2006－AL－105（8）
2006／3／17

属性付きグラフマッチングアルゴリズムの効率的な実装

森田　昭広†　　古賀　久志†　　渡辺　俊典†　　横山　貴紀†

† 電気通信大学 大学院 情報システム学研究科
182-8585 東京都調布市調布ケ丘 1-5-1
E-mail: †{a-morita,koga,watanabe,yokotaka}@sd.is.uec.ac.jp

概要　グラフのマッチング問題は一般に計算量が膨大であるが，問題固有の属性情報などを用いて効率的な探索を実現できる可能性がある．本研究では，グラフマッチング問題が入力 2 グラフから生成される積グラフの最大クリークを抽出する問題へ還元できることに着目し，その効率化のために 2 つの属性情報利用アルゴリズムを考案した．1 つ目はクリーク抽出の探索過程で属性情報を用いて探索領域を削減する方法，2 つ目は積グラフの生成時に属性情報を用いて積グラフの規模自体を抑制する方法である．これらを計算機実験によって比較検証した結果，双方共に有効であるが，特に後者の有効性が顕著であることを確認した．

An Efficient Implementation of Attribute-Graph Matching Algorithm

Akihiro MORITA†, Hisashi KOGA†, Toshinori WATANABE†, and Takanori YOKOYAMA†

† Graduate School of Information Systems
University of Electro-Communications
E-mail: †{a-morita,koga,watanabe,yokotaka}@sd.is.uec.ac.jp

Abstract　Graph matching problem has a very high computational complexity. But we can reduce it by exploiting domain-specific information such as object's attributes. In this research, where we solve the graph matching problem by reducing it into a maximum clique problem in a product graph generated from the two input graphs, we propose two algorithms, both exploiting attribute information. One is the method of decreasing the search space by using attribute information in the process of maximum clique search. The other is the method of decreasing the size of the product graph by using attribute information during the product graph generation. Through experiments we showed that, although both are effective, the latter dominates the former.

1. はじめに

現実問題の中にはグラフ上の問題として定式化し，解決できるものが存在することは広く知られている．その中でも検索や認識など，データ間の共通性を調査する必要がある場合は，グラフのマッチング問題に帰着して解くことが可能である．これらはグラフの同型性を判定するアルゴリズムなどによって支えられているが，その計算量が膨大であることもまた事実であり，より効率的な解法が求められる．

これまでにも J. R. Ullmann や L. P. Cordella らによって部分グラフ同型問題に対する高速なアルゴリズムが提案されてきた．この問題は，モデルとして与えるグラフと完全一致するグラフしか発見しないので，極めて正確なモデリングが要求され，曖昧な情報を多く含む対象には適用しづらい．しかしながら，画像処理などの現実の応用分野では，曖昧マッチングである類似検索がしばしば用いられており，モデルに対する柔軟性は重要である．

そこで本研究では，部分グラフ同型問題を拡張し，入力された 2 グラフそれぞれの部分グラフ間で同型調査を行い，同型となる部分グラフが頂点数最大になる組み合わせを求める問題を取り扱う．これを最大部分グラフ同型問題とここでは定義する．この拡張によってモデルが明らかになっていない状況でもグラフ間のマッチングが適用可能になる．

一方で，最大部分グラフ同型問題の欠点として計算量の更なる増大があり，実問題に最大部分グラフ同型を適

用するには，問題固有の属性情報 (ラベル) を利用して，調査する組み合わせの数を減らすのが現実的である．ここではグラフの頂点に属性情報が付与されている場合の，効率的な属性情報の使い方について論じる．

本研究では最大部分グラフ同型問題が，入力された 2 グラフから生成される積グラフにおいて最大クリークを抽出する問題に還元できることに着目し，2 種類の属性情報利用アルゴリズムを考案した．1 つ目はクリーク抽出の探索過程で探索領域削減に属性情報を用いる方法，2 つ目は積グラフの生成時に属性情報を用いる方法である．前者は定義通りに積グラフを生成し，そこから最大クリークを抽出する段階で属性情報を利用して探索領域を削減する．それに対して，後者は最大クリークの要素となり得ない孤立頂点を排除し，積グラフの頂点数自体を抑制する手法となる．

