最小コストフロー問題の高速解法とその VLSI コンパクション問題への適用

重弘 裕二† Itthichai ARUNGSRISANGCHAI†† 白川 功† 高橋 浩光†††

† 大阪大学 工学部 情報システム工学科
565 吹田市山田丘 2-1
†† Department of Electronic, King Mongkut's Institute of Technology
Ladkrabang, Bangkok, 10520 Thailand
††† 岡山県立大学 情報工学部 情報通信学科
719-11 総社市窪木 111
E-mail: sigehiro@ise.eng.osaka-u.ac.jp

VLSI のコンパクション処理に線形計画法を用いることができるが, その時に扱われる線形計画問題は最小コストフロー問題と双対であるので, 最小コストフローアルゴリズムを用いた手法が構築できるはずである. そこで, 本文ではコンパクション処理に適した最小コストフローアルゴリズムについて, 特にこれまで主単体法ほどには研究が行われていなかった主双対法に着目し, それから演繹される高速化手法について考案する. 各種最小コストフローアルゴリズムをコンパクション処理に適用した結果, コンパクション処理では, 本手法は主単体法以上に速いことがわかった.

A Fast Minimum Cost Flow Algorithm and Its Application to VLSI Layout Compaction

Yuji SHIGEHIRO† Itthichai ARUNGSRISANGCHAI†† Isao SHIRAKAWA†
Hiromitsu TAKAHASHI†††

† Department of Information Systems Engineering, Faculty of Engineering, Osaka University
2-1 Yamada-Oka, Suita, 565 Japan
†† Department of Electronic, King Mongkut's Institute of Technology
Ladkrabang, Bangkok, 10520 Thailand
††† Department of Communication Engineering
Faculty of Computer Science and System Engineering, Okayama Prefectural University
111 Kuboki, Souja, 719-11 Japan

In the layout design of custom VLSI, a number of effective compaction algorithms have been constructed on the basis of linear programming. Since the problem is dual to the minimum cost flow problem, flow algorithms can be applicable to the layout compaction. In this paper, an existing flow algorithm is investigated for the layout compaction, and a fast flow algorithm based on the primal-dual method is proposed. Experimental results show that, for the compaction problem, the dual-simplex method is the fastest of the existing algorithms, and the proposed method is more effective than them.

1 まえがき

集積回路技術の進歩にともない, VLSI はますます大規模化し, もはや人手だけに頼る設計は不可能となっている. これに対処する CAD 算法に関する研究も進み, 各設計工程における自動化が急速に進行している. レイアウト設計は, VLSI 設計の全工程の中で依然として主要な部分を占め, これまで多くの研究が進められてきた. なかでも, レイアウトの最適化を行うコンパクション処理に対しては数多くの手法が提案されている.

通常, コンパクション手法では, レイアウト要素の座標間の制約を表現した制約グラフ[1]に基づいて処理を行う. すなわち, まず, レイアウトを表現する制約グラフを構築し, 次に, その制約グラフを基に最適化されたレイアウトを生成する. 特に後者の処理は, 最終的なレイアウトの質を大きく左右するため, 制約グラフの各枝に対して各種のコストを導入することにより, コンパクション問題を線形計画問題として定式化し, 線形計画法により最適解を求めるという手法が採られている. これらの手法では, 特殊な制約を扱う場合を除き, グラフ論的主単体法 (primal-simplex method) が広く用いられている[2-4].

このような, グラフに関する線形計画問題は, これまで, 最小コストフロー問題という形で研究されてきた. それらは, 多項式時間アルゴリズムと非多項式時間アルゴリズムに分類できる. まだ実用の域には達していないといわれている多項式時間アルゴリズムに対し, 非多項式時間アルゴリズムは古くから知られており, 理論面および実用面からさまざまな研究が行われてきた[5-7].

コンパクション問題は, 一種の最小コストテンション問題[8]とみなすことができるため, 双対な最小コストフロー問題に変換することができる. したがって, 最小コストフロー問題に対する種々の解法を用いてコンパクション問題の解を求めることができるはずであるが, そのような応用はあまり例を見ない.

