アルゴリズム 43－8
（1995．1．23）

外平面グラフ上の最大流量を求める並列アルゴリズム

中山 慎一　　増山 繁

豊橋技術科学大学知識情報工学系

概要　本論文では，外平面グラフ G における最大流問題を CRCW PRAM 上で，$O(n^2/\log n)$ 個のプロセッサを用いて $O(\log n)$ 時間で解く並列アルゴリズムを提案する．但し，n は G の節点数とする．外平面グラフは直並列グラフの部分クラスであり，もし外平面グラフの始点，終点が直並列グラフの両端点に一致するならば He, Yesha の並列アルゴリズム [4] を利用して外平面グラフ上の最大流量を求めることができるが，一致しない場合にはそのアルゴリズムでは求めることができない．それに対し，本並列アルゴリズムは，任意の始点，終点の組に対して外平面グラフ上の最大流量を求めることができる．

A Parallel Algorithm for Finding The Maximum Flow in Outerplanar Graphs.

Shin-ichi Nakayama　Shigeru Masuyama

Department of Knowledge-Based Information Engineering,
Toyohashi University of Technology

Abstract　This paper proposes a parallel algorithm for solving the maximum flow problem in outerplanar graphs which runs in $O(\log n)$ time using $O(n^2/\log n)$ processors on a CRCW PRAM, where n is the number of vertices in a graph G. If a source and a sink of an outerplanar graph correspond to the terminal vertices of a series-parallel graph, the maximum flow in the outerplanar graph can be obtained by He, Yesha's parallel algorithm[4]. However if a source and a sink do not correspond to the terminal vertices as a series-parallel graph, their algorithm cannot be applied. On the other hand, our parallel algorithm gives the maximum flow for any source-sink pair in an outerplanar graph.

1 まえがき

$G=(V,E)$ は節点の集合 V, 辺の集合 E からなるグラフとし, 相異なる 2 節点 s,t がそれぞれ始点, 終点としてあらかじめ指定され, かつ, 各辺 (i,j) は, その上を流れる流量の上限値を表す容量 $c(i,j)$ を有しているとする. 最大流量問題とは, s から t へ「もの」を流すとき, これらの流れの中でその流量の値が最大のものを求める問題であり, 今までに数々の研究がなされてきた [9]. この問題を並列アルゴリズムの観点からながめると, 一般のグラフに関して, 最大流量問題は P 完全であることが既に知られている [8]. また, 平面グラフに関しては, G の節点数を n とすると, CREW PRAM[3][8] 上で, $O(n^4)$ 個のプロセッサを用いて $O(\log^3 n)$ 時間 (または, $O(n^6)$ 個のプロセッサを用いて $O(\log^2 n)$ 時間) で解を求める並列アルゴリズム [7] が知られているが, 非常に多くのプロセッサを必要とする.

アルゴリズム設計において, 一般のグラフに対し計算困難な問題, 又は, 計算量が非常に大きい問題であっても, ある特定のクラスに属するグラフ上ならば計算可能であったり, 非常に速く解けたりすることがよくある. 様々な問題に対し, これまでにもいくつかの興味あるクラスに限定した問題を解く効率の良いアルゴリズムが数々開発されてきた [9]. そのようなクラスの 1 つに外平面グラフがある [6]. 外平面グラフとは, 全ての節点を無限面 (グラフを平面に埋め込んだ時, グラフの外側の無限に広がった領域) の境界上に乗るように平面埋め込み可能なグラフである (図 1(a) 参照). 外平面グラフは st-直並列グラフの部分クラスなので, 始点, 終点の組が st- 直並列グラフの両端点 s, t にそれぞれ一致するなら, He, Yesha[4] の木の縮約を利用した並列アルゴリズムで最大流量を求めることができるが, 一致しない場合には, 最大流量を求めることが出来ない.

本論文では, CRCW PRAM 上で, $O(n^2/\log n)$ 個のプロセッサを用いて $O(\log n)$ 時間で, 外平面グラフ G 上の任意に指定した 2 節点 s,t 間の最大流量問題を求める並列アルゴリズムを提案する. なお, 外平面グラフを直並列グラフとみたとき, 始点, 終点の組が直並列グラフの両端点にそれぞれ一致するならば, He, Yesha[4] の並列アルゴリズムを用いて最大流量を求められるが, 我々の並列アルゴリズムは, 始点, 終点の組を直並列グラフの両端点に限定せず任意に選択して解くことができるので, [4] を利用して求める場合よりも適用範囲が広い.

2 準備

ネットワーク $N=(G=(V,E),s,t,c)$ を次のように定義する. (本論文で取り扱うグラフ G は無向グラフであるが, まず, グラフ G が有向グラフの場合について定義する.) G は, 多重辺を持たない連結有向グラフであり, 節点数 $|V|$ を n, 辺数 $|E|$ を m で示す. s と t $(s \neq t)$ は V に属する特別な節点で, それぞれ, 始点, 終点と呼ばれる. 各辺 $(i,j) \in E$ には, その上を流れる流量の上限値を表す容量 $c(i,j) \geq 0$ を有しているとする. この時, 次の性質をもつ各節点対 $u,v \in V$ に対して定義される実数値関数を流量関数 $f(u,v)$ という.

