



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

(Citation)

電子情報通信学会論文誌. D-I, 情報・システム, I-コンピュータ, J82-D-1(1):111-120

(Issue Date)

1999-01-20

(Resource Type)

journal article

(Version)

Version of Record

(Rights)

Copyright (c) 1999 IEICE

(URL)

<https://hdl.handle.net/20.500.14094/90000835>



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^{†,††}

Parallel Network Organization Algorithm for Graph Matching and Subgraph
Isomorphism Detection

Keita MAEHARA[†] and Kuniaki UEHARA^{†,††}

あらまし グラフによるデータ表現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も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検索するなど、基本的な処理を高速に行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。本論文では、ラベル付き有向グラフの集合から複数のグラフが共通してもつ部分構造を発見し、発見された構造に基づいてネットワークとして組織化するアルゴリズムについて述べる。なお、本アルゴリズムでは、MDL (Minimum Description Length) 基準を用いたヒューリスティックスにより処理を高速化している。

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

1. ま え が き

グラフ表現はデータの表現形式として柔軟で汎用性が高いため、回路設計、知識表現、画像認識などさまざまな分野で用いられている。このようなグラフ表現をデータベース化して、効率的に管理することは非常に大きな意義がある。また、グラフ表現から有益な情報を発見したり、特定の構造をもつグラフを検索することは重要な課題である。例えば、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。本論文では、ラベル付き有向グラフの集合から複数のグラフが共通してもつ部分構造を発見し、発見された構造に基づいてネットワークとして組織化するアル

ゴリズムについて述べる。なお、本アルゴリズムでは、MDL (Minimum Description Length) 基準 [6] を用いたヒューリスティックスにより処理を高速化している。

2. 基本的な考え方

2.1 グラフ集合の組織化

与えられたグラフ集合の中から、特定の部分構造をもったグラフを検索することを考える。この場合、与えられたすべてのグラフに対して、対象となる部分構造とのマッチングを行う必要がある。しかしながら、グラフ集合が大規模になるとマッチングのコストが増大し、現実的な時間での検索が不可能となる。したがって、検索を効率的に行うためには、マッチングにおける冗長な繰返しを抑えることが必要である。

エキスパートシステムでは、作業記憶内の要素とルールの条件部のマッチングを高速化するために、RETE アルゴリズム [3] と呼ばれる手法が提案されている。RETE アルゴリズムは、ルールの条件部の各要素から RETE ネットワークを生成して、作業記憶内の各要素とのマッチングをたかだか 1 回で済ませて高速化を図る手法である。ここで、個々のグラフをプロダクションルール、部分グラフを作業記憶内の要素に対応

[†] 神戸大学工学部情報知能工学科, 神戸市
Department of Computer and Systems Engineering, Faculty
of Engineering, Kobe University, Nada-ku, Kobe-shi, 657-
8501 Japan

^{††} 神戸大学都市安全研究センター, 神戸市
Research Center of Urban Safety and Security, Kobe Uni-
versity, Nada-ku, Kobe-shi, 657-8501 Japan

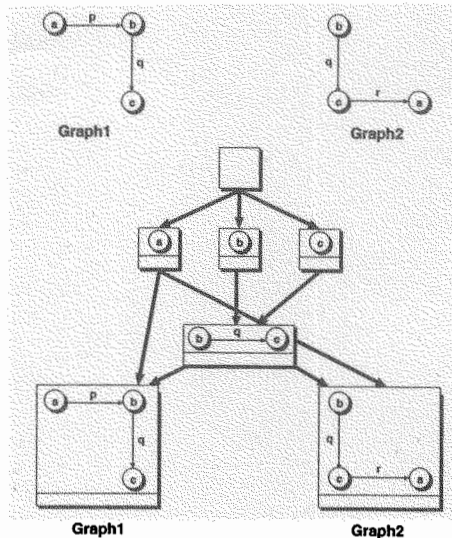


図1 グラフ表現のネットワーク化
Fig.1 Network structure for graph retrieval.

