台形グラフにおける全域森構築のための最適並列アルゴリズム

本間 宏利*

An Optimal Parallel Algorithm for Constructing Spanning Forest of a Trapezpid Graph

Hirotoshi HONMA

Abstract: Let $G=(V,E)$ be a simple graph with n vertices, m edges and p connected components. The problem of constructing a spanning forest is to find a spanning tree for each connected component of G. For a simple graph, Chin et al.[1] demonstrated that a spanning forest can be found in $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ processors. In this paper, we propose an $O(\log n)$ time parallel algorithm, which belongs to class NC and is optimal, with $O(n/\log n)$ processors on the EREW PRAM for constructing a spanning forest on trapezoid graphs.
Key words: Parallel Algorithm, Trapezoid Graph, Spanning Forest

1 はじめに

n 個の節点, m 本の辺，そして p 個の連結成分から構成される単純グラフ $G=(V,E)$ が与えられたとき，このグラフ G の各連結成分に対して，全域木を構築する問題を全域森問題という．もし，G の連結成分数が 1，すなわち G が連結であるならば，この問題は全節点を含んだ木を構築する全域木問題と等価である．これらの問題は電力供給問題，ネットワークデザイン等，様々な分野で応用されている．全域森問題は深さ優先探索等の手法を用いて線形時間内で解かれることはよく知られており，近年それらのアルゴリズムの並列化の研究もなされている．例えば，全域木問題は CRCW PRAM (Concurrent-Read Concurrent-Write Parallel Random Access Machine) 上で $O(\log n+m)$ 個のプロセッサを用いて，$O(\log n)$ 時間で解かれることが Klein らによって発表されている [6]．さらに， Chin らによって $O(n^2/\log^2 n)$ 個のプロセッサを用いて $O(\log^2 n)$ 時間で実行可能な並列アルゴリズムも提案された [1]．一般的に，問題の対象とするグラフの形状を制限することにより，より効率的な逐次，並列アルゴリズムの構築が可能であることはよく知られている．例として，グラフを *permutation graphs* に制限した場合，EREW PRAM (Exclusive-Read Exclusive-Write Parallel Random Access Machine) 上で $O(n/\log n)$ 個のプロセッサを用いた $O(\log n)$ 時間で終了する最適アルゴリズムが提案されている．また，Honma らによって対象とするグラフを台形グラフ（*trapezoid graphs*）に限定した EREW PRAM 上で $O(n)$ 個のプロセッサを用いて $O(\log n)$ 時間で実行可能な並列アルゴリズムも提案されている [4]．本研究では上記の研究を引き継ぎ，対象とするグラフを台形グラフに限定して，EREW PRAM 上で $O(n/\log n)$ 個のプロセッサを用いて $O(\log n)$ 時間で実行可能な並列アルゴリズムを提案する．この並列アルゴリズムはプロセッサ数と実行時間の積がデータ数に対して線形的になることから最適な並列アルゴリズムであることを意味する．

釧路高専情報工学科．

2 定義

台形グラフの紹介の前に，台形ダイアグラム (*Trapezoid Diagram*) を定義する．2 本の水平に平行な線分を配置し，上方の線分を *top channel*，下方を *bottom channel* と定義する．それぞれの channel には左から 1,2,...,2n（n は台形数）の整数値が割り付けられている．台形 T_i は top channel 上の 2 点，a_i, b_i $(a_i<b_i)$，bottom channel 上の 2 点，c_i, d_i $(c_i<d_i)$ からなる 4 つの角点（*corner points*）$[a_i,b_i,c_i,d_i]$ によって定義される図形であり，これらの角点は他の台形の角点と重複することはない．また，各台形はそれぞれが持つ角点 b の大きさによって，$b_i<b_j$ ならば，$i<j$ となるように順番付けされる．これらの台形の集合からなる幾何学的図形表現を台形ダイアグラムと呼ぶ [7]．図 1 に 17 個の台

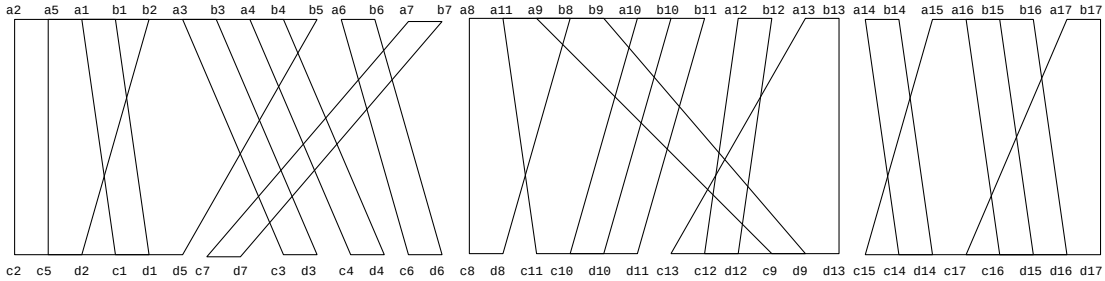

図 1: Trapezoid diagram T.

