

$$\delta(x;T_i)=\sum_{X_i}P(X_i|T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)\delta(x;T_i,X_i)=\sum_{X_i}\frac{P(T_i,X_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}{P(T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}\delta(x;T_i,X_i)$$

β の場合とほとんど同様に,

$$\frac{\pi(\mathcal{S}[x])}{P(T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}\frac{\partial P(T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}{\partial\pi(\mathcal{S}[x])}=\delta(x;T_i)$$

と計算される. これの類推から, $\widetilde{\delta}$ も同様に計算される.

$$\widetilde{\delta}(x;T_i)=\sum_{X_i}Q(X_i|h)\delta(x;T_i,X_i)$$

以上より,

$$\widetilde{\delta}(x;T_i)=\frac{\widetilde{\pi}(\mathcal{S}[x])}{\mathcal{Z}(T_i)}\frac{\partial\mathcal{Z}(T_i)}{\partial\widetilde{\pi}(\mathcal{S}[x])}=\frac{\widetilde{\pi}(\mathcal{S}[x])}{\mathcal{Z}(T_i)}b^1_{T_i}(x;\widetilde{\boldsymbol{\beta}},\widetilde{\boldsymbol{\pi}},h) \tag{4.76}$$

となる.

4.14 付録：$\mathcal{F}_h[Q]$ 値の計算

$\mathcal{F}_h[Q]$ の値は式 4.77 で表される.

$$\mathcal{F}^*_h[Q]=\sum_{i=1}^{N}\log\mathcal{Z}(T_i|h)-\int d\boldsymbol{\beta}Q^*(\boldsymbol{\beta}|h)\log\frac{Q^*(\boldsymbol{\beta}|h)}{P(\boldsymbol{\beta}|h)}-\int d\boldsymbol{\pi}Q^*(\boldsymbol{\pi}|h)\log\frac{Q^*(\boldsymbol{\pi}|h)}{P(\boldsymbol{\pi}|h)} \tag{4.77}$$

この式は以下のように導出される.

$$\begin{aligned}
\mathcal{F}^*_h[Q] &= \sum_{\mathbf{X}}\int d\boldsymbol{\beta}d\boldsymbol{\pi}q^*(\mathbf{X},\boldsymbol{\beta},\boldsymbol{\pi}|h)\log\frac{P(\mathcal{D},\mathbf{X},\boldsymbol{\beta},\boldsymbol{\pi}|h)}{Q^*(\mathbf{X},\boldsymbol{\beta},\boldsymbol{\pi}|h)}\\
&= \sum_{\mathbf{X}}\int d\boldsymbol{\beta}d\boldsymbol{\pi}Q^*(\boldsymbol{\beta}|h)Q^*(\boldsymbol{\pi}|h)Q^*(\mathbf{X}|h)\log\frac{P(\mathcal{D},\mathbf{X}|\boldsymbol{\beta},\boldsymbol{\pi},h)P(\boldsymbol{\beta}|h)P(\boldsymbol{\pi}|h)}{Q^*(\boldsymbol{\beta}|h)q^*(\boldsymbol{\pi}|h)Q^*(\mathbf{X}|h)}\\
&= \int d\boldsymbol{\beta}Q^*(\boldsymbol{\beta}|h)\log\frac{P(\boldsymbol{\beta}|h)}{Q^*(\boldsymbol{\beta}|h)}+\int d\boldsymbol{\pi}Q^*(\boldsymbol{\pi}|h)\log\frac{P(\boldsymbol{\pi}|h)}{Q^*(\boldsymbol{\pi}|h)}\\
&- \sum_{\mathbf{X}}Q^*(\mathbf{X}|h)\log Q^*(\mathbf{X}|h)\\
&+ \langle\log P(\mathcal{D},\mathbf{X}|\beta,\pi,h)\rangle_{Q^*(\mathbf{X}|h),Q^*(\boldsymbol{\beta}|h),Q^*(\boldsymbol{\pi}|h)}
\end{aligned} \tag{4.78}$$