づけると、同様の手法を用いてグラフマッチングの効率化を図ることができる。例えば、図1のグラフ集合 Graph1, Graph2 が与えられたとき、共通な部分グラフに着目すれば、下部に示したネットワークを構築することができる。

グラフの共通構造を抽出して組織化を行えば、検索の高速化を実現することができる。しかしながら、グラフのもつ適切な共通構造がネットワークに反映されていない場合には、検索を効率化することはできない。例えば、図2は同一のグラフ集合から別のネットワークを構築した場合を示している。この図では、二つのグラフに共通な部分構造がネットワークノードとして抽出されていないため、組織化を行っているにもかかわらず、マッチングのコストは全く軽減されない。このように、ネットワークを構築する際には、以下の3点から考察を行う必要がある。

- 検索効率の向上

組織化によって検索効率を向上させるためには、ネットワークの構造を最適化することが必要である。このためには、すべてのグラフの構造を把握して適切な部分構造を抽出し、それに基づいて組織化を行わなければならない。

- グラフ集合の効率的記述

空間計算量の観点からいえば、可能な限り少量の情報でもとのデータを表現できることが好ましい。すな

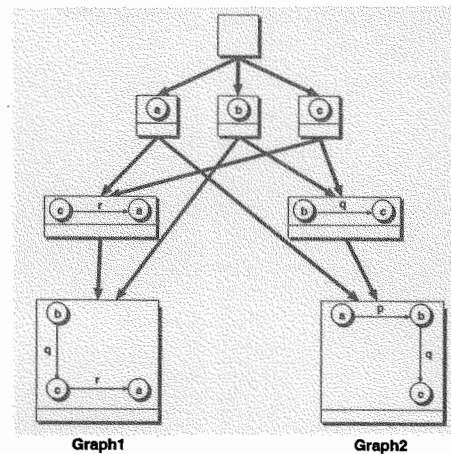


図2 非効率なグラフ表現のネットワーク化
Fig.2 Inefficient network structure.

わち、ネットワークの記述コストを小さくすることが記憶容量の効率化につながる。一般的なデータ圧縮アルゴリズムで用いられている手法と同様に、グラフの処理においても、複数のグラフに繰り返し現れる構造に基づいて、より少ない情報量でもとのデータを表現することを考えなければならない。

- 重要な部分構造の発見

グラフ集合中に繰り返し現れる部分構造は、概念的にも重要な意味をもっているものと考えられる。したがって、このような構造を抽出することができれば、応用上も非常に有益である。この観点からも、複数のグラフ間に共通する部分構造に着目することは重要である。

3. グラフ集合の共通構造

3.1 MDL 基準による部分構造の発見

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3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3(a)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。また、図3(b)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。

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。例えば、図3で示した四つのグラフが与

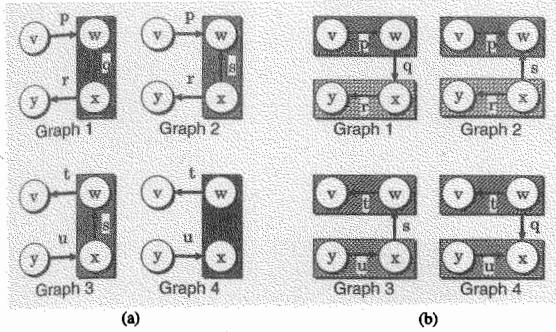


図3 グラフ集合と部分グラフ
Fig. 3 Graphs and their subgraphs.

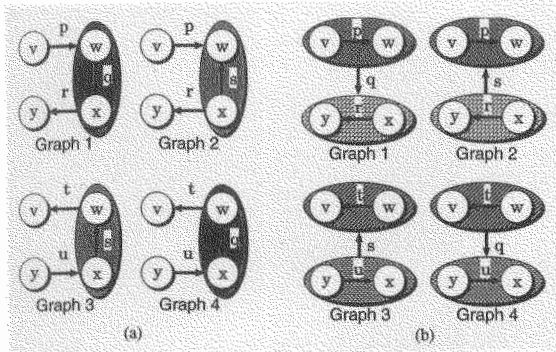


図4 グラフ集合の効率的な記述
Fig. 4 Efficient description of graphs.