(i) 容量制約: $0 \leq f(u,v) \leq c(u,v)$,

(ii) 流量保存則: s, t 以外の全ての節点 v において, v に入ってくる流量と v から出ていく流量が等しい, つまり, $v \neq s,t$ なる各 $v \in V$ について $\sum_{u:(u,v)\in E} f(u,v) = \sum_{u:(v,u)\in E} f(v,u)$ が成り立つ.

最大流量問題は, それぞれの辺の流量関数 $f(u,v)$ が, 容量制約と流量保存則を共に満たすという条件のもとで, 始点 s から終点 t への流量値を最大にするものを求める問題である.

本論文で取り扱うグラフ G は無向グラフである. 無向グラフの場合には, 無向辺 (u,v) を, それと同一の容量をもつ互いに逆方向の 2 つの有向辺 (u,v), (v,u) に置き換えることによって有向グラフの場合と同じように定義できる. この場合に, 無向グラフにおける任意の辺 (u,v) の流量 f' は, 有向グラフの流量関数 $f(u,v)$ を用いて次のように計算できる.

$f'(u,v) = \max\{0, f(u,v) - f(v,u)\}$,

$f'(v,u) = \max\{0, f(v,u) - f(u,v)\}$

以下で紹介するアルゴリズムは, 必ずしも, 実際に無向グラフのすべての辺を 2 本の互いに逆方向の有向辺で置き換えるわけではないが, 求める最大流は上記の意味においてのものである.

系列 v_0, e_1, v_1, e_2, v_2, ..., e_p, v_p は, $v_i \in V$, $i=0,1,\ldots,p$, および, $e_i = \{v_{i-1}, v_i\} \in E$, $i=1,2,\ldots,p$, の条件を満たすとき, v_0 から v_p への路であるといい, $v_0 = v_p$ の場合, 閉路という.

G より $v \in V$ を取り除くと非連結になるようなカット節点 v が存在しない, すなわち, 2 連結な外平面グラフ G は, 全ての節点を 1 度だけ通る閉路 H を無限面との境界として埋め込み, 閉路 H を構成しない残りの辺を互いに交差しないように閉

路 H の内側に埋め込むような平面埋め込みを持つ [2]. 無限面との境界をなす辺のことを周辺と呼び, 他の辺を対角辺と呼ぶ.

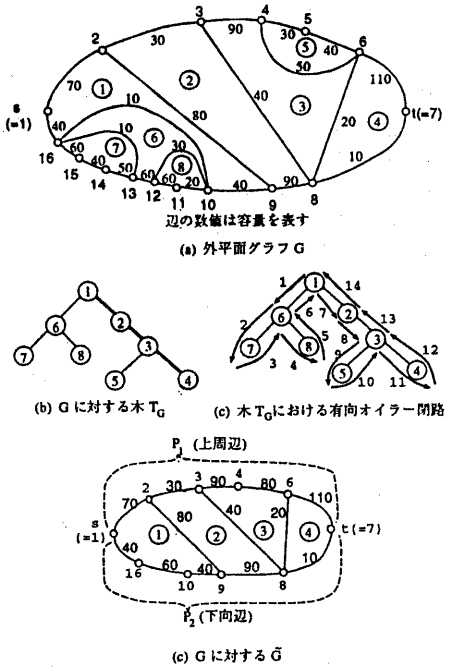

図 1: 外平面グラフ G, および, G に対する木 T_G, グラフ $\tilde{G}$

3 外平面グラフの最大流量を求める並列アルゴリズム

ここでは, 入力として与えられる外平面グラフ G は 2 連結グラフであると仮定する. 紙面の都合上, 入力として与えられるグラフ G が 2 連結でない連結外平面グラフの場合の詳細については省略するが, G の 2 連結成分を求め, 各連結成分に対し, 以下で述べる並列アルゴリズムを適用することにより, 容易に最大流量を求めることができる. はじめにアルゴリズムの概略を述べる.

まず, 与えられた外平面グラフ G の周辺を求める. 外平面グラフは, 全ての節点が無限面の境界上に乗るように平面埋め込み可能なグラフなので, 2 連結な外平面グラフには必ずハミルトン閉路が存在する. そのハミルトン閉路を構成する辺が周辺になる. 周辺の求め方は文献 [2] による. また, 始点 s を節点番号 1 として, 周辺 (ハミルトン閉路) に沿って節点番号 $1,\cdots,n$ を付ける.