本稿では，これら 2 手法の実行時間を調査することで実験的に比較した結果，後者の方がより有効な手段であることを検証した．

2. グラフ同型

グラフ間のマッチングについて議論するには，まずグラフ同型 (graph isomorphism) 問題への言及が必要であろう．

任意のグラフ G_1，G_2 が同型であるとは，頂点 i と j を結ぶ辺を (i,j) で表したときに，$V(G_1)$ から $V(G_2)$ への全単射 $\varphi : V(G_1) \to V(G_2)$ が存在し，任意の i, $j \in V(G_1)$ に対して

$$(i,j) \in E(G_1) \Leftrightarrow (\varphi(i),\varphi(j)) \in E(G_2)$$

が成り立つことを言う．またこのとき，φ を G_1 から G_2 への同型写像と呼ぶ．図 1 には同型である G_1 と G_2，それらと同型でない G_3 の具体例を示す．グラフ同型問題とは，2 つのグラフが入力データとして与えられたとき，その 2 グラフの頂点間の対応関係を求める問題である．なお，グラフ同型問題の計算量は，単純なアルゴリズムを用いた場合，最悪で $O(|V(G_1)|!)$ となる．

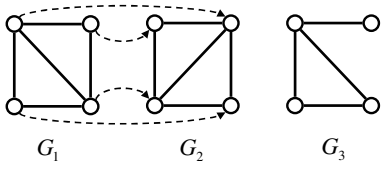


図 1　グラフ同型

これを更に，与えられた G_1 が G_2 のいずれかの部分グラフと同型であるか，という問題に発展させたものが部分グラフ同型 (subgraph isomorphism) 問題である．これはグラフ同型問題を最悪で ${}_{|V_2|}C_{|V_1|}$ 回解くことに相当し (ただし，$|V_1| < |V_2|$)，モデルとなるグラフを与えて同じパターンを発見する，いわゆるパターンマッチングの一手法である．この解法としては，Ullmann のアルゴリズム [1] や L. P. Cordella らによる VF [2]，VF2 [3] といったアルゴリズムなどが高速なアルゴリズムとして知られている．図 2 には部分グラフ同型である G_1 と G_2，そうでない G_3 の例を示す．

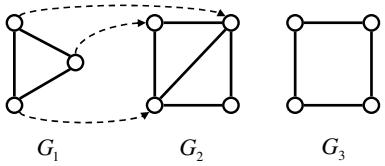


図 2　部分グラフ同型

部分グラフ同型問題は片方のグラフだけの部分グラフを考えるが，より一般性のあるグラフ間マッチングを実現するには，入力された両グラフの部分グラフを考慮する必要がある．つまりこれは，G_1，G_2 それぞれの部分グラフで同型性を判定する問題であり，その中で部分グラフの頂点数が最大となる写像を求める問題を最大部分グラフ同型 (maximum subgraph isomorphism) 問題と位置付ける．

ここで，$iso(G_1,G_2)$ を以下のように定義すると，

$$iso(G_1,G_2) = \begin{cases} V(G_1) & (G_1,G_2\text{が同型のとき}) \\ 0 & (\text{それ以外のとき}) \end{cases}$$

最大部分グラフ同型問題は

$$max_{subG_1,subG_2}|iso(subG_1,subG_2)|$$

を求める問題であると定義できる．これは入力された 2 つのグラフに対して，各々どの部分グラフを選択して対応付けるのが最適か，という問題でもある．なお，この問題はグラフ同型問題を最悪で $\sum_k {}_{|V_1|}C_k \bullet {}_{|V_2|}C_k$ 回解くことに相当する．図 3 には G_1 と G_2 の最大部分グラフ同型の例を示す．

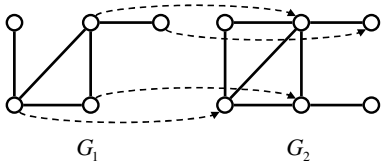


図 3　最大部分グラフ同型

3. 積グラフ上のクリーク抽出

この節では本研究で着目する最大部分グラフ同型問題に対して，グラフの積とクリークの抽出を用いた解法のフレームワークを述べる．

グラフ G は頂点集合 $V(G)$ と辺集合 $E(G)$ によって構成されているが，入力された 2 グラフに対し，グラフの積によって得られる積グラフの各頂点は，$V(G_1)$ と $V(G_2)$ の全ての組み合わせに相当している．そのため，2 グラフ間の対応を求めるには積グラフを利用することができると考えられる．