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4）。

図4(a)では、図3(a)で着目した部分グラフが一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(b)では、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も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一つとして、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 基準は、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。

すなわち、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もつ部分グラフの組合せの一つを $SGList$ とすると、MDL 基準により $SGList$ の評価関数 $Eval(SGList)$ は、式(1)で与えられる。

$$Eval(SGList) = DL(SGList) + DL(GSet|SGList) \quad (1)$$

なお、 $DL(SGList)$ は $SGList$ の記述長、 $DL(GSet|SGList)$ は $SGList$ を用いて $GSet$ を書き換えた際の記述長である。

$DL(SGList)$ は以下のように定義される。部分グラフ SG の節点数を v 、節点のもつラベル数を l_v 、辺数を e 、辺のもつラベル数を l_e としたとき、節点、辺をエンコードするのに必要なビット数はそれぞれ

$$b_v = \log_2 v + v \log_2 l_v \text{ [bits]} \quad (2)$$

$$b_e = \log_2 e + e \log_2 l_e \text{ [bits]} \quad (3)$$

で与えられる。更に、それぞれの辺の両端がどの節点に接続されているかという情報をエンコードするのに必要なビット数は

$$b_c = 2e \log_2 v \text{ [bits]} \quad (4)$$

で与えられる。したがって、

$$\begin{aligned} DL(SG) &= b_v + b_e + b_c \\ &= \log_2 v + v \log_2 l_v + \log_2 e \\ &\quad + e \log_2 l_e + 2e \log_2 v \end{aligned} \quad (5)$$

$$DL(SGList) = \sum_{i=1}^n DL(SG_i) \quad (6)$$

更に、グラフ G の部分グラフ SG を一つの節点とみなして置換した場合の G の記述長 $DL(G|SG)$ を、以

下のように定義する。SG に相当する部分グラフが G に n_{SG} 個含まれているものとし、置換後の節点数が v' 、辺数が e' になったものとする。このときグラフの辺の両端がどの節点に接続されているかという情報を失うことなくエンコードすると

$$DL(G|SG) = \log_2 v' + v' \log_2 (l_v + n_{SG}) + \log_2 e' + e' \log_2 l_e + 2e' \log_2 v' \quad (7)$$

$$DL(G|SGList) = \sum_{j=1}^n DL(G|SG_j) \quad (8)$$

$$DL(GSet|SGList) = \sum_{i=1}^N DL(G_i|SGList) \quad (9)$$

となる。

評価関数 $Eval(SGList)$ は、値が小さいほどもの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。例えば、図 3(a) と図 3(b) の記述長は式 (5) 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、式 (1) より図 4(a) の場合は 86.2 [bits]、図 4(b) 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(b) の 4 種類の部分グラフを用いればよいことがわかる。

3.2 探索範囲の絞り込み

MDL 基準を採用しても、もとのグラフをより効率的に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しなければならない。例えば、 n 節点のグラフが与えられたとき、そこから節点を採用する方法は $\sum_{i=1}^n nC_i$ 通り存在する。更に、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もつ部分グラフの組合せについても考える必要があり、問題の規模が大きくなると可能な組合せは膨大になる。

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図っている。例えば、図 5 は Graph1, Graph2, Graph3 の三つのグラフに共通な Subgraph1 が既に解として選択されており、次に選択される候補として

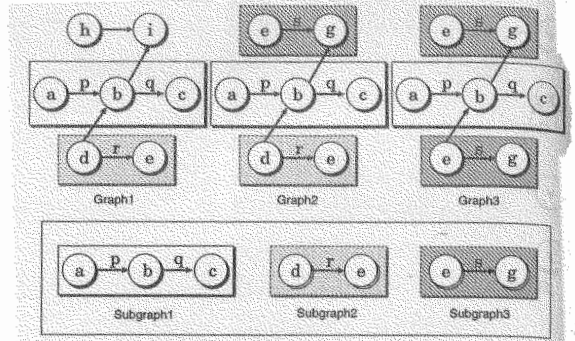


図 5 部分グラフの選択基準
Fig.5 Selection policy of subgraph.

