TTSP項グラフに対する マッチングアルゴリズムの改良

An Improvement of Pattern Matching Algorithm for TTSP Graph Patterns

鈴木 祐介
Yusuke Suzuki
広島市立大学 情報科学研究科
Email: y-suzuki @ hiroshima-cu.ac.jp

内田 智之
Tomoyuki Uchida
広島市立大学 情報科学研究科
Email: uchida @ hiroshima-cu.ac.jp

宮原 哲浩
Tetsuhiro Miyahara
広島市立大学 情報科学研究科
Email: miyares13 @ hiroshima-cu.ac.jp

***Abstract*—TTSP (Two-Terminal Series Parallel) graphs are used as data models for electric networks and scheduling. We use a TTSP term graph which is a TTSP graph having structured variables, that is, a graph pattern over a TTSP graph. We present an improvement of a polynomial time pattern matching algorithm which decides whether or not a given TTSP graph is obtained from a given TTSP term graph by replacing appropriate TTSP graphs with variables in the TTSP term graph.**

I. はじめに

TTSP グラフ (Two-terminal series parallel graph) は，電気ネットワークやスケジューリングをコンピュータで扱う際にデータモデルとして用いられることが多い．TTSP グラフは並列操作，直列操作と呼ばれる操作を帰納的に繰り返して得られるサイクルを持たない有向グラフである．TTSP 項グラフは TTSP グラフに変数の概念を導入したもので，変数には任意の TTSP グラフを代入可能である．本稿では，高味ら [1] が提案した TTSP 項グラフの多項式時間マッチングアルゴリズムの改良と実装を行ったので，その結果を報告する．

II. TTSP 項グラフ

次の $(1), (2)$ で帰納的に定義されるサイクルを持たない辺ラベル付き有向グラフを**TTSP 項グラフ**という．(1) 2つの頂点 u, v と u から v への1つの辺または変数から成る有向グラフは，ソースとして u を持ち，シンクとして v を持つ TTSP 項グラフである．(2) G_1, G_2 をソース s_1, s_2 とシンク t_1, t_2 をそれぞれ持つ TTSP 項グラフとする．このとき，s_1 と s_2，t_1 と t_2 を同一視する操作を**並列操作**，s_2 と t_1 を同一視する操作を**直列操作**という．並列操作，直列操作によって得られるグラフは TTSP 項グラフである．変数を持たない TTSP 項グラフを，単に **TTSP グラフ**と呼ぶ．TTSP 項グラフ g と TTSP グラフ G に対し，g の変数を適切な TTSP グラフで置き換えることで G が得られるならば，g と G は**マッチする**という．例えば，図1のように，TTSP 項グラフ g の変数 x を TTSP グラフ G_1 で，変数 y を TTSP グラフ G_2 で置き換えると TTSP グラフ G が得られるので，g と G はマッチする．

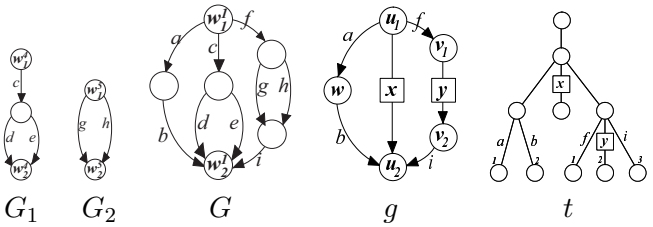

図 1. TTSP グラフ G_1,G_2,G，TTSP 項グラフ g，g の構文木 t

TTSP 項グラフを生成する過程を木で表現したものを TTSP 項グラフの**構文木**という．TTSP 項グラフとその構文木は 1 対 1 対応することが示されている [1]. 図 1 に，TTSP 項グラフ g に対応する構文木 t を示す．構文木の内部頂点には子に順序がついている頂点（直列操作を行うことを示している）と，子に順序がついていない頂点（並列操作を行うことを示している）が存在する．構文木の内部頂点 v に対して，v の子に順序がついているなら $kind(v) = o$，子に順序がついていないなら $kind(v) = u$ とする．また頂点 v の深さを $d(v)$ で表す．

III. マッチングアルゴリズム

TTSP 項グラフの照合問題を次のように定義する．

TTSP 項グラフの照合問題
入力：TTSP 項グラフ g，TTSP グラフ G
問題：G と g がマッチするか否かを判定せよ

この問題に対して高味ら [1] は，TTSP グラフと TTSP 項グラフを構文木へ変換し，構文木の照合問題として解く多項式時間マッチングアルゴリズム MTT-MATCHING を提案した．本研究では，MTT-MATCHING をもとに TTSP 項グラフの構文木では変数が葉にしか現れない点に着目し，頂点の深さを用い

た，頂点間の対応を調べる改良を行った．改良したアルゴリズム PT-MATCHING を図 2 に示す．アルゴリズム PT-MATCHING の動作を説明する．アルゴリズム PT-MATCHING は入力として TTSP グラフの構文木 T と TTSP 項グラフの構文木 t をとる．PT-MATCHING はまず初めに t の全ての頂点に**頂点識別子**と呼ばれる互いに異なる番号を割り当てる．t の頂点 v_t の頂点識別子を $I(v_t)$ で表す．T の頂点 v_T の**対応集合**は t の頂点の頂点識別子の集合であり，$CS(v_T)$ で表す．次に t の各頂点の親子関係と深さから $Rule(t)$ を作成する．t の頂点 u に対し，t の全ての**対応集合貼り付けルール**の集合 $Rule(t)$ を次のように定める．

$$Rule(t) \quad = \quad \bigcup_{u \in V_t} \{I(u)^{\overset{kind(u),d(u)}{\longleftarrow}} H(c_1), \cdots, H(c_m)\}$$