次に, 文献 [2] のアイディアを利用して, G に対応する有向グラフ G' における, 周辺に付けられた節点番号に基づいたいくつかの有向閉路を構成する (この各閉路は, G における各面に対応する). 但し, G に対応する有向グラフ $G' = (V', E')$ とは, ハミルトン閉路 C を構成している辺 $(i, i+1)$, $i = 1,\cdots,n-1$, および, $(n,1)$ を, それぞれ, i から $i+1$, $i = 1,\cdots,n-1$, n から 1 への有向辺に置き換え, 残りの各対角辺 (j,k) を, それぞれ, 2 つの逆向きの並列有向辺 (j,k), (k,j) に置き換えることによって構成されるグラフである. G' は, 各節点で入次数と出次数が等しい強連結グラフなので, 有向オイラーグラフであり, 各節点への入力辺と出力辺をそれぞれ 1 対 1 に対応させることにより, いくつかの有向閉路が得られる [2][3]. 文献 [2] では, 閉路の構成法, いわゆる, 節点 v での入力辺に続く出力辺の決定を次のように行なっている. いま, C が無限面の境界となるように置かれ, 節点番号 $1,\cdots,n$ の節点が時計回りに並んでおり, その他の交差辺が C の内面 (無限面でない方の面) に置かれていると考え, v からみて (i,v) から反時計回りに最初に出現した出力辺 (v,j), 但し, $i \neq j$, を v の入力辺 (i,v) に続く出力辺とする. このように有向閉路を構成することにより, 各閉路 C_i には, (i,j), $i > j$ であるような辺が必ず 1 本存在するので, これを閉路 C_i の指標とする辺とみなし主辺と呼ぶ [2][3].

また, 各閉路 C_i を節点 c_i, 主辺 $(n,1)$(節点番号 1 は始点 s である) を含む閉路 C_1 の節点 c_1 を木の根とし, 閉路 C_i が主辺 (i,j) をもち, かつ, C_j が辺 (j,i) をもつとき, C_i, C_j に対応する節点 c_i, c_j を接続する辺 (c_i,c_j) を付けることにより根付木 T_G を構成する [2][3]. 図 1(a) のグラフ G における根付木 T_G を図 1(b) に示す. 主辺 (i,j) を含む閉路 C_v に対応する T_G^* の節点 v の親を, 辺 (j,i) を含む閉路 C_u に対応する T_G^* の節点 u とし, また, v を u の子供という. 子供をもたない節点を葉と呼ぶ.

有向グラフ G' の終点 t に入る周辺を含む閉路 C_l に対応する T_G の節点を c_l とし, T_G における根 c_1 から c_l への路 p に着目する. この路 $p = c_1, c_2, \cdots, c_l$ 上の辺を T_G の主線, 路 p 上の端点も含む節点を主節点と呼び, それ以外の T_G の辺を支線, 節点を支節点と呼ぶ. 図 1(b) において, 主線は $(1,2)$, $(2,3)$, $(3,4)$ であり, 主節点は $1,2,3,4$ であ

る．主線の求め方の詳細については，アルゴリズムの概略を説明した後に述べる．

以下において，有向閉路 C_i に対して，辺の向きを無視して得られる無向閉路を $\tilde{C}_i$ で表す．主節点 $c_1, c_2, \cdots, c_l$ に対応する G の閉路 $\tilde{C}_1, \tilde{C}_2, \cdots, \tilde{C}_l$ のみからなる G の部分グラフ $\tilde{G}$ に着目する．$\tilde{C}_1$ は s, $\tilde{C}_l$ は t を含み，$\tilde{C}_i$, $i = 2, \cdots, l-1$, は $\tilde{C}_{i-1}$, $\tilde{C}_{i+1}$ のみと節点を共有しているので，$\tilde{G}$ は，s から t への G 上の2本の辺素な路 P_1, P_2 と，一方の端点が P_1 上にあり，他方の端点が P_2 上にあるような辺のみからなる．図 1(a) の G に対応する $\tilde{G}$ を図 1(d) に示す．最終的に，この $\tilde{G}$ 上での最大流量を求めることにより，G の最大流量を求める．そのために，まず，$\tilde{G}$ 以外の部分グラフ (T_G の支線に対応する G の部分グラフ) の最大流量を決定する．詳しく述べると次のようになる:

T_G より主節点，および，それに接続する辺を除去すると，いくつかの部分木 T_1, $T_2, \cdots, T_k$ からなる森となる．各部分木 T_i において，主節点と接続していた節点を木 T_i の根とする．T_i の節点に対応する閉路からなる G の部分グラフ (T_i の節点に対応する閉路の節点と辺からなり，2つの無向閉路 $\tilde{C}_i$, $\tilde{C}_j$ の共通辺のうち1本を除去したグラフ) を G_i で表す．G_i と $\tilde{G}$ のグラフの関係より，G_i と $\tilde{G}$ との共有節点は2つである．G_i と $\tilde{G}$ との2つの共有節点を x_i, y_i とすると，G_i における x_i, y_i 間の最大流量 $f(x_i, y_i)$ をまず求め，$\tilde{G}$ の辺 (x_i, y_i) の容量 $c(x_i, y_i)$ を $f(x_i, y_i)$ とする．

例えば，図 1(b) において，T_G より主節点，および，それに接続する辺を除去することにより，節点6を根とする部分木 T_1 と節点5のみからなる部分木 T_2 を得る．そこで，まず最初に $\tilde{G}$ と G_1 の共有節点 $x_1 = 10, y_1 = 16$ 間の G_1(図 3(a)) 上の最大流量，および，G_2 上 (閉路 $\tilde{C}_5$) の $x_2 = 4, y_2 = 6$ 間の最大流量を並列に求めることになる．また，G_1 における $x_1 = 10, y_1 = 16$ 間の最大流量 $f(10,16) = 60$ が，$\tilde{G}$ の辺 $(10,16)$ の容量 $c(10,16) = f(10,16) = 60$ となる．