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。

4. ネットワーク生成アルゴリズム

本章では、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、発見された部分グラフに対応している。また、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、グラフ検索を効率的に行うことが可能となる。

4.1 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ぶ。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、それぞれに対応する節点ノードが生成される。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 (図 6)。

図 7 は、この手順を擬似コードによって示したものである。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 *vertex* のラベルと同じラベルをもつ節点ノード *newNode* を新たに作成し、

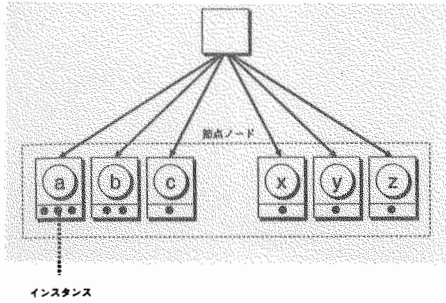


図6 節点ノードの生成
Fig.6 Generation of vertex nodes.

```

Procedure CreateVertexNodes(graphSet)
begin
  for each vertex in graphSet
    begin
      if (vertex-node for vertex
        has not been created) then
        begin
          Create a vertex-node newNode;
          Link newNode to input-node
        end;
        Store an instance of vertex into the vertex-node
      end
    end
  end.

```

図7 節点ノードの生成アルゴリズム
Fig.7 Vertex-node generation algorithm.

節点 *vertex* のインスタンスを格納する^(注1)。

次に中間ノードが生成される。中間ノードは、節点ノードや他の中間ノードの組合せによって生成される部分グラフを表現している。図8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うことができるため、並列に処理を行うことにより高速化が実現できることになる。

図9はこの手順を擬似コードによって示したものである。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 *instance1* と *instance2* の二つに注目する。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。更に、MDL基準に基づくヒューリスティクスに従い、もとのグラフ集合の記述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、高い評価を得る組合せから新しいノード *newNode* を

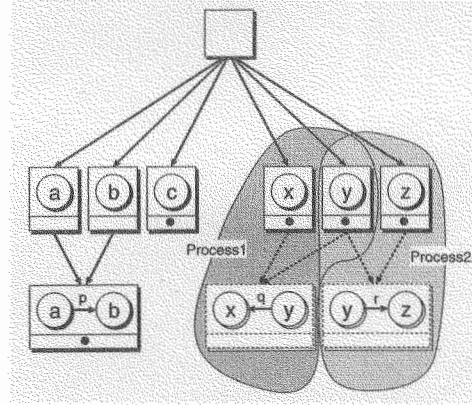


図8 中間ノードの生成 (a)
Fig.8 Generation of inter nodes (a).

```

Procedure CreateInternodes(graphSet);
begin
  local nodeSet, newNode, instance1, instance2;
  while (all graph-nodes for graphSet
    have not been created)
    begin
      nodeSet := vertex-nodes and inter-nodes
        which have their instances;
      Select instance1 from nodeSet;
      begin parallel
        for each instance2 where an edge
          between instance1 and instance2
          does exist
          begin
            Evaluate the description length of
              graphSet with respect to
              both instance1 and instance2;
          end
        end parallel
        if (the combination of instance1
          and instance2 exists) then
          begin
            Select the best combination
              of instance1 and instance2;
            Create a new inter-node newNode
              for the combination;
            Delete instance1 and instance2;
            Store an instance into the new inter-node;
          end
        end
      end
    end.

```

図9 中間ノードの作成アルゴリズム
Fig.9 Inter-node generation algorithm.