本文では, コンパクション問題を高速に解くための最小コストフローアルゴリズムについて, 特に, 主双対法 (primal-dual method) の高速化手法を中心に考案する. 従来の主単体法, 双対単体法, 並びに本手法を実現し, コンパクション問題に適用した結果, 双対単体法が主単体法より速いこと, コンパクション問題に関して, 本手法が双対単体法と同等以上の速度であることを示す. さらに, 本手法に基づく多項式時間アルゴリズムを実現し, その評価をも行う.

2 定義

$G=(V,E)$ は, 節点集合 V と 枝集合 E からなる有向グラフを表す. **フロー** f とは, E を定義域とする実関数で

Kirchhoff の電流則: 任意のカットセット Γ に対して,

$$\sum_{e\in\Gamma}\kappa(\Gamma,e)f(e)=0 \qquad (1)$$

ただし $\kappa(\Gamma,e)$ は, 枝 e が Γ と同じ向きであれば $+1$, そうでなければ -1

を満たすものをいう. **許容フロー** とは, 各枝 $e\in E$ に対して定められた**フローの下限** $\alpha_f(e)$ と**上限** $\beta_f(e)$ に対して $\alpha_f(e)\le f(e)\le\beta_f(e)$ $(e\in E)$ を満たすフロー f をいう. **最小コストフロー** とは, 各枝 $e\in E$ に対して指定された**単位フロー当たりのコスト** を $c(e)$ とするとき, すべての許容フローの中で**フローのコスト** $c(f)\triangleq\sum_{e\in E}c(e)f(e)$ を最小とするような許容フロー f をいう.

テンション t とは, E を定義域とする実関数で

Kirchhoff の電圧則: 任意の閉路 Λ に対して,

$$\sum_{e\in\Lambda}\varepsilon(\Lambda,e)t(e)=0 \qquad (2)$$

ただし $\varepsilon(\Lambda,e)$ は, 枝 e が Λ と同じ向きであれば $+1$, そうでなければ -1

を満たすものをいう. **許容テンション** とは, 各枝 $e\in E$ に対して定められた**テンションの下限** を $\alpha_t(e)$ と**上限** $\beta_t(e)$ に対して $\alpha_t(e)\le t(e)\le\beta_t(e)$ $(e\in E)$ を満たすテンション t をいう. **最小コストテンション** とは, 各枝 $e\in E$ に対して指定された**単位テンション当たりのコスト**を $d(e)$ とするとき, すべての許容テンションの中でテンションのコスト $c(t)\triangleq\sum_{e\in E}d(e)t(e)$ を最小とするような許容テンション t をいう.

3 定式化

3.1 コンパクション問題の定式化

配置とは, 平面上に置かれた各レイアウト要素の位置の集合をいう. 便宜上, レイアウト要素はすべて矩形であり, その各辺は水平または垂直に置かれるものとする. 制約グラフによるコンパクションでは, x 方向と y 方向の処理は全く同じであるので, x 方向についてのみ考察する.

配置が与えられたとき, 対応する水平制約グラフ[1] $G=(V,E)$ とは, その各節点 $v_i \in V$ が矩形の垂直辺 i を表し, 各枝 $e=(v_i,v_j) \in E$ が v_i, v_j に対応する 2 つの辺 i, j の間に位置に関する制約が存在することを表す, 有向グラフである. $v_i \in V$ に対応する辺 i の位置を示す x 座標を $x(v_i)$ で表したとき, 長さ l は, E を定義域とする実関数で $l(e) \triangleq x(v_j)-x(v_i)$ $(e=(v_i,v_j) \in E)$ であるものとする. 各枝 $e \in E$ に対応する位置制約は, e に対して定義されているレイアウトパラメータ (例えば, 矩形間の最小間隔, 矩形の最小幅, 等) を $r(e)$ とするとき, 不等式 $r(e) \leq l(e)$ $(e \in E)$ として与えられる. 制約グラフに基づくコンパクション問題の解法[2-4]とは与えられた制約グラフから定式化される次の最小化問題を解くことである.

