Scientific Research Publishing

Journal of Applied Mathematics and Physics, 2018, 6, 1649-1658
http://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

An Optimal Parallel Algorithm for Constructing a Spanning Tree on Proper Circle Trapezoid Graphs

Hirotoshi Honma, Yoko Nakajima, Shino Nagasaki, Atsushi Sasaki

National Institute of Technology, Kushiro College, Kushiro, Japan
Email: honma@kushiro-ct.ac.jp, yoko@kushiro-ct.ac.jp, sasaki@kushiro-ct.ac.jp

How to cite this paper: Honma, H., Nakajima, Y., Nagasaki, S. and Sasaki, A. (2018) An Optimal Parallel Algorithm for Constructing a Spanning Tree on Proper Circle Trapezoid Graphs. *Journal of Applied Mathematics and Physics*, **6**, 1649-1658.
https://doi.org/10.4236/jamp.2018.68141

Received: July 4, 2018
Accepted: August 11, 2018
Published: August 14, 2018

Abstract

Given a simple graph G with n vertices and m edges, the spanning tree problem is to find a spanning tree for a given graph G. This problem has many applications, such as electric power systems, computer network design and circuit analysis. For a simple graph, the spanning tree problem can be solved in $O(\log n)$ time with $O(n+m)$ processors on the CRCW PRAM. In general, it is known that more efficient parallel algorithms can be developed by restricting classes of graphs. In this paper, we shall propose a parallel algorithm which runs $O(\log n)$ time with $O(n/\log n)$ processors on the EREW PRAM for constructing on proper circle trapezoid graphs.

Keywords

Design and Analysis of Parallel Algorithms, Proper Circle Trapezoid Graphs, Spanning Tree

1. Introduction

Given a simple connected graph G with n vertices, the *spanning tree problem* is to find a tree that connects all the vertices of G. The spanning tree problem has applications, such as electric power systems, computer network design and circuit analysis [1]. A spanning tree can be found in $O(n+m)$ time using, for example, the depth-first search or breadth-first search. In recent years, a large number of studies have been made to parallelize known sequential algorithms. For simple graphs, Chin *et al.* presented that the spanning forest can be found in $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ processors [2]. Moreover, for a connected graph, Klein and Stein demonstrated that a spanning tree can be found in

$O(\log n)$ time with $O(n+m)$ processors on the CRCW (Concurrent Read Concurrent Write) PRAM (Parallel Random Access Machine) [3].

In general, it is known that more efficient algorithms can be developed by restricting classes of graphs. For instance, Wang *et al*. proposed an optimal parallel algorithm for constructing a spanning tree on *permutation graphs* that run in $O(\log n)$ time using $O(n/\log n)$ processors on the EREW (Exclusive Read Exclusive Write) PRAM [4]. Wang *et al*. proposed optimal parallel algorithms for some problems including the spanning tree problem on *interval graphs* that can be executed in $O(\log n)$ time with $O(n/\log n)$ processors on the EREW PRAM [5]. Bera *et al*. presented an optimal parallel algorithm for finding a spanning tree on *trapezoid graphs* that take in $O(\log n)$ time using $O(n/\log n)$ processors on the EREW PRAM [6]. In addition, Honma *et al*. developed parallel algorithms for finding a spanning tree on *circular permutation graphs* [7] and *circular trapezoid graphs* [8]. Both of them take in $O(\log n)$ time using $O(n/\log n)$ processors on the EREW PRAM.

Felsner *et al*. first introduced *circle trapezoid graphs* [9]. They also provided an $O(n^2)$ time algorithm for solving maximum independent set problem and $O(n^2 \log n)$ time algorithm for solving maximum clique problem. Recently, Lin showed that circle trapezoid graphs are superclasses of trapezoid graphs [10].

In this study, we propose a parallel algorithm for spanning tree problem on a proper circle trapezoid graph. It can run in $O(\log n)$ time with $O(n/\log n)$ processors on the EREW PRAM. The rest of this paper is organized as follows. Section 2 describes some definitions of circle trapezoid graphs and models and introduces the extended circle trapezoid model, as well as some notation. Section 3 presents some properties on circle trapezoid graphs, which are useful for finding a spanning tree in an efficient manner. Section 4 describes our parallel algorithm for the spanning tree problem and its complexity. Finally, Section 5 concludes the paper.