生成するようにしている。

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い、その中で最も評価の高いものを実際に生成

(注1): 同一のラベルをもつ節点が多数存在する場合、一つの節点ノード内に相当数のインスタンスが格納される。

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。例えば、図 8 では、プロセス Process1, Process2 がそれぞれ節点 y から節点 x への辺 q をもつグラフと、節点 y から節点 z への辺 r をもつグラフの評価を行っている。その結果、節点 y から節点 x への辺 q をもつグラフの評価が高かったとすると、図 10 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。

新しいノードが生成される際には、二つの親ノードから対応するインスタンスが削除され、新しい一つのインスタンスとしてそのノードに格納される。図 10 では、節点 y から節点 x への辺 q をも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一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、更に新たなノードを作成する際の候補となる。図 10 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。

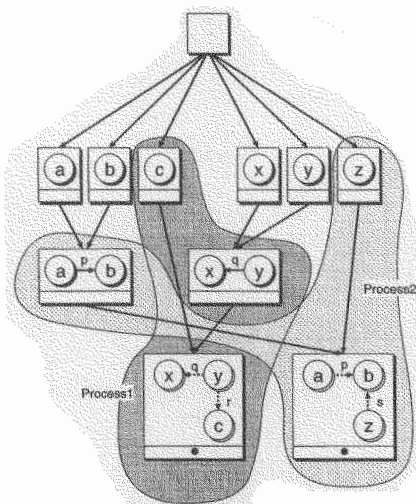


図 10 中間ノードの生成 (b)
Fig. 10 Generation of inter nodes (b).

5. 実 験

5.1 MDL 基準に基づくヒューリスティックスの有効性

3.2 で述べたヒューリスティックスの有効性を示すために、総節点数 12, 14 の三つのグラフ集合及び図 3 で示した四つのグラフ集合（総節点数 16）を入力し、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合せの探索を行う実験を行った。表 1 は、部分グラフの組み合わせ探索に際して、完全にランダムな場合 (random), 3.2 で述べたヒューリスティックスを用いた場合 (heuristic), 及びすべての部分グラフの組合せを考慮した場合 (brute force) の処理時間と式 (1) による評価値を示している。なお各結果は 10 回ずつ実験を行った平均値及び分散値である。

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一つの候補を選択して式 (1) による評価を行う。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、その候補を解の一つとし、更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、再度式 (1) による評価を行う。この評価値が以前のものよりも小さくなっていけば、新たに選択された候補も解の一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くなるまで行っている。一方、探索方針として heuristic を採用した際には、式 (1) による部分グラフの評価値が小さいものから候補の選択を優先的に行っている。

表 1 部分グラフ探索方式の比較 (総節点数 12, 14, 16)
Table 1 Comparison of three subgraph search methods (the number of vertices are 12, 14, 16).

	search method	processing time	evaluation value
12	random	1s (0)	75.0 bits (59.0)
	heuristic	1s (0)	67.4 bits (12.7)
	brute force	16s	65.3 bits
14	random	1s (0)	88.1 bits (49.0)
	heuristic	1s (0)	80.9 bits (17.1)
	brute force	71s	73.1 bits
16	random	1s (0)	97.4 bits (73.1)
	heuristic	1s (0)	72.6 bits (21.3)
	brute force	-	-

表2 部分グラフ探索方式の比較 (総節点数 257)
Table 2 Comparison of two subgraph search methods
(the number of vertices are 257).

search method	processing time	evaluation value
random	139 s (1756.3)	4467.8 bits (18299.0)
heuristic	103 s (77.8)	4237.6 bits (18188.2)

表1からもわかるように、brute force ではすべての組合せを考慮しているために、節点数16ほどで組合せの爆発が生じ、測定不能となっている。このため、総節点数257の五つのグラフ集合に対しては、random と heuristic に対してのみ同様の実験を行った。なお各結果は10回ずつ実験を行った平均値及び分散値である。この実験結果から、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がわかる。また、それぞれの分散値からもわかるように、random の処理時間には大きなばらつきがあるのに比べて、heuristic の処理時間は比較的小さな範囲内で収まっている。更に、ランダムに評価を行った場合に比べて、与えられたグラフ集合を効率よく記述できていることがわかる。しかしながら、いずれの場合も評価値には大きなばらつきがあり、最適な部分構造が発見されているとはいえない。