式 4.53 と 4.54 から, 変形して,

$$\log Q^*(X_i|h) = \int d\pi d\beta Q^*(\pi|h)Q^*(\beta|h)\log P(T_i, X_i|\beta, \pi, h) - \log \mathcal{Z}(T_i)$$

$$\log Q^*(\mathbf{X}|h) = \int d\pi d\beta Q^*(\pi|h)Q^*(\beta|h)\log P(\mathcal{D}, \mathbf{X}|\beta, \pi, h) - \sum_{i=1}^{N}\log \mathcal{Z}(T_i)$$

これを式 4.78 に代入して，

$$\mathcal{F}_h^*[Q] = \int d\beta Q^*(\beta|h)\log\frac{P(\beta|h)}{Q^*(\beta|h)} + \int d\pi Q^*(\pi|h)\log\frac{P(\pi|h)}{Q^*(\pi|h)} + \sum_{i=1}^{N}\log \mathcal{Z}(T_i)$$

$$\mathcal{F}_h^*[Q] = \sum_{i=1}^{N}\log \mathcal{Z}(T_i) - \int d\beta Q^*(\beta|h)\log\frac{Q^*(\beta|h)}{P(\beta|h)} - \int d\pi Q^*(\pi|h)\log\frac{Q^*(\pi|h)}{P(\pi|h)}$$

となる．Digamma 関数を用いることで，以下のように表すことが出来る．

$$\begin{aligned}
\int d\beta Q^*(\beta|h)\log\frac{Q^*(\beta|h)}{P(\beta|h)} &= \sum_{x\in H}\left\{\sum_{\alpha\in\mathcal{R}_r[H]}\left(\log\Gamma(b_{\mathcal{S}[x]\to\alpha}) - \log\Gamma(\breve{b}^*_{\mathcal{S}[x]\to\alpha})\right)\right. \\
&+ \left.\log\Gamma\left(\sum_{\alpha\in\mathcal{R}_r[H]}\breve{b}^*_{\mathcal{S}[x]\to\alpha}\right) - \log\Gamma\left(\sum_{\alpha\in\mathcal{R}_r[H]}b_{\mathcal{S}[x]\to\alpha}\right)\right\} \\
&+ \sum_{x\in H}\sum_{\alpha\in\mathcal{R}_r[H]}\left(\breve{b}^*_{\mathcal{S}[x]\to\alpha} - b_{\mathcal{S}[x]\to\alpha}\right) \\
&\times \left\{\psi(\breve{b}^*_{\mathcal{S}[x]\to\alpha}) - \psi\left(\sum_{\alpha'\in\mathcal{R}_r[H]}\breve{b}^*_{\mathcal{S}[x]\to\alpha'}\right)\right\}
\end{aligned}$$

$$\begin{aligned}
\int d\pi Q^*(\pi|h)\ln\frac{Q^*(\pi|h)}{P(\pi|h)} &= \sum_{x\in H}\left(\log\Gamma(a_{\mathcal{S}[x]}) - \log\Gamma(\breve{a}^*_{\mathcal{S}[x]})\right) \\
&+ \log\Gamma\left(\sum_{x\in H}\breve{a}^*_{\mathcal{S}[x]}\right) - \log\Gamma\left(\sum_{x\in H}a_{\mathcal{S}[x]}\right) \\
&+ \sum_{x\in H}(\breve{a}^*_{\mathcal{S}[x]} - a_{\mathcal{S}[x]})\left\{\psi(\breve{a}^*_{\mathcal{S}[x]}) - \psi\left(\sum_{x'\in H}\breve{a}^*_{\mathcal{S}[x']}\right)\right\}
\end{aligned}$$

