知能と複雑系 110－5
(1998. 1. 22)

並列組織化アルゴリズムによるグラフマッチングと部分構造の発見手法

前原 恵太 † 上原 邦昭 ‡

† 神戸大学 工学部 情報知能工学科
‡ 神戸大学 都市安全研究センター

〒657 兵庫県神戸市灘区六甲台町 1-1
E-mail: maehara@jedi.cs.kobe-u.ac.jp uehara@jedi.seg.kobe-u.ac.jp

あらまし グラフによるデータの表現には非常に柔軟性があり，さまざまな分野で用いられている．中でも，グラフ集合の中から重要な意味を持つ構造を発見したり，与えられたグラフにマッチする部分グラフを多数のグラフ集合から検索するなど，基本的な処理を高速に行なうアルゴリズムを開発することは非常に重要である．しかし，対象となるグラフ数が増大すると，処理に必要な計算コストが爆発するという問題がある．本稿では，MDL (Minimum Description Length) 基準を使用して複数のグラフが共通して持つ構造を発見するとともに，発見された構造に基づいてグラフ集合をネットワークへと組織化して，処理を高速化するための並列アルゴリズムについて述べる．

キーワード グラフマッチング MDL 基準 並列処理 組織化アルゴリズム

Parallel Organization Algorithm for Graph Matching and Subgraph Isomorphism Detection

Keita Maehara † Kuniaki Uehara ‡

† Department of Computer and Systems Engineering,
Faculty of Engineering, Kobe University
† Research Center for Urban Safety and Security, Kobe Unversity
Rokkodai-cho 1-1, Nada-ku, Kobe-shi, Hyogo, 657 Japan
E-mail: maehara@jedi.cs.kobe-u.ac.jp uehara@jedi.seg.kobe-u.ac.jp

Abstract

Graph representation of data is very flexible and used in various applications. Detecting interesting substructures and searching a given graph from a set of graphs is a very important problem. But the computational cost of graph-based operation such as graph matching is intractable, especially if their size is very large. In this paper, we will propose a parallel graph organization algorithm based on MDL (Minimum Description Length) criterion. Graphs are organized into a hierarchical network according to their common substructures and thus matching cost can be reduced by using this hierarchical network.

key words Graph Matching MDL Criterion Parallel Processing Organization Algorithm

1 はじめに

データの表現形式には様々なものが存在するが，中でもグラフ表現は柔軟で汎用性が高く，回路設計や画像認識などさまざまな分野で用いられている．このようなグラフの中から有益な情報を発見したり，特定の構造を持ったグラフを探し出すことには非常に大きな意義がある．たとえば，回路の中に繰り返し現れる構造を取り出すことができれば，その構造に対応した回路素子を用いて効率的に回路を構成することができる．しかしながら，一般に，グラフ表現を用いたアルゴリズムの計算コストは高く，より高速なアルゴリズムが必要となることが多い．本研究では，ラベル付き有向グラフで記述されたデータ集合の中から興味深い構造を発見するとともに，発見された構造に基づいてグラフ集合を組織化し，検索に役立てるための並列アルゴリズムについて述べる．

2 グラフ集合の共通構造

2.1 部分構造の発見

与えられたグラフ集合の中から，複数のグラフに共通する部分グラフを抽出することを考える．図 1 および図 2 は，同一のグラフ集合に対し，異なる部分グラフに着目した場合の例である．図 1 は Graph1 と Graph4 に共通な部分グラフ，Graph2 と Graph3 に共通な部分グラフの計 2 種類のグラフに着目した場合である．

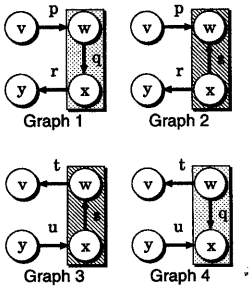

図 1: グラフ集合と部分グラフ (a)