この各部分グラフ G_i, $i = 1, \cdots, k$, 上の x_i, y_i 間の最大流量の求め方の詳細については，アルゴリズムの概略の説明後に述べるが，G_i に対応する木 T_i, $i = 1, \cdots, k$, の性質を利用して求める．

各 $G_i, i = 1, \cdots, k$, 上の $\tilde{G}$ との共有節点 x_i, y_i 間の最大流量が求まれば，最後に $\tilde{G}$ の最大流量を求める．この値が G の最大流量になる．$\tilde{G}$ の各辺 (x_i, y_i) の容量 $c(x_i, y_i)$ は，部分グラフ G_i, $i = 1, \cdots, k$, との共有節点 x_i, y_i 間においては G_i における x_i, y_i 間の最大流量 $f(x_i, y_i)$ が代入され，それ以外の辺 (x, y) については G の容量 $c(x, y)$ のままである．

$\tilde{G}$ の周辺 $s(= 1), \cdots, i, i+1, \cdots, t, \cdots, j, j+1, \cdots, n$ において，路 $s(=1), \cdots, i, i+1, \cdots, t$ 上の辺を上周辺と呼び，路 $t, \cdots, j, j+1, \cdots, s$ 上の辺を下周辺と呼ぶ．$\tilde{G}$ の双対グラフ [6] を求め G^* とする．但し，本来，平面グラフの無限面に対する双対グラフの節点は1つであるが，ここでは，$\tilde{G}$ の無限面に対応する G^* の節点は v^*_{up}, v^*_{down} の2つとし，上周辺に対応する G^* の辺は v^*_{up} に接続させ，下周辺に対応する G^* の辺は v^*_{down} に接続させる．図 2に例を示す．

G^* の各辺 (i^*, j^*) の長さ $l(i^*, j^*)$ を，対応する $\tilde{G}$ の辺 (i, j) の容量 $c(i, j)$ とし，G^* における v^*_{up} から v^*_{down} への最短距離を求める．この最短距離が G の最小カット [9] の値であり，最大流量 - 最小カットの定理 [9] により，G の最大流量 val が求まる．

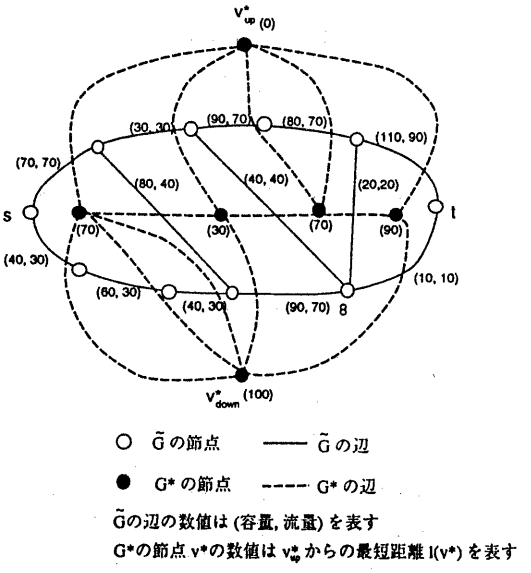

図 2: $\tilde{G}$ に対する双対グラフ G^*

以上をまとめると，次の並列アルゴリズム Procedure **Maximum-Flow(G)** を得る．

Procedure **Maximum-Flow(G)**

begin
(Step 1) Gの周辺を求め，始点sを節点番号1とし，周辺に沿ってGの各節点(すべて周辺上にある)に節点番号$1,\cdots,n$を付ける[2][3].
(Step 2) 節点番号に基づき，Gの有向グラフG'における有向閉路$C_1,\cdots,C_k$を構成する[2][3].
(Step 3) 有向閉路$C_1,\cdots,C_k$を節点とする木T_Gを構成する[2][3].
(Step 4) T_Gの主線，支線を求める.
(Step 5) 支線にあたる各部分木をT_i, $i=1,\cdots,k$, とする.
for *all* i, $1\leq i\leq k$ **in parallel do**
T_iに対応するGの部分グラフG_iと$\tilde{G}$との共有節点x_i,y_i間の最大流量を求める.
(Step 6) $\tilde{G}$の双対グラフG^*を求める.{ただし，$\tilde{G}$の無限面に対応するG^*の節点はv^*_{up}, v^*_{down}の2つとし，上周辺に対応するG^*の辺は節点v^*_{up}に接続し，下周辺に対応するG^*の辺は節点v^*_{down}に接続させる.}
(Step 7) G^*の各辺(i^*,j^*)の長さ$l(i^*,j^*)$を，対応する$\tilde{G}$の辺(i,j)の容量$c(i,j)$とし，G^*における節点v^*_{up}から節点v^*_{down}への最短距離を求める.(この最短距離がGの最小カット[9]の値であり，Gの最大流量valが求まる.)
end.

以下，*Procedure Maximum-Flow*(G)の*Step* 4, 5, 7について詳細に述べる.

