グラフデータベースにおける Top-k キーワード検索方式の改良と評価

蒋　吏君†　　大森 匡†　　星 守†

電気通信大学大学院情報システム学研究科‡

1　研究背景と目的

近年, データベースの研究としてグラフデータベースにおけるキーワード検索が注目されている. 関係データベースのタプルと XML のタグをノード, 関係データベースの外部キー参照と XML の階層関係をエッジと見なせば, あらゆるデータを一つの巨大なグラフ G に統合して表せる. 本稿はグラフデータベースにおける Top-k キーワード検索問題を扱う.

グラフにおけるキーワード検索は GST-k(Group Steiner Tree) を求める問題である. GST-k を求める問題は本質的に最小集合被覆問題と同等であり, NP 困難である. 従来の研究として, BANKS-I/II や DPBF などのアルゴリズムは top-k 解を近似的に計算している. 本稿では冗長列挙と top-k 最適解問題を解決した冗長抑制付き MDPBF アルゴリズムを提案する.

2　DPBF 法 [1]

2.1　DPBF 法の概要

DPBF 法 [1] は, 任意のデータグラフから全てのキーワードを含む top-k 個の最小コスト連結木を求めるための手法である. ノードごとに成長操作と併合操作を同時に行うことにより, steiner 木としてのコスト (グラフのエッジ重みの総和) 最小の正確な解を計算できる. コストが小さいほど, ノード間の関係が強いとする.DPBF 法の (最悪) 計算量はキーワード l に対して, $O(3^l n + 2^l((l + \log n)n + m))$($l$: キーワード数、n : ノード数、m : エッジ数) となる.[1]

DPBF 法は以下の漸化式の計算を, キーワード全集合 $\mathbf{P}(=\{p_1, p_2, ..., p_l\})$ を含む連結木が k 個見つかるまで繰り返すことによって,GST-k の解を求める.($\mathbf{p}, \mathbf{p_1}, \mathbf{p_2}, ...$ は $\mathbf{P}$ の, 空でない任意の部分集合とする)

(ノード v がキーワード部分集合 $\mathbf{p}$ を直接含むとき)

$$T_o(v, \mathbf{p}) = v \tag{1}$$

(それ以外の場合)

$$T_o(v, \mathbf{p}) = min\{TG(v, \mathbf{p}) \cup TM(v, \mathbf{p})\} \tag{2}$$

where

$$TG(v, \mathbf{p}) = \{(v, u) \oplus T_o(u, \mathbf{p}) | v \in N(u)\} \tag{3}$$

$$TM(v, \mathbf{p}) = \{T_o(v, \mathbf{p_1} \oplus T_o(v, \mathbf{p_2}) | (\mathbf{p_1} \cap \mathbf{p_2} = \phi) \wedge (\mathbf{p_1} \cup \mathbf{p_2} = \mathbf{p})\} \tag{4}$$

$\oplus$ は, 2 つの連結木（または, 1 つの連結木とエッジ）をマージして新しい連結木を作る演算子を表す. $N(v) = \{u \mid (v, u) \in E(G)\}$ は,v の隣接ノードの集合を表す.

式 (1) は, ノード v がキーワード部分集合 $\mathbf{p}$ を直接含んでいるとき,$T_o(v, \mathbf{p})$ をノード v とすることを表す. このとき T_o のコストは 0 となる. 式 (2) は,$TG(v, \mathbf{p})$ または $TM(v, \mathbf{p})$ に含まれる連結木のうち, 最小コストを持つ連結木が $T_o(v, \mathbf{p})$ となることを表す. 式 (3) はノード u をルートとする連結木から, ノード v へ成長することを表す (u, v を $V(G)$ の任意のノードとする). 式 (4) は $(\mathbf{p_1} \cup \mathbf{p_2} = \mathbf{p}) \wedge (\mathbf{p_1} \cap \mathbf{p_2} = \phi)$ を満たす $\mathbf{p_1}, \mathbf{p_2}$ に対して $T(v, \mathbf{p_1})$ と $T(v, \mathbf{p_2})$ を併合して新しく $T(v, \mathbf{p})$ を作ることを表す ($\mathbf{p_1}, \mathbf{p_2}$ はキーワード集合 $\mathbf{P}$ の部分集合).

2.2　DPBF の問題

DPBF 法では冗長列挙問題を解決していないし, さらに原理的に最適な top-1 しか出せず, 上位 2 番目以降は近似解になる. 我々が考えた冗長解とすべきケースは二つある. 一つはルートノートだけが異なって連結木のノード集合が全く同じの連結木である. これは DPBF-k の求める連結木が根付き木であるので, ルートノートさえ異なると,$|V(T)|$ 個の冗長な解が出力されうる ($|V(T)|$ とは連結木 T に含まれる頂点の個数である). 図 1-(a) で示す. もう一つは top-k の解となる連結木から成長させて得られる連結木である. 図 1-(b) で示す.