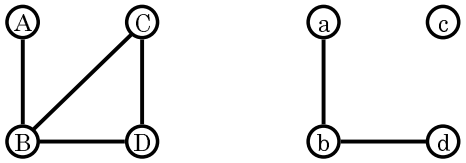
積グラフの辺についてはいくつかの定義があるが，最大部分グラフ同型問題を解決するためには，辺の存在の完全な一致性を調査する必要があり，これを実現するために本稿では EA 積 (Existence Agreement product) を以下のように定義する．

定義: G_1 と G_2 の EA 積 $G_1 \ {}_{EA}\ G_2$

$$
\begin{aligned}
V(G_1 \ {}_{EA}\ G_2) &= V(G_1) \times V(G_2) \\
E(G_1 \ {}_{EA}\ G_2) &= \{((u_1,u_2),(v_1,v_2)) : \\
&\quad (u_1,v_1) \in E(G_1) \text{かつ} (u_2,v_2) \in E(G_2), \\
&\quad \text{or } (u_1,v_1) \notin E(G_1) \text{かつ} (u_2,v_2) \notin E(G_2)\}
\end{aligned}
$$

この計算によって生成される EA 積グラフにおいて，各頂点は 1 つの写像 (1 組のノード対) を意味しており，各辺はそれらの写像が同時に成立するかを表している．

図 4 は EA 積グラフの隣接行列であり，例えば $\{Aa, Bb\}$ という写像は同時に成立しうるので，その値は “1” となっていることが確認できる．



G_1 G_2

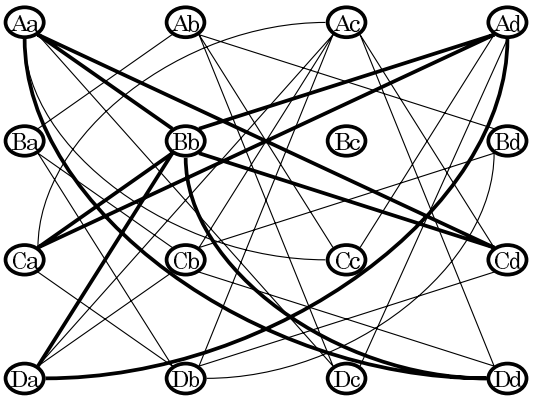
		A				B				C				D			
		a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
A	a	0	0	0	0	0	1	0	0	0	0	1	1	0	0	1	1
	b	0	0	0	0	1	0	0	1	0	0	1	0	0	0	1	0
	c	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1
	d	0	0	0	0	0	1	0	0	1	0	1	0	1	0	1	0
B	a	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
	b	1	0	0	1	0	0	0	0	1	0	0	1	1	0	0	1
	c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	d	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
C	a	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0
	b	0	0	1	0	1	0	0	1	0	0	0	0	1	0	0	1
	c	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	d	1	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0
D	a	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0
	b	0	0	1	0	1	0	0	1	1	0	0	1	0	0	0	0
	c	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	d	1	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0

$G_1 \ {}_{EA}\ G_2$

図 4　EA 積グラフ生成の例

また，上記の計算によって生成される EA 積グラフの様子を図 5 に示すが，4 つの最大クリークを構成する $\{Aa, Bb, Cd\}$，$\{Aa, Bb, Dd\}$，$\{Ad, Bb, Ca\}$，$\{Ad, Bb, Da\}$ が，確かに G_1 と G_2 の最大部分グラフ同型の解となる写像であることが分かる．よって，この EA 積グラフから最大クリークを抽出することで，最大部分グラフ同型問題の要求に応えるアルゴリズムを構築することが可能である．

このフレームワークの利点は，2 グラフ間のマッチングを既に数多くのアルゴリズムが開発されている最大クリーク抽出問題に還元して扱えることにある．この最大クリーク抽出問題に対しては，各所から様々な厳密解アルゴリズムが発表されており ([4], [5], [6])，今後も新たな手法が提案される可能性があるため，その効果を取り込むことができるシステムにしておくことは重要である．なお，本研究では筆者らが提案した MCQ” [7] を用いる．



