Vol.2018-ARC-230 No.35
Vol.2018-SLDM-183 No.35
Vol.2018-EMB-47 No.35
2018/3/8

逆方向カット・エッジのない最小カットを求めるアルゴリズムの改良

神保 潮[1,a)]　五島 正裕[2]

概要：FF を用いた回路をラッチを用いた回路に変換する問題は，最小カット問題の一種に帰着する．ただしこの際，始点から終点に至るすべての道にカット・エッジをただ 1 つ含むという制約がある．そこで我々は，逆方向カット・エッジのない最小カットを見つけるアルゴリズムを提案した．これは容量 ∞ の逆平行エッジを追加したうえで，既存の最大フロー・アルゴリズムを適用するというものであった．しかし，最大フロー・アルゴリズムが ∞ の容量を扱えるかどうかは自明ではない．本稿では，この容量を有限の値に変更する改良を提案する．これにより，一般の最大フロー・アルゴリズムの利用が可能となる．

キーワード：グラフ・アルゴリズム，最小カット，最大フロー最小カット定理

1. はじめに

同期回路における同期動作を実現する方式には，フリップ・フロップ (**FF**) を用いたものの他に，二相の**ラッチ**を用いたものがある．前者に比べて後者は，タイミング制約が緩いという利点がある [1–4] が，設計はより煩雑である．そこで，前者を入力として，後者を自動的に生成することが考えられる．

逆相ラッチ挿入問題

そのためには，**図 1**（上）から（中）に示すように，

(1) FF をラッチに変更する と同時に，

(2) FF と次の FF とに挟まれたロジックの中央に**逆相の**ラッチを挿入する

という変換を行えばよい．ただし逆相ラッチを挿入する際には，以下の制約がある：ロジック内のあらゆるパスに対して，逆相ラッチはただ 1 つ挿入されなければならない．

また，逆相ラッチの挿入位置に関しては，以下の 2 つの評価基準がある：

(1) ラッチの挿入個数は少ないほどよい．

(2) クリティカル・パスを短縮するため，挿入位置はロジック内の各パスの遅延を等分することが望ましい．

グラフ・カット

これらの評価基準に対して最適な挿入位置を求めるにあたって，回路は，図 1（下）に示すようなグラフに写像することができる：

- ロジックの，FF やゲートなどのインスタンスを頂点，ネットをエッジとする．
  インスタンスとネットには，信号の流れる向きがあるから，グラフ（エッジ）は有向となる．
- エッジのコストは，ステージ内の各パスの中央ほど低く，両端ほど高く設定する．

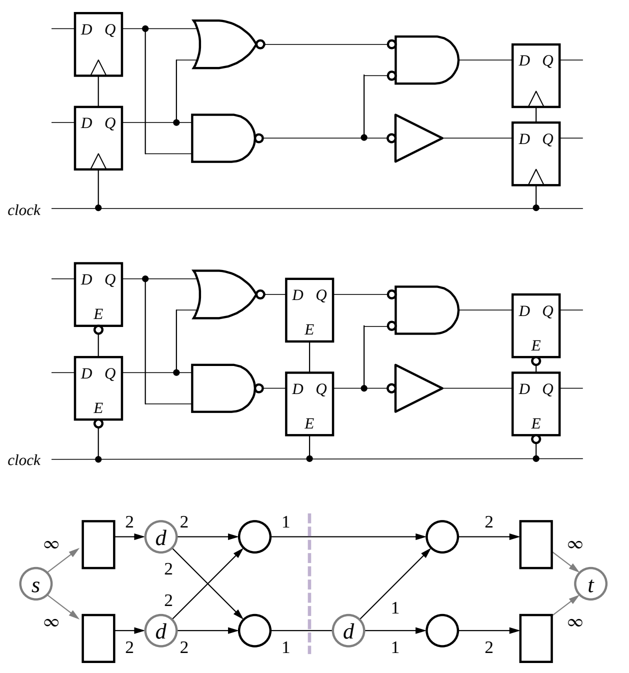


図 1　FF（上），二相ラッチを用いた回路（中）と，そのグラフ（下）
Fig. 1　Circuits with FFs (upper), with two-phase latches (middle), and their graph (lower).