また，図 2 は Graph1 と Graph2 に共通な 2 種類の部分グラフ，Graph3 と Graph4 に共通な 2 種類の部分グラフの計 4 種類のグラフに着目した場合である．

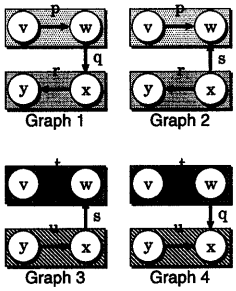

図 2: グラフ集合と部分グラフ (b)

いずれも同一のグラフ集合であるが，図 1 の部分グラフの取り方では，各グラフについて 2 節点ずつしかカバーできていないのに対して，図 2 の部分グラフの取り方ではすべての節点がカバーされている．このように，複数のグラフが共通して持つ部分グラフの選択の仕方はただ一通りに定まるものではなく，ある部分グラフの選択の仕方が他の部分グラフの選択の仕方に大きな影響を与えている．

ここで，選択した部分グラフを一つの節点として表現すると，グラフ集合をより効率的に記述することが可能となる．たとえば，図 1 や図 2 で示した 4 つのグラフが与えられたとき，各々の図において着目している部分グラフを一つの節点として表現すると，グラフ集合の節点や辺の数を減少させることができる (図 3，図 4)．図 3 では，図 1 で着目した部分グラフが 1 つの節点で置換され，各グラフの節点数は 3，辺数は 2 に減少している．図 4 では，図 2 で着目した部分グラフが 1 つの節点で置換され，各グラフの節点数は 2，辺数は 1 に減少している．

このように，複数のグラフが共通して持つ部分グラフにはさまざまな可能性があるため，何らかの採用基準を定めなければならない．このような採用基準の 1 つとして，より多くのグラフに共通に含まれる部分グラフを採用するとい

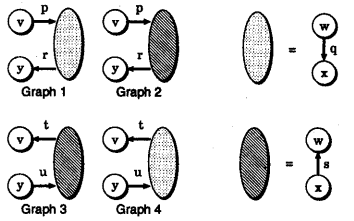

図 3: グラフ集合の効率的な記述 (a)

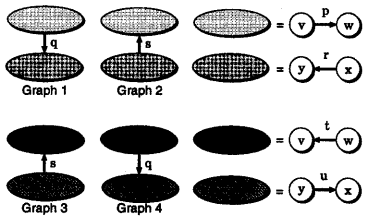

図 4: グラフ集合の効率的な記述 (b)

う方針が考えられる．この場合，多くのグラフを同一の部分グラフを用いて記述できるという意味では理想的である．しかし，多くのグラフに共通に含まれているような部分グラフのサイズは一般に小さいため，この方針が最善であるとはいえない．また，逆にサイズが大きい部分グラフを採用するという方針をとることも考えられるが，グラフをより簡単な形で記述することができるという意味では理想的であるものの，サイズが大きい部分グラフが共通に含まれているようなグラフの数は一般に少ないため，この方針も最善であるとはいえない．

本研究で採用している MDL (Minimum Description Length) 基準 [1] は，これらの方針の中間にあたる．すなわち，MDL 基準における最良のモデルとは，データを説明するモデル自身の記述長と，そのモデルを用いてデータを記述したときの記述長との和が最小となるようなモデルである．これをグラフ集合の記述の問題に当てはめると，ある部分グラフの記述長と，それを用いてグラフ集合を記述した際の記述長の和を最小にするような部分グラフがモデルとして最も理想的であるということになる．

本研究では，グラフ G の節点数を v，節点の持つラベル数を l_v，辺数を e，辺の持つラベル数を l_e としたとき，グラフ G の記述長 $DL(G)$ を以下のように定義している．

$$DL(G) = \log_2 v + v \log_2 l_v + \log_2 e + e \log_2 l_e + 2e \log_2 v \quad (1)$$