第5章　結論

5.1　まとめ

本論文では，GP-Hard な問題を対象とした，確率的な関数進化アルゴリズムを提案した．GP はその適用範囲の広さから，広範な適用が行われている．その一方で，統計的手法に基づく関数最適化アルゴリズムの研究はあまり行われていなかった．既に提案されている GP-EDA に関しても，それらの研究が焦点を当てているのは，「通常の GP でも比較的効率的に解ける問題を，より少ない適合度評価回数と解く」という側面が強い．一方，本論文が対象としているのは，GP では解きにくい（非常に多くの適合度評価回数がいる，又はほとんど解けない）という問題である．GA は，その対象とする問題が主にパラメータ最適化であり，比較的適用しやすい．一方で，GP は表現力が GA より遥かに高いという利点があると同時に，適用方法の難しさの問題がある．それゆえ，GP の実問題への適用は，GA の場合と比較して進んでいない．GA のような固定長一次元配列では，依存関係を扱う EDA の登場により，従来型の GA では解きにくい問題に対して EDA が適用されるようになってきている．人工知能へのニーズが高まるにつれ，自動プログラミング手法は一層重要になると考えられる．より広範な分野への適用を視野に入れた場合，従来 GP では解けないような問題が出現する可能性は非常に高い．本論文で提案した二つのアルゴリズムは，そのような問題に対しての一つの答えとなる可能性がある．

本論文では，GP-Hard なベンチマークに対して有効な提案手法を二つ提案した．プロトタイプ木に基づく POLE は表現手法として，拡張構文木を用い，ベイジアンネットワーク推定を用いることで，ノード間の依存関係を推定している．POLE が示した重要な点は，拡張構文木を用いることで，ベイジアンネットワークなどのグラフィカルモデルを木構造に適用可能であると示した点にある．3.6.3.4 節では，通常の構文木と拡張構文木を用い，双方にベイジアンネットワークを適用した場合の比較を行っているが，拡張構文木を用いた場合にのみ，深さ 6 以上の場合のネットワーク推定に成功している．このように，提案手法が，拡張構文木とグラフィカルモデルの組み合わせの有効性を示したことで，プロトタイプ木に基づく GP-EDA に新しい発展を示したと言える．

第二の提案手法である PCFG-LA を用いた PAGE は，位置非依存の部分構造の推定に適している．PCFG は文脈自由であるため，本来は位置非依存のモデルであるが，実際には，深さなどを考慮しているモデルを用いた GP-EDA が非常に多かった．提案手法 PAGE では，潜在アノテーションを用いることで位置に依存しない部分構造を推定することが可能となる．これにより，部分構造の抽出により焦点を当てた関数進化アルゴリズムの研究に弾みがつくと期待される．

5.2 将来研究の方向性

木構造に対するEDAに関する研究は比較的新しく，実用化のためにはさらなる研究が必要があると考えられる．特に以下の点は，これからの重要課題であると考えている．

- イントロンの影響
- 収束性などの理論的検証
- 確率モデルの選択
- キラーアプリケーション

イントロンは，適合度には影響を及ぼさない構造であるが，確率モデルの学習には影響を及ぼす構造である．イントロンを多く有する集団に対してモデル学習を適用しても，イントロンの影響で正しく部分構造を推定できない可能性がある．関数同定問題やパリティ問題など，GPが対象とする問題の多くはイントロンを含む問題である．GP-EDAがこのような問題に対しても正しく働くためには，イントロンを適切に扱うための方法が必要である．考えられる方法はいくつかあるが，イントロンをヒューリスティクスで除去する方法などが考えられる．この場合，あらかじめルールを設定し，そのルールを適用する形でイントロンを除去する方法が考えられる．しかし，この場合，問題毎にルールの設定が必要な点など，問題もある．

理論的な収束性についても行う必要がある．現在の形のGP-EDAでは，収束性に関する考察は一切行われていない．GA-EDAの場合より，確率モデルが複雑であるため，理論的解析が難しいことが理由である．GA-EDAであるUMDAにおいては，必ず最適解を獲得出来ることが示されている．GP-EDAにおいても同様の理論的解析が必要である．

この論文で述べているように，EDAではモデルの仮定が必要である．通常のGAやGPが，問題に対してほとんど仮定を要しないアルゴリズムであるのに対して，EDA，特にGP-EDAは事前のモデルの選択が重要である．将来的には，モデルの選択もある程度自動的に行えるようにすることが必要である．