問題 1

各枝 $e \in E$ に対して単位長さ当たりのコスト $c(e)$ が与えられたとき, 制約 $r(e) \leq l(e)$ $(e \in E)$ を満たし, $\sum_{e \in E} c(e)l(e)$ を最小化するような長さ l を求めよ.

ここで, 任意の長さ l と任意の閉路 $\Lambda = (v_1, e_1, v_2, e_2, \ldots, v_n, e_n, v_1)$ に対して,

$$\begin{aligned}\sum_{e \in \Lambda} &\varepsilon(\Lambda,e)l(e) \\ &= \sum_{1 \leq i \leq n} \varepsilon(\Lambda,e_i)l(e_i) \\ &= (x(v_2)-x(v_1))+(x(v_3)-x(v_2))+\cdots \\ &\quad +(x(v_n)-x(v_{n-1}))+(x(v_1)-x(v_n)) \\ &= 0\end{aligned}$$

である. したがって, 長さはテンションであり, **問題 1** はテンションに関する次の最小化問題と等価である.

問題 2

各枝 $e \in E$ に対して定められた単位テンション当たりのコスト $d(e)$ に対して, 制約 $\alpha_t(e) \leq t(e)$ $(e \in E)$ を満たし, $\sum_{e \in E} d(e)t(e)$ を最小とするようなテンション t を求めよ.

3.2 コンパクション問題と最小コストフロー問題の等価性

グラフ G の節点数を n, 枝数を m, 零度を $\mu\ (=m-n+1)$, 階数を $\rho\ (=n-1)$ で表す. また, $(\mu \times \mu)$ の単位行列を $\mathbf{1}_\mu$, $(\rho \times \rho)$ の単位行列を $\mathbf{1}_\rho$ とする. G の木 T を考え, 各枝に対し, 補木枝が $e_1, \cdots, e_\mu$, 木枝が $e_{\mu+1}, \cdots, e_m$ となるように番号付けを行う. いま, 各 $e \in E$ に対して関数 $w(e)=t(e)-\alpha_t(e)$ を定義し, $\boldsymbol{w}_{\bar{T}}$, $\boldsymbol{w}_T$ はそれぞれ行ベクトル $(w(e_1), \cdots, w(e_\mu))$, $(w(e_{\mu+1}), \cdots, w(e_m))$ を表すものとし, $\boldsymbol{w}^t$ は $\boldsymbol{w}$ の転置を, $(\boldsymbol{w}_{\bar{T}} \vdots \boldsymbol{w}_T)$ は行ベクトル $(w(e_1), \cdots, w(e_\mu), w(e_{\mu+1}), \cdots, w(e_m))$ を表すものとする (他のベクトル, 行列についても同様).

任意の閉路は基本閉路の一次結合により表されるから, 補木枝 $e_c \in \bar{T}=E-T$ により定まる基本閉路 $\Lambda(e_c)$ に対して

$$\sum_{e \in \Lambda(e_c)} \varepsilon(\Lambda(e_c),e)t(e)=0 \quad (e_c \in \bar{T})$$

が成り立つならば, t がテンションであることは明らかである. したがって **問題 2** は

$$(\boldsymbol{w}_{\bar{T}} \vdots \boldsymbol{w}_T) \geq \boldsymbol{0} \tag{3}$$

$$\begin{aligned}\sum_{e \in \Lambda(e_c)} &\varepsilon(\Lambda(e_c),e)w(e) \\ &= -\sum_{e \in \Lambda(e_c)} \varepsilon(\Lambda(e_c),e)\alpha_t(e) \quad (e_c \in \bar{T})\end{aligned} \tag{4}$$

$$(d(e_1), \cdots, d(e_m))(\boldsymbol{w}_{\bar{T}} \vdots \boldsymbol{w}_T)^t \rightarrow min \tag{5}$$

という線形計画問題に変換できる.