$G_1 \ {}_{EA}\ G_2$

図 5　EA 積グラフと 4 つの最大クリーク

4. 属性情報の効率的な利用法

最大部分グラフ同型問題は，実問題への適用する場合，何らかの属性情報を付与できるため，これを組み合わせ数の制限に用いることが妥当である．例えば分子の構造をグラフとしてモデリングする場合には，その分子を構成する原子の種類を頂点に与えることなどが考えられる．本稿では頂点に属性を与えられた場合に，その情報を用いた制約を加えて，最大クリーク抽出に必要な探索領域・実行時間の削減が期待できる以下の 2 手法を検討する．

4.1 手法 1: 分枝限定時の利用

最大クリーク抽出の効率化として広く用いられている手法に，何らかの条件を利用して探索に必要な領域を削減する分枝限定法がある．本研究で用いている MCQ” も例外ではなく，このアルゴリズムはグラフの彩色数が最大クリークサイズの上界を与える，といった事実を探索領域の削減に活かしていることが最も重要な要素である．実装上は深さ優先探索の各状態で，グラフの彩色数による制約条件を満たしていなければその先の探索を打ち切るという仕組みになっているが，ここに属性の一致という制約を加えることで探索領域の更なる削減を実現する．図 6 にこのフレームワークを示す．

前節の図 4 に挙げたグラフの頂点に 3 種類の属性 (○, △, □) を与えることにすると (図 7)，これらの EA 積グラフは定義により図 5 と全く同型であるが，属性の一致しない組み合わせに遭遇した場合は，その先の探索を打ち切ることで効率化を実現できる．

4.2 手法 2: 積グラフ生成時の利用

この手法は，積グラフの生成段階において予め上記の属性情報による制約条件を用い，最大クリーク抽出を行

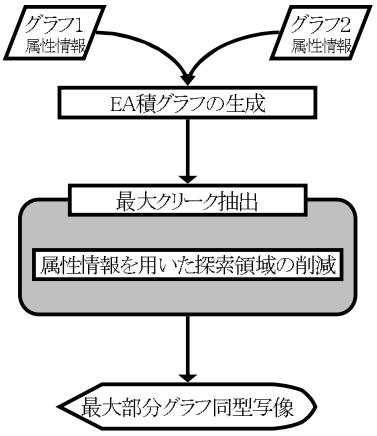


図 6　分枝限定時の属性情報利用

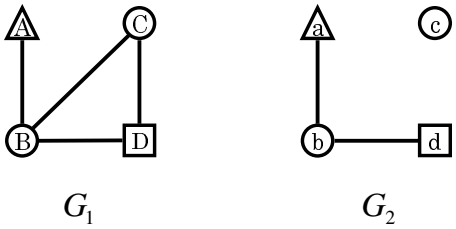


図 7　3 種類の属性を与えたグラフ

う積グラフの頂点数自体を抑制する．全体の流れとしては，まず属性情報が一致せず，マッチングの可能性がない組み合わせを予め除外しておくことで積グラフの辺数を減らす．これによって積グラフにおける孤立頂点の数を増加させ，クリークの成分となる可能性がないこれらの頂点を除去することで，生成されるグラフの頂点数を抑制し，その結果，最大クリーク抽出に要する実行時間を短縮させることが期待できる．図 8 にこのフレームワークを示す．

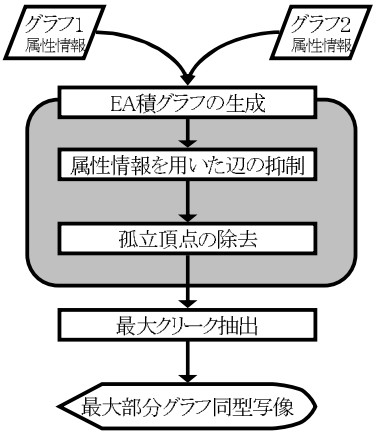


図 8　積グラフ生成時の属性情報利用

実際に図 7 の 2 グラフからどのような積グラフが生成されるかを示す．まず EA 積グラフを生成する際に，EA 積の定義に加え，属性の一致という条件を満たす辺のみを残す．つまり，$u_1, v_1 \in V(G_1)$ と $u_2, v_2 \in V(G_2)$ について，u_1 と u_2 の属性が一致，かつ，v_1 と v_2 の属性が一致という条件を満たさない限り，その辺は削除されることになる．なお，属性が一致する確率を $P(u, v)$ で表すと，EA 積グラフ上の辺が残る割合は $P(u_1, v_1) \times P(u_2, v_2)$ である．この処理によって積グラフが疎になり，最大クリークの構成要素と成り得ない孤立頂点の数が増加しているため，それらを除去することで規模を抑制した積グラフを生成することができる．図 9 には上記の操作によって生成された積グラフ (実線) と，削除された辺と頂点 (破線) の様子を示す．