[1] 総合研究大学院大学 複合科学研究科
School of Multidisciplinary Sciences, SOKENDAI
[2] 国立情報学研究所 アーキテクチャ科学研究系
Systems Architecture Research Division, NII
[a)] ushio@nii.ac.jp

また，以下のようなダミーの頂点とエッジを追加する：

- 入/出力側の FF の前/後に，始点 s/終点 t を追加する．s/t と FF を結ぶエッジのコストは ∞ とし，それらがアルゴリズムに影響を与えることのないようにする．
- ネットの分岐にダミーの頂点を挿入する（図中 d）．

するとグラフは，ラッチが挿入されたエッジにおいて，s 側と t 側に二分される．このような二分割をグラフの**カット**という．二つの分割にまたがるエッジ（この問題ではラッチが挿入されるエッジ）を**カット・エッジ**という．カット・エッジのコストの総和をカットの**サイズ**という．すると問題は，サイズ最小のカットを求める**最小カット**の問題（の一種）に帰着される．同図の例では，破線で示すカットが，サイズ $1 + 1 = 2$ で最小である．

なお，図中下側のカット・エッジのように，ネットの分岐に対して挿入したダミー d に対しては，その入力側を選ぶことで，複数の出力先をまとめて 1 個のラッチを挿入することを表現することができる．

単一カット・エッジ制約

逆相ラッチ挿入問題では，前述したように，ラッチはロジック内のあらゆるパスにただ 1 つ挿入されなければならない．この制約は，グラフの言葉では以下のようになる：

単一カット・エッジ制約　始点 s から終点 t へ至るあらゆる道上にカット・エッジが 1 つ存在する

逆相ラッチ挿入問題では，この制約を満たすカットは必ず存在する；すなわち，入力側ラッチの直後，もしくは，出力側ラッチの直前で分割したものである．特に前者の場合，逆相ラッチを挿入した回路は元の FF の回路と等価となる．

既存のアルゴリズム

既存の最小カット・アルゴリズムは，道上のカットの数を意識せず，この問題にそのまま用いることはできない．実際，次章の例に見られるように，単一カット・エッジ制約を満たさないカットが選ばれることがある．

そこで我々は，この単一カット・エッジ制約を満たすカットのうちでサイズ最小となるものを見つけるアルゴリズムを提案した [5]．このアルゴリズムは，すべてのエッジに対して容量 ∞ の逆平行エッジを追加したうえで，既存の最大フロー・アルゴリズムを適用するというものである．

しかし, 既存の最大フロー・アルゴリズムが容量 ∞ のエッジを扱えるかどうかは自明ではない．[5] では，最も実用的なアルゴリズムの 1 つであるエドモンズ・カープのアルゴリズムが容量 ∞ のエッジを扱えることを証明した．しかし，論文にとって必ずしも本質的ではないその証明が紙面の少なくない部分を占めることとなった．さらに，他のアルゴリズムが容量 ∞ のエッジを扱えるかどうかはなお自明ではない．

本稿の提案

そこで本稿では，この容量を有限の値へと変更する．有限の容量を持つ逆平行エッジを付け加えたところで何ら特殊性は生じないため，一般の最大フロー・アルゴリズムを証明なしに用いることができる．

以下，本稿では，2 章で既存のアルゴリズムについてまとめた後，3 章で提案のアルゴリズムについて詳しく述べる．本稿は，基本的には，[5] を参照しなくてもよいように構成してある．ただし，本稿における変更は実行性能にほとんど影響を及ぼさないため，実行性能の評価については [5] を参照されたい．

2. 最大フロー・アルゴリズム

最小カットを求めるには，グラフをフロー・ネットワークとみなし，その最大フローを求めることが一般的である．最大フロー最小カット定理によって，最小カットは最大フローと同時に求めることができる．フロー・ネットワークにおいては，エッジに与える重みはフローの**容量**と考えるが，前章におけるコストを容量と読み替えればよい．

本章では，そのような最大フロー・アルゴリズムの例として，最も基本的なフォード・ファルカーソンのアルゴリズムを概説する．

2.1 フォード・ファルカーソンのアルゴリズム

フォード・ファルカーソンのアルゴリズム (Ford-Fulkerson algorithm) [6] では，元のフロー・ネットワークに対して**残余ネットワーク** (residual network) というネットワークを生成する．

