冗長性を持たせたグラフ分解による部分グラフ検索手法の提案と評価

西村将太郎† 片山 薫† 石川 博†

† 東京都立大学大学院工学研究科 〒 192–0397 東京都八王子市 1–1
E-mail: †nishi-no-heya_2@msj.biglobe.ne.jp, ††{katayama,ishikawa}@eei.metro-u.ac.jp

あらまし 二つのグラフが与えられた時，一方のグラフが他方のグラフに含まれるかどうかを判定する問題を部分グラフ同型判定問題という．この問題は NP 完全であるので，大量のグラフを扱う際には膨大な計算コストが必要である．Messmer らはグラフ集合を，グラフの分解によって得られる特別なデータ構造に変換後，部分グラフ同型判定を行うことによって，この問題を効率的に解くアルゴリズムを提案している．しかし，グラフの分解によって非連結グラフが生成されると，計算コストが非常に増加するという問題がある．我々はグラフの分解において，分解するグラフに含まれる共有可能な部分グラフが発見された際，その部分グラフを引いて出来たグラフが非連結グラフになるときは共有しないとし，必ず連結グラフが生成されるようにすることでこの問題を解決したが，共有できるグラフが限られた．そこで，非連結グラフが生成されたとき，非連結部分を連結するような頂点，枝を非連結グラフに加えることで，共有範囲を減らすことなく連結グラフを生成するよう改良した．更に，代表的な部分グラフ同型判定アルゴリズム VF2 と比較し，提案手法の有効性を示す．

キーワード 情報検索，科学 DB，知識発見

Detecting Subgraphs Efficiently by Redundant Graph Decomposition

Shotaro NISHIMURA†, Kaoru KATAYAMA†, and Hiroshi ISHIKAWA†

† Graduate School of Engineering, Tokyo Metropolitan University, 1-1 Minami-Osawa,Hachioji-shi Tokyo, 192-0397
E-mail: †nishi-no-heya_2@msj.biglobe.ne.jp, ††{katayama,ishikawa}@eei.metro-u.ac.jp

Abstract The subgraph isomorphism problem is the problem of whether one graph is contained in another graph, given two graphs. An enormous calculation cost is necessary for this problem when large quantities of graphs are handled because it is NP complete. Messmer et al. propose the algorithm to solve this problem efficiently based on graph decomposition. However, when unconnected graph is generated by decomposition of graph, the calculation cost increases very much. We propose that subgraph does not share, when the sharable subgraph contained in the graph to decompose is discovered, and the graph was generated by subtracting subgraph is unconnected graph. This problem is solved when connected graph certainly being generated. However, sharable graph decreased by this method. When unconnected graph was generated, we proposed that connected graph generated without size of shared subgraph become small, by adding the vertice and edge to unconnected graph in order to connects an unconnected portion. Furthermore, we show effectiveness of proposed technique by compared with typical subgraph isomorphism algorithm VF2.

Key words Information retrieval, Science DB, Knowledge discovery

1. はじめに

二つのグラフが与えられた時，一方のグラフがもう一方のグラフに含まれるかどうかを判定する問題を部分グラフ同型判定問題という．部分グラフ同型判定は，パターン認識やコンピュータビジョンなどの情報学の分野の他，化学や生物学などの様々な分野において，広く応用されグラフマッチングと呼ばれる．しかし，この問題は NP 完全であるので，大量なグラフを扱うさいには膨大な計算コストが必要である．

これまでの研究として，検索範囲を著しく減少させる手続きである，バックトラックに基づくアルゴリズムが Ullman[2] によって提案されている．グラフ同型性と，部分グラフ同型判定の両方のためのアルゴリズムであり，今日まだ正確なグラフマッチングとして一般に使われるものの 1 つである．さらに，

グラフをカノニカルなフォームに変形してから同型を判定する Nauty[3] algorithm がある．VF[4] は，グラフ同型，およびサブグラフ同型判定手法で，複雑で大きなグラフを扱うことができる．

Messmer[1] らはグラフ集合を，グラフの分解によって得られる特別なデータ構造に変換後，部分グラフ同型判定を行うことによって，この問題を効率的に解くアルゴリズムを提案している．グラフ分解を利用したアプローチは，例えば画像データデータベース中の各画像が，必要な複数の特徴を持っているかを調べるといったような，多くのデータに対して同じ問い合わせを繰り返す必要のある処理に有効である．しかし，Messmer らの手法はグラフの分解によって非連結グラフが生成されると，計算コストが非常に増加するという問題がある．

西村ら [5] は，まずグラフの分解において，分解するグラフに含まれる共有可能な部分グラフが発見された際，その部分グラフを引いて出来たグラフが非連結グラフになるときは共有しないで分解するとし，必ず連結グラフが生成されるようにすることでこの問題を解決した．

しかし，その方法では共有できるグラフが頂点数の少ないグラフに限られる．Messmer らの手法の利点は，異なるグラフによって共有される部分グラフが入力グラフとマッチングされるのは一度だけということである．つまり共有するグラフが大きいほど効率が良くなる．共有できるグラフが小さいグラフに限られることで，利点である共有範囲が減少する．

そこで我々は，非連結グラフが生成されたとき，非連結部分を連結するような頂点，枝を非連結グラフに加えることで，共有範囲を減らすことなく連結グラフを生成するよう改良する．更に，代表的な部分グラフ同型判定アルゴリズム VF2 と比較し，提案手法の有効性を示す．

