最小遅延パス問題に対する近似アルゴリズムの実験的評価

中野 慎太郎† 浅野 孝夫‡

中央大学大学院 理工学研究科 情報工学専攻†
中央大学 理工学部 情報工学科‡

1 はじめに

無向グラフにおいて, 与えられた始点から終点まで, 与えられた点数以上の点を通るようなパスで長さ最小のものを求める問題を**最小遅延パス問題** (Minimum Latency Path problem, MLP) という. これは, 最小スパンニング木問題 (Minimum Spanning Tree problem, MST), 巡回セールスマン問題 (Traveling Salesman Problem, TSP), 賞金収集シュタイナー木問題 (Prize Collecting Steiner Tree problem, PCST) などに応用することができる問題で, NP-困難であることが証明されている. Chaudhuri, Godfrey, Rao, Talwar [1] は 2003 年に最小遅延パス問題に対する, 同期的プライマルデュアル法を用いた近似アルゴリズムを提案した. 本稿ではそのアルゴリズムを実装し, 計算機実験を通してその近似性能を評価する. さらに, 近似性能や計算時間などの観点からより効率的なアルゴリズムを目指し, 改善を試みる.

2 問題定義

重みつき無向グラフ $G=(V,E,c)$ に対してパス $P=(v_1,...,v_n)$ を考える. V は点集合, E は辺集合, c はコストをそれぞれ表している. パス P における点 $v_i \in P$ の**遅延** (latency) $l_{v_i,P}$ とは, 点 v_i に到達するまでのパス P のコスト, すなわち $l_{v_i,P}=c((v_1,...,v_i))$ と表すことができる. MLP の目的は, グラフ G において, 各ノードの遅延の総和が最小になるようなパスを求めることである. 本稿で扱う問題は, **k-パス問題** (k-path problem) と呼ばれるものであり, 始点 s から終点 t まで, 少なくとも k 個以上のノードを通るパスのうち, 最短のものを求める問題である.

3 Chaudhuri らのアルゴリズム [1]

3.1 線形緩和問題

Chaudhuri らのアルゴリズムは, Goemans and Williamson の PCST に対するアルゴリズム [2] に基づくものである. アルゴリズムは, s から t までの k-パス問題に対する線形緩和問題とその双対問題を用いている. 以下に線形緩和問題 (主問題) を記す.

$$
\begin{aligned}
\min \quad & \sum_{e\in E} c_e x_e \\
\text{s.t.} \quad & \sum_{e\in\delta(S)} x_e \geq 2x_v \qquad \forall S\subseteq V\backslash\{s,t\}, \forall v\in S \\
& \sum_{e\in\delta(U)} x_e \geq 1, \quad \forall U\subseteq V : t\in U, s\notin U \\
& \sum_{v\in V\backslash\{s,t\}} x_v \geq k-2 \\
& 0\leq x_v \leq 1 \qquad \forall v\in V\backslash\{s,t\} \\
& x_e \geq 0 \qquad \forall e\in E
\end{aligned}
$$

x_e は辺 e がパスに含まれるなら 1, そうでなければ 0 をとる変数で, x_v はパス上の s,t を除く $(k-2)$ 個以上の点で 1, それ以外では 0 をとる変数を表している. $\delta(S)$ は一方の端点のみが点集合 S に含まれるような辺の集合を示している.

また, 上記の主問題に対する双対問題は p, 各点 v に対して p_v, $S\subseteq V\backslash\{s,t\}$ と $v\in S$ の各対に対して $y_{v,S}$, そして $t\in U$ となるような各点集合 $U\subseteq V\backslash\{s\}$ に対して $y_{t,U}$ という変数を含むものである.

3.2 アルゴリズム

入力としてグラフ $G=(V,E)$, 始点 $s\in V$, 終点 $t\in V$ 及びパラメータ λ が与えられる. アルゴリズムは 2 つのフェイズ, 反復を繰り返し辺を追加していき木を求める Growth Phase と木から余分な辺を除去する Delete Phase より構成されている.

Growth Phase

アルゴリズムにおいて, グラフは**活性集合** (active component) か **不活性集合** (inactive component), どちらかに分割されるものとする. 各点 v は非負の**予算** (budget) b_v を持ち, v が含まれる集合の成長に寄与する. 正の予算を持つ点を含み, かつ s を含まない集合, もしくは t を含む集合が活性集合となり, すべての点が予算 0, もしくは s を含む集合が不活性となる. 初期状態においては, 各点はそれぞれ集合を構成しており, s は予算 0, t は予算 ∞, その他すべての点は予算 λ をそれぞれ持つ. なお, 変数 $y_{v,S}$ は各点 v が集合 S に含まれているときに, どれだけ「支払った」かを表しており, その初期値は 0 である.

以下にアルゴリズムを記す.

1. 小さな値 ϵ と各活性集合 S から点 v_S を選ぶ.

An Experimental Study of Approximation Algorithms for the Minimum Latency Path Problem

†Shintaro NAKANO, Information and System Engineering Course, Graduate School of Science and Engineering, Chuo University

‡Takao ASANO, Department of Information and System Engineering, Faculty of Science and Engineering, Chuo University

2. ϵ だけすべての点の λ を減らす.
   ϵ だけ各 v_S に対応する $y_{v,S}$ を増やす.

このとき, ϵ は以下の 1 か 2 どちらかを満たすように選ばれるものとする.