Step 4は，根付木T_Gの主線，支線を求めている.つまり，T_G上の根c_1から節点c_lへの路を求める.(c_1は，G'上の主辺$(n,1)$を含む閉路C_1に対応するT_Gの節点であり，c_lは，終点tに入る周辺を含む閉路C_lに対応するT_Gの節点である.)ここでは，根付木T_Gに対し，オイラーツアー法(Euler tour technique)[3]を利用する.オイラーツアー法について簡単に説明する:

まず，木の各辺を互いに逆向きの2本の有向辺に置き換える.　すると，各節点において入次数と出次数が等しくなるので有向オイラー閉路が存在する.有向オイラー閉路中の木の根rに戻る辺(v,r)の先端の節点rでオイラー閉路を切断して路とし，それを木の各辺が要素であるリストLで表現する.図1(c)に木T_Gに対するリストLを示す(辺の数値はランクを表す).このリストにダブリング法(doubling method)[3]を適用して，リストL中の各要素のランク，つまり，リストにおいて先頭から何番目かを並列に求める.リストL中には，元の木T_Gの辺(c_i,c_j)に対応する2本の有向辺(c_i,c_j)と(c_j,c_i)が出現する.　この2つの有向辺のうち，ランクの小さい方を前進辺，大きい方を後退辺とよぶ.リストLの先頭を辺(c_1,c_2)とし，(c_1,c_2)から出発して，c_lに接続する辺のうちランク最小の辺(c_{l-1},c_l)で終了する路pに着目すると，p中には，(c_i,c_j)の前進辺，後退辺が共に出現する場合，前進辺のみ出現する場合，共に出現しない場合が存在する.元の木T_Gの辺のうち，p中に前進辺のみ出現する辺(c_i,c_j)が主線となり，それ以外の辺が支線となる.

Step 5では，T_Gの支線にあたる各部分木T_iに対応するGの部分グラフG_iと$\tilde{G}$との共有節点x_i,y_i間の最大流量を求めている.そのために，G_iにおけるx_i-y_i最小カットを求める.以下に，x_i-y_i最小カットの求め方について述べる.

まず，G_iに対応する部分木T_iに次に示すように節点，及び，辺を加えて木T_i^*を構成する.　追加する節点は，v_r^*，及び，G_iの周辺の数n'個の節点$v^*_{1'},\cdots,v^*_{n'}$である.　(節点$v^*_{1'},\cdots,v^*_{n'}$は，G_iの周辺$e_{1'},\cdots,e_{n'}$に対応する.)追加する辺は，v_r^*とT_iの根，及び，閉路C_jに対応するT_iの節点v_jとC_jの周辺$e_{k'}$に対応する追加した節点$v_{k'}$を接続する辺である.　図3(a)に部分木T_1(部分グラフG_1)に対応する木T_1^*を示す.

このように構成したT_i^*をG_iの双対グラフとみなす(本来，双対グラフはGの無限面に節点1つを対応させるが，ここでは，無限面と境界をなす辺(周辺)の数だけ節点を準備している点が異なる).T_i^*の根v_r^*からT_i^*の各葉までの距離を計算し，その距離の最小値がx_i-y_i最小カットになる.

v_r^*からT_i^*の各葉までの距離を求めるのに，先に説明したオイラーツアー法[3]を再度利用する.先と同様に，各要素がT_i^*の有向オイラー閉路の辺であるリストL'を得る.　ここでは，リストL'の各要素(辺)を次のように数値に置き換える.　T_i^*の辺(u^*,v^*)に対応するG_iの辺を(u,v)とすると，リストL'における要素が(u^*,v^*)の前進辺の場合には，辺(u,v)の容量$c(u,v)$で置き換え，後退辺の要素の場合には$-c(u,v)$で置き換える.例えば、図3(b)に示したT_1^*の場合，リストL'は，$c(v_r^*,6)$,$c(6,7)$,$c(7,v^*_{1'})$,$c(v^*_{1'},7)$, $\cdots$,$c(6,v_r^*)$, つまり，$10,10,60,-60,\cdots,-10$となる.　次に，根v_r^*を先頭とするリストL'に対し，プレフィクス和を求める.　プレフィクス和とは，リストに

数列 $x_1, x_2, \cdots, x_n$ が入っているとき，x_1，$x_1 + x_2$，$x_1 + x_2 + x_3$，$\cdots$，$x_1 + x_2 + \cdots + x_n$ の値を計算する手法であり，効率良く求める並列アルゴリズムが知られている [3][8]．T_i^* における v_r^* から各節点 v^* への距離は，v^* の前進辺までのリスト L' のプレフィクス和で求められる．図 3(b) に示す例の場合，$x_1, x_2, x_3, \cdots, x_8$ は，それぞれ $10, 10, 60, \cdots, -10$ に対応するので，プレフィクス和の結果は $10, 20, 80, \cdots, 0$ となる．