形からなる台形ダイアグラム T の例を示す．

続いて台形グラフ G を定義する．台形ダイアグラム T において，T 上の各台形を節点に一対一対応させ，ダイアグラム上の 2 つの台形 T_i, T_j が交差 (intersect) する場合，かつその時に限り，それらの台形に対応する節点 i, j 間に辺が存在するグラフ G を台形グラフと定義する [7]．台形ダイアグラムは top channel 上に割り当てられた角点の配列 $T_T[1:2n]$ と角点の持つ整数値の配列 $P_T[1:2n]$，および，bottom channel 上に割り当てられた角点の配列 $T_B[1:2n]$ と角点の持つ整数値の配列 $P_B[1:2n]$ によって入力される．表 1 に図 1 で示した台形ダイアグラム T の入力 $T_T[1:2n]$, $P_T[1:2n]$, $T_B[1:2n]$, $P_B[1:2n]$ を示す．また，これに対応する台形グラフ G の例を図 2 に示す．

3 並列アルゴリズム

台形グラフの全域森構築のための並列アルゴリズム CSF(Construction of Spanning Forest) について記述する．このアルゴリズムはポインタジャンピング技法 [3][5] と並列プレフィックス演算技法 [3][5] を用いて構成されている．

Algorithm **CSF**
Input: Arrays $T_T[1:2n]$, $P_T[1:2n]$, $T_B[1:2n]$, $P_B[1:2n]$.
Output: A spanning forest F^* of G. Initially F^* be a graph with n vertices and no edge.
(Step 1) [Construction of arrays P_a,P_b,P_c,P_d.]

(1) If $T_T[i]$ is corner point 'a_j', $P_T[i]$ is stored to $P_a[j]$, otherwise (i.e., $T_T[i]$ is 'b_j') $P_T[i]$ is stored to $P_b[j]$ in parallel for i,$1 \leq i \leq 2n$.

(2) If $T_B[i]$ is corner point 'c_j', $P_B[i]$ is stored to $P_c[j]$, otherwise (i.e., $T_B[i]$ is 'd_j') $P_B[i]$ is stored to $P_d[j]$ in parallel for i,$1 \leq i \leq 2n$.

表 2 は表 3 の入力データに上記 Step 1 の処理を適用した結果を示している．$P_a[1:n]$,$P_b[1:n]$,$P_c[1:n]$,$P_d[1:n]$ は台形ダイアグラム T 上のそれぞれの台形の角点 'a','b','c','d' に割り当てられた整数値の配列である．

(Step 2) [Construction of arrays L_a,L_c,R_d.]

(1) Let $L_a[i]$ be min($P_a[n]$,$P_a[n-1]$,...,$P_a[i]$) in parallel for i, $1 \leq i \leq n$.

(2) Let $L_c[i]$ be min($P_c[n]$,$P_c[n-1]$,...,$P_c[i]$) in parallel for i, $1 \leq i \leq n$.

(3) Let $R_d[i]$ be max($P_c[1]$,$P_c[2]$,...,$P_c[i]$) in parallel for i, $1 \leq i \leq n$.

(Step 3) [Construction of arrays S_a, C.]
Initially $C[i] := 0$ for all i.

(1) If $P_a[i] = L_a[i]$, let $S_a[i]$ be a pointer to i (*self-loop*), otherwise, let $S_a[i]$ be a pointer to $i+1$ in parallel for i, $1 \leq i \leq n$.
Then, we apply pointer jumping technique to $S_a[i]$ in parallel for i, $1 \leq i \leq n$.

(2) If $P_b[i] > L_a[i+1]$, then $C[i] := S_a[i+1]$ and $F^* := F^* \cup \{(i, S_a[i+1])\}$ in parallel for i,$1 \leq i \leq n-1$.

(Step 4) [Construction of array S_c.]