さらに，グラフ G の部分グラフ SG を 1 つの節点と見なして置換した場合の G の記述長 $DL(G|SG)$ は，置換後の節点数を v'，辺数を e'，グラフのラベル数を l'_v とすると，

$$DL(G|SGList) = \log_2 v' + v' \log_2 l'_v + \log_2 e' + e' \log_2 l_e + 2e' \log_2 v' \quad (2)$$

で与えられる．

以上の定義から，グラフ集合 $GSet$ を考えた場合，$GSet$ に含まれるグラフが共通に持つ部分グラフの組み合わせの 1 つを $SGList$ とすると，MDL 基準により $SGList$ の評価関数 $Eval(SGList)$ は式 (3) で与えられる．

$$Eval(SGList) = DL(SGList) + DL(GSet|SGList) \quad (3)$$

なお，$DL(SGList)$ は部分グラフの組み合わせ $SGList$ に対する記述長，$DL(GSet|SGList)$ は $SGList$ を用いてグラフ集合 $GSet$ を記述した際の記述長である．

評価関数 $Eval(SGList)$ は，値が小さいほど元のグラフ集合が効率的に表現されているという尺度になっている．たとえば，図 1 と図 2 における，もとのグラフ集合の記述長は (1) 式によりともに 113.4 [$bits$] であるが，もとのグラフ集合を部分グラフに着目して記述したときの評価値は，(3) 式により図 3 の場合は 86.2 [$bits$]，図 4 の場合は 59.0 [$bits$] となる．したがって，MDL 基準の観点から見れば，与えられたグラフ集合をより効率的に記述するためには図 4 の 4 種類の部分グラフを用いればよいことがわかる．

2.2 探索範囲の絞り込み

MDL 基準を採用しても，より効率的にもとのグラフを表現できるものを発見するためには，あらゆる部分グラフに対して評価を行なわなければならない．たとえば，n 節点のグラフが与えられたとき，その中から節点を採用する方法は $\sum_{i=1}^{n} {}_nC_i$ 通り存在する．さらに節点の連結パターンを考慮に入れると，評価すべき部分グラフの数が非常に大きくなることは明らかである．また，実際には複数のグラフが共通して持つ部分グラフの組み合わせについて考える必要があり，問題の規模が大きくなるとその数は膨大になる．このため，本研究では「あるグラフを用いてグラフ集合を記述した際の記述長が小さければ，別のグラフと組み合わせてグラフ集合を記述した際の記述長も小さい」というヒューリスティクスを利用して処理の高速化をはかっている．

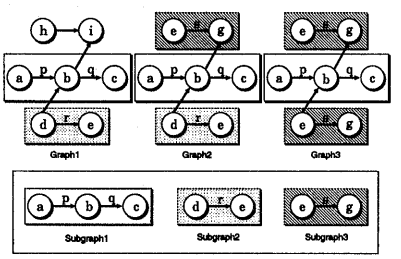

図 5: 部分グラフの選択方針

たとえば，図 5 は Graph1, Graph2, Graph3 の 3 つのグラフに共通な Subgraph1 がすでに解として選択されており，次に選択される候補として Subgraph2 と Subgraph3 が存在するときの状態を示している．Subgraph2 と Subgraph3 の節点数や辺数は同一であるが，Subgraph2 はグラフ集合中に 2 度，Subgraph3 は 3 度現れているため，それぞれ単体でグラフ集合を記述した場合の記述長は Subgraph3 の方が小さくなる．したがって，Subgraph1 と組み合わせてグラフ集合を記述する際には Subgraph3 が採用される．

3 ネットワーク生成アルゴリズム

本節では，MDL 基準を用いたグラフ集合の並列組織化アルゴリズムについて説明する．このアルゴリズムにより，与えられたグラフ集合から効率的な検索を行なうためのネットワークが構築される．ネットワーク中の各ノードは，MDL 基準によって発見された部分グラフに対応しており，その部分グラフに対するマッチングに利用することができる．複数のグラフに共通な部分グラフに対応するノードはネットワーク中で共有されているため，マッチングの重複を避けることができ，グラフの検索を効率的に行なうことが可能となる．