残余ネットワーク

容量 $c(u, v)$ のエッジ $u \to v$ にフロー $f(u, v)$ を流したとき，残余ネットワークにおいては，u，v 間に順方向と逆方向の 2 つの有向エッジを張る：

順方向　まだあと $c_f(u, v) = c(u, v) - f(u, v)$ だけ流せるという意味で，容量 $c_f(u, v)$ の順方向エッジ

逆方向　逆に，$c_b(u, v) = f(u, v)$ だけ減らすことができるという意味で，容量 $c_b(u, v)$ の逆方向エッジ

なお，$c_f(u, v) + c_b(u, v) = c(u, v) - f(u, v) + f(u, v) = c(u, v)$ である．

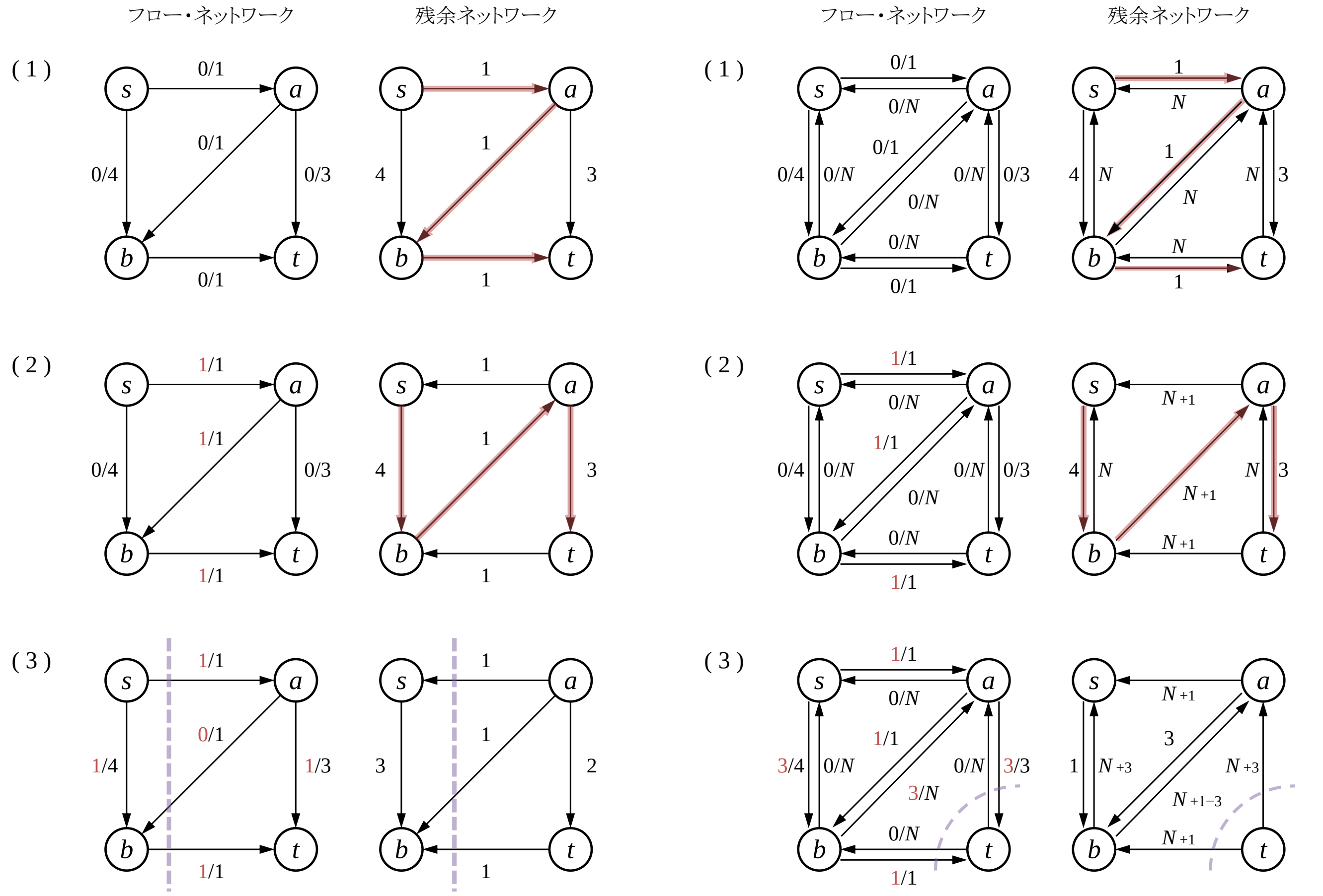
増加道

残余ネットワークにおいて s から t へと至る道を**増加道** (augmenting path) という．s から t へのフローは，増加道の容量の最小値（すなわち，増加道を形成するすべてのエッジの容量のうちの最小値）だけ，増加させることができる．

増加道に上述した逆方向エッジが含まれる場合には，そのエッジのフローを逆に減少させることになる．このおかげで，増加道をグリーディに見つけて行っても最大フローが求まるというのがフォード・ファルカーソンのアルゴリズムの要諦である．証明は，[7] などを参照されたい．

アルゴリズムの動作例

図 2（左）の例を用いてフォード・ファルカーソンのアルゴリズムの動作を説明する．同図中，（1）～（3）の各段階において，左がフロー・ネットワーク，右がそのフロー・ネットワークに対応する残余ネットワークを示す．一般に，フロー・ネッ



（フロー）/（容量）

図 2 オリジナル（左）と提案（右）のフォード・ファルカーソンのアルゴリズムの動作例

Fig. 2 Behavior of Ford-Fulkerson algorithm in original (left) and proposal (right).

トワークにおいては，エッジに「(フロー)/(容量)」と付す．また，残余ネットワークにおいては，容量 0 となったエッジは除去する [*1]．

アルゴリズムは，以下のように進む：