T の基本閉路行列を $\boldsymbol{B}_f = (\mathbf{1}_\mu \vdots \boldsymbol{B}_\tau)$ とし, $\boldsymbol{\alpha}_t \triangleq (\alpha_t(e_1), \cdots, \alpha_t(e_m))\boldsymbol{B}_f{}^t$ とおくと (4) 式は,

$$(\mathbf{1}_\mu \vdots \boldsymbol{B}_\tau)(\boldsymbol{w}_{\bar{T}} \vdots \boldsymbol{w}_T)^t = -\boldsymbol{\alpha}_t{}^t \tag{6}$$

と記述できる. さらに, (6) 式を用いて (5) 式から $\boldsymbol{w}_{\bar{T}}$ を消去し, 整理すると

$$(d(e_1), \cdots, d(e_m))(-\boldsymbol{B}_\tau{}^t \vdots \mathbf{1}_\rho)^t \boldsymbol{w}_T{}^t \rightarrow min \tag{7}$$

となるが, $\boldsymbol{C}_f = (-\boldsymbol{B}_\tau{}^t \vdots \mathbf{1}_\rho)$ は T の基本カットセット行列であるから,

$$\boldsymbol{d} \triangleq (d(e_1), \cdots, d(e_m))\boldsymbol{C}_f{}^t$$

と定義すると (7) 式は

$$\boldsymbol{d}\boldsymbol{w}_T{}^t \rightarrow min \tag{8}$$

となる.

(3), (6), (8) 式がシンプレックスタブロー表現になっていることに注意すると, この問題の主実行可能条件は $\boldsymbol{\alpha}_t \leq \boldsymbol{0}$, 双対実行可能条件は $\boldsymbol{d} \geq \boldsymbol{0}$ であることがわかる. さらに, (3), (6) 式からも $\boldsymbol{w}_{\bar{T}}$ を消去しまとめると, 結局

$$\boldsymbol{w}_T \geq \boldsymbol{0} \tag{9}$$

$$\boldsymbol{B}_\tau \boldsymbol{w}_T{}^t \leq -\boldsymbol{\alpha}_t{}^t \tag{10}$$

$$-\boldsymbol{d}\boldsymbol{w}_T{}^t \rightarrow max \tag{11}$$

という線形計画問題が得られる.

ここで, フローに関する次の最小化問題

問題 3

各枝 $e \in E$ に対して定められた単位フロー当たりのコスト $\alpha_t(e)$ に対して, 制約 $f(e) \leq d(e)$ $(e \in E)$ を満たし, $\sum_{e \in E} \alpha_t(e) f(e)$ を最小化するようなフロー f を求めよ.

を考え, **問題 2** の場合と同様に主実行可能条件, 双対実行可能条件を導く.

任意のカットセットは基本カットセットの一次結合により表されるから, 木枝 $e_t \in T$ により定まる基本カットセットを $\Gamma(e_t)$ とすると,

$$\sum_{e \in \Gamma(e_t)} \kappa(\Gamma(e_t), e) f(e) = 0 \quad (e_t \in T)$$

であることと, f がフローであるということとは同値である.

ここで $y(e) = d(e) - f(e)$ とおくと **問題 3** は,

$$(\boldsymbol{y}_{\bar{T}} \vdots \boldsymbol{y}_T) \geq \boldsymbol{0} \tag{12}$$

$$(-\boldsymbol{B}_\tau{}^t \vdots \boldsymbol{1}_\rho)(\boldsymbol{y}_{\bar{T}} \vdots \boldsymbol{y}_T)^t = \boldsymbol{d}^t \tag{13}$$

$$(\alpha_t(e_1), \cdots, \alpha_t(e_m))(\boldsymbol{y}_{\bar{T}} \vdots \boldsymbol{y}_T)^t \rightarrow max \tag{14}$$

となり, (13) 式 を用いて (14) 式 より $\boldsymbol{y}_T$ を消去することにより

$$\boldsymbol{\alpha}_t \boldsymbol{y}_{\bar{T}}{}^t \rightarrow max \tag{15}$$

を得る.