3.1 アルゴリズムの詳細

アルゴリズムの初期状態においては，ただ一つのノードのみが存在している．このノードを入力ノードと呼び，与えられたグラフは必ず最初にこの入力ノードに受け渡される．グラフは入力ノードで節点集合に分解され，それぞれ対応する節点ノードが生成される．

さらに，節点ノードは入力ノードの子ノードとして，入力ノードとリンクされる．初期状態では，入力ノードの子ノードは全く存在しないため，分解された節点のラベルと同じラベルを持つ節点ノードを新たに作成し，そのノードに節点を格納する．同じラベルを持つ節点はすべて同じ節点ノードに格納されていくことになる．ここで節点ノードに格納された節点を，その節点ノードのインスタンスと呼ぶ．

次に中間ノードが生成される．中間ノードとは，節点ノードや他の中間ノードの組み合わせによって生成されるもので，もとのグラフの部分グラフを表現しているノードである．具体的な手順としては，ある時点でネットワーク中のノードに格納されているインスタンスの中から，もとのグラフで互いに連結されていたもの 2 つに注目する．そして，それらのインスタンスを組み合わせたときにできる部分グラフによって，もとのグラフ集合を記述したときのグラフ集合の記述長を計算する．もとのグラフ集合の記述

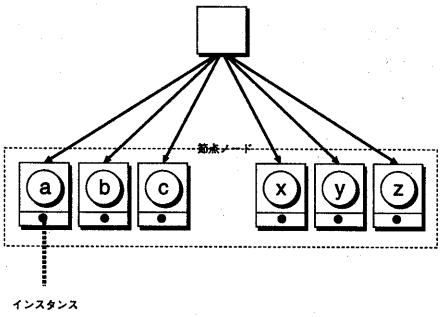

図 6: 節点ノード

長を小さくする部分グラフは，他の部分グラフと組み合わせたときにも記述長をより小さくすると考えられるため，この処理において高い評価を得る組み合わせから新しいノードを生成すれば，そのノードの評価も高くなる．

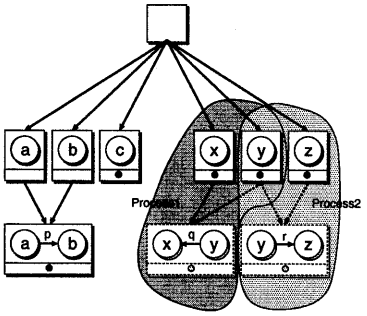

図 7: 中間ノードの生成 (a)

図 7 は，中間ノードの生成の様子の一例である．ノードの評価自体はネットワークの各部で独立して行なうことができるため，並列に処理を行なうことにより高速化が実現できることになる．本研究では，複数のプロセッサが独立にノードの評価を行ない，その中で最も評価の高いものを実際に生成するようにしている．これは，与えられたグラフ集合に含まれるあらゆる部分グラフを並列に探索し，より評価の高いものを選択することに相当している．たとえば，図 7 では，別々のプロセス Process1 と Process2 がそれぞれ節点 y から節点 x への辺 q を持つグラフと，節点 y から節点 z への辺 r を持つグラフの評価を行なっている．その結果，節点 y から節点 x への辺 q を持つグラフの評価が高かったとすると，図 8 のように，実際にそのグラフに対応したノードが生成される．

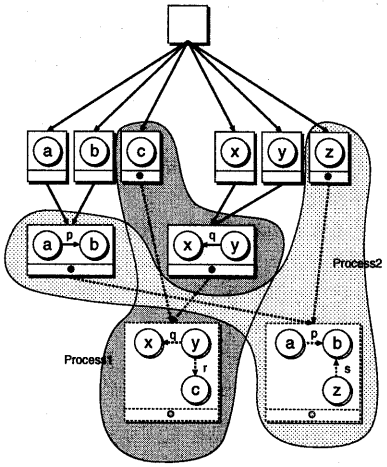

図 8: 中間ノードの生成 (b)