(1) 初期状態では，フローは 0 とする．
したがって右の残余ネットワークは，左のフロー・ネットワークの容量をそのまま写したものとなる．
この残余ネットワークにおいて，s から t へ至る増加道を探す．辞書順だと，$s \to a \to b \to t$ が見つかる（太矢印）．

(2) この増加道に最大のフローを流す．この増加道を形成するエッジ $s \to a$，$a \to b$，$b \to t$ の容量はすべて 1 であるから，それぞれに容量一杯のフロー 1 を流すことになる．すると，左のフロー・ネットワークが得られる．
更にこのフロー・ネットワークから右の残余ネットワークを得る．エッジ $s \to a$，$a \to b$，$b \to t$ のそれぞれに容量一杯のフロー 1 を流したため，残余はそれぞれ 0 となる．これらのエッジに対しては逆に，フローを 1 だけ減らすことができるという意味で，容量 1 の逆向きのエッジを張る．
この残余ネットワークにおいては，増加道 $s \to b \to a \to t$ が見つかる（同じく太矢印）．

(3) この増加道に最大容量である 1 のフローを流すと，左のフロー・ネットワークが得られる．エッジ $a \to b$ には，先ほど 1 のフローを流したが，同じく 1 のフローを今度は逆向きに流したので，相殺されてフローは 0 に戻る．
更にこれから右の残余ネットワークが得られる．この残余ネットワークにおいては，s から t へ至る増加道はもはや見つからないので，アルゴリズムは終了する．

最大フロー

増加道 $s \to a \to b \to t$ と $s \to b \to a \to t$（図 2 中の太矢印 2 本）によって 1 ずつ増加されたので，最大フローは合計 2 となる．

物理的なフローは，これらの 2 つの増加道の重ね合わせである；すなわち，道 $s \to a \to t$ と $s \to b \to t$ に 1 ずつ流されている．相殺されるので，エッジ $a \to b$ にはフローは流されない（0 である）．

*1 主に図の見やすさのため．要は容量 0 のエッジを含む増加道を見つけないようにすればよいので，プログラムでは容量 0 のエッジとして残しておいた方が実装が容易であろう．

