Messmerらによるグラフ分解を利用した部分グラフ同型判定手法の改良

西村　将太郎† 　片山　薫† 　太田　学† 　石川　博†

†東京都立大学大学院工学研究科 〒192-0397 東京都八王子市南大沢 1-1

E-mail: †nishi-no-heya_2@msj.biglobe.ne.jp, ‡{katayama,ohta,ishikawa}@hikendbs.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．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．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．我々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と，グラフ分解によって得られる情報を部分グラフ同型判定に利用することで, より効率的に部分グラフ同型判定を行う手法を提案する．

キーワード　部分グラフ同型，グラフ分割，グラフマッチング

Improvement of Messmer's Approach to Subgraph Isomorphism Detection

Shotaro　NISHIMURA† 　Kaoru KATAYAMA† 　Manabu OHTA† 　Hiroshi ISHIKAWA†

†Graduate School of Engineering, Tokyo Metropolitan University　1-1 Minami-Osawa, Hachioji-shi Tokyo, Japan 192-0397

E-mail: †nishi-no-heya_2@msj.biglobe.ne.jp, ‡{katayama,ohta,ishikawa}@hikendbs.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. We improve the Messmer's approach so that connected graphs are formed in the decomposition of each graph and information produced by graph decomposition is used in subgraph isomorphism detection.

Keyword　Subgraph isomorphism, Graph decomposition, Graph matching

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．

関連研究としては，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 Ullmann [2]によって提案されている．グラフ同形性と，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．さらに同型をチェックする前にカノニカルなフォームのグラフとマッチするためにグラフを変形する 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．この論文では，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する．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．

2. 提案手法の概要

2.1 Messmer らの部分グラフ同型判定手法

グラフの集合 $G_1,\ldots,G_n$ 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，$G_1,\ldots,G_n$ 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 の各モデルグラフに順次入力グラフをマッチするアルゴリズム等が考えられるが，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．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複数回現れる部分グラフ（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．更に，S_1 がもし G_I の部分グラフでなかったとすると，それを含む G_1 も部分グラフではないことがわかり，より小さいグラフを部分グラフ同型判定するだけで結果が得られる．

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 らの手法の欠点は，頂点の個数が多いモデルグラフを分解する際に，枝が切られ，すべての頂点が枝で結ばれていない（連結グラフではない）部分グラフが生成されることである．それによって計算コストが非常に増加してしまう．詳しくは次の 2.2 章で示している．

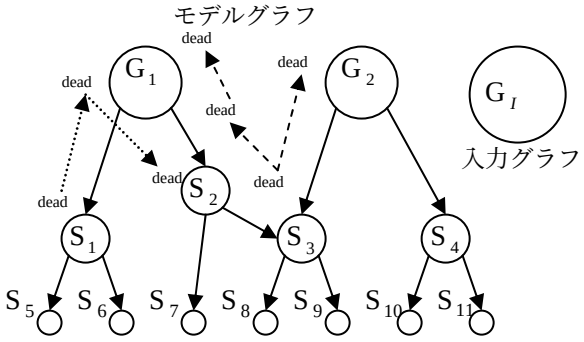

図 1　Messmer らの方法

2.2 グラフ分解方法の改良点

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たり，すべての頂点が枝で結ばれていない（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してしまう．特に疎グラフは枝が少ないので，このような現象が起きやすい．

例を図 2 に示す．丸の中の数字は頂点のラベルで，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,..., 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．モデルグラフの分解は，あるモデルグラフの分解の中から枝のない部分だけを抜き取ってきたものである．S_3 は頂点のラベルが 2 なので，G_I においてラベルが 2 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 0 なので G_I の ID3,6 の頂点が割り当てられ，S_5 は頂点のラベルが 1 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 ID3,9 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．実際は，より頂点の数，モデルグラフの個数が多いので，非常に多くの割り当てが生成されてしまうことがある．

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 2 参照）に分割，(2)ランダムに分割，の二つの分け方がある．我々の方法では，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．