(12), (13), (15) 式 がシンプレックスタブロー表現になっていることに注意すると, この問題の主実行可能条件は $\boldsymbol{d} \geq \boldsymbol{0}$, 双対実行可能条件は $\boldsymbol{\alpha}_t \leq \boldsymbol{0}$ であることがわかる. さらに, (12), (13) 式 から $\boldsymbol{y}_T$ を消去し, まとめると, 結局

$$\boldsymbol{y}_{\bar{T}} \geq \boldsymbol{0} \tag{16}$$

$$\boldsymbol{B}_\tau{}^t \boldsymbol{y}_{\bar{T}}{}^t \geq -\boldsymbol{d}^t \tag{17}$$

$$-\boldsymbol{\alpha}_t \boldsymbol{y}_{\bar{T}}{}^t \rightarrow min \tag{18}$$

という線形計画問題が得られる.

明らかに (9), (10), (11) 式 からなる線形計画問題は (16), (17), (18) 式 からなる線形計画問題と互いに双対である. したがって, **問題 2** と **問題 3** は等価である.

4 コンパクション手法

4.1 既存アルゴリズムの適用

従来の最小コストフロー問題の解法は大きく以下の 3 つに分けることができる.

1. 主単体法 (primal-simplex method)
   まず主実行可能解を求め, それを, 主実行可能であることを保ちながら, 最適解になるように改良する.
2. 双対単体法 (dual-simplex method)
   まず双対実行可能解を求め, それを, 双対実行可能であることを保ちながら, 最適解になるように改良する.
3. 主双対法 (primal-dual method)
   与えられた問題に対し, 部分問題 (主問題) とその双対問題に対する最適解を順次求めることにより, 元の問題の最適解を求める.

[6] では, これらの手法の実行効率について詳しい評価実験を行い, 主単体法が最も効率良く, 主双対法がそれに続き, 双対単体法が最も非効率であると報告している.

3 章で述べたように, コンパクション問題と最小コストフロー問題は双対である. それらの主実行可能条件, 双対実行可能条件を比較すると, 後者における主単体法, 双対単体法は, それぞれ, 前者における双対単体法, 主単体法に相当することがわかる. したがって, コンパクション問題に対しては, 広く用いられているグラフ論的主単体法[2-4]よりも, 全く活用事例が報告されていない双対単体法の方が効率が良いものと予想される.

4.2 既存アルゴリズムの高速化

主単体法と双対単体法は枢軸変換 (pivoting) を繰り返すことにより実行される. [6] では, 特にこれら 2 つの手法に関し, 枢軸変換の回数と所要時間について詳しい評価を行っている. それによると, 最小コストフロー問題における主単体法は, 双対単体法と比べ, 枢軸変換の回数は多いが, 1 回の枢軸変換に要する時間が極めて短いために, 全体の実行時間が短い.

一方, 主双対法はフロー増分 (flow argumentation) を繰り返すことにより実行される. [6] では特に言及していないが, 著者らがその回数を実験により評価したところ, 後述するように, 双対単体法の枢軸変換の回数よりも多く, 主単体法のそれよりも少ないという結果が得られた.

主双対法や双対単体法は, 枢軸変換やフロー増分操作の回数が主単体法のそれよりも少ないため, これらの操作を高速化できれば主単体法以上に高速化できる可能性がある. 以下では, フロー増分操作の効率化による主双対法の高速化手法について考察する.

主単体法では, グラフの木を用いて基底変数を表現し, 木の初等変換により枢軸変換を行う. データ構造を工夫することにより, 一部の枝のデータのみを変更することで初等変換を行えるため, 効率良く枢軸変換を実行できる.

一方, 主双対法では, フロー増分可能経路に

フローを流す操作を繰り返すことにより解を得る．フローを流すと，その経路上のいくつかの枝が飽和し，増分不可能となり，フロー増分可能経路が変化する．そのため，フロー増分毎にフロー増分可能経路の探索を行わなければならない．一般に，探索はグラフ全体に対して行われるため，多くの時間を要する．しかし，フロー増分により飽和する枝は増分経路上に限定され，さらに，多くの場合，同時に多数の枝が飽和することはないと考えられるので，フロー増分可能経路の変化が局所的である可能性が高い．探索の結果をフロー増分木 (森) として保持しておき，フロー増分毎に，フロー増分可能経路が変化する可能性のある部分だけを再探索して，フロー増分木 (森) を再構築すれば，実行速度が劇的に向上する可能性がある．