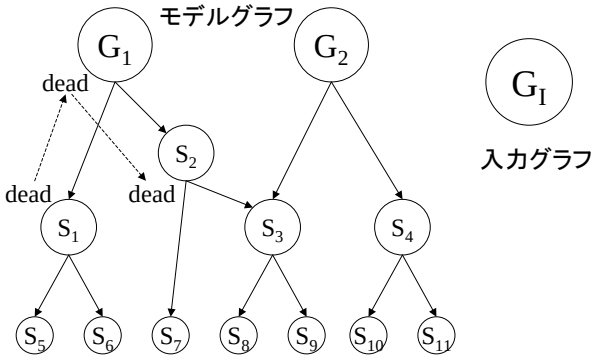

図 1.Messmer らの方法

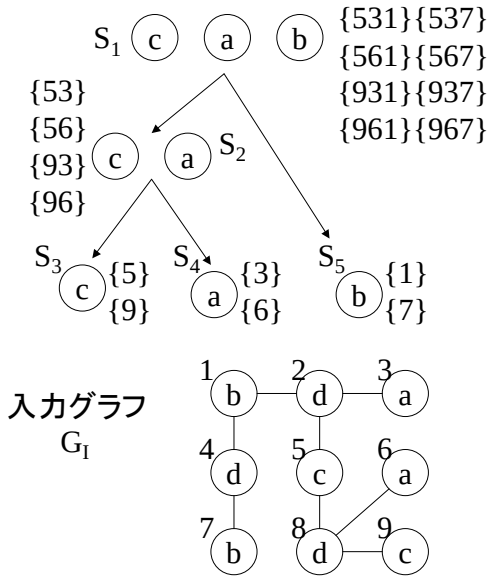

図 2. 枝がないモデルグラフに対する部分グラフ同型判定

2. 提案手法の概要

2.1 Messmer らの部分グラフ同型判定手法とその問題

グラフの集合 $G_1, ..., G_n$ が与えられた時，それらがグラフ G_I に含まれるかどうかを判定する問題を考える．本稿では，$G_1, ..., G_n$ をモデルグラフ，G_I を入力グラフと呼ぶことにする．単純な方法としては，例えば Ullman のアルゴリズムを利用して各モデルグラフに順次入力グラフをマッチする等が考えられるが，この問題は NP 完全であり，大量のグラフを扱うさいには膨大な計算コストが必要である．

Messmer らはこの問題を解くため，以下のようなアルゴリズムを考案した．入力グラフを個々に各モデルグラフとマッチングする代わりに，図 1 に示すようにまずモデルグラフ G_1，G_2 を部分グラフに分解する．あるグラフ G_i（又は S_i）を分解する際，それ以前の分解によって生成されたグラフの中に，グラフ G_i に含まれるグラフ（部分グラフ）があったなら，G_i をその部分グラフと G_i から部分グラフを引いてできたグラフに分解する．そのような部分グラフがなかったらランダムに分解する．図 1 では，G_1 がランダムに分解され S_1 と S_2 が生成される．更に S_1 は分解され S_5と S_6 が生成される．G_2 の分解では，G_2 の部分グラフ S_3 が発見され，G_2 は S_3 と G_2 から S_3 を引いたグラフ S_4 が生成される．それから，入力グラフ G_I と部分グラフをマッチングして，最終的に完全なモデルグラフとの部分グラフ同型を検出する．この方法による利点は，分解によって得られた異なるグラフによって共有される部分グラフ（例えば図 1 の S_3）が，入力グラフとマッチングされるのは一度だけであるということである．

この方法は 2 つのパートからなる．まずモデルグラフが分解されるプロセスで，得られた部分グラフは図 1 のような特別なデータ構造で表される．次のプロセスで，前プロセスで生成されたデータ構造で表されたモデルグラフと入力グラフをマッチングする．

Messmer らのアルゴリズムは，グラフの分割において切られる枝について考慮されていないので，たくさんの枝が切られ，連結グラフではない部分グラフが生成される可能性がある．このような部分グラフが生成されると，部分グラフ同型判定に要する計算コストが非常に増加する．

例を図 2 に示す．丸の中のアルファベットは頂点のラベルで，入力グラフ G_I の頂点の横の数字は頂点の ID である．分解されたモデルグラフの部分グラフ $S_1, \cdots, S_5$ の横の { }で囲まれた数字は，割り当てられた入力グラフ G_I の頂点の ID である．上の図は，枝を持たない 3 つの頂点だけからなるグラフ S1 の分解例である．まず S_3 は頂点のラベルが c なので，G_I においてラベルが c である ID が 5,9 の頂点が割り当てられる．同様に S_4 は頂点のラベルが a なので G_I の ID3,6，S_5 は頂点のラベルが b なので G_I の ID1,7 の頂点が割り当てられる．S_2 は枝がないので，G_I において割り当てられた頂点の間にも枝がなければ必ず部分グラフである．G_I の ID5,9 の頂点と ID3,6 の頂点の間には枝がない．よって S_2 は 4 通りの割り当てができる．さらに，G_I の ID5,3,6,9 の頂点と ID1,7 の頂点のそれぞれの間には枝がなく，S_1 にも枝がないので 8 通りの割り当てが生成される．頂点の数，モデルグラフの個数が更に多いと，非常に多くの割り当てが生成されることがある．

2.2 西村らによる改良

西村ら [5] は，グラフの分解において必ず連結グラフが生成されるようにすることを考えた．グラフ G を分割する際，Messmer らの方法では (1) 今までの分解で生成されたグラフの中の最大の部分グラフ S_{max} と，G − S_{max}（G と S_{max} の差異；定義 3 参照）に分割，(2) ランダムに分割，の二つの分け方がある．この両方の場合において，分解された部分グラフが必ず連結グラフになるようにする．(1) では，G − S_{max} が，連結グラフになるような S_{max} をとり，(2) においては，Kernighan/Lin アルゴリズムに，分解されたグラフが必ず連結グラフになるという条件を加えた分解方法で，切られる枝が少なく，かつ分解されたグラフが連結グラフになるようにする．

Messmer らの部分グラフ同型判定のプロセスは，分解のプロセスにおける最大の部分グラフを探すアルゴリズムとほぼ同じである．図 1 においてもし S_1 が G_I の部分グラフではない（部分グラフでないものを"dead"と呼ぶ），つまり"dead"であったなら，当然 S_1 を含んでいる G_1 も"dead"であり計算を省略することができる．

