割り当て問題に対するランダマイズド・アルゴリズムの実験的検証

5L－6

中野 淳　徳山 豪
日本IBM 東京基礎研究所

Abstract

n 人の人員を k ヶ所に費用が最小となるように割り当てる問題を，$n \gg k$ の場合に効率良く解くランダマイズド・アルゴリズムの性能を実験によって確認する．

1 Introduction

最小費用割り当て問題とは $n+k$ 個の頂点と kn 本の枝を持つ完全二部グラフ上の最小費用流問題であり，次のように定義される (図1参照)．

$$\begin{aligned} &\text{minimize} \quad \textstyle\sum_{i=1}^{n}\sum_{j=1}^{k}\alpha_{ij}y_{ij} \\ &\text{subject to} \quad \textstyle\sum_{i=1}^{n} y_{ij} = \lambda_j,\ \sum_{j=1}^{k} y_{ij} = 1, \\ &\qquad y_{ij} = 0 \text{ or } 1 \ (1 \le i \le n,\ 1 \le j \le k) \end{aligned}$$

この問題は供給量の総和と需要量の総和が等しいとき (すなわち $\sum_{j=1}^{k}\lambda_j = n$) にのみ解を持つが，そうでない場合でも供給点または需要点を1つ余分に追加することにより上記の標準形に変換することができる．また問題の対称性から，一般性を失うことなく $n \ge k$ とできることに注意しておく．Fredman-Tarjan のアルゴリズム [1] はこの問題を $O(kn^2 + n^2 \log n)$ 時間で解くが，これは k が n に比べて非常に小さい場合でも $\Omega(n^2)$ 時間かかってしまうという点で不満足である．本論文ではまず最小費用割り当て問題をそれと等価な幾何学的問題に置き換え，$n \gg k$ であるようなアンバランスなネットワークに対してその非対称性を利用して効率良く解くランダマイズド・アルゴリズム [2] を紹介し，その有効性を計算機実験によって確認する．

2 幾何学的問題への変換

各供給点 i $(1 \le i \le n)$ に対し，それと需要点を結ぶ k 本の枝のコストを並べてできる $\mathbf{R}^k$ 内の点 $p(i) = (\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{ik})$ を考える．また各需要点 j $(1 \le j \le k)$ には capacity λ_j を持った領域 $T(G;j) = \{(x_1, x_2, \ldots, x_k) \in \mathbf{R}^k \mid \text{for } \forall \ell \ne j, x_j - g_j \le x_\ell - g_\ell\}$

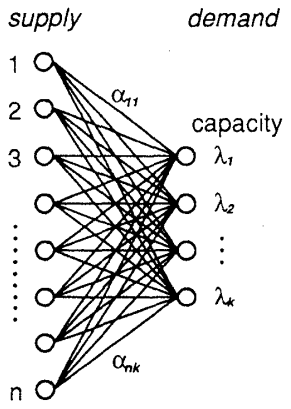

図 1: 最小費用割り当て問題

を対応させる ($G = (g_1, g_2, \ldots, g_k)$ は $\mathbf{R}^k$ 内のある点—以降 *splitter* と呼ぶ)．図2に $k = 3$ の場合の配位を平面 $x_1 + x_2 + x_3 = 0$ に正射影したものを示す．基本的なアイディアは *splitter* G が与えられた時，$p(i) \in T(G;j)$ なら供給点 i を需要点 j に割り当てる，というものである．ここでもし各 j $(1 \le j \le k)$ に対して領域 $T(G;j)$ 内の点の数がその領域の capacity λ_j に等しいような *splitter* G を見つけることができれば，もとのネットワーク上での対応するフローは我々が求めようとしている最小費用流になっていることが示される．以下ではこの *splitter finding problem* について考える．

3 アルゴリズム

各領域 $T(G;i)$ $(1 \le i \le k)$ ごとに $k-1$ 個のプライオリティー・キュー $A(i,j)$ $(j \ne i)$ を用意し，各供給点 $(x_1, \ldots, x_k) \in T(G;i)$ をキュー $A(i,j)$ $(1 \le j \le k;\ j \ne i)$ に $x_j - x_i$ をキーとして蓄えておく．基本となるステップは1点を追加したときにある領域でオーバーフローが生じたら，splitter を移動してそれを解消する操作であり，それは $O(k^2 \log n)$ 時間で実行できることが示される (これは k 頂点の完全グラフ上での最短路木，各キューにつき高々1回の削除操作と高々1回の挿入操作などからなる)．従って逐次添加法を用

Experimental study of a randomized minimum-cost assignment algorithm
Jun NAKANO and Takeshi TOKUYAMA
IBM Research, Tokyo Research Laboratory