4.3 提案手法

本文では，[9] で提案された多項式時間アルゴリズム RHS(Right Hand Side)-scaling algorithm を簡略化した主双対法 (以下，**RHS** と呼ぶ) に着目し，それに基づくコンパクションアルゴリズムを構築する．以下では，RHS の概要，および高速化手法について述べる．なお，RHS には次のような特徴がある．

1. 非多項式時間アルゴリズムであり，主双対法の中では高速である．
2. [9] にしたがって多項式時間，強多項式時間のアルゴリズムに容易に変更できる．

RHS は，次のような形式の最小化問題を扱う．

問題 4

各節点 $v \in V$ に対して定められた供給量 $b(v)$ に対して，(1) 式 の代わりに

$$\sum_{e\in\omega^+(v)} f(e) - \sum_{e\in\omega^-(v)} f(e) = b(v) \qquad (v \in V) \quad (19)$$

を満たすフロー f に注目する．ただし，$\omega^+(v)$ は v を始点とする枝の集合，$\omega^-(v)$ は v を終点とする枝の集合である．各枝 $e \in E$ に対して定められた単位フロー当たりの非負のコストを $c(e)$ とするとき，制約

$$f(e) \geq 0 \quad (e \in E) \qquad (20)$$

を満たし，$\sum_{e\in E} c(e)f(e)$ を最小化するようなフロー f を求めよ．

なお，一般に，負のコストの枝を持つ最小コストフロー問題は，[10] の手法により，全てコストが非負であるような問題に変換できる．また，**問題 3** は容易に **問題 4** の形式に変換できる．したがって，コンパクション問題は **問題 4** の形式に変換できる．

procesure RHS:
begin
1 $f(e) \leftarrow 0$ $(e \in E)$;
2 **while** 流入過多の節点がある **do begin**
3 枝のコストを距離とみなして，流入過多の節点から他の節点への最短フロー増分可能経路を探索する;
4 その結果を用いて，[10] の手法により，すべてのフロー増 (減) 分可能枝のコストが非負 (正) になるようにする;
5 流入過多の節点から流入不足の節点への最短経路を 1 つ選び，流せるだけのフローを流す;
end
end

図 1 RHS の概要

E を定義域とする実関数で，(20) 式 を満たすが (19) 式 を満たさないものを**疑似フロー** (pseudoflow) と呼ぶ．疑似フローにおいては，フローの流入量と流出量が等しくない節点 v が存在する．その値を

$$\delta(v) = b(v) - \sum_{e\in\omega^+(v)} f(e) + \sum_{e\in\omega^-(v)} f(e)$$

と定義する．また，$\delta(v) > 0$ のとき v を**流入過多** (excess)，$\delta(v) < 0$ のとき v を**流入不足** (deficit) という．

RHS の概要を 図 1 に示す．これを高速化するためのいくつかの手続きについて以下に述べる．

1. 前述のように，図 1, 3 行目 のフロー増分可能経路の再探索を局所的に行う．
2. 探索の結果，複数のフロー増分森が存在し得る場合，その後の再探索の範囲が狭くなるものを選ぶ．
3. [9] では，一つの流入過多節点から最短経路探索を行うが，本手法では，すべての流入過多節点群から最短経路探索を行う．その結果，フロー増分森に含まれる個々の木の大きさが小さくなり，再探索時の探索範囲が狭くなる．
4. [10] の手法を適用すると，多くの枝のコストが 0 になる．最短経路探索には heap を用いるが，コストが 0 の枝を例外処理することにより，heap に関する処理を高速化する．

表 1 実験結果

入力	制約グラフ		枢軸変換/フロー増分 (回)				CPU 時間 (秒)			
	節点数	枝数	Psl	Dsl	Rhs	RhsS	Psl	Dsl	Rhs	RhsS
na21x	151	425	80	246	167	335	0.04	0.02	0.02	0.21
na21y	256	828	88	407	242	436	0.11	0.06	0.04	0.29
an2or3x	400	1377	201	753	380	696	0.36	0.14	0.11	0.87
an2or3y	416	1716	246	738	384	625	0.40	0.15	0.11	1.11
dffx	872	3511	535	1739	877	1693	4.42	0.57	0.27	4.95
dffy	891	4314	464	1733	935	1475	2.59	0.50	0.45	6.24
ff1x	1270	9209	864	3077	1399	2714	17.36	1.41	1.29	15.04
ff1y	1692	8332	928	3507	1751	2927	5.68	1.28	1.03	13.76

5 実験結果

以下のプログラムを Sun SPARC station 2 上で C 言語により実現した. それらをコンパクション問題に適用し, その結果を比較する.

