6F-07 断線したネットワークの復旧問題に対する近似アルゴリズム

西村 知洋　　宮崎 修一　　岩間 一雄
京都大学 情報学研究科

1 はじめに

現在, インターネットは情報の伝達手段として必要不可欠なものになっている. そのため, 事故や災害などによってネットワークが断線した場合における復旧は非常に重要な問題である. 災害時には, 復旧にあてることのできる予算, 人材などには限りがあり, 全ての破壊箇所を同時に復旧させることは困難である. このため, そのような資源を限定した上での効率の良い復旧に関する研究が盛んに行なわれている [GMNS99]. 本研究では, 制限された予算内でできるだけ多くのノードの連結性を復旧する問題（ネットワーク復旧問題）を考え, この問題に対する近似アルゴリズムを考える.

本研究で取り扱うネットワーク復旧問題は, 与えられた枝のコスト内でできるだけ多くの頂点を連結する問題であるが, 逆に, 与えられた頂点数を連結するためにかかる枝のコストの和を最小にする問題として, k- 最小全域木問題（k-MST 問題）がある [AK00, AR98, Gar96]. k-MST 問題の研究は非常に盛んに行なわれており, 最も新しい結果は, 任意の正の定数 ϵ に対して $2+\epsilon$- の多項式時間近似アルゴリズムである [AK00]. 本研究では, k-MST 問題に対する近似アルゴリズムを利用してネットワーク復旧問題に対する近似アルゴリズムを構築し, その近似度を評価した. その結果, ネットワーク復旧問題に対する 7- 近似アルゴリズムを構築することができた.

2 ネットワーク復旧問題と k- 最小全域木問題

本研究では, 以下の問題を取り扱う.

ネットワーク復旧問題
入力: 枝コスト付き無向グラフ $G(V,E)$, 整数 B.
問: コスト B 以内の全域木で頂点数最大のものをみつける.

この問題をネットワーク復旧問題と呼ぶ理由は以下の通りである. グラフ G で与えられるネットワークの枝が全て断線したとする. 枝を復旧するコストは枝に付けられたコストであるとする. このとき, 与えられた予算 B 以内で, できるだけ多くの頂点を連結させようというのが本問題である.

k- 最小全域木問題 (k-MST 問題)
入力: 枝コスト付き無向グラフ $G(V,E)$, 整数 k.
問: k 頂点からなる全域木でコスト最小のものをみつける.

3 近似アルゴリズム

k-MST 問題に対する $2+\epsilon$- 近似アルゴリズム (以下, Approx-k-MST と表記する) を利用して, ネットワーク復旧問題を解く近似アルゴリズムを示す.

アルゴリズム 1

ステップ 1: $k=1$ とする.
ステップ 2: Approx-k-MST を適用して k 頂点の木 T を得る.
ステップ 3: (T のコスト) $\leq B$ ならば $k=k+1$ としてステップ 2 へ. (T のコスト) $> B$ ならば $t=k$ としてステップ 4 へ.
ステップ 4: $k=t-1$ のときの近似解を出力する.

4 近似度の解析

まず, 以下の用語を定義する.

定義. T を任意の木とする. T から 1 頂点 v を選び, v につながる枝を 2 つのグループ (E_1,E_2) に分ける. v と E_1 から生成される部分木を T_1, v と E_2 から生成される部分木を T_2 とする. これを T を T_1 と T_2 に**分割する**という.

定義. T に分割を繰り返して頂点数 n 以上の部分木 m 個に分割できるとき, **T から n 頂点の部分木が m 個とれる**という.

アルゴリズム 1 の近似度について以下の定理が得られる.

“An approximation algorithm for network recovery problem”
Tomohiro Nishimura, Miyazaki Shuichi, Kazuo Iwama
School of Informatics, Kyoto University

定理. アルゴリズム 1 の近似度は 7 以下である.