5.2 並列化の効果

並列処理による効果を検証するため、日本昔ばなしの「おむすびころりん」をもとにストーリーの内容記述を行い、Schankにより提案された概念依存表現[5]を用いてグラフ表現した。対象としたデータは、総節点数257, 514, 771及び1285からなる4種類のグラフ集合である。なお、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: 512 Mbytes, OS: IRIX 6.4) を使用している。このときの使用プロセッサ数と処理時間の関係を図11に示す。各結果は10回ずつ実験を行った平均値である。新たに生成するネットワークのノードの評価を並列的に行うことにより、図11では、プロセッサ数の増加に対する明確な処理速度の向上が見られる。

既に述べたように、図9の中間ノードを生成する処理は、もとのグラフにおいて辺で連結されているインスタンスの組合せに対して、グラフ集合の記述長を計算するノード評価処理と、ノード評価処理で評価されたインスタンスの組合せの中から評価値が最も小さいものを選び、その組合せで実際に中間ノードを生成す

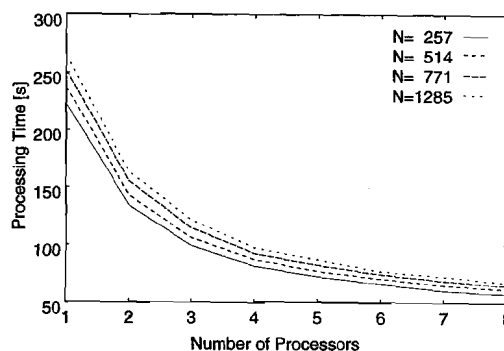


図11 並列化の効果

Fig. 11 Experimental result of parallel network organization algorithm.

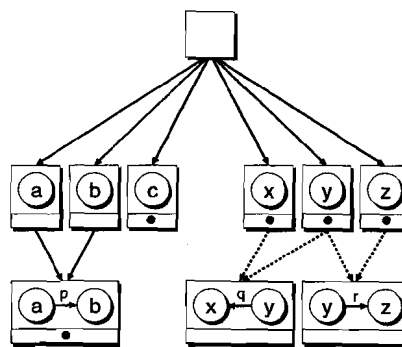


図12 インスタンスの受け渡し

Fig. 12 Instance passing in the network.

るノード決定・生成処理からなる。

このうちノード評価処理の部分は、各プロセッサによって独立に行われているため、同一のノードに対する評価が複数回行われる可能性がある。このような重複を排除することができれば、よりいっそうの高速化を図ることが可能になる。

ノード決定・生成処理の部分では、ノードを生成した二つの親ノードに格納されている各インスタンスを削除し、新たに生成された子ノードに新たなインスタンスを1個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う場合には注意が必要である。例えば、図12の点線部は、節点yから節点xへの辺qをもつグラフと、節点yから節点zへの辺rをもつグラフの二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12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

使用して新たな子ノードを生成してしまうと、本来は一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。このため、本来のグラフで1ノードであったものが、あたかも複数存在したかのようなネットワークとして表現されてしまうという問題がある。言い換えると、ノード評価処理は、ネットワークの各部で独立して行うことができるが、ノード決定・生成処理を行うプロセッサは常に1個でなくてはならない。すなわち、ノード決定・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。

また、ノード評価処理はグラフの規模が大きくなると組合せ数が増大し、処理時間も大きくなるが、ノード決定・生成処理は、評価した組合せの中から一つのノードを生成するのみで、グラフの規模にかかわらずほぼ無視できる程度である^(注2)。したがって、グラフの規模が大きくなってもクリティカルリージョンでの計算に要する時間は小さく、並列化に与える影響も小さいと考えられる。