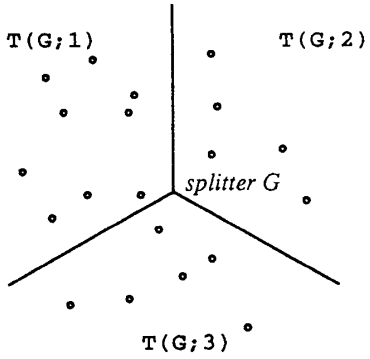

図 2: $k = 3$ の場合の splitting の平面 $x_1 + x_2 + x_3 = 0$ への投影図

いると合計では最悪の場合 $O(k^2 n \log n)$ 時間かかるが，これは Fredman-Tarjan のアルゴリズムと比較して，$n > k \log k$の場合に改善となっている．

一方，ランダマイズド・アルゴリズムではまず $p(i)$ $(1 \leq i \leq n)$ の中からランダムに m 点を選びそれを $(m\lambda_1/n, m\lambda_2/n, \ldots, m\lambda_k/n)$ に分割する splitter を求める．m を十分大きくとればこの splitter は最適の splitter の良い近似となっているはずである．次に残りの $n - m$ 点を splitter を更新せずに追加し，オーバーフローが生じたら後処理でこれを解消する．m の値を適当に選ぶことにより計算時間は $O(kn + k^{7/3} n^{2/3} \log^{4/3} n)$ となる (Fibonacci ヒープを使った場合)．これは $n > k \log k$なら前述の逐次添加アルゴリズムより速く，特に $n > k^4 \log^4 k$ の場合には最適である $O(kn)$ (= input size) 時間のアルゴリズムとなっている．

4 実験結果

問題のインスタンス (すなわち，コストと容量) は乱数によって生成し，前もって行った収束性を見るための実験結果から各n, kの組について 100 回ずつ試行した．$n = 100, 200, 400, 800, \ldots, 12800$; $k = 2, 4, 8, \ldots, 64$ の各組合せについて実験を行った結果を図 3に示す．このグラフ上で回帰平面を求めることにより，計算時間のパラメータ依存性は逐次添加アルゴリズムは $k^{1.43} n^{0.91}$ (理論値 $k^2 n \log n$)，ランダマイズド・アルゴリズムは $k^{1.45} n^{0.76}$ (理論値 $kn + k^{2.33} n^{0.67} \log n$) となっている．すなわち，オーバーフローによる splitter の更新回数は最悪ケースよりはかなり少なく，計算時間は点のキューへの挿入時間 (kn) の方に引っ張られ

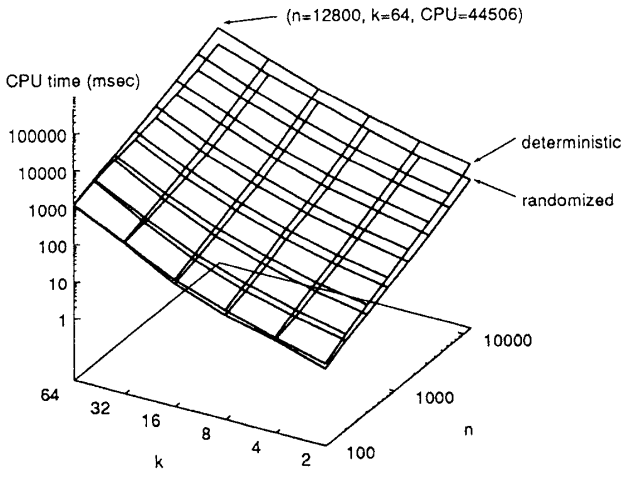

図 3: CPU time の n, k依存性

てより良い n, k依存性を持っていることが見てとれる．また，ランダマイズド・アルゴリズムは n, kの値が大きくなるほど有効で，実験した範囲内では逐次添加アルゴリズムと比べて 1.02 倍から 2.18 倍の高速化になっていることが確認された．

5 おわりに

今回の実験では，我々のアルゴリズムは予想された通りの性能を発揮し，またランダマイゼーションの効果も確認された．ランダムに生成されたものではなく，実際のアプリケーションで生じるインスタンスに対する性能評価という課題は残されているものの，その場合でも我々のランダマイズド・アルゴリズムはその性質上，問題のインスタンスによる影響を受けにくく，ますます逐次添加アルゴリズムに対する優位性を発揮すると期待される．

参考文献

[1] M. Fredman and R. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, *J. ACM., 34 (1987)*, pp.596–615.

[2] T. Tokuyama and J. Nakano, Geometric Algorithms for the Minimum Cost Assignment Problem, *Proc. 7th ACM Symposium on Computational Geometry*, pp.262–271, (1991).