証明. ネットワーク復旧問題の任意の例題を G, B とする. G に対する k-MST 問題の最適解のコストを $f(k)$ で表す. また, 同様に G に Approx-k-MST を適用させて得られた近似解のコストを $g(k)$ で表す. 例題 G, B にアルゴリズム 1 を適用して得られた近似解のサイズ (頂点数) を s とするとき, アルゴリズム 1 の性質より $g(s+1) > B, g(s+1) \leq (2+\epsilon)f(s+1)$ となる. 従って, $f(s+1) > \frac{B}{2+\epsilon}$ となる. すなわち, G からどのように $s+1$ 頂点の全域木をとっても, そのコストは $\frac{B}{2+\epsilon}$ よりも大きい.

次に, G, B のネットワーク復旧問題に対する最適解を T とし, そのサイズを t とする.

補題 1. T から $s+1$ 頂点の部分木を 3 個とることはできない.

証明. T から $s+1$ 頂点の部分木を 3 個とることができたとする. T は G の部分木であるから, これら 3 つの部分木のコストはいずれも $\frac{B}{2+\epsilon}$ より大きい. 従って, T のコストは $3 \times \frac{B}{2+\epsilon} > B$ となり, T がネットワーク復旧問題の解であることに矛盾する. □

ここで, $s+1$ 頂点の木を 3 個とれない木の中で頂点数が最大のものを U とする. 補題 1 より, (T の頂点数) $\leq$ (U の頂点数) となる.

補題 2. U の頂点数は $7s$ 以下である.

証明. 次節で述べる. □

従って, $\frac{t}{s} \leq \frac{7s}{s} = 7$ となり, アルゴリズム 1 の近似度が 7 以下であることが示された. □

5 補題 2 の証明

補題 2 を証明するために, 以下の補題 3 を証明する.

補題 3. a を自然数とするとき, a 頂点の部分木を 2 個とることのできない木の中で最大の頂点数を持つものの頂点数は $3a-5$ である.

証明. サイズ a の部分木が 2 個とれない最大の木を T とする. このとき, T をどの頂点でどのように分割しても小さい方のサイズは明らかに $a-1$ 以下である. 以下の一連の補題の証明は略す.

補題 3.1. T はサイズ $a-1$ と a 以上の 2 個の木に分割できる点を持つ.

補題 3.1 より T にはサイズ $a-1$ と a 以上の木に分割する点が存在するので, その点を T の「中心」と呼ぶことにする.

補題 3.2. T はただ 1 つの中心を持つ.

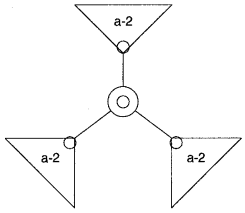

図 1: サイズ a の部分木が 2 個とれない最大の木

補題 3.3. T の中心は 3 分岐である.

補題 3.1 から補題 3.3 より中心が 1 個, その分岐数 3 で最大になるものが存在する. ここで, 図 1 の木はどこに頂点を 1 個加えてもサイズ a の部分木が 2 個とれる. よって, この木が求める木である. これよりサイズ a の部分木が 2 個とれない木の最大サイズは, $3(a-2)+1 = 3a-5$ である. □

補題 3 より, $s+1$ の木を 2 個とることのできない最大の木は, $3(s+1)-5 = 3s-2$ 頂点を持つことがわかる. 証明は略すが, 補題 2 で述べた U の頂点数は, この $3s-2$ 頂点からなる 2 個の木を, 長さ s のパスでつないだ木の頂点数で押さえられる. 従って, U の頂点数は $2\times(3s-2)+s = 7s-4 \leq 7s$ 以下となる. □

6 おわりに

今回は最適解の上限をサイズ $s+1$ の部分木が 3 個とれない木の最大サイズで押さえたが, 実際にはもう少し低く押さえることができると予想される. 今後はその証明を考える.

参考文献

[AK00] S. Arora and G. Karakostas, "A 2+ϵ approximation algorithm for the k-MST problem," *Proc. SODA'00*, to appear.

[AR98] S.Arya, H.Ramesh, "A 2.5-factor approximation algorithm for the k-MST problem," *Information Processing Letters 65*, pp. 117-118, 1998.

[Gar96] N.Garg, "A 3 factor approximation algorithm for the minimum spanning k vertices," *Proc. FOCS'96*, pp. 302-309, 1996.

[GMNS99] S.Guha, A.Moss, J.Naor, B.Schieber, "Efficient recovery from power outage," *Proc. STOC'99*, pp. 574-582, 1999.