新しいノードが生成される際には，2 つの親ノードから対応するインスタンスが削除され，新しい 1 つのインスタンスとしてそのノードに格納される．図 8 では，節点 y から節点 x への辺 q を持つグラフの評価が最も高かったため，節点ノード y と節点ノード x に格納されていたインスタンスが削除され，新しく生成されたノードに新しい 1 つのインスタンスとして格納されている．このインスタンスは，さらに新たなノードを作成する際の候補となる．図 8 では，生成されたノードが Process1 による評価の対象となっている．このようにして，ネットワークの各部において次々と新しいノードが生成されていく．

以上の処理を繰り返し続けることにより，最終的にはもとのグラフ集合の個々のグラフに対応したノードがネットワークの最下部に生成されることになる．すべてのグラフに対応したノー

ドが生成された時点で組織化処理は終了する．

4 実験

4.1 ヒューリスティクスの有効性

2.2 節でも述べたように，ネットワークの作成の際には MDL 基準を用いたヒューリスティクスを利用している．このヒューリスティクスの有効性を示すために，総節点数 257 の 5 つのグラフ集合から 400 の部分グラフを生成して，もとのグラフ集合をより効率的に記述できる部分グラフの組み合わせの探索を行なう実験を行なった．表 1 は，実験の結果を示している．表中の「探索方針」は部分グラフの組み合わせを探索する際の方針であり，「処理時間」は最終的な結果を得るために要した時間，「評価値」は，もとのグラフ集合を，最終的に選びだされた部分グラフの組み合わせによって記述したときの (3) 式による評価値である．

探索方針	処理時間 [s]	評価値 [bits]
random	266	2921.8
heuristic	153	2428.2

表 1: 部分グラフ選択方針の比較

探索方針として “random” を採用した際には，まず最初に生成された部分グラフの中からランダムに 1 つの候補を選択して，(3) 式による評価を行なう．この評価値がもとのグラフ集合の記述長よりも小さくなる場合には，その候補を解の 1 つとし，さらに別の候補をランダムに選択する．次に，解として選ばれた部分グラフと，新たに選択された候補とを組み合わせて再度 (3) 式による評価を行なう．この評価値が以前のものよりも小さくなっていれば，新たに選択された候補も解の 1 つとして付け加える．このような手順を，評価値がそれ以上小さくならないところまで行なっている．一方，探索方針として “heuristic” を採用した際には，候補の選択をランダムに行なうのではなく，個々の部分グラフの (3) 式による評価の小さいものから優先的に行なっている．この実験結果より，上記のヒューリスティクスを使用すれば処理時間が短縮されることだけではなく，より適切な部分構造を発見することもできることが示されている．

4.2 並列化の効果

グラフ数 50，総節点数 1285 のグラフ集合に対して本組織化アルゴリズムを適用して実験を行なった．実験にはシリコングラフィックス社の Origin 2000 (CPU: R10000 × 8，メインメモリ: 384Mbytes，OS: IRIX 6.4) を使用した．このときの使用プロセッサ数と処理時間の関係を図 9 に示す．

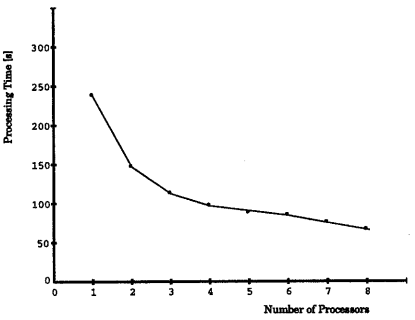

図 9: 並列化の効果

ネットワークのノードを新たに生成する際には，ノードを生成した 2 つの親ノードに格納されている各インスタンスを，新たに生成されたノードの新たな 1 つのインスタンスとして受け渡さなければならない．しかしながら，本アルゴリズムのように複数のプロセッサがこのような処理を行なう可能性がある場合には注意が必要である．たとえば，図 10 は，節点 y から節点 x への辺 q を持つグラフと，節点 y から節点 z への辺 r を持つグラフの 2 つのグラフを複数のプロセッサが同時に生成しようとする状況を示している．図 10 のように，親ノードからインスタンスが消去される前に別のプロセッサが同一のインスタンスを使用して新たなノードを生成してしまうと，もともとは 1 つであったインスタンス (この例では，節点 y に対応す