本論文での有効性の検証は，人工的なベンチマークテストで行われた．このようなベンチマークテストは，探索メカニズムの解析には非常に適している．一方で，探索アルゴリズムが工学的に意味を持つためには，実問題への適用が不可欠である．「工学的に意味がある」というのは二つあり，GPより探索が速い，もしくはGPにはない特徴がある，のどちらかが必要である．前者に関しても重要であると考えるが，特に後者の点に考えてみたい．GP-EDAで推定されるのは，解のみならず確率分布も学習される．もし，学習したベイジアンネットワークや，確率分布それ自体が重要な情報となる問題があった場合，GPではそのような情報を得ることが出来ないため，提案手法のキラーアプリケーションになると考えられる．このような問題においては，GP-EDAは単なる探索アルゴリズムではなく，データマイニングも同時に行うことが出来るアルゴリズムと考えられる．近年，GPはゲノムデータなどの大規模データに対する適用が行われている．このようなデータ

5.2. 将来研究の方向性

に対して，解構造に関する確率分布などの情報が加われば，解析が容易になることが期待される．

参考文献

[Angeline 98] Angeline, P. J.: Multiple Interacting Programs: A Representation for Evolving Complex Behaviors, *Cybernetics and Systems*, Vol. 29, No. 8, pp. 779–806 (1998)

[Attias 99] Attias, H.: Inferring parameters and structure of latent variable models by variational Bayes, in *the 15th Conference of Uncertainty in Artificial Intelligence*, pp. 21–30, Stockholm, Sweden (1999), Morgan Kaufmann

[Baker 79] Baker, J. K.: Trainable grammers for speech recognition, in Wolf, J. J. and Klatt, D. H. eds., *Speech communication papers presented at the 97th meeting of the Acoustical Society of America*, pp. 547–550, Cambridge, MA (1979), MIT press

[Baluja 94] Baluja, S.: Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, Technical Report CMU-CS-94-163, Pittsburgh, PA (1994)

[Baluja 97] Baluja, S. and Davies, S.: Combining Multiple Optimization Runs with Optimal Dependency Trees (1997), Carnegie Mellon Report, CMU-CS-97-157, 1997c.

[Beal 03] Beal, M. J.: *Variational Algorithms for Approximate Bayesian Inference*, PhD thesis, Gatsby Computational Neuroscience Unit, University of College London (2003)

[Bonet 96] Bonet, J. S. D., Isbell, C., and Viola, P.: MIMIC: Finding Optima by Estimating Probability Densities, in Michael Jordan, M. M. and Perrone, M. eds., *Advances in Neural Information Processing*, Vol. 9, Denver 1996 (1996), MIT Press, Cambridge

[Bosman 99] Bosman, P. A. N. and Jong, de E. D.: An algorithmic framework for density estimation based evolutionary algorithms, Technical Report UU-CS-1999-46, Utrecht University (1999)

[Bosman 04] Bosman, P. A. N. and Jong, de E. D.: Grammar Transformations in an EDA for Genetic Programming, Technical Report UU-CS-2004-047, Institute of Information and Computing Sciences, Utrecht University (2004)

[Cooper 92] Cooper, G. and Herskovits, E.: A Bayesian method for the induction of probabilistic networks from Data, *Machine Learning*, Vol. 9, pp. 309–347 (1992)

[Cramer 85] Cramer, N. L.: A Representation for the adaptive generation of simple sequential programs, in J, G. J. ed., *Proceedings of an International Conference on Genetic Algorithms and the Applications*, pp. 183–187, Carnegie-Mellon University, Pittsburgh, PA (1985)

[Dempster 77] Dempster, A., Laird, N., and Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society, Series B*, Vol. 39, No. 1 (1977)

[Etxeberria 99] Etxeberria, R. and Larrañaga, P.: Global optimization using Bayesian networks, *Proc. 2nd Symposium on Artificial Intelligence (CIMAF-99)*, pp. 332 – 339 (1999)