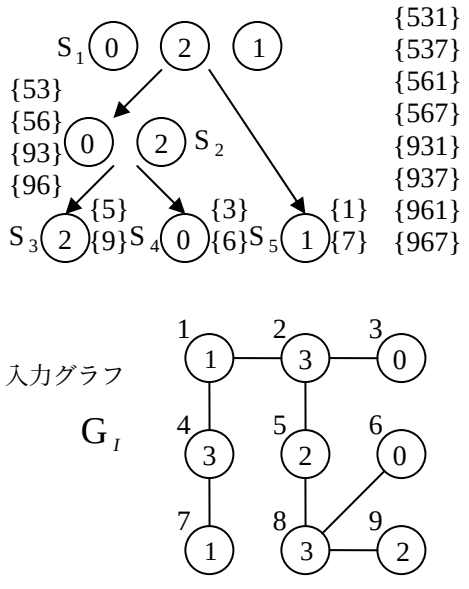

図 2　枝がない分解における部分グラフ同型判定

2.3 部分グラフ同型判定法の改良点

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．

Messmer らの手法はここまでだが，我々は S_2 がたとえ G_I の部分グラフであったとしても，G_1 がすでに“dead”であるということが解っているので，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．

部分グラフ同型判定において，入力グラフと分解されたモデルグラフを部分グラフ同型判定するとき，どの分解されたグラフから部分グラフ同型判定を行うかについて考える．更に，図 1 において，モデルグラフ G_1，G_2 両方の部分グラフである S_3 を最初に部分グラフ同型判定し，“dead”であったなら G_1 ，G_2 も“dead”であることがわかる．よって，S_8，S_9，S_3 のみを部分グラフ同型判定するだけで結果が得られる．このように，分解されたグラフを部分グラフ同型判定するときの順序を考慮することによって，効率化されることが期待できる．

4 章で示される分解のアルゴリズムにおいて，分解されたグラフのデータ（分解データ）が生成された順に部分グラフ同型判定を行う場合（詳細は 4 章）と，多数のモデルグラフの部分グラフであるデータから順に部分グラフ同型判定をする場合を考え，6 章の実験において比較する．

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 は頂点のラベルの集合
- ν は枝のラベルの集合

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. $E_s = E \cap (V_s \times V_s)$
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 の分解によって得られたグラフであり，$S_2=G-S_1$
2. F 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 $D_{u1},\ldots,D_{uj}$ 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={G1,…,Gn}，D=φとする
2. for i=1 to n      Decompose(Gi)
```

図 3.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終了
4. S_max=Max_subgraph(G,D)
5. S_max が G と同型ならば終了
6. S_max が見つからなかったなら，
    (a)G を Kernighan/Lin Algorithm で S_1,S_2 に
    分解し，S_max=S_1 とする
    (b)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 4.分解の手続き

4. モデルグラフの分解

4.1 分解のアルゴリズム

図 3 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 4 の分解の手続き Decompose は，それぞれのモデルグラフ G1,…,Gn に対して順次呼び出される．Decompose では，ステップ 4 で図 5 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 の部分グラフで，かつ $G-S_{max}$ が連

結グラフになるグラフの中で最大のグラフである．ステップ 4 で S_{max} が見つからなかったなら，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，それぞれ連結グラフになっているとは限らない．そこで，我々は 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”という条件を付け足した．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，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（部分グラフ同型判定に関しては，更に効率の良いと思われる判定処理の順序を提案する）．

```
Max_subgraph(G,D)
1. D のすべての Di を“unsolved”にする
2. Smax=φ
3. for(すべての Di について；i=0,1,2…)
    (a)Di が指す Ds1,Ds2 が“alive”なら
        (1) Gi が頂点 1 つからなるなら
            (ⅰ)Fs=vertex_test(G,Di)
            (ⅱ)Fs=φなら Di は“dead”,
            Fs≠φなら Di は“alive”
        (2) Gi が頂点 2 つ以上からなるなら
            (ⅰ)Fs=Combine(Di)
            (ⅱ)Fs=φなら Di は“dead”,
            Fs≠φなら Di は“alive”
    (b) Di が指す Ds1 または Ds2 が“dead”なら，Di は“dead”
    (c)Di が“alive”かつ Gi が Smax より大きく，かつ G-Gi が連結グラフであるならば Smax=Gi
4. Smax を返す
```

図 5. S_{max} を探すアルゴリズム

図 5 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 6 の vertex_test を呼ぶ．G_i が頂点 2 つ以上から成るなら，図 7 の Combine を呼ぶ．S_{max} は G－G_i が連結グラフになる部分グラフの中で最大のグラフである．