西村らは S_2 が G_I の部分グラフであったとしても，G_1 がすでに"dead"であるので，S_2 は計算する必要がないことを利用し計算量を減らす．

2.3 提 案 手 法

西村らの手法ではモデルグラフの分解において，今までの分解で生成されたグラフの中の最大の部分グラフ S_{max} と，G − S_{max} に分解する時，必ず G − S_{max} が連結グラフになるような S_{max} をとるとすると，S_{max} となり得るグラフが限られる．特に頂点数の多いグラフを S_{max} とすると G − S_{max} が非連結グラフになりやすいので，S_{max} が頂点数の少ないグラフになる．あるモデルグラフ G_1 と G_2 を分解するときの例を図 3 に示す．G_2 を分解する時，G_2 の最大の部分グラフは S_2 で，G_2 から S_2 を引いて出来たグラフ S_5 が非連結グラフになったとする．西村らの手法では，S_5 が非連結グラフになったら，分解せずに次に大きい部分グラフを探す．そして，次に大きい部分グラフ S_3 が発見され，G_2 から S_3 を引いたグラフ S_4 が連結グラフであったら S_3 を共有して分解する．このように Messmer らの方法では破線（赤線）で囲まれた範囲が共有されていたが，西村らの手法では非連結グラフが生成されるのを回避するために実線（黒線）で囲まれたところへ共有範囲が小さくなる．つまり，異なるモデルグラフよって共有されてる部分グラフが小さいグラフに限られることで，共有されている部分グラフが入力グラフとマッチングされるのは一度だけという Messmer らの利点である共有範囲が減少することになる．

そこで我々は，G_2 を分解する時，最大の部分グラフ S_2 が発見され，，G_2 から S_2 を引いて出来たグラフ S_5 が非連結グラフになったなら S_5 を連結グラフにすることを考える．あるグラフ G を分解する時，今までの分解で生成されたグラフの中の最大の部分グラフ S_{max} と G − S_{max} に分割し，G − S_{max} が非連結グラフであったら，G − S_{max} が連結グラフになるような G 上の頂点，枝を G − S_{max} に加えることで G − S_{max} を連結グラフにするとする．これにより，Messmer らの欠点である非連結グラフが生成される問題を解決しつつ，利点である異なるグラフよって共有されている部分グラフの範囲を減少させることなく分解することができる．

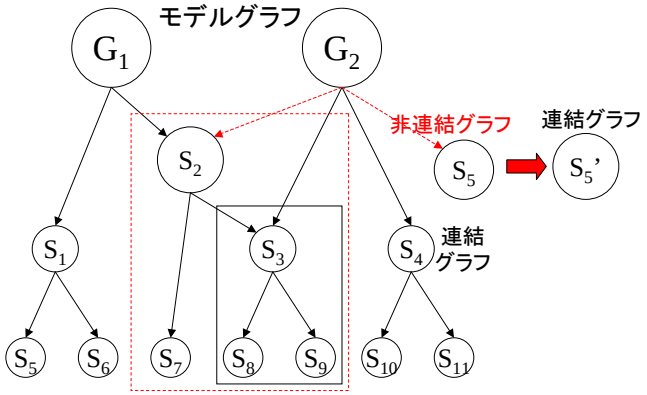

図 3. 西村らの手法の問題点と提案手法

3. 定　　義

［定義 1］ グラフ G は 4 つの要素 $G = (V, E, \mu, \nu)$ からなる

- V は頂点の集合
- E は枝の集合
- μ は V からラベルの集合への関数
- ν は E からラベルの集合への関数

［定義 2］ グラフ $G = (V, E, \mu, \nu)$ が与えられると部分グラフ $S = (V_s, E_s, \mu_s, \nu_s)$ は以下のように定義される

（1） $V_s \subseteq V$

（2） $E = E \cap (V_s \times V_s)$

（3） すべての $v \in V_s$ において，$\mu_s(v) = \mu(v)$

（4） すべての $v \in V_s$ において，$\nu_s(v) = \nu(v)$

［定義 3］ グラフ $G = (V, E, \mu, \nu)$，G の部分グラフ $S = (V_s, E_s, \mu_s, \nu_s)$ とすると，G と S の差異は，頂点 $V - V_s$ の G の部分グラフであり，$G - S$ と定義する．

［定義 4］ グラフ G と G の部分グラフ S が存在するとき，頂点の割り当て f は以下の条件を満たす．

（1） すべての $v \in V_s$ において $\mu(v) = \mu'(f(v))$ であり

（2） すべての $e_s = (v_1, v_2) \in E_s$ に対する $\nu(e_s) = \nu(e)$ のような $e = (f(v_1), f(v_2)) \in E$ が存在し，かつすべての $e = (f(v_1), f(v_2)) \in E$ に対する $\nu(e) = \nu(e_s)$ のような $e_s = (v_1, v_2) \in E_s$ が存在する

［定義 5］ $D_i = (G_i, S_1, S_2, F_i)$ はグラフの分割によって得られたデータ集合であり分解データと呼ぶ．D は D_i の集合である．

（1） S_1，S_2 は G_i の分解によって得られたグラフであり，$S_2 = G - S_1$

（2） F_i は f の集合である

D_i の S_1 を G_i とする分解データを D_s1，D_i の S_2 を G_i とする分解データを D_s2 とする．D_i の G_i を S_1 または S_2 としている分解データを $D_{u1}, \cdots, D_{uj}$ とする（j は D_i が指された順）．例えば，図 1 において S_3 の位置にあるデータを D_i とすると，D_{s1} は S_8，D_{s2} は S_9 であり，本論文では D_i が D_{s1} と D_{s2} を指しているという．さらに，D_{u1} は S_2，D_{u2} は G_2 であり，D_i は D_{u1}と D_{u2} に指されているという．

Decomposition(B)
1. $B = G_1, \cdots, G_n$，$D = \phi$ とする
2. $for i = 1 to n Decompose(G_i)$