る節点ノードに格納されていたインスタンス)が多重に取り扱われてしまうため，最終的に得られるネットワークがもとのグラフ集合を反映したものではなくなるという問題がある．

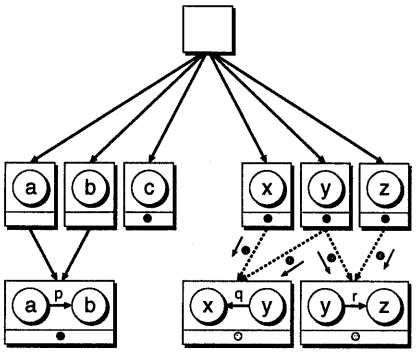

図 10: インスタンスの受け渡し

したがって，ネットワーク構築の際のノードの評価はネットワークの各部で独立して行なうことができるが，ノードの生成を行なうプロセッサは常に 1 つでなくてはならない．すなわち，ノードの生成処理は本アルゴリズムにおけるクリティカルリージョンとなっている．しかし，実際にはノードの生成にかかる時間よりも，生成するノードを決定するための評価にかかる時間の方が大きく，グラフの規模が大きくなるとその影響も小さいと予想される．図 9 からもわかるように，この処理が並列化の妨げとなることは少ないものと考えられる．

5 関連研究

SEQUITUR [2] は，記号列を先頭から順に走査しながら，その中に繰り返し現れる部分列に注目して記号列の文法ルールを発見するアルゴリズムである．SEQUITUR は文法ルールを発見する際に “Digram Uniqueness” および “Rule Utility” という 2 つの制約を設けている．“Digram Uniqueness” は隣接する 2 つの記号が複数のルールに現れてはいけないという制約である．もし，このような部分列が発見されると，その部分列は即座に新たな非終端記号によって置換される．これは，本研究における「複数のグラフに共通して現れる小さい部分構造を組み合わせ，より大きな部分構造を発見する」という手続きと同等であるとみなすことができる．また，“Rule Utility” は発見されたルールが 2 回以上使用されていなければならないという制約である．この制約は，ルールが有効に利用されることを保証するために設けられている．これは，本研究において使用している「より多くのグラフが共通して持つ部分構造を発見する」という基準と同等であるとみなすことができる．したがって，SEQUITUR の対象を，記号列からより高度なグラフ表現へと一般化したものが，本研究のアルゴリズムであると考えることもできる．

また，MDL 基準を利用してグラフの持つ階層的な部分構造を発見するアルゴリズムとしては SUBDUE [3] があげられる．SUBDUE は，部分構造の評価の際に，まずグラフの一つの節点を部分構造として採用する．そして，その節点と連結している各節点からなる部分グラフのそれぞれについて評価を行なう．その中で最も評価の高いものを採用し，これを新たな部分構造としている．このような手順を繰り返し行ない，グラフに含まれる一つの部分構造を拡張していくという方針を取っている．したがって SUBDUE では，あるグラフから発見される部分構造の数は常に一つである．そのため，複数の部分グラフの組み合わせによる元のグラフの記述を得られないという欠点がある．