```
Vertex_test(G,Di)
1. i=φ，l=μ(vi)は Gi の頂点のラベル
2. すべての G の頂点 v において
    (a)l=μ(v)なら f(vi)=v として，F に f を加える
3. F を返す
```

図 6.頂点の手続き

図 6 に頂点の手続き Vertex_test を示す．入力は分解されるグラフ G と 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）．

```
Combine(G,Di)
1.S1=(V1,E1,μ1,ν1), S2=(V2,E2,μ2,ν2), F=Φとする
2.f1∈F1，f2∈F2 のすべてのペア(f1,f2)において
  (a) f1(V1)∩f2(V2)=Φ
  (b) すべてのペア v1，v2，v1∈V1，v2∈V2 において，ν(ei)=ν(e)かつ Gi の枝 ei=(v1,v2) に対する G の枝 e=(f1(v1),f2(v2))が存在する
  (c) すべてのペア f1(V1)，f2(v2)，f1(v1)∈f1(V1)，f2(v2)∈f2(V2)において，G の枝 e=(f1(v1),f2(v2))に対する Gi の枝 ei=(v1,v2)が存在する
  (d) (a)(b)(c)が真ならば f は以下のように定義する
      f(v)=f1(v1)∪f2(v2)
           v1∈V1，v2∈V2
      F に f を加える
3.F を返す
```

図 7.結合の手続き

図 7 に結合の手続き Combine を示す．入力は分解されるグラフ G と 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．

4.2 分解の例

我々のアルゴリズムを例証するために，例を図 11 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数字は頂点のラベル，モデルグラフの G1 と入力グラフ GI_1，GI_2 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 の i は，分解データ 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) S_1={1,2,3},S_2={4,5}, (2) S_1={1,4}，S_2={2,3,5}，(3) S_1={3.5}，S_2={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（S_1，S_2は分解によって生成されたグラフ，｛｝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 11 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 D_i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は部分グラフではない．D_8は部分グラフであるが，G－G_8が連結グラフではないので S_{max}ではない． D_7は部分グラフで，G－G_7が連結グラフなので S_{max}である．D_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_7のグラフ G_7となり，G2 は G_7と G2－G_7に分割される．D_{14}は分解データ D に S_{max}がないので，Kernighan/Lin アルゴリズムによって分解される．G2 はさらに分割され，図 11 のようなデータが生成される．

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 8 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 6 の vertex_test，G_iが頂点 2 つ以上からなるなら，図 7 の Combine を呼ぶ．D_iが“dead”になったら図 9 の Msrk_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 Msrk_up する．ステップ 2，3 で図 10 の Mark_down を呼び，それぞれの

```
subgraph(G,D)
5. D のすべての D_i を“unsolved”にする
6. for(すべての D_i について；i=0,1,2…)
   (ア) D_i が“unsolved”で
        ① G_i が頂点 1 つから成るなら，
           Fs=vertex_test(G,D_i)
        ② G_i が頂点 2 つ以上から成るなら，
           Fs=Combine(D_i)
   (イ)Fs=φなら D_i は“dead”，Fs≠φなら
        D_i は“alive”
   (ウ)D_i が“dead”になったなら，
        for(j=0,1,2)  Mark_up(D_uj)
```

図 8.部分グラフ同型判定の手続き

```
Mark_up(D_i)
1. for(すべての D_ui について；i=N,…,2,1)
   (a) D を“dead”にする
   (b) 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 9.上方マーク手続き

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

図 10.下方マーク手続き

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 9 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．

図 8 の subgraph のステップ 6 において，D_iを i=0,1,2…の順に部分グラフ同型判定，つまり D に分解データが格納された順に部分グラフ同型判定する方法と，多数のモデルグラフの部分グラフである分解データから順に判定する方法を考える．後者は，グラフの頂点数が少ない分解データから判定し，さらに，グラフの頂点数が同じ分解データの中でより多数のモデルグラフの部分グラフである分解データから順に部分グラフ同型判定する．しかし，頂点が 1 つから成るグラフが“dead”となることは少ないので，頂点が 2 つ以上から成るグラフから部分グラフ同型判定を始め，頂点が 1 つから成るグラフについては必要なときに行う．