(1) If $P_c[i] = L_c[i]$, let $S_c[i]$ be a pointer to i (*self-loop*), otherwise, let $S_c[i]$ be a pointer to $i+1$ in parallel for i, $1 \leq i \leq n$.
Then, we apply pointer jumping technique to $S_c[i]$ in parallel for i, $1 \leq i \leq n$.

(2) If $P_d[i] > L_c[i+1]$ and $C[i] = 0$, then $C[i] := S_c[i+1]$ and $F^* := F^* \cup \{(i, S_c[i+1])\}$ in parallel for i, $1 \leq i \leq n-1$.

(Step 5) [Construction of array S_d.]

(1) If $P_d[i] = R_d[i]$, let $S_d[i]$ be a pointer to i (*self-loop*), otherwise, let $S_d[i]$ be a pointer $i-1$ in parallel for i, $1 \leq i \leq n$.
Then, we apply pointer jumping technique to $S_d[i]$ in parallel for i, $1 \leq i \leq n$.

(2) If $R_d[i] > L_c[i+1]$ and $C[i] = 0$, then $C[S_c[i+1]] := S_d[i]$ and $F^* := F^* \cup \{(S_c[i+1], S_d[i])\}$ in parallel for i, $1 \leq i \leq n-1$.

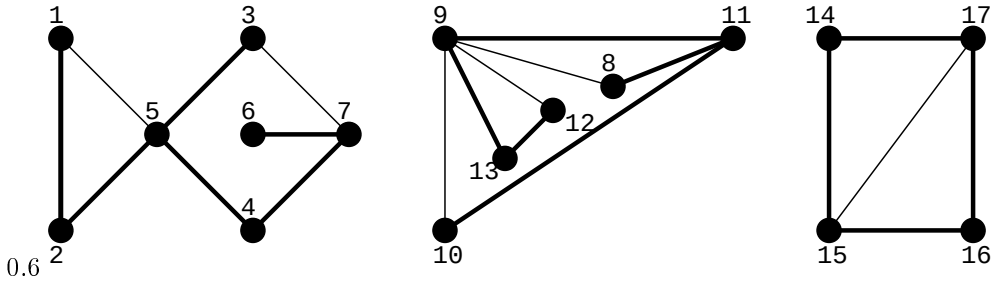

図 2: A spanning forest F^* for Trapezoid graph G.

表 1: Arrays T_T, P_T, T_B, P_B.

T_T	a_2	a_5	a_1	b_1	b_2	a_3	b_3	a_4	b_4	b_5	a_6	b_6	a_7	b_7	a_8	a_{11}	a_9
P_T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
T_B	c_2	c_5	d_2	c_1	d_1	d_5	c_7	d_7	c_3	d_3	c_4	d_4	c_6	d_6	c_8	d_8	c_{11}
P_B	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
T_T	b_8	b_9	a_{10}	b_{10}	b_{11}	a_{12}	b_{12}	a_{13}	b_{13}	a_{14}	b_{14}	a_{15}	a_{16}	b_{15}	b_{16}	a_{17}	b_{17}
P_T	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
T_B	c_{10}	d_{10}	d_{11}	c_{13}	c_{12}	d_{12}	c_9	d_9	d_{13}	c_{15}	c_{14}	d_{14}	c_{17}	c_{16}	d_{15}	d_{16}	d_{17}
P_B	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

(3) Change F^* to be an undirected graph by neglecting the direction of each edge in F^*.

表 3 は表 2 に上記の Steps 2，3，4，5 を実行した時の結果を示している．図 2 は，このアルゴリズムによって構築された全域森の一例である．

4　アルゴリズム CSF の正当性

並列アルゴリズム CSF の正当性を示すため，いくつかの補題を挙げる．

Lemma 1　$1 \leq i < j \leq n$ なる i,j において，$P_b[i] > L_a[j]$ が成立するならば，$(i, S_a[j])$ は台形グラフ G の辺である．

$1 \leq i < j \leq n$ なる i,j において，$P_d[i] > L_c[j]$ が成立するならば，$(i, S_c[j])$ は台形グラフ G の辺である．