Step 7 は，G^* における v_{up}^* から v_{down}^* への最短距離を求めている．まず最初に $\tilde{G}$ の内面に対応する G^* の節点 $v_1^*, \cdots, v_l^*$ のみからなるグラフ，つまり，辺 $(v_1^*, v_2^*), (v_2^*, v_3^*), \cdots, (v_l^*, v_l^*)$ のみからなる 1 本の路の各節点間 v_i^*, v_j^*，$i, j = 1, \cdots, l$, の距離 $l(v_i^*, v_j^*)$ を求める．これは，プレフィクス計算を用いれば求まる．配列 $L^*(i+1)$，$i = 1, \cdots, l-1$, に辺の長さ $l(v_i^*, v_{i+1}^*)$ を代入する．ただし，$L^*(1) = 0$ とする．配列 $L^*(i)$ でのプレフィクス和 $L^*(1)$，$L^*(1) + L^*(2)$，$L^*(1) + L^*(2) + L^*(3), \cdots, L^*(1) + L^*(2) + \cdots + L^*(l)$ を並列に計算して，$L^*(1)$ から $L^*(j)$ までの和を配列 $LEN(j)$，$j = 1, \cdots, l$, に格納する．配列 $LEN(j)$ を求めておけば，節点間 $v_k^*, v_{k'}^*$，$(k < k')$ の距離は，$LEN(k') - LEN(k)$ により定数時間で求めることができる．

次に，節点 v_{up}^* と節点 v_i^*，$i = 1, \cdots, l$, の辺の長さ，v_i^* と v_{down}^* を結ぶ辺の長さをそれぞれ $l(v_{up}^*, v_i^*)$, $l(v_i^*, v_{down}^*)$ とすると，

$$\min_{i,j=1,\cdots,l} \{ l(v_{up}^*, v_i^*) + l(v_i^*, v_j^*) + l(v_j^*, v_{down}^*) \}$$

により，節点 v_{up}^* から節点 v_{down}^* への最短距離が求まる．

次に Procedure Maximum-Flow(G) の計算時間とプロセッサ数の解析を行なう．ここでの並列計算機モデルとしては，CRCW PRAM[3][8] を採用する．また，外平面グラフの辺数 m は，$2n - 3$ 以下 [6] なので $m = O(n)$ である．

Step 1 について．文献 [2] に述べられているように，G から辺 e_i のみ除去したグラフ $G - e_i$ が 2 連結でない場合，その辺 e_i は周辺である．よって，各辺 e_i，$i = 1, \cdots, m$, に対し，$G - e_i$ が 2 連結であるかどうかを調べればよい．2 連結性を判定する並列アルゴリズム [1] は，$O(\log n)$ 時間，$O(n\alpha(m,n)/\log n)$ 個のプロセッサで実行可能である．ここで $\alpha(m,n)$ は、Ackermann 関数

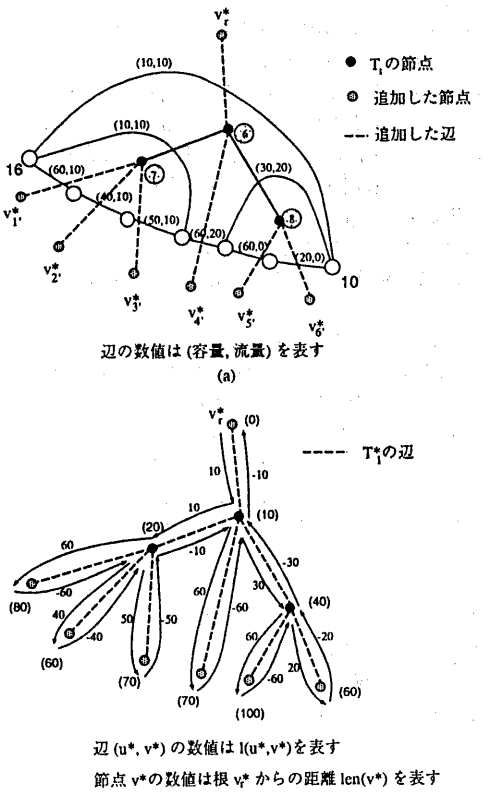

図 3: (a)T_i^* の構成，(b)T_i^* に対する有向オイラー閉路

の逆関数で，m と n に関して非常にゆっくりと増加する関数であり，実用的な m，n の値に対しては 4 を越えない定数である [10]．よって，全体として $O(\log n)$ 時間，$O(n^2\alpha(m,n)/\log n)$ 個のプロセッサで周辺が求まる．しかし，合計で q 回の計算を何台かのプロセッサを用いて t 時間で並列計算可能な場合，p 台のプロセッサを用いて，$O(q/p + t)$ 時間で並列計算可能である，という Brent の定理 [3] により，次のようにプロセッサ数を削減できる．まず，辺を $\lceil m/\log n \rceil$ 個のブロック $M_i, i = 1, \cdots, \lceil m/\log n \rceil$ に分割する．ブロック M_1 に対し，M_1 に属する辺 ($O(n/\log n)$ 本) が周辺であるかどうかを $O(n\alpha(m,n)/\log n) \times O(n/\log n) = O(n^2\alpha(m,n)/\log^2 n)$ 個のプロセッサ，$O(\log n)$