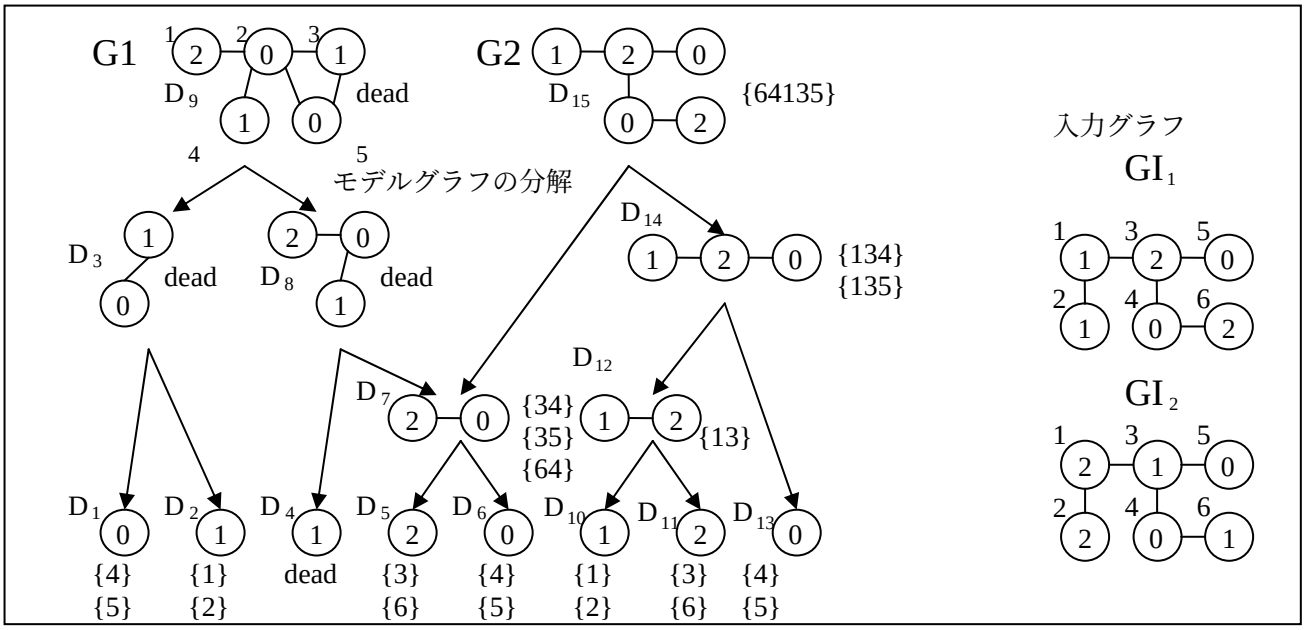

図 11　分解と部分グラフ同型判定の例

5.2 部分グラフ同型判定の例

図 11 にモデルグラフ G1, G2 の分解と入力グラフ GI_1 を示す．分解されたそれぞれのグラフの横に，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又は計算不要の場合は“ dead ”が示されている．

まず, D_i の i=0,1,2…の順に部分グラフ同型判定する方法について考える. D_1 のラベルは 0 なので, 入力グラフのラベルが 0 である頂点{4},{5}が割り当てられる．D_2 も同様にして頂点{1},{2}が割り当てられる. D_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 D_3 を指している D_9 も必ず部分グラフにはならないので“ dead ”である．更に D_9 に指されている D_8 も，指されているグラフ（ここでは D_9 のみ）のすべてが“ dead ”なので“ dead ”である．D_4 も同様に“ dead ” である．D_7 は，D_8 が“ dead ”であるが，D_{15} は“ unsolved ”なので“ unsolved ”のままである．
次に D_4 は“ dead ”なので D_5, D_6 にそれぞれ{3},{6}と{4},{5}が割り当てる．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．D_7 の頂点の間には枝があり，GI_1 の ID3 と 4，3 と 5，6 と 4 の間にも枝があるので，D_7 には{34}，{35}，{64}が割り当てられる．D_8，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_{14} まで計算する．D_{15} は D_7 と D_{14} を結合する. 同じ頂点 ID が割り当てられていない組み合わせは{64}と{135}だけである．D_{15} において D_7 と D_{14} の間には，4 と 3 が割り当てられた頂点の間にだけ枝がある. GI_1 においても, {64}と{135}の間にある枝は ID3 と 4 の頂点の間だけである．よって，G2 に{64135}が割り当てられ，G2 は入力グラフ GI_1 の部分グラフであることがわかる．