1. $b_{v_S} = 0$ となるように ϵ を選ぶ. なお, $b_{v_S} = 0$ となったときは, 他の $b_v > 0$ の点 $v \in S$ が選ばれる. そのような点がなければ S は不活性となる.
2. $y_{v,S}$ が辺 $e \in E$ に「支払う」ことで集合 C_1 と C_2 が併合できるように ϵ を選ぶ. このとき併合した集合に s が含まれるなら不活性となる.

Growth Phase は以上の操作を繰り返し, 各段階で選ばれた辺を記憶しておき, すべての集合が不活性となった時点で終了する.

Delete Phase

Growth Phase により得られた s を含む集合を T とする. 得られた T は木となっている. ここで, Growth Phase の中で不活性集合を形成したことのあるすべての部分木 $S \subseteq T \backslash \{s\}$ を除去する. 除去することで得られた木 T_{k_λ} と $y_{v,S}$ を返して終了する.

4 提案手法

Chauduri らのアルゴリズムは Goemans and Williamson のアルゴリズムに基づいたものであることは既に述べた通りだが, 最も大きな違いは Chauduri らが予算という概念を与えたことにある. 3.2 においては, s と t 以外のすべての点に対して一律に同じ予算 λ をパラメータとして与えている. そこで, λ を一律ではなく, 距離などの条件を与えて変化させることにより, 解の改善及び計算時間の短縮化を図った.

5 実験的性能評価

5.1 入力データ

実験にあたって, 本稿では配送計画問題 (Vehicle Routing Problem, VRP) に対する著名な例題である, Solomon のベンチマーク* を入力データとして用いた. 顧客が密集している配置の C タイプ, ランダムな配置の R タイプの 2 タイプの例題をそれぞれ入力とした実験を行った.

また, s 及び t からある距離以内の点集合を **near**: V_N, それ以外を **far**: V_F として, V_N と V_F への λ の与え方に差異をつけ, それぞれの集合の個数を変化させ実験を行い, パスの長さ及び計算時間の計測を行った.

5.2 実験結果

表 1 は様々な入力データに対して得られたパスの長さと計算時間をまとめたものである. 入力タイプはベンチマークのタイプ (C タイプ, R タイプ) を示している. V_N の個数, V_F の個数はそれぞれ個数の比率が約 $1:1, 2:8, 8:2$ となるように s と t からの距離を与えた. また, λ は入力として与えたパラメータに対し, V_N に含まれる点の予算の $1.25, 1.5, 1.75, 2$ 倍, V_F に含まれる点の予算の $1/1.25, 1/1.5, 1/1.75, 1/2$ 倍, といったようにそれぞれ差異をつけた.

表 1 各入力に対するパスの長さと計算時間の比較

入力タイプ	V_Nの個数	V_Fの個数	λの与え方	パスの長さ	計算時間(秒)
R			Normal	2265.34839	5.83
R	22	76	n*2.0 f/1.0	2085.38564	10.14
R	22	76	n/1.25 f*1.0	672.982677	5.58
R	55	43	n*1.5 f*1.0	1785.0753	2.06
R	55	43	n/1.25 f*1.0	727.272013	5.2
R	85	13	n*1.5 f*1.0	1848.83526	2.17
R	85	13	n/1.75 f*1.75	2039.21783	21.88
C			Normal	2543.78057	5.94
C	15	83	n*1.5 f*1.0	2979.16091	6.33
C	15	83	n/2.0 f*2.0	2212.24199	4.09
C	48	52	n*1.5 f*1.5	891.382133	22.78
C	48	52	n/1.25 f*1.0	1032.26941	5.74
C	81	17	n*2.0 f/2.0	1569.57148	3.78
C	81	17	n/2.0 f*2.0	4510.77958	3.55

R タイプ

R タイプの入力に対しては, V_N と V_F の個数比が 1 : 1 の場合はパスの長さ, 計算時間ともにどのような場合でも改善されていることが多かった. V_N の個数が多いときにはパスの長さの改善は全般的になされているが, 計算時間については特に V_N の点に予算を多く与えることで大幅な改善ができた. 逆に, V_F の個数が多い場合は V_F の点に多く予算を与えることで良い解が得られている.

C タイプ

C タイプの入力に対しては, V_N と V_F の個数がほぼ同数のとき, V_N の点に多い λ を与えることにより, パスの長さについては大きく改善することができた. V_N の個数を多くした場合は, V_N の予算の比率を高めることがパスの長さの改善に直結し, 計算時間も短縮されている. これは R タイプでも同様だが, V_N に多く予算を与えることで, s と t に近い点が早い段階で数多く活性集合に含まれ, その結果としてアルゴリズムが早く終了すると考えられる. 一方, V_F の個数を多くとった場合は, V_F の点に多く λ を与えることでパスは短くなった.

6 結論

入力データタイプに関わらず, V_N と V_F, どちらか個数の多い方に予算 λ を与えることがパスの長さを減らす上では有効であることがわかった. また, 双方の個数を同数にした場合にもパスの長さの改善には成功している. 以上のことから λ の与え方によってはかなりの改善を果たし, 有効であるといえる.

今後の課題としては, さらなる λ の与え方の分析, そして大規模データでの実装と実験, 及びそれを実現するためのアルゴリズムの高速化が挙げられる.

参考文献

[1] K. Chaudhuri, B. Godfrey, S. Rao, K. Talwar: Paths, Trees, and Minimum Latency Tours. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, Massachusetts, 2003, pp.36-45.

[2] M. X. Goemans and D. P. Williamson: A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2), 1995, pp. 296-317.

*M. M. Solomon http://w.cba.neu.edu/ msolomon/home.htm