Vol.2018-ARC-230 No.35
Vol.2018-SLDM-183 No.35
Vol.2018-EMB-47 No.35
2018/3/8

2.2 最大フロー最小カット定理

フォード·ファルカーソンのアルゴリズムにおいては，最小カットは，アルゴリズム終了時の残余ネットワークにおいて，s から到達可能な頂点とそれ以外の頂点への分割として与えられる．図 2 の場合，終了時の残余ネットワーク，すなわち，(3) 右において，s から到達可能であるのは b のみであるので，最小カットは $\{s, b\}$ と $\{a, t\}$ である．

2 つの部分をまたがるカット·エッジのうち，$s \to a$ と $b \to t$ は，s 側から t 側へ向かう**順方向カット·エッジ**であり；$a \to b$ は逆に，t 側から s 側へ向かう**逆方向カット·エッジ**である．

最大フロー最小カット定理によれば，順方向カット·エッジのフローはそれぞれの容量一杯であり，同時に，逆方向カット·エッジのフローはすべて 0 であるとき，最大フローは達成される．上述した最大フロー 2 に対する最小カットは，順方向カット·エッジの容量の総和，すなわち，$c(s, a) + c(b, t) = 1 + 1 = 2$ で与えられ；逆方向カット·エッジの容量，すなわち，$c(a, b) = 1$ は含めない．

この最小カットでは，道 $s \to a \to b \to t$ を構成する 3 つのエッジ $s \to a$，$a \to b$，$b \to t$ はすべてカット·エッジとなっており，単一カット·エッジ制約は満たされていない．

3. 提案アルゴリズム

本稿で求めるべきカットには，始点から終点へ至るすべての道にカット·エッジがただ 1 つ現れるという単一カット·エッジ制約がある．実はこの制約は，逆方向カット·エッジがないという逆方向カット·エッジなし制約と等価である．そこで本稿では，逆方向カット·エッジを含まない最小カットを求めるアルゴリズムを考える．まず，3.1 節で，単一カット·エッジ制約と逆方向カット·エッジなし制約が等価であることを証明する．その後，3.2 節と 3.3 節で，アルゴリズムの手順と動作例を示す．アルゴリズムの正しさの証明は，改めて 3.4 節で行う．最後に 3.5 節で，提案アルゴリズムの計算量は採用した最大フロー·アルゴリズムより増加しないことを述べる．

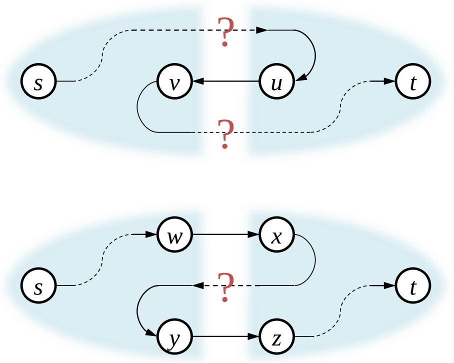


図 3 逆方向カット·エッジを含むカット
Fig. 3 Cut with a backward cut edge.

3.1 逆方向カット·エッジなし制約

定理 1. フロー·ネットワークにおいて，以下は等価である：

単一カット·エッジ制約 始点 s から終点 t へ至るすべての道にカット·エッジがただ 1 つ現れる．

逆方向カット·エッジなし制約 逆方向カット·エッジがない，すなわち，カット·エッジはすべて順方向である．

証明. 単一カット·エッジ制約 ⇒ 逆方向カット·エッジなし制約を，背理法で証明する．s から t へ至る道にカット·エッジがただ 1 つ現れ，かつ，それが逆方向であると仮定する．その逆方向カット·エッジを $u \to v$ とする（**図 3** 上）．すると，s から u，v から t へ至る道は（順方向の）カット·エッジを含むことになり，ただ 1 つという仮定と矛盾する．

逆も，同じく背理法で証明する．カット·エッジはすべて順方向エッジであり，かつ，s から t へ至る道のうち，カット·エッジが 2 つ以上現れるものがあると仮定する．2 つの順方向カット·エッジを $w \to x$ と $y \to z$ とする（図 3 下）．すると，道 $x \to \cdots \to y$ は，逆方向カット·エッジを含むことになり，カット·エッジがすべて順方向であるという仮定と矛盾する． □