証明. 台形グラフの定義より，台形ダイアグラム T 上の台形 T_i と T_j が交差していれば，台形グラフ G 上の節点 i, j 間に辺 (i, j) が存在する．ここで，台形 T_i と T_j が交差するならば，それは top channel 上で $P_b[i] > P_a[j]$，または，bottom channel 上で $P_d[i] > P_c[i]$ のどちらかが成立することを意味する．したがって，式 (1) が成立するとき，台形グラフ G 上の節点 i, j 間に辺 (i, j) が存在する．

$$(i-j)(P_b[i]-P_a[j]) < 0 \; or \; (i-j)(P_d[i]-P_c[j]) < 0. \quad (1)$$

また，$i < j$，$P_b[i] > L_a[j]$ より次の式が成立する．

$$(i-j)(P_b[i]-L_a[j]) < 0. \quad (2)$$

Step 4-(1) 実行後，$S_a[j]$ は $L_a[j] = P_a[k_1]$ を満たす k_1 $(k_1 \geq j)$ を値として保持する．また，定義 $L_a[j]$ =min$(P_a[j], P_a[j+1], ..., P_a[n])$ より次式が成立する．

$$S_a[j] \geq j,$$

$$L_a[j] = L_a[S_a[j]] = P_a[S_a[j]].$$

さらに，式 (2) より次式が成り立つ．

$$(i - S_a[j])(P_b[i] - P_a[S_a[j]]) < 0. \quad (3)$$

式 (3) は G 上の節点 i と $S_a[j]$ 間に辺が存在することを意味している．よって辺 $(i, S_a[j])$ は台形グラフ G 上の辺である．同様に $(i, S_c[j])$ も G 上の辺である．□

Lemma 2　*Step 4* 実行後，$C[1:n]$ が q 個の *'0'* の要素を含むとする．このとき，F^* は n 個の節点, $n-q$ 本の辺，q 個の連結成分からなる全域森を構成する．

表 2: Arrays P_a, P_b, P_c, P_d.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
P_a	3	1	6	8	2	11	13	15	17	20	16	23	25	27	29	30	33
P_b	4	5	7	9	10	12	14	18	19	21	22	24	26	28	31	32	34
P_c	4	1	9	11	2	13	7	15	24	18	17	22	21	28	27	31	30
P_d	5	3	10	12	6	14	8	16	25	19	20	23	26	29	32	33	34

表 3: Arrays $L_a, L_c, R_d, S_a, S_c, S_d, C$.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
P_a	3	1	6	8	2	11	13	15	17	20	16	23	25	27	29	30	33
L_a	1	1	2	2	2	11	13	15	16	16	16	23	25	27	29	30	33
S_a	2	2	5	5	5	6	7	8	11	11	11	12	13	14	15	16	17
P_b	4	5	7	9	10	12	14	18	19	21	22	24	26	28	31	32	34
P_c	4	1	9	11	2	13	7	15	24	18	17	22	21	28	27	31	30
L_c	1	1	2	2	2	7	7	15	17	17	17	21	21	27	27	30	30
S_c	2	2	5	5	5	7	7	8	11	11	11	13	13	15	15	17	17
P_d	5	3	10	12	6	14	8	16	25	19	20	23	26	29	32	33	34
R_d	5	5	10	12	12	14	14	16	25	25	25	25	26	29	32	33	34
S_d	1	1	3	4	4	6	6	8	9	9	9	9	13	14	15	16	17
C	2	5	5	5	0	7	4	11	11	11	0	13	9	15	16	17	0

証明. Step 4 実行後，$C[n]$ の内容は明らかに '0' である．ここで，$C[i] = 0, C[i+1], C[i+2], ..., C[n-1] \neq 0$, $C[n] = 0$ なる節点 i を想定する．そのような i が存在しないとするなら G は連結である．今，G は非連結すなわち $p > 1$ だと想定する．すると，$C[n-1] \neq 0$ より，節点 $n-1$ と n に接続する辺 $(n-1, n)$ が存在する．また，$C[n-2] \neq 0$ より，節点 $n-2$ と節点 $n-1$ と n のどちらかの節点に接続する辺が存在することになる．このように，$i+1 \leq j \leq n-1$ なる j に対して，節点 j と節点 $j+1$,$j+2$,...,n の中のどれかの節点に 1 本の辺が存在することがわかる．反対に，$C[i] = 0$ なので，節点 i と j ($j \geq i+1$) 間にはどんな辺も存在しない．従って，$i+1$ から n に対して，$n-1$ 個の節点，$n-i-1$ 本の辺を有する連結グラフが構築されることになる．木の定義より，このような部分グラフは木であることがわかる．同様に，他の連結成分に対しても部分木が構築される．補題 1 より，Steps 3,4 によって構成された辺は台形グラフ G の辺であることがいえる．よって，F^* は q 個の連結成分からなる G の部分グラフであり，n 個の節点，$n-q$ 本の辺を持ち，かつ，各成分が木を構成することが証明される．□