時間で調べる．同様の操作を $O(\log n)$ 回繰り返すことにより，$M_i, i = 2, \cdots, \lceil m/\log n \rceil$ が調べられるので，全体として $O(n^2\alpha(m,n)/\log^2 n)$ 個のプロセッサ，$O(\log n + \log n) = O(\log n)$ 時間で求めることができる．

Step 2 は，文献 [2][3] にあるようにソーティングが必要であるが，バケットソートは $O(n \log\log n/\log n)$ 個のプロセッサを用いて $O(\log n)$ 時間で求まる [5]．

Step 3 は，木 T_G の構成なので $O(1)$ 時間，$O(n)$ 個のプロセッサで実行できる．

Step 4 は，木 T_G のオイラーツアー法を利用しているので $O(\log n)$ 時間，$O(n/\log n)$ 個のプロセッサで求まる [3][8]．

Step 5 は，オイラーツアー法，及び，プレフィクス計算を用いているので，$O(\log n)$ 時間，$O(n/\log n)$ 個のプロセッサで実行できる [3][8]．

Step 6 の双対グラフの構成は，$O(1)$ 時間，$O(n)$ 個のプロセッサで実行できる．

Step 7 は，$O(n^2)$ 個の値の最小値を求めている．単純に $O(n^2)$ 個のプロセッサ，$O(\log n)$ 時間で求まるが，Brent の定理 [3] により，$O(n^2/\log n)$ 個のプロセッサ，$O(\log n)$ 時間で求めることができる．($l(v_i^*, v_j^*)$ の値は，先に述べたように，プレフィクス計算を用いて求めた配列 LEN を基に，定数時間で求めることができる．)

以上から次の定理が得られる．

[定理 1] **Procedure Maximum-Flow(G)** により，***CRCW PRAM*** 上で全体として，$O(n^2/\log n)$ 個のプロセッサを用いて $O(\log n)$ 時間で 2 連結外平面グラフ G の最大流量 val を求めることができる．但し，n は G の節点の個数である．□

4 辺の流量を求める並列アルゴリズム

前章では，外平面グラフ G の最大流量 val を求めた．ここでは，最大流量 val を基に，G の各辺の流量を決定する並列アルゴリズムについて述べる．前章の最大流量を求めるアルゴリズムでは，まず，各部分グラフ G_i 上での $\tilde{G}$ との共有節点 x_i, y_i を始点，終点とする最大流量を求め、その値を $\tilde{G}$ の辺の容量として，$\tilde{G}$ における最大流量を求めることにより G の最大流量を求めた．これとは逆に，辺の流量を求めるアルゴリズムでは，まず，G の最大流量 val を基に $\tilde{G}$ の各辺の流量 $f(x,y)$ を決定し，次に，$\tilde{G}$ における $\tilde{G}$ と各部分グラフ G_i の共有辺 (x_i, y_i) の流量 $f(x_i, y_i)$ を基に，G_i の各辺の流量を求めることにより，G の各辺の流量を決定する．

以下に並列アルゴリズムを示す．

Procedure Flow-Determination
begin
(Step 1) { $\tilde{G}$ の各辺の流量を決定する．}
$\tilde{G}$ の内面に対応する G^* の節点を $v_1^*, \cdots, v_l^*$ とする．G^* の各辺 (i^*, j^*) の長さ $l(i^*, j^*)$ を，対応する $\tilde{G}$ の辺 (i,j) の容量 $c(i,j)$ とし，v_{up}^* から各 v_i^*, $i = 1, \cdots, l$, への最短距離を求め，その値を $l(v_i^*)$, $i = 1, \cdots, l$, に格納する．ただし，v_{up}^*, v_{down}^* の値は，$l(v_{up}^*) = 0$, $l(v_{down}^*) = val$ とする．(図 2参照．)

$\tilde{G}$ の辺 (u,v) の流量 $f(u,v)$ は，対応する (交わる)G^* の辺を (u^*, v^*) とすると，

$$f(u,v) = |l(u^*) - l(v^*)| \qquad (1)$$

になる．流れの方向は次のようになる．G^* を平面に埋め込んだとき，G^* の辺 (u^*, v^*) の端点の値 $l(u^*)$, $l(v^*)$ の大きい方を上，小さい方を下とみなせば，(u^*, v^*) に交わる $\tilde{G}$ の辺 (u,v) の右側の節点から左側の節点へ流れる．

(Step 2) { 部分グラフ G_i の各辺の流量を求める．各辺の流量は，G_i に対応する木 T_i^* の根から各節点への距離を用いて求める．}

for *all* i, $1 \le i \le k$ **in parallel do**

$\tilde{G}$ と G_i の共有節点を x_i, y_i とし，Step 1 で求めた $\tilde{G}$ における辺 (x_i, y_i) の流量を $f(x_i, y_i)$ とする．T_i^* の辺 (u^*, v^*) において，u^* を v^* の親とする．T_i^* の根から u^*, v^* への距離をそれぞれ $len(u^*)$, $len(v^*)$ とする．(u^*, v^*) に対応する G_i の辺 (u,v) の流量 $f(u,v)$ は，

$$f(u,v) = \begin{cases} len(v^*) - len(u^*): & len(v^*) \le f(x_i, y_i) \text{ のとき} \\ f(x_i, y_i) - len(u^*): & len(v^*) > f(x_i, y_i) \text{ かつ} \\ & len(u^*) < f(x_i, y_i) \text{ のとき} \\ 0: & len(v^*) > f(x_i, y_i) \text{ かつ} \\ & len(u^*) \ge f(x_i, y_i) \text{ のとき} \end{cases} \qquad (2)$$