したがって，1 章の問題などのために，始点から終点へ至るすべての道にカット·エッジがただ 1 つ含まれるようなカットを求めるためには，逆方向カット·エッジのないカットを求めればよい．

3.2 提案アルゴリズムの手順

提案の逆方向カット·エッジを含まない最小カット·アルゴリズムは，以下のとおりである：

(1) **前処理** 元のフロー·ネットワークの全てのエッジに対して，容量 N の逆平行エッジを追加する．

(2) **最大フロー** このフロー·ネットワークに対して，任意の最大フロー·アルゴリズムによって最小カットを求める．

N は，十分に大きい値，具体的には，元のフロー·ネットワークにおけるカット·サイズの最大値より大きい値であればよい．そしてこの最大値は，元のフロー·ネットワークにおけるすべてのエッジの容量の総和で抑えられる．

証明は 3.4 節で行うが，直感的には，このアルゴリズムは以下のようにして逆方向カット·エッジを避ける：あるエッジを逆方向カット·エッジとして選ぼうとすると（元のエッジではなく）追加された逆平行エッジの容量 N がカット·サイズに加算され，最小カットとして選ばれない十分に大きな値になる．

3.3 動作例

本節では主に，前節における (2)，すなわち，最大フローを求める部分の動作を説明する．

動作例

図 2 (右) に，提案アルゴリズムの動作を示す．元のフロー·

ネットワークは同図（左）のものと同一である．同図（右）では，すべてのエッジに対して容量 N の逆平行エッジが追加されている．元のフロー・ネットワークのすべてのエッジの容量の総和は 10 であるので，$N = 11$ とした．

最大フロー・アルゴリズムとしては，同図（左）と同じフォード・ファルカーソンのアルゴリズムを用いている．したがって，同図（左）/（右）では，同一の最大フロー・アルゴリズムが異なる初期フロー・ネットワークに対してどのように振る舞うかを見ることになる．アルゴリズムは以下のように進むが，実際，(2) までは，同図（左）と変わらない：

(1) 増加道 $s \to a \to b \to t$ が見つかる（太矢印）．

(2) この増加道に最大容量である 1 のフローを流すと，フロー/残余ネットワークが得られる．この残余ネットワークにおいては，増加道 $s \to b \to a \to t$ が見つかる（太矢印）．

(3) ただし，この増加道のフローは（左）とは異なる．この増加道のフローは，(左) では $b \to a$ の 1 によって制限されていた．(右) では，容量 $N+1$ の逆平行エッジ $b \to a$ が追加されたため，$b \to a$ ではなく，$a \to t$ の 3 によって制限されることになる．

この結果，$a \to t$ は飽和し，残余ネットワークにおいて増加道はもはや見つからず，アルゴリズムは終了する．

最大フロー最小カット

最大フローは $1 + 3 = 4$ となり，元のフロー・ネットワークにおける最大フロー 2 より大きい．これは，元のフロー・ネットワークに存在しない容量 N の逆平行エッジ $b \to a$ に 2 のフローを流すことによって達成されていることに注意する必要がある．

2.1 節で述べたとおり，フォード・ファルカーソンのアルゴリズムでは，最小カットはアルゴリズム終了時の残余ネットワークにおいて s から到達可能な頂点とそれ以外の頂点への分割として与えられる．同図の場合，s から b，そして a へ到達可能であるので，最小カットは $\{s, a, b\}$ と $\{t\}$ となる．カット・エッジは，$a \to t$ と $b \to t$ であり，逆方向エッジは含まれない．また，s から t へ至るあらゆる道上でカット・エッジは 1 つである．

最小カットは，最大フローと同じ 4 である．これは，次節で証明するように，元のフロー・ネットワークにおいて逆方向カット・エッジを含まないカットのうちで最小のものである．

3.4 提案アルゴリズムの正しさと停止性

定理 2. 提案アルゴリズムによって得られるカットは，元のフロー・ネットワークのカットの中で，逆方向カット・エッジなし制約（単一カット・エッジ制約でも等価）を満たすものがあれば，それらのうちでサイズ最小のものである．