Lemma 3　$1 \leq i < j \leq n$ なる i, j に対して，$R_d[i] > L_c[j]$ が成り立つならば，$(S_c[j], S_d[i])$ は G の辺である．

証明. 定義，$i < j$，$R_d[i] > L_c[j]$ より次式が成立する．

$$(i-j)(R_d[i] - L_c[j]) < 0. \tag{4}$$

Step 5-(1) より，$S_c[j]$ は $L_c[j] = P_c[k_2]$ を満たす $k_2 (k_2 \geq j)$ を保持する．また，$S_d[i]$ は $R_d[i] = P_d[k_3]$ を満たす $k_3 (k_3 \leq i)$ を保持する．さらに，定義 $L_c[j]$=min$(P_c[j], P_c[j+1], ..., P_c[n])$, $R_d[i]$=max$(P_d[1], P_d[2], ..., P_d[i])$ より次式を得る．

$$S_c[j] \geq j,$$

$$S_d[i] \leq i,$$

$$L_c[j] = L_c[S_c[j]] = P_c[S_c[j]],$$

$$R_d[i] = R_d[S_d[i]] = P_d[S_d[i]].$$

式 (4) より，次式が成り立つ．

$$(S_d[i] - S_c[j])(P_d[S_d[i]] - P_c[S_c[j]]) < 0. \tag{5}$$

式 (5) は G の節点 $S_c[j]$ and $S_d[i]$ 間に辺が存在することを意味している．よって，$(S_c[j], S_d[i])$ は G の辺である．□

Lemma 4　*Step 5* 実行後，F^* は G の全域森を構築する．

5 計算量解析

並列アルゴリズム CSF の計算量を解析する．Step 1 は Brent のスケジューリング原理 [3] によって $O(n/\log\ n)$ 個のプロセッサを利用して $O(\log\ n)$ 時間で実行可能である．Step 2 は並列プレフィックス演算 [5] により，$O(n/\log\ n)$ 個のプロセッサを利用して $O(\log\ n)$ 時間で実行可能である．Steps 3,4,5-(1) はポインタジャンピング技法 [8] により，$O(n/\log\ n)$ 個のプロセッサを利用して $O(\log\ n)$ 時間で実行可能である．Steps 3,4,5-(2) は Brent のスケジューリング原理 [3] によって $O(n/\log\ n)$ 個のプロセッサを利用して $O(\log\ n)$ 時間で実行可能である．これらの並列技法は EREW PRAM 計算機モデル上で実現可能である．よって以下の定理が成立する．

Theorem 1　並列アルゴリズム *CSF* は台形グラフの全域森を *EREW PRAM* 計算機モデル上で，$O(n/\log\ n)$ 個のプロセッサを利用して $O(\log\ n)$ 時間で構築する．

6 考察

台形グラフの全域森構築の並列アルゴリズムは文献 [4] では $O(n)$ 個のプロセッサを利用して $O(\log\ n)$ 時間で実行可能であった．この並列アルゴリズムは効率的ではあるが最適なものではなかった．この障害となっていたのがポインタジャンピング手法にて使用するプロセッサ数が $O(n)$ であったことが原因であるが，今回は文献 [8] で紹介されている $\log n$ 規定集合を利用したポインタジャンピング手法の改善により最適なプログラムに改良することを可能にした．この並列アルゴリズムは最適な並列アルゴリズムであることが定理からも理解できる．

参考文献

[1] F. Y. Chin, J. Lam and I. Chen, Efficient parallel algorithms for some graph problems, *Communications of the ACM*, **25**, 9 (1982).

[2] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York (1988).

[3] A. Gibbons and W. Rytter, *Efficient parallel algorithms*, Cambridge University Press (1988).

[4] H. Honma and S. Masuyama, *An $O(\log\ n)$ time parallel algorithm for constructing a spanning forest on Trapezoid graphs*, 釧路高専紀要第 31 号, (1997) 79-84.

[5] J. JáJá, *An Introduction to parallel algorithms*, Addison-Wesley Publishing Company (1992).

[6] P. Klein and C. Stein, A parallel algorithm for eliminating cycle in undirected graphs, *Information processing. letters.*, **34** (1990) 307-312.

[7] Y. D. Liang, Dominations in trapezoid graphs, *Information processing. letters.*, **52** (1994) 309-315.

[8] S. Miyano 並列アルゴリズム，近代科学社 (1993).