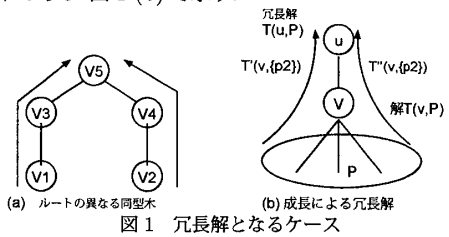

図 1　冗長解となるケース

同型木の排除については, 同型の連結木の数が多くないので, 解と確定される前, 既に解になった同じコストの連結木と木構造を比較すればすぐに排除できる.

図 1-(b) の冗長性を利用して解となる連結木を含む冗長解を排除することが出来る. 図 1-(b) で説明すると, 連結木 $T'(u, \{p1\})$ と $T''(u, \{p2\})$ がノード u で併合されて $T_m(u, \{p1, p2\})$ という候補解を探索キューに入れて確定を待つ. その後, コスト昇順の確定により解 $T(v, \{p1, p2\})$ が確定し, これをノード u へ成長させて冗長解というマークを付いた連結木 $T_g(u, \{p1, p2\})$ が候補としてキューに入る. 当然ながら $T_g(u)$ は $T_m(u)$ のコストより小さいので $T_m(u)$ は $T_g(u)$ に入れ換えられる.$T_g(u)$ は冗長解マークが付いているので, 探索処理には入るが, 解としては避けられる. この操作を繰り返していくと,$T_m(u)$ のような冗長解を全て回避できる.

Exact Top-k Keyword Search On Graph Databases

†L.Jiang, T.Ohmori, M.Hoshi

‡The University of Electro-Communications

3 MDPBF 法 [2]

3.1 MDPBF の概要 [2]

DPBF[1] の考えに基づいて, 正確な top-k 解を求めるためには, 任意の $v, \mathbf{p}$ について, $T_1(v,\mathbf{p}), \ldots, T_k(v,\mathbf{p})$ までを保存して計算する. そこで, 以下の式 (5)–(8) を計算するように, DPBF 法を改良する.

(v がキーワード $\mathbf{p}$ を直接含むとき)

$$T_1(v,\mathbf{p}) = v \tag{5}$$

(それ以外の場合, $1 \le n \le k$)

$$T_n(v,\mathbf{p}) = \text{min\_n}\ (\ TG(v,\mathbf{p}) \cup TM(v,\mathbf{p})\ ) \tag{6}$$

where

$$TG(v,\mathbf{p}) = \{\ (v,u) \oplus T_i(u,\mathbf{p}) \mid u \in N(v),\ 1 \le i \le k\ \} \tag{7}$$

$$TM(v,\mathbf{p}) = \{\ T_i(v,\mathbf{p_1}) \oplus T_j(v,\mathbf{p_2}) \mid \mathbf{p_1} \cap \mathbf{p_2} = \emptyset \wedge \mathbf{p_1} \cup \mathbf{p_2} = \mathbf{p},\ i \times j \le k\ \} \tag{8}$$

ここで, $T_n(v,\mathbf{p})$ は, v をルートとし, キーワード $\mathbf{p}$ を含む連結木の中で, n 番目に小さいコストを持つ連結木を表す. min_n は, 引数として与えられた連結木の集合から, n 番目にコストの小さい連結木を返す関数を表す.

式 (5) は式 (1) と同じことを表す. 式 (6) は, $TG(v,\mathbf{p})$ または $TM(v,\mathbf{p})$ に含まれる連結木のうち, n 番目 ($1 \le n \le k$) に小さいコストを持つ連結木が, $T_n(v,\mathbf{p})$ となるということを表す. ここで, $TG(v,\mathbf{p})$ は, ノード v の隣接ノード u をルートとする連結木 $T_i(u,\mathbf{p})$ ($1 \le i \le k$) から, ノード v へ grow した連結木の集合である (式 (7)). $TM(v,\mathbf{p})$ は, $(\mathbf{p_1} \cup \mathbf{p_2} = \mathbf{p}) \wedge (\mathbf{p_1} \cap \mathbf{p_2} = \emptyset)$ となる $\mathbf{p_1}, \mathbf{p_2}$ に対して, $T_i(v,\mathbf{p_1})$ と $T_j(v,\mathbf{p_2})$ ($i \times j \le k$) を merge した連結木の集合である (式 (8)).

式 (5)–(8) は, 任意の $v, \mathbf{p}$ について, 高々 top-k 個の連結木しか考慮しないで計算を行うが, $T_{k+1}, T_{k+2}, \ldots$ を部分木として持つ連結木は, それぞれの $v, \mathbf{p}$ について top-k 個の中に入りえない (GST-k の解になりえない) ので, $T_{k+1}, T_{k+2}, \ldots$ は top-k の計算 (GST-k の計算) に必要なく, 式 (5)–(8) によって, 確かに, GST-k の厳密解を求められる.