証明. **図 4** の上の列は，それぞれ，あるフロー・ネットワークのすべてのカットを，それらのカット・サイズに従って昇順に並べたものである．図中のアイコンは，楕円は元のフロー・ネットワークを，矢印はエッジを，破線はカットを，それぞれ模式的に表している．特に，左向き矢印が破線に重なっている場合，この矢印は逆方向カット・エッジとなっている．

これらのカットに対して，仮に，提案手法の前処理を施した，すなわち，すべてのエッジに容量 N の逆方向エッジを追加した場合のカット・サイズを求めよう．下の列は，上の列のカットを，この新たに求めたカット・サイズの昇順に並べ直したものである．同じカットを異なるサイズに従って並べ直しただけであるから，上/下の列のカット間には，1 対 1 の関係がある．上下を結ぶ矢印は，この 1 対 1 の関係を示している．

これらのカットの中には，元のフロー・ネットワークにおいてカット・エッジに逆方向エッジを含むものと含まないものがある．その結果，下では以下のような順序の変化がある：

逆方向カット・エッジを含むカット 前処理を施した場合には，追加された逆平行エッジの容量 N が加算される．N は，カット・サイズの最大値より大きい値としたことに注意されたい．したがって，N が加算されたカットは，下の列では最後尾に移動することになる．

逆方向カット・エッジを含まないカット 前処理を施した場合でも，追加された逆平行エッジの N の容量が加算されないため，カット・サイズは変化しない．その結果，下の列でもその位置に残されている．

前処理後に任意の最大フロー・アルゴリズムを適用する．

まず，その最大フロー・アルゴリズムの停止性が保証されているなら，提案アルゴリズム全体でも停止することは明らかである．

そして，その最大フロー・アルゴリズムが最小カットを見つけられるならば，図中，下の列において一番左，○を付けたカットが選ばれる．

ここで，逆方向カット・エッジを含まないカットのみに注目しよう．前処理を施した場合でもカット・サイズは変化しないため，それらの間の順序は上下の列の間で変化しないことが分かる（上下を結ぶ直線の矢印）．

したがって，提案アルゴリズムで得られた最小カットは，上の列においては，左から二番目，同じく○を付けて示さ

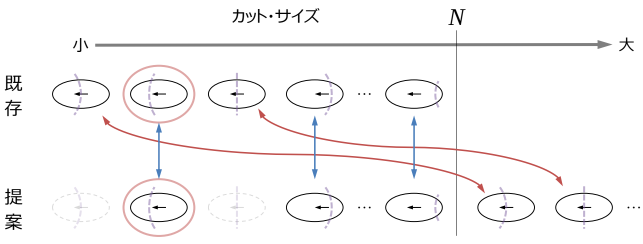


図 4 既存（上）/提案（下）によって計算されたサイズによるカットの昇順列

Fig. 4 Sequences of cuts in ascending order of their sizes computed by existing (upper)/proposed (lower) algorithms.

Vol.2018-ARC-230 No.35
Vol.2018-SLDM-183 No.35
Vol.2018-EMB-47 No.35
2018/3/8

れたカットと全く同一のものである．これは，元のフロー・ネットワークのカットの中で逆方向カット・エッジを含まないもののうち，サイズ最小のものである． □

3.5 逆平行エッジ追加による計算量の変化

最大フロー・アルゴリズムは数多く存在する [8] が，[5] の実験では，より実用的な**エドモンズ・カープのアルゴリズム** (Edmonds-Karp algorithm) [9] を用いた．エドモンズ・カープのアルゴリズムの実行時間は，頂点数を V，エッジ数を E とすると，$O(VE^2)$ である [7].

エドモンズ・カープのアルゴリズムを用いた場合，提案では逆方向エッジを追加する前処理によってエッジ数 E は 2 倍になるので，実行時間は $2^2 = 4$ 倍になるように思われるが，実際にはそうではない．[5] の評価では，1.5 倍程度にとどまっている．それは，以下の理由による．

エドモンズ・カープのアルゴリズムをはじめ，フォード・ファルカーソンのアルゴリズムをベースとする最大フロー・アルゴリズムでは，残余ネットワークの逆平行エッジも平行エッジと区別なく扱われる．$O(VE^2)$ という実行時間は，この逆平行エッジを考慮に入れたものである．