多数のモデルグラフの部分グラフである分解データから順に判定する方法について考える．入力グラフは GI_2 とする．頂点が二つから成るグラフの中で最も多数のモデルグラフの部分グラフであるのは D_7 であるので，まず D_5，D_6 について部分グラフ同型判定すると，それぞれ{12}，{45}が割り当てられる．次に D_7 を判定し，部分グラフではないので“ dead ”となる．D_7 は G1，G2 双方の部分グラフなので G1，G2 も“ dead ”となり，GI_2 の部分グラフではないことがわかる．

もし, 入力グラフを GI_1 としたら, D_i の i=0,1,2… の順で部分グラフ同型判定したときと同じ計算量になる．しかし，GI_2 を D_i の i=0,1,2…の順で部分グラフ同型判定すると，D_1，D_2，D_3，D_4，D_5，D_6，D_7 と判定するので，多数のモデルグラフの部分グラフである分解データから順に判定する方法のほうが，D_1，D_2，D_3，D_4 の分だけ少ない計算量でよい．

2 章でも触れたが，図 2 にエッジが切られ，なくなってしまった分解の一部を示す．このような箇所では入力グラフで部分グラフ同型判定を行うと，図のように多数の割り当てが生成されてしまう．よって我々の手法では必ず枝が残るように分解する．

6. 性能評価実験

Messmer らのアルゴリズムと我々のアルゴリズムを比較した．一つのグラフの平均頂点数，一つ

のグラフの平均頂点数が 50 個のときと，20 個のときのモデルグラフの個数，頂点ラベル数を変化させたときに，モデルグラフの分解，および部分グラフ同型判定をするのに要した処理時間を測定した．更に，部分グラフ同型判定において D_i の i=0,1,2…の順（分解データ D に格納された順）で部分グラフ同型判定する方法（手順 1）と，多数のモデルグラフの部分グラフである分解データから順に判定する方法（手順 2）を比較した．

	モデルグラフ			入力グラフ		
	D	V	T	D	V	T
図 12	100	10	—	500	10	—
図 13	—	10	20	—	10	30
図 14	—	10	50	—	10	60
図 15	300	—	70	300	—	80
表 2	300	10		5000	10	
表 3	300	10	100	5000	10	100

表 1　グラフのパラメーター

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．

生成されるモデルグラフ，入力グラフのパラメーターは表 1 に示される．D はグラフの個数，V は頂点のラベル数，T は一つのグラフの平均頂点数である．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．

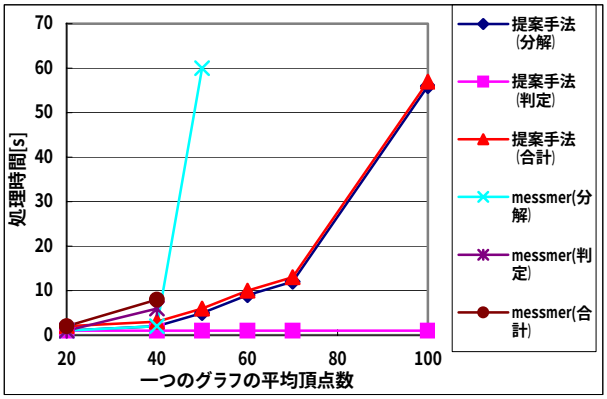

図 12　平均頂点数とグラフ分解，部分グラフ同型判定に要する時間

6.2 実験結果

図 12 は，グラフ集合に含まれる一つのグラフの平均頂点数と，それらをグラフ分解，部分グラフ同型判定をするのに要する時間との関係を示している．我々の方法は一つのグラフの平均頂点数が増えるにしたがって，処理時間もなだらかに増加するが，Messmer らの方法では一つのグラフの平均頂点数が 50 個から急激に増え，50 個以上では測定不可能（メモリオーバー）となった．Messmer らの手法では，一つのグラフの平均頂点数が 50 個以上では図 2 のような枝がない分解が生成され，急激に処理時間が増加したと考えられる．メモリオーバーとなったのは，割り当てられた頂点のデータ F が大きすぎたためである．