V_t は t の全ての頂点集合，$c_1, \cdots, c_m$ は t の頂点 u の全ての子とする．c_i が変数の要素なら $H(c_i) = (I(c_i))$ であり，それ以外なら，$H(c_i) = I(c_i)$ である．

T の各頂点 u に対し，手続き CS-ATTACHING を根から再帰的に適応し，対応集合を貼り付けていく．T の各頂点 u に対する手続き CS-ATTACHING$(u, Rule(t))$ の動作を説明する．$c_1, \ldots, c_n$ を u の全ての子とする．$Rule(t)$ 中の各ルール $I(u')^{\overset{kind(u'),d(u')}{\longleftarrow}} H(c'_1), \cdots, H(c'_m)$ に対して，$kind(u') = kind(u)$ かつ $d(u') = d(u)$ となるものだけを適用する．u の子の対応集合 $CS(c_1), \ldots, CS(c_n)$ が頂点識別子 $H(c'_1), \cdots, H(c'_m)$ を含むならば，$CS(u)$ に頂点識別子 $I(u')$ を加える．これは，T の根に対応集合が貼り付けられるまで繰り返される．貼り付けられた T の根の対応集合に t の根の頂点識別子が含まれるなら，PT-MATCHING=“yes” となる．

IV. 実験

前節で提案したアルゴリズム PT-MATCHING を計算機上に実装し，評価実験を行った．実験ではアルゴリズム MTT-MATCHING 及び PT-MATCHING を，主記憶メモリが 4.00GB，OS が Microsoft Windows7 である 3.16GHz の Xeon プロセッサを持つ計算機上に言語 Java で実装した．頂点数 n の TTSP 項グラフにマッチする頂点数 N の TTSP グラフを生成し，MTT-MATCHING と PT-MATCHING を行い，その実行時間を計測した．n は 10，N は 100 から 100 ずつ 2000 まで増やして実験を行った．実験は各 30 回行い，その平均実行時間を計測した．実行結果を図 3 に示す．結果より，提案したアルゴリズム PT-MATCHING が MTT-MATCHING よりも高速であることがわかる．これは，対応集合貼り付けルールに頂点の深さの情報を加えることで，適用するルールの候補数を減らすことができ，さらにそれによって対応集合の要素数を減らせるためと考えられる．

V. 結論

本研究では，高味ら [1] の提案した TTSP 項グラフの多項式時間マッチングアルゴリズムをもとに，TTSP 項グラフの構文木では変数が葉にしか現れない点に着目し，深さを利用して対応集合の判定をする改良を行った．また提案したアルゴリズムをコンピュータ上に実装し，その評価実験を行い，結果を報告した．今後の課題としては，提案したアルゴリズムのデータマイニングシステムへの応用などが考えられる．

Algorithm PT-MATCHING(T, t);
input T:TTSP グラフの構文木,
　　t:TTSP 項グラフの構文木;
begin
　r_T を T の根,r_t を t の根とする;
　$Rule(t)$ を作成する;
　PT-MATCHING-SUB(T, t, r_T);
　if $CS(r_T)$ に $I(r_t)$ が含まれる **then return** yes;
　else return no;
end.

Procedure PT-MATCHING-SUB(T, t, u);
input T:TTSP グラフの構文木,
　　t:TTSP 項グラフの構文木, u: T の頂点;
begin
　if u が葉 **then** $CS(u)$ に t の各葉の
　　頂点識別子を加える;
　else begin
　　$c_1, \ldots, c_n$ を u の全ての子とする;
　　foreach c_i **do** PT-MATCHING-SUB(T, t, c_i);
　　CS-ATTACHING$(u, Rule(t))$
　end;
end.

図 2.　アルゴリズム PT-MATCHING(T, t)

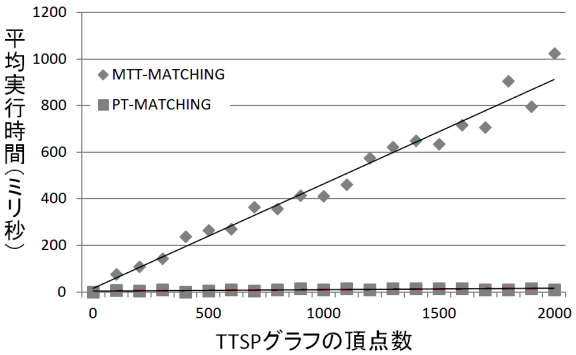

図 3.　マッチングの平均実行時間

参考文献

[1] Ryoji Takami, Yusuke Suzuki, Tomoyuki Uchida, Takayoshi Shoudai: Polynomial Time Inductive Inference of TTSP Graph Languages from Positive Data. IEICE TRANSACTIONS on Information and Systems, E92-D(2), pp.181–190, (2009)

問い合わせ先
〒 731-3194　広島市安佐南区大塚東 3-4-1
広島市立大学 情報科学研究科　鈴木 祐介