5.3 部分構造の発見

図13は、前述のデータに対して本組織化アルゴリズムを適用した結果を示したものである。もとのグラフ表現では、オブジェクト間の関係が複雑であり、構造を把握することが非常に困難であるが、組織化を行うことによって、データがもつ階層的な部分構造やそれらの関係が非常に明確になり、場面の検索などに有効に利用することができる。また、図13では、「おじいさんが箸を持っているシーン」といった小さなシーンから「ねずみの娘が料理を手に持って、部屋の外からおじいさんの前までやってくるシーン」といった比較的大きなシーンまで、さまざまな場面に相当する構造が発見されている。ただし、本アルゴリズムではグラフ集合の記述長のみに注目しているために、意味的に重要な部分構造が発見されているとは限らない。この問題に対しては、部分構造を採用する基準としてMDL基準のほかに背景知識を用いて、意味的に重要な構造を発見するという方法が考えられる。

(注2)：グラフの規模が大きくなると作成される中間ノードが増加するため、インスタンスの処理により若干ノード生成処理の時間は増加するが、全体の処理時間に比べて無視できるほどである。

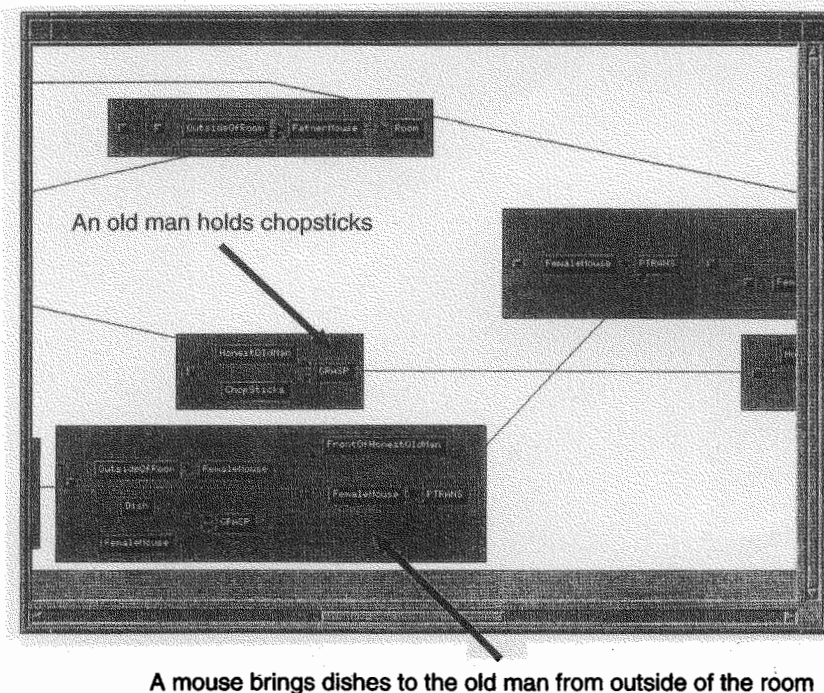


図13 グラフ表現から発見された部分構造
Fig.13 Interesting subgraphs discovered from network.

6. 関連研究

本論文で示したアルゴリズムと類似した、大規模なデータ中から重要な部分構造を発見したり、部分構造に基づく組織化を行うアルゴリズムは、既にいくつか提案されている。このうち、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 の利用例としては、複数の言語で書かれた同一の文章から、単語の構造や同じ意味をもつ単語の組を発見したり、楽譜の中で繰り返されるフレーズを抽出する例などがある。

SEQUITUR は、文法ルールを発見する際に Digram Uniqueness 及び Rule Utility という二つの制約を設けている。Digram Uniqueness は隣接する二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も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 の対象を単なる記号列から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られる。

MDL 基準を利用してグラフのも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 [1] がある。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とっている。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 1 個である。また、本論文で示したアルゴリズムとは異なり、グラフの検索などは考慮されていない。