Psl : 主単体法[2-4]

Dsl : 双対単体法[11]

Rhs : 主双対法 (RHS)

RhsS : 多項式時間アルゴリズム[9]

表 1 に実験に用いた問題の大きさ, 枢軸変換/フロー増分の回数, および CPU 時間を示す.

まず, 非多項式時間アルゴリズムである Psl, Dsl, Rhs に関する実験結果を比較すると, 枢軸変換/フロー増分の回数は Psl が最も少なく Dsl が最も多いこと, および, CPU 時間は Psl が最も遅く Rhs が Dsl より若干速いことがわかる.

RhsS は Rhs と比較してかなり遅い. これは, (1) フロー増分回数が増えたこと, および (2) RhsS の性質上, フロー増分経路の再探索範囲が広くなることが原因である.

6 むすび

本文では, まず, コンパクション問題と最小コストフロー問題が双対であり, 最小コストフローアルゴリズムがコンパクションに適用可能であることを述べた. また, 現在コンパクションで主に用いられているグラフ論的主単体法が非効率的であることを導き, 計算機実験により確認した. さらに, 主双対法の高速化について考察し, コンパクション問題に適用した.

実験結果は, コンパクション問題に対しては双対単体法が適していること, および, 効率を追求した主双対法は双対単体法以上に高速に動作することを示している. 主双対法は, これまで双対単体法 (ネットワーク問題における主単体法) ほど高速化に関する研究が行われていないため, 今後の研究により, より一層の高速化が期待される.

参考文献

[1] M. Y. Hsueh: "Symbolic layout and compaction of integrated circuits", *ERL Memo. UCB/ERL M79/80*, Univ. of California, Berkeley (Dec. 1979).

[2] T. Yoshimura: "A graph theoretical compaction algorithm", *Proc. Intl. Symp. on Circuits and Systems*, pp. 1455–1458 (Jun. 1985).

[3] S. L. Lin and J. Allen: "Minplex - a compactor that minimizes the bounding rectangle and individual rectangles in a layout", *Proc. 23th Design Automation Conf.*, pp. 123–130 (Jun. 1986).

[4] Y. Shigehiro, T. Nagata, I. Shirakawa and T. Kambe: "Optimal layout recycling based on graph theoretic linear programming approach", *Proc. Intl. Conf. on Very Large Scale Integration*, pp. 25–34 (Sep. 1993).

[5] D. R. Fulkerson: "An out-of-kilter method for minimal-cost flow problems", *J. Society for Industrial and Applied Mathematics*, **9**, 1, pp. 18–27 (Mar. 1961).

[6] F. Glover, D. Karney and D. Klingman: "Implementation and computational comparisons of primal, dual and primal-dual computer codes for minimum cost network flow problems", *Networks*, **4**, pp. 191–212 (1974).

[7] A. I. Ali, R. V. Helgason, J. L. Kennington and H. S. Lall: "Primal simplex network codes: state-of-the-art implementation technology", *Networks*, **8**, pp. 315–339 (1978).

[8] M. Iri: "Network Flow, Transportation and Scheduling", Academic Press, New York (1969).

[9] J. B. Orlin: "A faster strongly polynomial minimum cost flow algorithm", *Proc. 20th Annual ACM Symp. on Theory of Computing*, pp. 377–387 (May 1988).

[10] J. Edmonds and R. M. Karp: "Theoretical improvements in algorithmic efficiency for network flow problems", *J. ACM*, **19**, 2, pp. 248–264 (Apr. 1972).

[11] 茨木俊秀, 福島雅夫: "FORTRAN77 最適化プログラミング", 岩波書店 (1991).