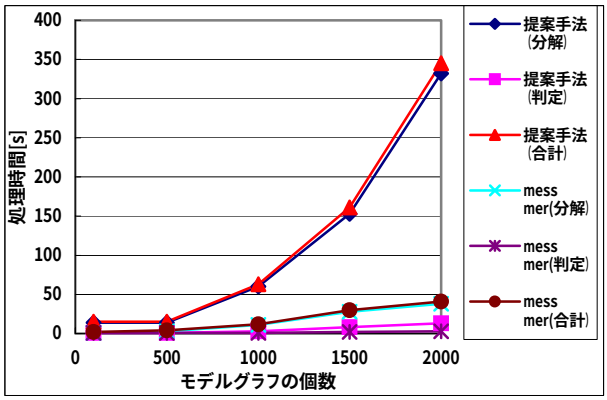

図 13　平均頂点数がモデルグラフ 20 個，入力グラフ 30 個におけるモデルグラフの個数と,モデルグラフの分解，部分グラフ同型判定に要する処理時間

図 13 は，一つのグラフの平均頂点数がモデルグラフ 20 個，入力グラフ 30 個におけるモデルグラフの数と，それをグラフ分解，部分グラフ同型判定するのに要する時間との関係を示している．提案手法の方が常に処理時間が上回っている．一つのグラフの平均頂点数が少ないと図 2 のような枝がないグラフが生成されにくく，割り当てられる入力グラフの頂点も少ないので，枝を残す工夫をしている我々の提案手法より，Messmer らの手法のほうが早くなったと考えられる．

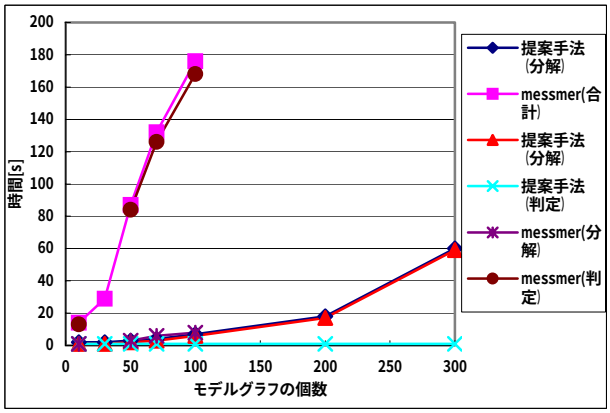

図 14 平均頂点数がモデルグラフ 50 個，入力グラフ 60 個におけるモデルグラフの個数と，モデルグラフの分解，部分グラフ同型判定に要する処理時間

図 14 は一つのグラフの平均頂点数がモデルグラフ 50 個，入力グラフ 60 個におけるモデルグラフの数と，それをグラフ分解，部分グラフ同型判定するのに要する時間との関係を示している．図 14 においてモデルグラフの個数が 100 を越えると測定不可能であった．我々の手法は，平均頂点数が増えても処理時間が急激に増加することはなく処理を行った．

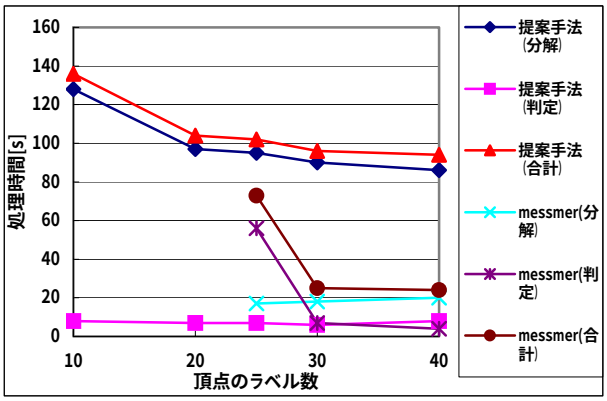

図 15 頂点ラベル数と，モデルグラフの分解，部分グラフ同型判定に要する処理時間

図 15 は頂点のラベル数と，それをグラフ分解，部分グラフ同型判定するのに要する時間との関係を示している．Messmer らの手法は，ラベル数が多いと早いが，ラベル数が少ないと測定不可能になる．我々の手法は頂点のラベル数が少なくても処理時間の増加はなだらかである．