で求まる．(図 3(a) 参照．)

G_i の各辺の流れの方向は，Step 1 で求めた $\tilde{G}$ における辺 (x_i, y_i) 上の流れの向き，および，節点 x_i, y_i の節点番号の大小関係で決まる．つまり，辺 (x_i, y_i) 上の流れの向きが節点 x_i から y_i で，かつ，節点番号に関して $x_i < y_i$ ならば，G_i の各辺 (u,v) 上には，節点番号の小さい節点から大きい節点へ $f(u,v)$ 流れ，逆に，節点番号に関して $x_i > y_i$ ならば，G_i の各辺 (u,v) 上には，節点番号の大きい節点から小さい節点へ $f(u,v)$ 流れる．

end.

Procedure **Flow-Determination** の正当性を示すために必要な2つの補題を導入する.

[補題 1] 式 *(1)* に示した流量の割り当てにより, $\tilde{G}$ における最大流量の割り当てが求まる. □

(証明略)

[補題 2] 式 *(2)* に示した流量の割り当てにより, G_i における流量 $f(x_i, y_i)$ の各辺への割り当てが求まる. □

(証明略)

Procedure **Flow-Determination** の計算時間とプロセッサ数について述べる.

Step 1 では, v^*_{up} から各 v^*_i, $i = 1, \cdots, l$, への最短距離 $l(v^*_i)$ を求めるために, 各 v^*_i に対し,

$$l(v^*_i) = \min_{j=1,\cdots,l} \{\, l(v^*_{up}, v^*_j) + l(v^*_j, v^*_i) \,\}$$

の計算を行なう. Procedure **Maximum-Flow(G)** の Step 7 と同様に, $O(\log n)$ 時間, $O(n^2/\log n)$ 個のプロセッサで求めることが可能である. ($l(v^*_j, v^*_i)$ の値は, 先に述べたように, プレフィクス計算を用いて求めた配列 LEN を基に, 定数時間で求めることができる.)

Step 2 は, Procedure **Maximum-Flow(G)** で求めた T^*_i に関する v^*_i から各節点 v^* の距離 $len(v^*)$ が与えられれば, 単位時間, $O(n)$ 個のプロセッサで求めることが可能である.

以上から次の定理が得られる.

[定理 2] Procedure **Maximum-Flow(G)** で2連結外平面グラフ G の最大流量 val を求めたならば, Procedure **Flow-Determination** により, $O(n^2/\log n)$ 個のプロセッサを用いて $O(\log n)$ 時間で G の各辺の流量を求めることができる. 但し, n は G の節点の個数である. □

5 あとがき

本論文では, 外平面グラフ G 上の任意に指定した2節点 s, t 間の最大流量を CRCW PRAM 上で, $O(n^2/\log n)$ 個のプロセッサを用いて $O(\log n)$ 時間で求める並列アルゴリズムを提案した.

今後の課題としては, 一般のグラフ上においてクラス P 完全 [3][8] に属する問題や, $\log n$ の多項式時間で求めるには非常に多くのプロセッサを必要とする問題について, 外平面グラフ上では, 線形のプロセッサ数で $\log n$ の多項式時間で求められる並列アルゴリズムが存在するかどうかを検討することがある.

参考文献

[1] R. Cole, U. Vishkin: "Approximate parallel scheduling, II: Applications to optimal parallel graph algorithms in logarithmic time" *Inform. Comput.*, **91**, pp.1-47, 1991.

[2] K. Diks: "A fast parallel algorithm for six-colouring of planar graphs", *LNCS 233*, Springer Verlag, pp.273-282, 1985.

[3] A. Gibbons and W. Rytter: *Efficient Parallel Algorithms*, Cambridge University Press, 1988.

[4] Xin He, Yaacov Yesha: "Binary tree Algebraic computation and parallel algorithms for simple graphs" *Journal of Algorithms*, **9**, pp.92-113, 1988.

[5] T. Hagerup: "Towards optimal parallel bucket sorting" *Inform. Comput.*, **75**, pp.39-51, 1987.

[6] F. Harary: *Graph Theory*, Addison-Wesley , 1969.

[7] D. B. Johnson: "Parallel Algorithms for Minimum Cuts and Maximum Flows in Planar Networks", *J. ACM*, Vol. 34, No. 4, pp. 950-967, 1987.

[8] Joseph JáJá: *An Introduction to parallel algorithms*, Addison-Wesley Publishing Company, 1992.

[9] J. van Leeuwen: *Graph Algorithms, in: J. van Leeuwen, eds. Handbook of Theoretical Computer Science*, Elsevier Science Publishers B.V., 1990.

[10] R. E. Tarjan: *Data Structures and Network Algorithms*, SIAM, Philadelphia, PA, 1983.