[Friedman 96] Friedman, N. and Yakhini, Z.: On the sample complexity of learning Bayesian networks, in *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 274–282 (1996)

[Gallagher 07] Gallagher, M., Wood, I., Keith, J., and Sofronov, G.: Bayesian Inference in Estimation of Distribution Algorithms, in *Proceeding of IEEE Congress on Evolutionary Computation (CEC) 2007*, pp. 127–133 (2007)

[Gathercole 96] Gathercole, C. and Ross, P.: An Adverse Interaction between Crossover and Restricted Tree Depth in Genetic Programming, in Koza, J. R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L. eds., *Genetic Programming 1996: Proceedings of the First Annual Conference*, pp. 291–296, Stanford University, CA, USA (1996), MIT Press

[Goldberg 93] Goldberg, D. E., Deb, D., and Kargupta, H.: Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms, in Forrest, S. ed., *Proc. of the Fifth Int. Conf. on Genetic Algorithms*, pp. 56–64, San Mateo (1993), Morgan Kaufman

[Hasegawa 06] Hasegawa, Y. and Iba, H.: Estimation of Bayesian Network for Program Generation, in *Proceedings of The Third Asian-Pacific Workshop on Genetic Programming*, pp. 35–46, Hanoi, Vietnam (2006)

[長谷川 07] 長谷川 禎彦, 伊庭 斉志：ベイジアンネットワーク推定による確率モデル遺伝的プログラミング, 人工知能学会論文誌, 22 巻, 1 号, pp. 37–47 (2007)

[長谷川 08] 長谷川 禎彦, 伊庭 斉志：潜在アノテーション推定を用いた確率文法による分布推定アルゴリズム, 人工知能学会論文誌, 23 巻, 1 号, pp. 13–26 (2008)

[Heckerman 94] Heckerman, D., Geiger, D., and Chickering, M.: Learning Bayesian networks: The combination of knowledge and statistical data, Technical Report MSR-TR-94-09, Redmond, WA: Microsoft Research (1994)

[Heckerman 95] Heckerman, D.: A tutorial on learning with Bayesian networks (1995)

[Holland 75] Holland, J. H.: *Adaptation in Natural and Artificial Systems*, University of Michigan press (1975)

[伊庭 01] 伊庭 斉志：遺伝的プログラミング入門, 東京大学出版会 (2001)

[樺島 05] 樺島 祥平, 上田 修功：計算統計II. 統計科学のフロンティア12, 岩波書店 (2005)

[北 99] 北 研二：言語と計算-4 確率的言語モデル, 東京大学出版 (1999)

[Koza 89] Koza, J. R.: Hierarchical genetic algorithms operating on populations of computer programs, in *Prodeedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89*, Vol. 1, pp. 768–774, San Francisco, CA (1989), Morgan Kaufmann

[Koza 92] Koza, J. R.: *Genetic Programming : On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, USA (1992)

[Kurihara 04] Kurihara, K. and Sato, T.: An Application of the Variational Bayesian Approach to Probabilistic Context-Free Grammars (2004), IJCNLP-04 Workshop Beyond shallow analyses, http://sato-www.cs.titech.ac.jp/en/publication/

[Langdon 97] Langdon, W. B. and Poli, R.: An Analysis of the MAX Problem in Genetic Programming, in Koza, J. R., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M., Iba, H., and Riolo, R. L. eds., *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pp. 222–230, Stanford University, CA, USA (1997), Morgan Kaufmann

[Langdon 02] Langdon, W. B. and Poli, R.: *Foundations of Genetic Programming*, Springer-Verlag (2002)

[Looks 05a] Looks, M.: Learning Computer Programs with the Bayesian Optimization Algorithm (2005), Master thesis, Washington University Sever Institute of Technology

[Looks 05b] Looks, M., Goertzel, B., and Pennachin, C.: Learning Computer Programs with the Bayesian Optimization Algorithm, in Beyer, H. G. and et al., eds., *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, Vol. 2, pp. 747–748, Washington DC, USA (2005), ACM Press