以上より，我々の手法が，Messmer らの手法の頂点数が多いときに枝がない分解が生成され，頂点の割り当てが急増し，メモリオーバーとなる問題を解決し，より大規模なグラフを扱えるようになった．

頂点数	25 個	50 個	75 個	100 個
手順 1	29 秒	49 秒	80 秒	110 秒
手順 2	27 秒	48 秒	77 秒	101 秒

表 2 平均頂点数と部分グラフ同型判定に要する処理時間

次に，D_i の i=0,1,2…の順（分解データ D に格納された順）で部分グラフ同型判定する方法（手順 1）と，多数のモデルグラフの部分グラフである分解データから順に判定する方法（手順 2）を比較した．これ以前の実験（図 12 から図 15 まで）は，手順 1 の方法で部分グラフ同型判定を行った．

表 2 に手順 1, 2 におけるモデルグラフ数 300 個，入力グラフ数 5000 個のときの平均頂点数と部分グラフ同型判定に要する処理時間を示す．手順 2 が，手順 1 より 1 秒から 9 秒（2%から 8%）処理時間が早い．

頂点数	1-5	6-10	11-15	16-20	21-25
手順 1	752375	2896	3219	3211	1749
手順 2	702818	2870	3197	3209	1739

頂点数	26-30	31-35	35-40	41-45	46-
手順 1	70	18	17	10	1
手順 2	70	18	17	10	1

表 3 処理したグラフの頂点数とその処理回数

表 3 に手順 1, 2 における処理したグラフの頂点数とその処理回数を示した．処理したグラフの頂点数が 1 から 5 個のグラフの処理回数は，手順 1 より手順 2 のほうがおよそ 5 万回(6.7%)少ない．6 個から 25 個では，手順 2 のほうが処理回数は少ないがその差は小さい．26 個以上ではまったく変わらない．

部分グラフ同型判定は，グラフの頂点数が多いほど処理時間がかかる．よって，処理したグラフの頂点数が 1 個から 5 個の小さいグラフの処理回数が多数減少したとしても，処理時間にはその影響が現れにくい．しかし，手順 1 の D_i の i=0,1,2…の順（分解データ D に格納された順）で部分グラフ同型判定する方法より，手順 2 の多数のモデルグラフの部分グラフである分解データから順に判定する方法の方が，常に処理回数が少なく，入力グラフとモデルグラフが部分グラフになりにくいデータにおいては，大きなグラフを判定する回数が減るので，効果は大きくなるだろう．逆に，入力グラフがモデルグラフに含まれやすいデータにおいては，効果は少ない．

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．必ず連結グラフになるように分解することによって，頂点数が多いときの Messmer らの手法の問題を解決し，部分グラフ同型判定処理においてもさらに効率的なアルゴリズムを提案した．頂点数が多いグラフにおいて我々の手法が有利であることが示された．

今後の課題は，代表的な部分グラフ同型判定アルゴリズム VF と我々の提案手法を比較することである．更に，部分グラフ同型判定のステップにおいて，入力グラフがモデルグラフに含まれるかという問題を扱えるよう拡張することを考えている．

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. Messner and Horst Bunke, ”Efficient Subgraph Isomorphism Detection: A Decomposition Approach ”IEE Trans. On Knowledge and Data Eng., 12(2), 2000.

[2] J.R. Ullmann, “An Algorithm for Subgraph Isomorphithms,” J. Assoc. 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”, IEE Transactions on PATTERN ANALYSIS AND MACHINE INTELLIGENCE, vol.26, no.10, pp.1367-1372, October 2004