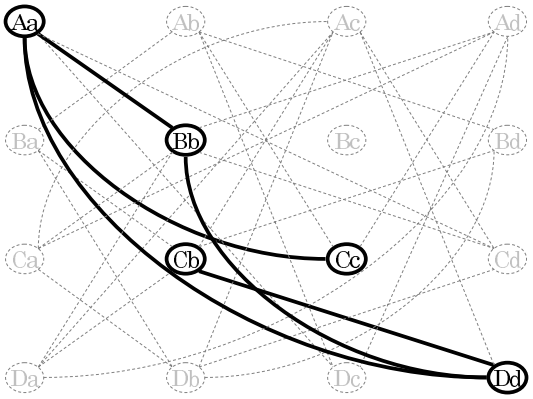


図 9　属性情報によって頂点数が抑制された積グラフ

5. 計算機実験

上記の手法 1 と手法 2 の比較と評価をするため，C 言語を用いて計算機上に実装し，辺の存在と頂点の属性を乱数によって設定したグラフに対して実行した．なお，使用した計算機環境は以下の通りである．

- OS: Linux
- CPU: Pentium4 1.8GHz
- メモリ: 512MB
- コンパイラ: gcc -O3

表 1〜表 6 には実行結果を示すが，表中の N_1，N_2 は入力された 2 グラフの頂点数を表しており，手法 1 と手法 2 の欄には各々のアルゴリズムを用いてマッチングを行った実行時間を掲載してある．また，頂点数に関しては手法 2 の孤立頂点除去によって，生成された積グラフの頂点数がどこまで減少したかを表している．なお，最右欄には手法 1 と手法 2 の実行時間比を示す．

これらのデータを見ると，手法 2 の実行時間は手法 1 と比べて大幅に短いことが明らかであり，効率的に解を得るためには積グラフの頂点数を抑制することが如何に重要であるかが分かる．また，図 10〜図 12 には，入力されたグラフの特性によって実行時間がどの様に変化するかをまとめた図を掲載する．まず，頂点数と実行時間の関係を見るために，辺密度と属性の種類を固定したものが図 10 であるが，当然のように頂点数の増加に伴い実行時間が急増している．しかし，手法 2 は孤立頂点の除

表 1　マッチングの所要時間
辺密度=0.1，属性の種類=5

N_1	N_2	手法 1[sec]	手法 2[sec]	頂点数	手法 2/手法 1
10	10	0.00091	0.00049	24	0.540
	20	0.0060	0.0020	46	0.338
	30	0.019	0.0046	62	0.241
	40	0.046	0.0085	84	0.184
	50	0.10	0.012	104	0.122
20	20	0.051	0.0090	94	0.176
	30	0.19	0.035	123	0.183
	40	2.38	0.18	169	0.076
	50	1.66	0.16	208	0.098
30	30	1.99	0.22	186	0.111
	40	7.42	0.68	243	0.091
	50	18.1	9.73	301	0.538
40	40	75.6	17.2	326	0.228
	50	1390	47.4	405	0.034
50	50	>10000	3026	510	-

表 2　マッチングの所要時間
辺密度=0.1，属性の種類=10

N_1	N_2	手法 1[sec]	手法 2[sec]	頂点数	手法 2/手法 1
10	10	0.0010	0.00049	18	0.485
	20	0.0065	0.0020	27	0.309
	30	0.021	0.0046	35	0.216
	40	0.052	0.0080	44	0.154
	50	0.11	0.013	55	0.114
20	20	0.055	0.0078	54	0.141
	30	0.2102	0.01837	69	0.087
	40	0.54	0.033	90	0.060
	50	1.10	0.053	109	0.048
30	30	0.70	0.042	108	0.061
	40	1.91	0.10	136	0.054
	50	4.39	0.16	163	0.038
40	40	5.83	0.26	180	0.044
	50	9.89	0.63	217	0.064
50	50	90.2	10.2	268	0.113