[MacKay 97] MacKay, D. J. C.: Ensemble Learning for Hidden Markov Models (1997), http://www.inference.phy.cam.ac.uk/mackay/abstracts/ensemblePaper.html

[Matsuzaki 05] Matsuzaki, T., Miyao, Y., and Tsujii, J.: Probabilistic CFG with latent annotations, in *In Proceedings of the 43rd Meeting of the Association for Computational Linguistics (ACL)*, pp. 75–82, Michigan, USA (2005), Morgan Kaufmann

[Mitchell 92] Mitchell, M., Forrest, S., and Holland, J. H.: The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance, in Varela, F. J. and Bourgine, P. eds., *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, 1991*, pp. 245–254, Paris (1992), A Bradford book, The MIT Press

[Mühlenbein 96] Mühlenbein, H. and Paa, G.: From recombination of genes to the estimation of distributions, in Eiben, A., Bäck, T., Shoenauer, M., and Schwefel, H. eds., *Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature PPSN IV*, pp. 178–187, Berlin (1996), Springer Verlag

[Mühlenbein 99a] Mühlenbein, H. and Mahnig, T.: The Factorized Dsitribution Algorithm for additively decomposed functions, in *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 752–759, IEEE press (1999)

[Mühlenbein 99b] Mühlenbein, H., Mahnig, T., and Rodriguez, A. O.: Schemata, Distributions and Graphical Models in Evolutionary Optimization, *Journal of Heuristics*, Vol. 5, No. 2, pp. 215–247 (1999)

[Neapolitan 04] Neapolitan, R. E.: *Learning Bayesian Networks*, Pearson Education (2004)

[Pelikan 99a] Pelikan, M., Goldberg, D. E., and Cantú-Paz, E.: BOA: The Bayesian Optimization Algorithm, in Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E. eds., *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, Vol. I, pp. 525–532, Orlando, FL (1999), Morgan Kaufmann Publishers, San Fransisco, CA

[Pelikan 99b] Pelikan, M. and Mühlenbein, H.: The Bivariate Marginal Distribution Algorithm, in Roy, R., Furuhashi, T., and Chawdhry, P. K. eds., *Advances in Soft Computing - Engineering Design and Manufacturing*, pp. 521–535, London (1999), Springer-Verlag

[Pelikan 02] Pelikan, M.: *Bayesian optimization algorithm: From single level to hierarchy*, PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL (2002), Also IlliGAL Report No. 2002023

[Peña 05] Peña, J. M., Lozano, J. A., and Larrañaga, P.: Globally Multimodal Problem Optimization Via an Estimation of Distribution Algorithm Based on Unsupervised Learning of Bayesian Networks, *Evolutionary Computation*, Vol. 13, No. 1, pp. 43–66 (2005)

[Punch 98] Punch, W. F.: How Effective are Multiple Populations in Genetic Programming, in Koza, J. R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B.,

Garzon, M. H., Goldberg, D. E., Iba, H., and Riolo, R. eds., *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pp. 308–313, University of Wisconsin, Madison, Wisconsin, USA (1998), Morgan Kaufmann

[Quesenberry 64] Quesenberry, C. P. and Hurst, D. C.: Large Sample Simultaneous Confidence Intervals for Multinomial Proportions, *Technometrics*, Vol. 6, pp. 191–195 (1964)

[Ratle 01] Ratle, A. and Sebag, M.: Avoiding the bloat with Probabilistic Grammar-guided Genetic Programming, in Collet, P., Fonlupt, C., Hao, J.-K., Lutton, E., and Schoenauer, M. eds., *Artificial Evolution 5th International Conference, Evolution Artificielle, EA 2001*, Vol. 2310 of *LNCS*, pp. 255–266, Creusot, France (2001), Springer Verlag

[Rissanen 78] Rissanen, J.: Modeling by Shortest Data Description, *Automatica*, pp. 465 – 471 (1978)

[坂元 83] 坂元 慶行, 石黒 真木夫, 北川 源四郎：情報量統計学, 共立出版株式会社 (1983)