図 4. 分解アルゴリズム

Decompose(G)
1. S_{max}=φ
2. D におけるすべての D_i の F をφにする
3. G がただ一つの頂点なら，$G - S_{max}$=φでステップ 8
4. S_{max}=Max_subgraph(G,D)
5. $G - S_{max}$ が非連結ならば，combination(G,S_{max})
6. S_{max}=φなら，
 6.1 G を Kernighan/Lin Algorithm で S_1, S_2 に分解し，
  $S_{max} = S_1$ とする
 6.2 Decompose(S_{max})
7. Decompose($G - S_{max}$)
8. D に D_i=(G,S_{max},$G - S_{max}$,F) を加える (F=φ)

図 5. モデルグラフ分解

4. モデルグラフの分解

4.1 分解のアルゴリズム

図 4 に分解のアルゴリズム Decomposition を示す．入力はモデルグラフの集合 B で図 5 のモデルグラフ分解の手続き Decompose が，それぞれのモデルグラフ $G_1, \cdots, G_n$ に対して順次呼び出される．Decompose では，ステップ 4 で図 6 の S_{max} を探すアルゴリズム Max_subgraph を呼ぶ．Max_subgraph が呼ばれる時までに生成された分解データ D の中から S_{max} を探す．ここで S_{max} は，Decompose において分解される G の部分グラフの中で最大のグラフである．S_{max} が発見され，G から S_{max} を引いた G − S_{max} が非連結ならば，図 9 の Combination を呼ぶ．Combination は $G - S_{max}$ を連結になるような G 上の頂点，枝を加えたグラフにする，または S_{max}=φにする．ここで $G - S_{max}$ は単純に G から S_{max} を引いたグラフではなく，$G - S_{max}$ を連結グラフにするためにいくつかの頂点，枝を S_{max} と $G - S_{max}$ で共有したグラフになる．S_{max} がなかったなら，ステップ 6 の (a) において Kernighan/Lin アルゴリズムを使って G を分解する．Kernighan/Lin アルゴリズムは，G を頂点の数が同数（奇数個ある時は，k 個と k+1 個）に分解するとき，切られる枝の数を最小にするような分解を見つけるアルゴリズムである．しかし，Kernighan/Lin アルゴリズムでは，分解されたグラフが連結グラフになっているとは限らない．S_1 と S_2 に分解したとき，もし S_1 が非連結グラフになったら，連結成分ごとに分けて，その中で小さい成分から順に S_2 へ持っていくことで連結グラフにする．逆も同様にして，必ず連結グラフになるようにする．

さらに，ステップ 6 の (b)，ステップ 7 において分解されたグラフを再帰的に分解する．最後に，ステップ 8 で分解によって得られた分解データ D_i を D に入れる．この時点でデータを格納することによって，S_{max} を探す際に無駄なデータを参照せずにデータが格納された順に S_{max} を探す，または，部分グラフ同型判定を行うことができる．

Max_subgraph(G,D)
1. D のすべての D_i を"unsolved"にする
2. S_{max}=φ
3. for(すべての D_i について；i=0,1,2…)
 6.1 D_i が指す D_{s1}, D_{s2} が"alive"なら
  6.1.1 G_i が頂点 1 つからなるなら
   6.1.1.1 Fs=vertex_test(G,D_i)
   6.1.1.2 Fs=φなら D_i は"dead"，Fs ≠φなら D_i は"alive"
  6.1.2 G_i が頂点 2 つ以上からなるなら
   6.1.2.1 Fs=Combine(D_i)
   6.1.2.2 Fs=φなら D_i は"dead"，Fs ≠φなら D_i は"alive"
 6.2 D_i が指す D_{s1} または D_{s2} が"dead"なら，
  D_i は"dead"
 6.3 D_i が"alive" かつ G_i が S_{max} より大きいなら
  $S_{max} = G_i$
4. S_{max} を返す

図 6.S_{max} 探索

図 6 に最大の部分グラフを探す手続き Max_subgraph を示す．入力は Decompose で分解されるグラフ G と分解データ D である．ステップ 3 の D_i(i=0,1,2,…) は分解データ D に

格納された順である．G_i が頂点 1 つからなるなら，図 7 の vertex_test を呼ぶ．G_i が頂点 2 つ以上から成るなら，図 8 の Combine を呼ぶ．S_{max} は分解データ D 中の G の部分グラフで最大のグラフである．

Vertex test(G,D_i)
1. F=φ，l=μ (v_i) は G_i の頂点のラベル
2. すべての G の頂点 v において
l=μ (v) なら f(v_i)=v として，F に f を加える
3. F を返す

図 7. 頂点判定

Combine(G,D_i)
1. $S_1=(V_1,E_1,\mu_1,\nu_1)$, $S_2=(V_2,E_2,\mu_2,\nu_2)$,
F=Φとする
2. $f_1\in F_1$, $f_2\in F_2$ のすべてのペア (f_1,f_2) において
2.1 $f_1(V_1)\cap f_2(V_2)=\phi$
2.2 すべてのペア v_1, v_2, $v_1\in V_1$, $v_2\in V_2$ において，
$\nu(e_i)=\nu(e)$ かつ G_i の枝 $e_i=(v_1,v_2)$ に対する
G の枝 $e=(f_1(v_1),f_2(v_2))$ が存在する
2.3 すべてのペア $f_1(V_1)$, $f_2(v_2)$, $f_1(v_1)\in f_1(V_1)$,
$f_2(v_2)\in f_2(V_2)$ において，G の枝 e=$(f_1(v_1),f_2(v_2))$
に対する G_i の枝 $e_i=(v_1,v_2)$ が存在する
2.4 2.1，2.2，2.3 が真ならば f は以下のように定義する
$f(v)=f_1(v_1)\cup f_2(v_2)\quad v_1\in V_1$, $v_2\in V_2$
F に f を加える
3. F を返す