4 冗長抑制付き MDPBF

MDPBF 法 [2] では冗長列挙の抑制を考慮しなかった. そのため,MDPBF には DPBF[1] の冗長性を持っている. まず,2.2 節の冗長解抑制方法を MDPBF 法に適用する. そのうえさらに、MDPBF 特有の冗長性を排除する必要がある.

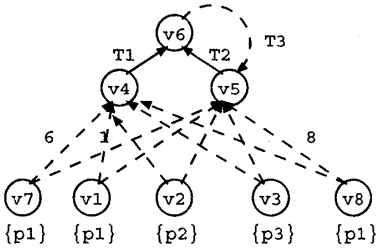

図 2 MDPBF の冗長解

すなわち,MDPBF ではノードごとに k 個の連結木を保存するので, 同じ葉ノード集合から異なる経路で同じルートノードに辿り着いてきた連結木が存在する可能性がある. これらを冗長解と見なして, 排除すべきである. この状況を図 2 で説明する. 同じ葉ノード集合を持つ連結木 $T_1(v4,\{p1,p2,p3\})$ と $T_2(v5,\{p1,p2,p3\})$ はノード $v6$ に成長してきて, たとえコストが違うとしても, 連結木に含まれる情報 (キーワードを含む葉ノード) が全く同じなので, 冗長の答えとする. これにより, ノード $v6$ から別の経路でノード $v5$ へ成長してきた連結木 T3 が $T(v5,\{p1,p2,p3\})$ と同じ葉ノードを持つので,top-10 を求めるとしても連結木 T3 を保存せずに済む. 以下, この冗長解抑制を入れた MDPBF 法を冗長抑制付き MDPBF と呼ぶ.

図 2 において, 冗長抑制付き MDPBF ではキーワード $p1$ を持つノード ($v1$,$v7$,$v8$) をそれぞれ葉ノードとする連結木は top-k の解として返せる. 冗長抑制付き DPBF ではノード $v1$ を葉ノードとするコスト最小の連結木 T($v4,\{p1,p2,p3\}$) しか返せない. なぜならノード $v4$,$v5$ でノード $v1$ から成長してきた連結木 $T(v4,\{p1\})$ のコストが一番小さいので, ノード $v7$、$v8$ から成長してきた連結木 $T(v4,\{p1\})$ が保存されないからである.

5 比較評価

文献 [1] の例題グラフ (図 3) で,{Query,DB,keyword,Jim} をキーワードとして冗長抑制付き DPBF と冗長抑制付き MDPBF により求めた top-3 解の比較を行った. 図 4 の (a)(b)(c) は冗長抑制付き DPBF で出せる top-3 の解である. 図 4-(d) の解は図 4-(c) よりいい解と考えられるが, 冗長抑制付き DPBF では出せない. また, 冗長抑制付き MDPBF では図 4 の (a)(b)(d) を top-3 の解として出せる.

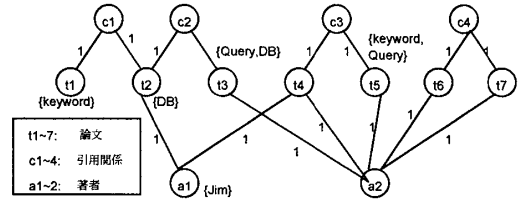

図 3 データグラフ

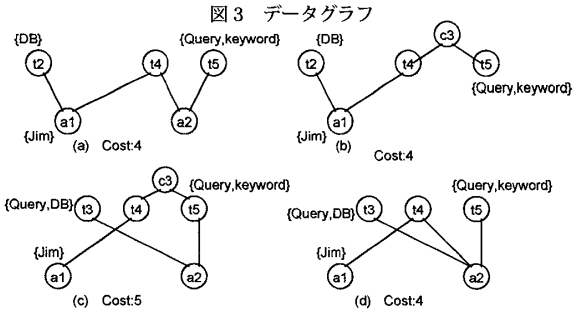

図 4 冗長抑制付き DPBF 法で出せる top-3 の解 (a)(b)(c) と出せない解 (d)

6 まとめ

本稿ではグラフデータベースにおける正確な top-k 解を検索できる冗長抑制付き MDPBF 法を提案した. 冗長抑制付き DPBF と冗長抑制付き MDPBF の実行処理効率について文献 [3] で述べる予定である.

参考文献

[1] B. Ding, etc. Finding Top-k Min-Cost Connected Trees in Databases. In *ICDE'07*,pp.836-845, 2007.

[2] 岡谷, 大森, 星, グラフデータベースにおけるキーワード検索方式の改良. *FIT Forum '09*, A-020, 2009.

[3] 蒋, 大森, 星, グラフデータベースにおける正確な Top-k キーワード検索方式. *DEIM '2010*, 投稿中.