[Sałustowicz 97a] Sałustowicz, R. P. and Schmidhuber, J.: Probabilistic Incremental Program Evolution, *Evolutionary Computation*, Vol. 5, No. 2, pp. 123–141 (1997)

[Sałustowicz 97b] Sałustowicz, R. P. and Schmidhuber, J.: Probabilistic Incremental Program Evolution: Stochastic Search Through Program Space, in Someren, van M. and Widmer, G. eds., *Machine Learning: ECML-97*, Vol. 1224, pp. 213–220, Springer-Verlag (1997)

[Santana 05] Santana, R.: Estimation of Distribution Algorithms with Kikuchi Approximations, *Evolutionary Computation*, Vol. 13, No. 1, pp. 67–97 (2005)

[Schwarz 78] Schwarz, G.: Estimating the Dimension of a Model, *The Annals of Statistics*, Vol. 6 (2), pp. 461 – 464 (1978)

[Shan 03] Shan, Y., McKay, R. I., Abbass, H. A., and Essam, D.: Program evolution with explicit learning: a New Framework for Program Automatic Synthesis, in Sarker, R., Reynolds, R., Abbass, H., Tan, K. C., McKay, B., Essam, D., and Gedeon, T. eds., *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pp. 1639–1646, Canberra (2003), IEEE Press

[Shan 04] Shan, Y., McKay, R. I., Baxter, R., Abbass, H., Essam, D., and Hoai, N. X.: Grammar Model-based Program Evolution, in Greenwood, G. ed., *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pp. 478–485, Portland, Oregon (2004), IEEE Press

[Shan 06] Shan, Y., McKay, R. I., Essam, D., and Abbass, H. A.: A Survey of Probabilistic Model Building Genetic Programming, in Pelikan, M., Sastry, K., and Cantú-Paz, E. eds., *Scalable Optimization via Probabilistic Modeling*, chapter 6, pp. 121–160, Springer (2006)

[Stolcke 94] Stolcke, A.: *Bayesian Learning of Probabilistic Language Models*, PhD thesis, University of California, Berkeley, CA (1994)

[Teller 94a] Teller, A.: Genetic Programming, Indexed Memory, The Halting Problem, And Other Curiosities (1994)

[Teller 94b] Teller, A.: Turing Completeness in the Language of Genetic Programming with Indexed Memory, in *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, Vol. 1, pp. 136–141, Orlando, Florida, USA (1994), IEEE Press

[Teller 95] Teller, A. and Veloso, M.: PADO: Learning Tree Structured Algorithms for Orchestration into an Object Recognition System, Technical Report CMU-CS-95-101, Pittsburgh, PA, USA (1995)

[Teller 96] Teller, A. and Veloso, M.: PADO: A New Learning Architecture for Object Recognition, in Ikeuchi, K. and Veloso, M. eds., *Symbolic Visual Learning*, pp. 81–116, Oxford University Press (1996)

[筒井 03] 筒井 茂義：エッジヒストグラムを用いる順序表現向き確率モデル GA の提案, 人工知能学会論文誌, 18 巻, 4 号, pp. 173–182 (2003)

[上田 01] 上田 修功：最良モデル探索のための変分ベイズ学習, 人工知能学会論文誌, 16 巻, 2 号 SP-F, pp. 299–308 (2001)

[Whigham 95] Whigham, P. A.: Grammatically-based genetic programming, in *Proceedings of the Workshop on Genetic Programming : From Theory to Real-World Applications*, pp. 44–41, Tahoe City, California USA (1995)

[Whigham 96] Whigham, P. A.: Search Bias, Language Bias, and Genetic Programming, in Koza, J. R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L. eds., *Genetic Programming 1996: Proceedings of the First Annual Conference*, pp. 230–237, Stanford University, CA, USA (1996), MIT Press

[Wineberg 94] Wineberg, M. and Oppacher, F.: A Representation Scheme to Perform Program Induction in a Canonical Genetic Algorithm, in Davidor, Y., Schwefel, H.-P., and Männer, R. eds., *Parallel Problem Solving from Nature III*, Vol. 866 of *LNCS*, pp. 292–301, Jerusalem (1994), Springer-Verlag