図 8. 結合処理

Combination(G,S_{max})
1. G − S_{max} を連結成分ごとに分ける
G− $S_{max}=(g_a,g_b,g_c,...)$
2. 成分が一つなら終了
3. 成分 g_a と g_b の S_{max} と枝で結ばれていた頂点
$V_a=(v_{a1},v_{a2},...,v_{am}),V_b=(v_{b1},v_{b2},...,v_{bn})$ を求める
4. for i=1 to m
for j=1 to n
v_{ai} と v_{bj} を繋ぐG上の最短経路 R_k を求める
5. $R_1,R_2,...$ の中で最も短い経路上で，
成分 g_a，g_b に含まれない頂点，枝を G − S_{max} に加える
6. もし G=G − S_{max} ならば，$S_{max}=\phi$ で終了
7. ステップ 1 に戻る

図 9. 連結グラフ生成

図 7 に頂点の判定 Vertex_test を示す．入力は分解されるグラフ G と頂点一つからなるグラフの分解データ D_i である．G_i と G の頂点ラベルを比較し，同じなら f(v_i) に v を入れる（v_i は G_i の頂点，v は G の頂点）．

図 8 に結合の手続き Combine を示す．入力は分解されるグラフ G と分解データ D_i である．S_1，S_2 は D_i の G_i を分解して生成されたグラフである．f_1 と f_2 に同じ値が含まれず，かつ G_i における S_1 と S_2 の間の枝が G に存在し枝のラベルも同じ，かつ G における頂点 $f_1(v_1)$ と $f_2(v_2)$ の間の枝が G_i に存在し枝のラベルも同じである，ならば F に f を加える

図 9 に連結グラフ生成の手続き Combination を示す．入力は分解されるグラフ G と S_{max} である．まず，G-S_{max} を連結成分ごとに分ける．g_a，g_b，g_c,... はそれぞれ連結グラフであり，g_a，g_b，g_c,... それぞれのグラフの間に枝はない．連結成分 g_a と g_b（適当な二つの成分）の S_{max} と枝で結ばれていた頂点 $V_a=(v_{a1},v_{a2},...,v_{am}),V_b=(v_{b1},v_{b2},...,v_{bn})$ を求める．グラフ G における V_a と V_b それぞれの間の最短経路を求め，その中で最も短い経路の成分 g_a，g_b に含まれない頂点，枝を G − S_{max} に加える．このとき $G-S_{max}$ は単純に G から S_{max} を引いたグラフではなく，$G-S_{max}$ を連結グラフにするためにいくつかの頂点，枝を S_{max} と $G-S_{max}$ で共有したグラフになる．もし頂点，枝を加えることで G − S_{max} が G と同じ頂点数になったら，S_{max}=φとして終了する．この操作を繰り返し連結成分が一つになったら終了する．

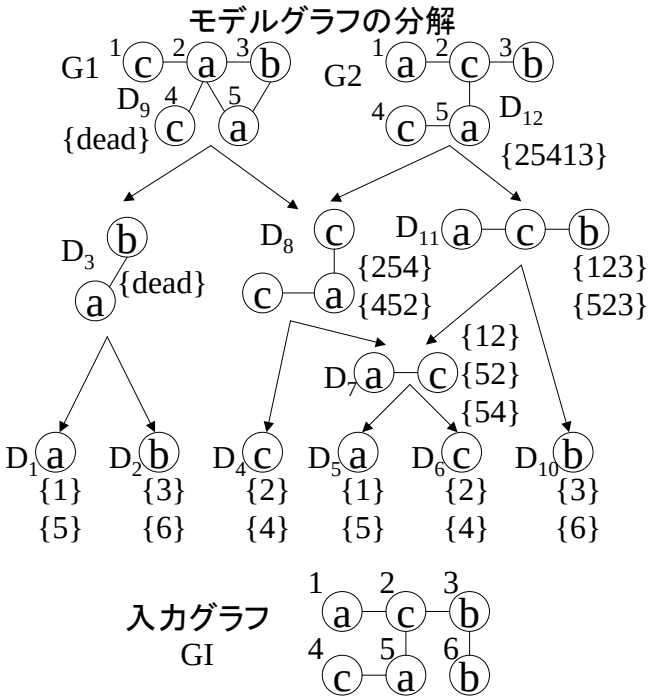

図 10. 分解と部分グラフ同型判定の例

4.2 分解の例

アルゴリズムを例証するために，例を図 10 に示す．ここでは枝のラベルは省略する．モデルグラフは G1 と G2 である．グラフの頂点の中のアルファベットは頂点のラベル，モデルグラフの G1，G2 と入力グラフ GI の頂点の横の数字は，説明のために用いる頂点の ID である．グラフの横の D_i は，G_i の分解によって生成されたデータ（分解データ）で，D_i=(G_i,S1,S2) で定義される．ここで，S1，S2 は G_i の分解によって生成され

たグラフである．D_i の i は，D_i が分解データ集合 D に入れられる順を示す．

まず，分解のプロセスを例証する．はじめは D=φなので G1 は Kernighan/Lin アルゴリズムで分解される．例えば，（実際にはすべての可能な分解について考えるが）G1 を (1) S1={1,2,3}，S2={4,5}，(2) S1={1,4}，S2={2,3,5}，(3) S1={3.5}，S2={1,2,4}の 3 つの場合に分解することについてだけを考える（{ }の中の数字は G1 の頂点の ID）．まず，Kernighan/Lin アルゴリズムによって，G1 を頂点の数が同数（奇数個ある時は，k 個と k+1 個）になるよう分解し，かつ切られる枝の個数を最小にするような場合を選択する．切られる枝の個数は，(1) は 3 個，(2) と (3) は 2 個なので (2)，(3) を選択する．次に，(2) は連結グラフではないので (3) のように分割し，図のようになる．

さらに G1 は分解されて，頂点 1 つになったとき初めて D_i が D に入れられる．ある分解データ D_i の G_i を頂点一つになるまで分解することによって生成されたすべての分解データが D に入れられてから，その分解データ Di 自身が D に入れられる（例えば，D_4，D_5，D_6，D_7 が D に入れられてから D_8 が入れられる）．