本研究と同様に，複数のグラフが共通して持つ部分グラフに着目して，グラフ集合をネットワークへと組織化するアルゴリズムとしては NA [4] があげられる．NA は，与えられたグラフ集合の中から，新たに入力されるグラフと同じ構造を持つグラフの検索を行なうシステムである．しかしながら，NA は逐次的に与えられるグラフを順番に処理していくインクリメンタルなアルゴリズムであり，数多くの部分グラフの組み合わせの中から適切な構造を選択するという方法をとっていないため，最終的に得られる構造がもとのグラフ集合に対する適切な部分構造になっているとは限らない．適切な構造

が発見されずに組織化が終了した場合，同一の部分構造に対するマッチングが重複して行なわれるため，組織化を行なっているにも関わらずマッチングのコストは軽減されない．

さらに，ユーザモデルの獲得 [5] などに応用されている「データに含まれる類型的ペアの逐次拡張」という考えに基づいてグラフの部分構造を抽出し，グラフからの分類規則学習を行なう手法として GBI (Graph Based Induction) 法 [6] が提案されている．GBI 法における逐次ペア拡張アルゴリズムでは，本アルゴリズムと同様，複数のグラフの組み合わせによって元のグラフ集合を記述することが可能である．一方，複数のグラフに現れる類型的構造を抽出する基準として，分類規則学習における統計的指標が用いられている点が，本アルゴリズムと異なっている点である．ここでの「類型性」とは，データ中における出現頻度の高さを指すが，本アルゴリズムにおける評価基準である MDL 基準では，最終的な記述効率を問題としている．これらの基準の違いが，最終的に得られる部分構造に与える影響については，今後，検討を行なう必要がある．

6 おわりに

本稿では MDL 基準に基づいて発見されたグラフ集合の部分構造から，グラフの検索のためのネットワークを構築するための並列アルゴリズムについて述べるとともに，アルゴリズムのマルチプロセッサ環境との親和性を示した．

しかしながら，現在のアルゴリズムでは，負荷の分散という観点において十分な考慮がなされていない．各プロセッサは完全に独立してノードの評価を行なうため，同一のノードに対する評価が複数回行われている可能性がある．このような重複をできる限り排除して，各プロセッサが行なう処理を効率化することにより，一層の高速化が可能となるものと考えられる．

また，別の課題としては，データの内部表現の検討などがあげられる．ネットワーク中の 2 つの中間ノードのインスタンスから，新たなノードを生成した場合の評価値を計算するには，それらのインスタンス間を接続する辺があるかどうかを，あらかじめ調べておかなければならない．たとえば，節点数 m のインスタンスと節点数 n のインスタンスを考えた場合，調べる対象となる節点の組み合わせの数は mn となる．現在はグラフの内部表現として，記憶領域の小さい隣接リストを表現を採用しているが，個々のグラフのサイズがあまり大きくない場合には，記憶領域の問題の影響はあまり大きくないものと考えられる．したがって，行列表現などを採用して各種の処理をビット演算に帰着させることによって，より高速な処理が実現できるものと考えられる．

参考文献

[1] 山西健司: MDL 入門: 計算論的学習理論の立場から，人工知能学会誌, Vol. 7, No. 3, pp. 435–442 (1992).

[2] Nevill-Manning, C.G. and Witten, I.H.: Identifying Hierarchical Structure in Sequences: A linear-time algorithm, Journal of Artificial Intelligence Research, Volume 7, pp.67-82 (1997).

[3] D. J. Cook and L. B. Holder: Substructure Discovery Using Minimum Description Length and Background Knowledge, Journal of Artificial Intelligence Research, 1, pp.231-255 (1994).

[4] Messmer, B.T. and Bunke, H.: A Network Based Approach to Exact and Inexact Graph Matching, Technical Report, LAM-93-021, University of Berne (1993).

[5] K. Yoshida and H. Motoda: Automated User Modeling for Intelligent Interface, International Journal of Human Computer Interaction (Invited), Vol. 8, No. 3, pp.237-258 (1996)

[6] 吉田健一，元田 浩, “逐次ペア拡張に基づく帰納推論,” 人工知能学会誌, Vol.12, No.1, pp.58-67 (1997)