提案における逆平行エッジは，通常なら増加操作によって随時追加されるものを，初期状態から追加するに過ぎない．したがって，残余ネットワークを用いる最大フロー・アルゴリズムを用いた場合，提案手法の実行時間はオーダー上は悪化しない．

特にプログラムにおいては，このような逆平行エッジは，必要に応じて追加/削除するのではなく，予めすべてのエッジに逆平行エッジを追加したうえで，その容量を 0 に初期化することによって実現することになろう．この場合，提案手法における前処理とは，逆平行エッジを追加するのではなく，この初期値を 0 から N に変更するに過ぎない．実際，[5] のプログラムでも，そのように実装されている [10].

3.4 節で述べたように，提案手法ではよりサイズの大きいカットを探すことになる．そのためには，より多くの増加道を探索する必要がある．実際の実行時間の差は，このために生じると考えてよい．すなわち，ベースの最大フロー・アルゴリズムと提案手法の実行時間の差は，オーダー上のものではなく，トポロジ的には同一のフロー・ネットワークにおける容量の違いによるものである．

4. おわりに

FF を用いた回路からラッチを用いた回路に変換する際などには，始点から終点に至るすべての道にカット・エッジを 1 つ含むという制約を満たすカットを見つける必要がある．我々は，この単一カット・エッジ制約を満たすカットのうちでサイズ最小となるものを見つけるアルゴリズムを提案した [5].

このアルゴリズムは，すべてのエッジに対して容量 ∞ の逆平行エッジを追加したうえで，既存の最大フロー・アルゴリズムを適用するというものであった．しかし，既存の最大フロー・アルゴリズムが容量 ∞ のエッジを扱えるかどうかは必ずしも自明ではない．

そこで本稿では，この容量を有限の値 N へと変更できることを示した．容量 N の逆平行エッジを付け加えたところで何ら特殊性は生じないので，一般の最大フロー・アルゴリズムを採用することができる．

謝辞 本研究の一部は，文部科学省科学研究費補助金 No. 16H02797 による．

参考文献

[1] Harris, D.: *Skew-Tolerant Circuit Design*, Morgan Kaufmann Publishers (2001).
[2] Jimbo, U., Yamada, J., Shioya, R. and Goshima, M.: Applying Razor Flip-Flops to SRAM Read Circuits, *IEICE Trans. Electron.*, Vol. E100-C, No. 3, pp. 245–258 (online), DOI: 10.1587/transele.E100.C.245 (2017).
[3] 神保 潮，山田淳二，五島正裕: 動的タイム・ボローイングを可能にするクロッキング方式の適用 (2017). *cross-disciplinary Workshop on Computing Systems, Infrastructures, and Programming* (*xSIG 2017*) に採択.
[4] 神保 潮，山田淳二，五島正裕: 動的タイム・ボローイングを可能にするクロッキング方式の適用，情報処理学会論文誌：コンピューティングシステム， Vol. 10, No. 2, pp. 1 – 12（オンライン），入手先 ⟨http://id.nii.ac.jp/1001/00183237/⟩ (2017).
[5] 神保 潮，五島正裕: 逆方向カット・エッジのない最小カットを求めるアルゴリズム，情報処理学会論文誌：コンピューティングシステム，Vol. 10, No. 4 (2017).（採録決定）.
[6] Ford, L. R. and Fulkerson, D. R.: Maximal flow through a network, *Canadian Journal of Mathematics*, pp. 399 – 404 (online), DOI: 10.4153/CJM-1956-045-5 (1956).
[7] Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C.: *Introduction to Algorithms*, The MIT Press (2009).
[8] Goldberg, A. V. and Tarjan, R. E.: A new approach to the maximum-flow problem, *Journal of the ACM*, Vol. 35, No. 4, pp. 921 – 940 (online), DOI: 10.1145/48014.61051 (1988).
[9] Edmonds, J. and Karp, R. M.: Theoretical improvements in algorithmic efficiency for network flow problems, *Journal of the ACM*, Vol. 19, No. 2, pp. 248 – 264 (1972).
[10] Sharaiha, E.: Edmonds Karp in C# (online), available from ⟨http://gist.github.com/Eyas/7520781⟩.