G2 の分解を考える．まず，G2 の S_{max} を探す（G1 を分解している時も，D ≠φの時点から S_{max} を探している）．探す順番は D_1，D_2，D_3,…である．しかし，頂点が 1 つから成るグラフは S_{max} とはみなさない．D_3 の G_3 は部分グラフではない．D_7 の G_7 は部分グラフである．D_8 の G_8 も部分グラフである．D_9 の G_9 は部分グラフではない．G2 を分解する時点で D に格納されている分解データ D_i は D_9 までなので，S_{max} は D_8 の G_8 となり，G2 は G_8 と G2 − G_8 に分割される．しかし，G2 − G_8 は ID1 の頂点と ID3 の頂点の二つの連結成分からなる非連結グラフになる．二つの連結成分の最短経路を求め，最短経路上の二つの成分に含まれない頂点，枝を G2 − G_8 に加える．ここでは，ID1 の頂点から ID2 の頂点を通り ID3 の頂点へ行くのが最短経路であるので，G2 − G_8 に ID1 と 2 の頂点の間の枝，ID2 の頂点，ID2 と ID3 の頂点の間の枝を加えたグラフを生成する．G_{11} は S_{max} である G_7 と G_{11}− G_7 に分解される．G2 はさらに分割され，図 10 のようなデータが生成される．

この例で G1，G2 を分解したとき，共有される頂点の個数は 3 個であった．もし G2 を分解する時，グラフに冗長性を持たせず，ただ GI − S_{max} が連結になるような S_{max} を取ったとすると，S_{max} は D_7 の G_7 となり，共有される頂点の個数は 2 個となる．

5. 部分グラフ同型判定

5.1 部分グラフ同型判定のアルゴリズム

図 11 に部分グラフ同型判定のアルゴリズムを示す．G_i が頂点 1 つから成るなら図 7 の vertex_test，G_i が頂点 2 つ以上からなるなら，図 8 の Combine を呼ぶ．D_i が"dead"になったら図 12 の Mark_up で D_i を指している D_{uj} をすべて"dead"にし，その D_{uj} を D_i として再帰的に Mark_up する．Mark_up はステップ 2，3 で図 13 の Mark_down を呼び，それぞれの D_{uj} を D_i としたときにその D_i が指している D_{s1}，が"unsolved"で，D_{s1} を指しているすべての D_{uj} が"dead"なら，その D_{s1} も"dead"とし再帰的に Mark_down する．D_{s2} も同様である．図 12 のステップ 1 で，D_{uj} が j=N,…,2,1 の順に見られることに注意する必要がある．

```
Subgraph(G,D)
1. D のすべての D_i を"unsolved"にする
2. for(すべての D_i について；i=0,1,2…)
 2.1 D_i が"unsolved"で
  2.1.1 G_i が頂点 1 つから成るなら，Fs=vertex_test(G,D_i)
  2.1.2 G_i が頂点 2 つ以上から成るなら，Fs=Combine(D_i)
 2.2 Fs=φなら D_i は"dead"，Fs ≠φなら D_i は"alive"
 2.3 D_i が"dead"になったなら，
  for(j=0,1,2,...) Mark_up(D_ui)
```

図 11. 部分グラフ同型判定

```
Mark_up(D_i)
1. for(すべての D_ui について；i=N,…,2,1)
 1.1 D_i を"dead"にする
 1.2 if(D_uj = φ) for(j=0,1,2,...) Mark_up(D_uj)
2. D_s1 が"unsolved"なら，Mark_down(D_s1)
3. D_s2 が"unsolved"なら，Mark_down(D_s2)
```

図 12, 上方マーク処理

```
Mark_down(D_i)
1. D_i のすべての D_ui が"dead"なら，D_i は“ dead ”
2. D_i が"dead"になったなら，
 2.1 D_s1≠φ，D_s1 が"unsolved"なら，Mark_down(D_s1)
 2.2 D_s2≠φ，D_s2 が"unsolved"なら，Mark_down(D_s2)
```

図 13. 下方マーク処理

5.2 部分グラフ同型判定の例

図 10 にモデルグラフ G1，G2 の分解と入力グラフ GI を示す．分解されたそれぞれのグラフの横の{ }内に，割り当てられた頂点が示されている．頂点が割り当てられない場合，又は計算不要の場合は"dead"が示されている．

D_i の i=0,1,2…の順に部分グラフ同型判定をする．D_1 の G_1 のラベルは a なので，入力グラフのラベルが a である頂点{1},{5}が割り当てられる．D_2 も同様にして頂点{3},{6}が割り当てられる．D_3 の G_3 は入力グラフの部分グラフではないので"dead"である．ここで G_3 を含んでいる G_9 も必ず部分グラフにはならないので"dead"である．

次に D_4，D_5，D_6 にそれぞれ{2},{4}と{1},{5}と{2},{4}を割り当てる．D_7 は D_5 と D_6 に割り当てられた頂点を結合して部分グラフになるものを選ぶ．G_7 の頂点の間には枝があり，GI の ID1 と ID2，ID5 と ID2，ID5 と ID4 の頂点間にも枝があるので，D_7 には{12}(D_5 の{1}と D_6 の{2}を結合しそのま

ま並べて示す)，{52}，{54}が割り当てられる．

D_8 は G_4 のラベル c の頂点と G_7 のラベル a の頂点の間に枝があり，GI の ID2 と 5，または ID4 と 5 の頂点の間に枝があるので{254}，{452}が割り当てられる．D_9 は"dead"であるので計算不要である．同様にして D_{10} から D_{11} まで計算する．D_{12} は D_8 と D_{11} を結合する．共有している頂点は GI の ID2 の頂点であるので G2 に{25413}が割り当てられ，G2 は入力グラフ GI の部分グラフであることがわかる．

6. 性能評価

提案手法を，西村らの手法，および，代表的な部分グラフ同型判定アルゴリズムである VF2 と比較し，提案手法の有効性を示す．提案手法と西村らの手法のモデルグラフの分解は前処理として，処理時間には加えずに比較する．VF2 は一対一のグラフに対して部分グラフ同型判定を行うアルゴリズムであるので，各モデルグラフが順次入力グラフに含まれるかどうかを判定される．