[矢吹 04] 矢吹太郎：進化計算による自動プログラミングのための表現の研究, PhD thesis, 東京大学大学院新領域創成科学研究科 (2004)

[Yanai 03] Yanai, K. and Iba, H.: Estimation of distribution programming based on Bayesian network, in Sarker, R., Reynolds, R., Abbass, H., Tan, K. C., McKay, B., Essam, D., and Gedeon, T. eds., *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pp. 1618–1625, Canberra (2003), IEEE Press

[柳井 05] 柳井 孝介, 伊庭 斉志：条件付確率に基づく分布推定アルゴリズムによるプログラム進化, 情報処理学会論文誌, Vol. 46, No. SIG10, pp. 111–123 (2005)

発表文献

論文誌

1. Yoshihiko Hasegawa and Hitoshi Iba: A Bayesian Network Approach to Program Generation, IEEE Transactions on Evolutionary Computation, in press
2. 長谷川 禎彦, 伊庭 斉志. 潜在アノテーション推定を用いた確率文法による分布推定アルゴリズム, 人工知能学会論文誌, 23巻1号, pp.13-26, 2008
3. 長谷川 禎彦, 伊庭 斉志. ベイジアンネットワーク推定による確率モデル遺伝的プログラミング, 人工知能学会論文誌, 22巻1号, pp.37-47, 2007
4. 長谷川 禎彦, 伊庭 斉志. 免疫系を用いた遺伝的プログラミングによる多峰性探索, 人工知能学会論文誌, 21巻2号, pp.176-183, 2006

国際会議

1. Yoshihiko Hasegawa and Hitoshi Iba : Estimation of Distribution Algorithm based on Probabilistic Grammar with Latent Annotations, Proceedings of IEEE Congress of Evolutionary Computation (CEC 2007), pp.1143-1150, 2007, Singapore
2. Yoshihiko Hasegawa and Hitoshi Iba : Estimation of Bayesian Network for Program Generation, Proceedings of The Third Asian - Pacific Workshop on Genetic Programming (ASPGP '06), pp.35-46, 2006, Hanoi, Vietnam
3. Yoshihiko Hasegawa and Hitoshi Iba : Optimizing Programs with Estimation of Bayesian Network, Proceedings of the 2006 World Congress on Computational Intelligence (WCCI 2006), pp.5527-5534, 2006, Vancouver, Canada
4. Yoshihiko Hasegawa and Hitoshi Iba : Multimodal Search with Immune Based Genetic Programming, Proceedings of the third International Conference (ICARIS 2004), pp.330-341, Springer, 2004, Catania, Italy

国内シンポジウム・ワークショップ等

1. 長谷川禎彦，伊庭斉志．ノード間依存推定による確率的プログラム生成手法，情報論的学習理論ワークショップ（IBIS2006）

2. 長谷川禎彦，伊庭斉志．構造推定を用いた遺伝的プログラミング，人工知能学会全国大会（21回），2007

3. 長谷川禎彦，伊庭斉志．条件付きエントロピーを用いた遺伝的アルゴリズムのリンケージ推定方法，情報処理学会数理モデル化と問題解決研究会，2005

謝辞

本論文をまとめるに当たり，修士以来，東京大学伊庭斉志教授に多くの御助言と御指導を賜り，様々な面で支えていただきました．非常に自由な環境で研究をさせていただき，心より感謝致します．

元伊庭研究室，現日立中央研究所の柳井孝介博士には，修士以来，研究に関する多くの御助言を頂きました．その時行われた多くの議論が，本論文執筆の際の助けとなりました．

また，伊庭研究室の方々には非常にお世話になりました．峠隆広氏，柳瀬利彦氏には，修士から現在に至るまで，長きに渡り非常にお世話になりました．クラウス・アランニャ氏には，英語の校正などを含め，研究面でお世話になりました．安藤大地氏，丹治信氏とは，非常に有意義な議論を行いました．皆様に心より感謝致します．