表 3　マッチングの所要時間
辺密度=0.5，属性の種類=5

N_1	N_2	手法 1[sec]	手法 2[sec]	頂点数	手法 2/手法 1
10	10	0.00081	0.00048	24	0.589
	20	0.0057	0.0020	46	0.353
	30	0.018	0.0045	62	0.247
	40	0.046	0.0082	84	0.177
	50	0.097	0.013	104	0.135
20	20	0.043	0.0077	94	0.178
	30	0.18	0.019	123	0.107
	40	0.44	0.041	169	0.093
	50	0.87	0.076	208	0.088
30	30	0.63	0.054	186	0.086
	40	1.51	0.12	243	0.081
	50	3.08	0.27	301	0.086
40	40	4.13	0.44	326	0.106
	50	9.73	1.31	405	0.134
50	50	27.1	4.35	510	0.161

表 4　マッチングの所要時間
辺密度=0.5，属性の種類=10

N_1	N_2	手法 1[sec]	手法 2[sec]	頂点数	手法 2/手法 1
10	10	0.00081	0.00048	18	0.596
	20	0.0064	0.0021	27	0.330
	30	0.020	0.0045	35	0.221
	40	0.049	0.0079	44	0.161
	50	0.11	0.012	55	0.112
20	20	0.050	0.0077	54	0.153
	30	0.20	0.018	69	0.089
	40	0.50	0.032	90	0.064
	50	0.95	0.050	109	0.053
30	30	0.69	0.040	108	0.058
	40	1.59	0.074	136	0.047
	50	3.06	0.12	163	0.039
40	40	3.6415	0.133	180	0.037
	50	6.79	0.23	217	0.034
50	50	13.4	0.40	268	0.030

表 5　マッチングの所要時間
辺密度=0.9，属性の種類=5

N_1	N_2	手法 1[sec]	手法 2[sec]	頂点数	手法 2/手法 1
10	10	0.00091	0.00048	24	0.530
	20	0.0073	0.0021	46	0.281
	30	0.023	0.0045	62	0.197
	40	0.055	0.0081	84	0.147
	50	0.11	0.013	104	0.119
20	20	0.044	0.0081	94	0.184
	30	0.22	0.028	123	0.128
	40	0.56	0.039	169	0.070
	50	1.01	0.065	208	0.064
30	30	1.07	0.16	186	0.146
	40	5.95	0.32	243	0.054
	50	9.86	9.43	301	0.956
40	40	629	60.5	326	0.096
	50	916	363	405	0.396
50	50	>10000	4358	510	-

表 6　マッチングの所要時間
辺密度=0.9，属性の種類=10

N_1	N_2	手法 1[sec]	手法 2[sec]	頂点数	手法 2/手法 1
10	10	0.00091	0.00048	18	0.529
	20	0.0064	0.0021	27	0.323
	30	0.021	0.0045	35	0.213
	40	0.053	0.0047	44	0.088
	50	0.11	0.012	55	0.113
20	20	0.050	0.0078	54	0.155
	30	0.21	0.018	69	0.086
	40	0.50	0.033	90	0.065
	50	0.99	0.051	109	0.051
30	30	0.70	0.041	108	0.058
	40	1.70	0.087	136	0.051
	50	3.31	0.14	163	0.043
40	40	9.49	1.08	180	0.114
	50	13.6	2.25	217	0.165
50	50	74.1	12.0	268	0.162