すべての実験において入力グラフの個数は 1000 個とする．つまり，いくらかのモデルグラフを分解してできたデータに対して，判定処理を 1000 回行ったということである．一つのグラフの平均頂点数とは，モデルグラフ，入力グラフを合わせたすべてのグラフの平均頂点数であり，中央値より頂点数が多いグラフを入力グラフ，少ないグラフをモデルグラフとした．モデルグラフ数が 1000 個における一つのグラフの平均頂点数を変化させたときの部分グラフ同型判定の処理時間，モデルグラフの平均頂点数が 30 個におけるモデルグラフ数を変化させたときの部分グラフ同型判定の処理時間をそれぞれについて比較する．VF2 との比較においては，提案手法に分解処理を含んだ場合についての比較も示す．

6.1 実験環境

Pentium(R)4 プロセッサ 3GHz，メモリ 1GB，OS として Windows XP を搭載した PC を使用し， Visual C++を使用して開発を行った．実験で使用するグラフデータは，蔵持らの開発したグラフ生成ソフトウェアを利用した．

6.2 実験結果

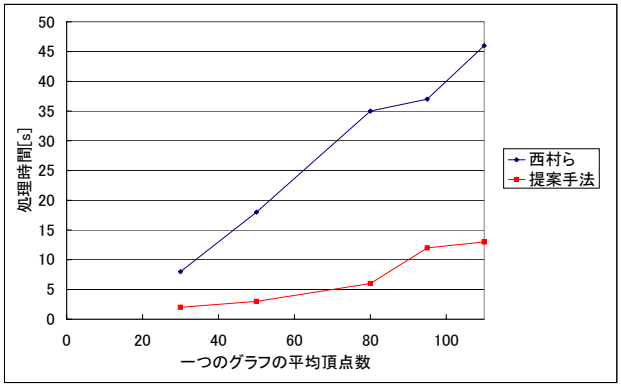

図 14. 平均頂点数と部分グラフ同型判定に要する時間 (西村らの手法との比較) モデルグラフ 1000 個

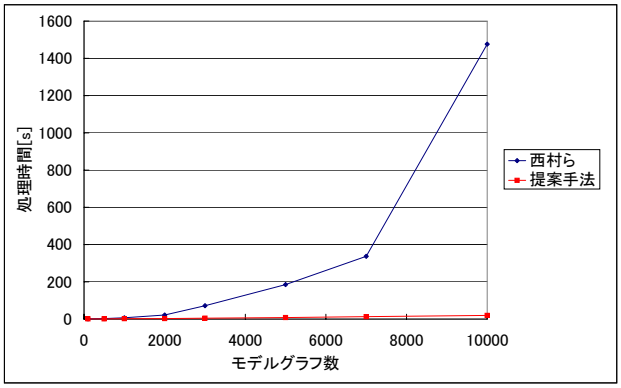

図 15. モデルグラフの個数と部分グラフ同型判定に要する時間 (西村らの手法との比較) 一つのグラフの平均頂点数 30 個

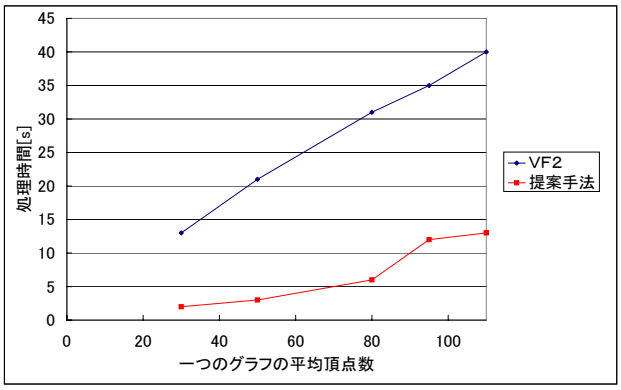

図 16. 平均頂点数と部分グラフ同型判定に要する時間 (VF2 との比較) モデルグラフ数 1000 個

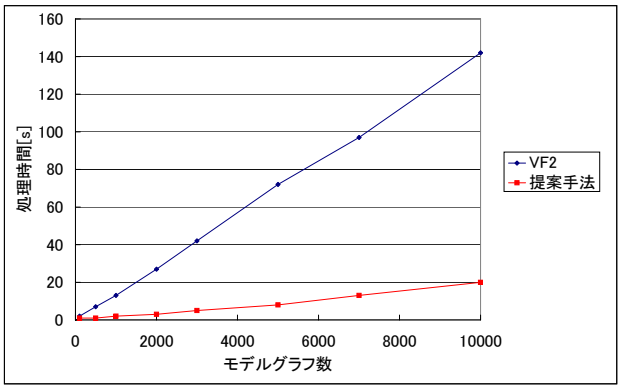

図 17. モデルグラフの個数と部分グラフ同型判定に要する時間 (VF2 との比較) 一つのグラフの平均頂点数 30 個

図 14 にモデルグラフ数が 1000 個における，グラフ集合に含まれる一つのグラフの平均頂点数と，それらを部分グラフ同型判定するのに要する時間との関係の西村らの手法と提案手法の比較を示している．西村らの手法は，2.3 章で述べた欠点から提案手法の 3 倍から 6 倍の処理時間がかかっている．

図 15 は，一つのグラフの平均頂点数が 30 個における，モデルグラフの個数と，部分グラフ同型判定するのに要した時間との関係の西村らの手法と提案手法の比較を示している．西村らの手法では，図 14 と同様に 2.3 章で述べた欠点からモデルグラフ数が増えるにしたがって処理時間が急激に増えていく．

図 14,15 の実験結果から，提案手法が西村らの手法から更に

効率化されたことが示される．

図 16 は，モデルグラフ数が 1000 個における，グラフ集合に含まれる一つのグラフの平均頂点数と，それらを部分グラフ同型判定するのに要する時間との関係の VF2 と提案手法の比較を示している．頂点数が 30 個と 50 個では VF2 が提案手法のおよそ 7 倍，頂点数 50 個ではおよそ 5 倍，頂点数 95 個と 110 個ではおよそ 3 倍の処理時間となった．頂点数が少ないグラフの方で提案手法がより効率的であることが分かる．