本研究と同様に、複数のグラフが共通してもつ部分

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もつグラフの検索を行うシステムである。NA は逐次的に与えられるグラフを順番に処理するインクリメンタルなアルゴリズムである。NA は、数多くの部分グラフの組合せの中から適切な構造を選択するという方法を用いていないため、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われるため、組織化を行っているにもかかわらず、マッチングのコストは軽減されないという問題が生じる可能性がある。

ユーザモデルの獲得 [7] 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う手法として GBI (Graph Based Induction) 法 [8] 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に、複数のグラフの組合せによってもとのグラフ集合を記述することが可能である。しかしながら、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を用いていることが、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、本アルゴリズムの評価基準である 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、検討を行う必要がある。

7. む す び

本論文では 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。

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。5. でも述べたように、各プロセッサは完全に独立してノードの評価を行うため、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、いっそうの高速化が可能となるものと考えら

れる。

今後の課題としては、データの内部表現の検討が挙げられる。ネットワーク中の二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。例えば、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合せの数は 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余り大きくない場合には、記憶領域の問題の影響は余り大きくないものと考えられる。したがって、行列表現を採用して各種の処理をビット演算に帰着させれば、より高速な処理が実現できるものと考えられる。

謝辞 本研究は、文部省科学研究費重点領域研究「高度データベース」及び日本学術振興会未来開拓学術研究推進事業における研究プロジェクト「マルチメディア・コンテンツの高次処理の研究」の援助を受けて行われたものである。また、本研究の追試実験を担当された本学大学院博士前期課程の中西剛紀君に感謝する。

文 献

- [1] D.J. Cook and L.B. Holder, "Substructure discovery using minimum description length and background knowledge," *Journal of Artificial Intelligence Research*, vol.1, pp.231-255, 1994.
- [2] C.G. Nevill-Manning and I.H. Witten, "Identifying hierarchical structure in sequences: A linear-time algorithm," *Journal of Artificial Intelligence Research*, vol.7, pp.67-82, 1997.
- [3] C.L. Forgy, "RETE: A fast algorithm for the many pattern/many object pattern match problem," *Artificial Intelligence*, vol.19, no.1, pp.17-37, 1982.
- [4] B.T. Messmer and H. Bunke, "A network based approach to exact and inexact graph matching," *Technical Report, LAM-93-021, University of Berne* 1993.
- [5] R.C. Schank and C.K. Riesbeck, 石崎俊監 訳, "自然言語理解入門—LISP で書いた 5 つの知的プログラム," pp.12-47, 総研出版 1986.
- [6] 山西健司, "MDL 入門: 計算論的学習理論の立場から," *人工知能誌*, vol.7, no.3, pp.435-442, 1992.
- [7] K. Yoshida and H. Motoda, "Automated user modeling for intelligent interface," *International Journal of Human Computer Interaction*, vol.8, no.3, pp.237-258, 1996.
- [8] 吉田健一, 元田 浩, "逐次ベア拡張に基づく帰納推論," *人工知能誌*, vol.12, no.1, pp.58-67, 1997.

(平成 10 年 5 月 6 日受付, 8 月 28 日再受付)

前原 恵太



平 8 神戸大・工・情報知能工卒。平 10 同大学院自然科学研究科情報知能工学専攻博士前期課程修了。現在、ヤマハ(株)勤務。在学中は主に機械学習、マルチメディアデータベースの研究に従事。

上原 邦昭



昭 53 阪大・基礎工・情報工卒。昭 58 同大学院博士後期課程単位取得退学。同大産業科学研究助手、講師、神戸大学工学部情報知能工学学科助教授を経て、現在同大学都市安全研究センター教授。同大学情報知能工学科を兼任。平 1~2 Oregon State University, Visiting Assistant Professor。平 6~8 同大学総合情報処理センター副センター長。工学博士。人工知能、特に機械学習、マルチメディアデータベース、自然言語によるヒューマンインターフェースの研究に従事。1990 年度人工知能学会研究奨励賞受賞。人工知能学会、情報処理学会、計量国語学会、日本ソフトウェア科学会、AAAI 各会員。