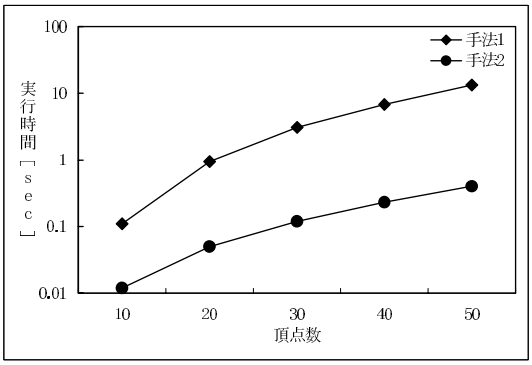


図 10　頂点数と実行時間の関係
辺密度=0.5，属性の種類=10

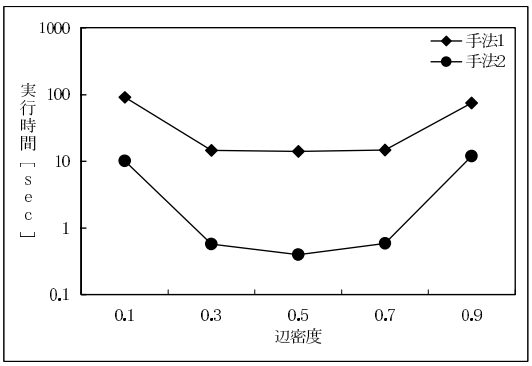


図 11　辺密度と実行時間の関係
頂点数=50，属性の種類=10

去によりグラフの規模を抑制していることで，頂点数増加への耐性が多少強まっていることが確認できる．続いて，入力グラフの辺密度と実行時間の関係を示しているのが図 11 である．辺密度の低い部分，あるいは高い部分では辺の存在が一致する組み合わせが多くなるため，積グラフの辺密度が高くなる．逆に，辺密度が中位の場合は積グラフの辺密度が比較的低くなるため，手法 2 の孤立頂点除去の効果が大きくなり，実行時間が短縮されている．最後に属性の種類と実行時間の関係を示しているのが図 12 であるが，これは属性の種類が増えることで頂点の組み合わせ数が減ることは自明なので，実行時間は徐々に減少していく．特に手法 2 はその効果を得られている様子が読みとれる．

6. むすび

本研究では，グラフマッチングの効率化のために属性情報をどう利用するべきであるか，という検証を行った．特に，ここで取り扱った最大部分グラフ同型問題は，柔軟なマッチングを可能にする反面，極めて急激な計算量の増加を招くため，効率的に属性情報を用いることが必須である．したがって，この解法の主幹である最大クリーク抽出と積グラフの生成という手続き内で属性情報を用いる 2 手法を考案したところ，双方共に効率化に有効であるが，後者の方が圧倒的に優位であることが分かった．

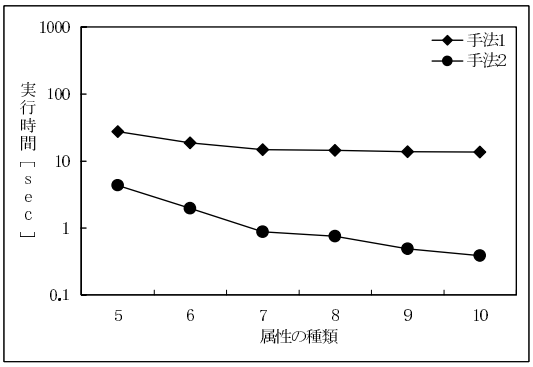


図 12　属性の種類と実行時間の関係
頂点数=50, 辺密度=0.5

つまり，予め頂点数を抑制しておくことが非常に重要であると言える．

提案手法を用いることで，最大クリーク抽出アルゴリズムの内部に立ち入ることなく効率化を実現できるため，グラフマッチングの目的・用途によって近似解アルゴリズムを使用することも容易である．ただし，この場合でも頂点数を抑えておくことは実行時間の短縮に貢献するものと予想される．

以上，グラフマッチングにおける属性情報の効率的利用法について述べてきたが，より大規模な問題の扱い方や適用分野の探求などを今後の課題とする．

謝　辞

本研究は科学研究費基礎研究 (C) 課題番号 17500061 の支援を受けて行った．

参 考 文 献

[1] J. R. Ullmann, “An Algorithm for Subgraph Isomorphism,” J. Assoc. for Computing Machinery, vol.23, pp.31-42, 1976.

[2] L. P. Cordella, P. Foggia, C. Sansone, M. Vento, “Performance Evaluation of the VF Graph Matching Algorithm,” 10th International Conf. Image Analysis and Processing, pp.1172-1177, 1999.

[3] L. P. Cordella, P. Foggia, C. Sansone, M. Vento, “A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol.26, No.10, pp.1367-1372, 2004.

[4] E. C. Sewell, “A branch and bound algorithm for the atability number of a sparse graph,” INFORMS J. Comput. 10, pp.438-447, 1998.

[5] T. Fahle, “Simple and fast: Improving a branch-and-bound algorithm for maximum clique,” European Symp. on Algorithms 2002, LNCS 2461, pp.485-498, 2002.

[6] P. R. J. Östergård, “A fast algorithm for the maximum clique problem,” Discrete Appl. Math. vol.120, pp.197-207, 2002.

[7] 森田昭広, 富田悦次, 亀田宗克, “最大クリーク抽出のより高速なアルゴリズム,” 科学技術レターズ（FIT2004）, pp.19-22, 2004.