図 17 は，一つのグラフの平均頂点数が 30 個における，モデルグラフの個数と，部分グラフ同型判定するのに要した時間との関係の VF2 と提案手法の比較を示している．常に VF2 が提案手法のおよそ 7 倍の処理時間となった．

図 16，17 の実験結果から，多数のグラフを扱う際には，提案手法が VF2 より効率的であることが示される．

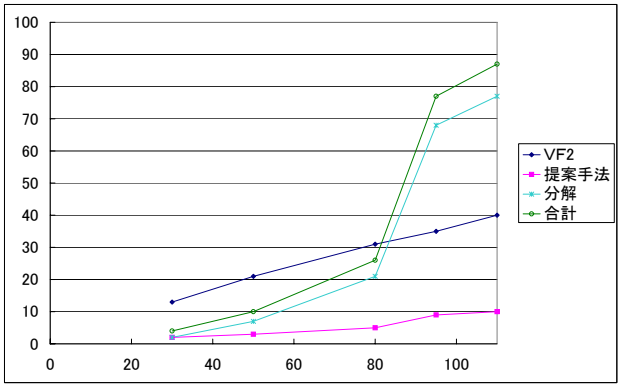

図 18. 分解処理時間込みの平均頂点数と部分グラフ同型判定に要する時間 (VF2 との比較) モデルグラフ数 1000 個

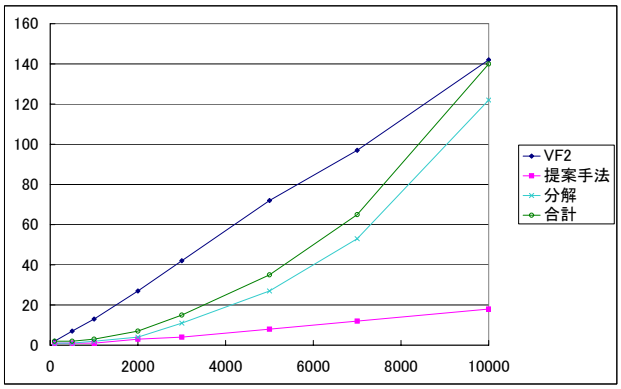

図 19. 分解処理時間込みのモデルグラフの個数と部分グラフ同型判定に要する時間 (VF2 との比較) 一つのグラフの平均頂点数 30 個

図 18 は，提案手法の処理時間にモデルグラフ分解処理の時間を加えたときの，モデルグラフ数が 1000 個における，グラフ集合に含まれる一つのグラフの平均頂点数と，それらを部分グラフ同型判定するのに要する時間との関係の VF2 と提案手法の比較を示している．モデルグラフの分解処理時間は，モデルグラフの頂点数が増えると判定処理時間の増加よりも急激に増加するので，あるところ以上からは提案手法が VF2 の処理時間を超える．ここでは平均頂点数 95 個で提案手法が VF2 を上回っている．

図 19 は，提案手法の処理時間にモデルグラフ分解処理の時間を加えたときの，一つのグラフの平均頂点数が 30 個における，モデルグラフの個数と，部分グラフ同型判定するのに要した時間との関係の VF2 と提案手法の比較を示している．図 18 と同様に，モデルグラフの分解処理時間は，モデルグラフ数が増えると増加するので，あるところ以上は提案手法が VF2 の処理時間を超える．ここではモデルグラフ数 10000 個でほぼ同じ処理時間なので，おそらく更に頂点数が増えると提案手法が VF2 の処理時間を超えるであろう．

図 18，19 より，モデルグラフの分解と部分グラフ同型判定を合わせた提案手法の処理時間と VF2 を比較すると，あるところ以上では VF2 の処理時間が速くなる．しかし，提案手法はあらかじめ多数のモデルグラフのデータが与えられ，事前にモデルグラフの分解が行えるような状況において，高速に部分グラフ同型判定を行うということを想定しているので，分解処理に時間が掛かっても良いとする。

7. まとめと今後の課題

Messmer らのアルゴリズムを元にした，より効率的な部分グラフ同型判定の手法を提案した．グラフ分解において非連結グラフが生成されたとき，非連結部分を連結するような頂点，枝をグラフに加えることで，共有範囲を減らすことなく連結グラフを生成し，Messmer らの手法における問題を解決した．代表的な部分グラフ同型判定アルゴリズムである VF2 と提案手法を比較し，グラフが多数あるときには提案手法の方が効率的であることを示した．今後の課題は，モデルグラフの分解処理の効率化と，応用分野の実データへ提案手法を適用することである．

謝　　辞

実験用グラフデータ生成ソフトウェアを提供頂いたミネソタ大学蔵持道広氏に感謝致します．本研究の一部は，(独) 日本学術振興会科学研究費補助金基盤研究 (B)(2)(課題番号:16300030) による．

文　　献

[1] Bruno T. Messmer and Horst Bunke, "Efficient Subgraph Isomorphism Detection: A Decomposition Approach" IEEE Transactions on Knowledge and Data Engineering, 12(2), 2000.

[2] J.R. Ullman, "An Algorithm for Subgraph Isomorphithms", Journal of Association for Computing Machinery, vol.23, pp.31-42, 1976.

[3] B.D. Mckay, "Practical Graph Isomorphism," Congressus Numerantium, vol. 30, pp. 45-87, 1981.

[4] Luigi.P.Cordella, Pasquale. Foggia, Carlo. Sansone, Mario. Vento, "A (sub)Graph Isomorphism Algorithm for Matching Large Graphs" , IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.26, no.10, pp.1367-1372, October 2004.

[5] 西村将太郎，片山薫，太田学，石川博：Messmer らのグラフ分解を利用した部分グラフ同型判定手法の改良，電子情報通信学会，第 16 回データ工学ワークショップ DEWS2005,2005.3