2. Preliminaries

2.1. Circle Trapezoid Model and Graph

We first illustrate the circle trapezoid model before defining the circle trapezoid graph. There is a unit circle C such that the consecutive integer i, $1 \le i \le 4n$ are assigned clockwise on the circumference (n is the number of circle trapezoids). Consider nonintersecting two arcs $A' = [a_i b_i]$ and $A'' = [c_i d_i]$ along the circumference of C. The point b_i (resp., d_i) is the last point encountered when traversing A' (resp., A'') clockwise. A circle trapezoid CT_i is the region in a circle C that lies between two non-crossing chords $\langle a_i d_i \rangle$ and $\langle b_i c_i \rangle$. Without loss of generality, each circle trapezoid CT_i has four corner points a_i, b_i, c_i, d_i, and all corner points are distinct. We assume that circle trapezoids are labeled in increasing order of their corner points a_i's, *i.e.*, $CT_i < CT_j$ if $a_i < a_j$. The geometric representation described above is called the *circle trapezoid model*.

Figure 1(a) illustrates an example of a circle trapezoid model M with eight circle trapezoids. The circle trapezoid with $a_i > d_i$ is called *feedback circle trapezoid*. Note that there exist two feedback circle trapezoids (CT_7, CT_8) in a circle trapezoid model M.

An undirected graph G is a *circle trapezoid graph* if it can be represented by the following circle trapezoid model; each vertex of the graph corresponds to a circle trapezoid in circle trapezoid model, and two vertices are adjacent in G if and only if their circle trapezoids intersect [9]. Figure 1(b) illustrates a circle trapezoid graph G corresponding to M shown in Figure 1(a). Table 1 shows the details of circle trapezoid model M of Figure 1.

2.2. Extended Circle Trapezoid Model

In the following, we introduce the *extended circle trapezoid model EM* constructed from a circle trapezoid model for making the problem easier. We first cut a circle trapezoid model M at point 1 on the circumference and next unroll onto the real horizontal line. Each circle trapezoid $CT_i = [a_i, b_i, c_i, d_i]$ in M is also changed to a pair of line segment $I_i = ([a_i, d_i], [b_i, d_i])$ called *interval pair* by executing the above process. Here, feedback circle trapezoid $CT_i = [a_i, b_i, c_i, d_i]$ in M is changed to interval pair $I_i = ([a_i, d_i + 4n], [b_i + 4n, d_i + 4n])$ for $a_i > b_i, c_i, d_i$. Moreover, copies I_{i-n} of I_i are created by shifting $4n$ to the left respectively, for each I_i, $1 \le i \le n$. Note that both interval pairs I_i and I_{i-n} in extended circle trapezoid model *EM* are corresponding to CT_i in M.

The following Algorithm CEM constructs an *EM* from a M. Figure 2 shows the *EM* constructed from the M illustrated in Figure 1. Table 2 shows the details of *EM* of Figure 2.

Algorithm CEM

Input: Corner points $[a_i, b_i, c_i, d_i]$ of CT_i in M.

Begin

For each non feedback circle trapezoid CT_i **pardo**

Create a interval pair $I_i = ([a_i, d_i], [b_i, d_i])$;

For each feedback circle trapezoid CT_i **pardo**

For each $b_i, c_i, d_i < a_i$

Create a interval pair $I_i = ([a_i, d_i + 4n], [b_i + 4n, d_i + 4n])$;

For $1 \le i \le n$ **pardo**

Create copies I_{i-n} by shifting $4n$ to the left for I_i;

End

3. Proper Circle Trapezoid Model and Graph

3.1. Definitions for Proper Circle Trapezoid Graph

In this study, we focus and treat a proper circle trapezoid graph. Graph G is a circle trapezoid graph corresponding to a circle trapezoid model M and an extended circle trapezoid model *EM* is constructed from M by executing Algorithm CEM. We consider circle trapezoid model M such that the extended

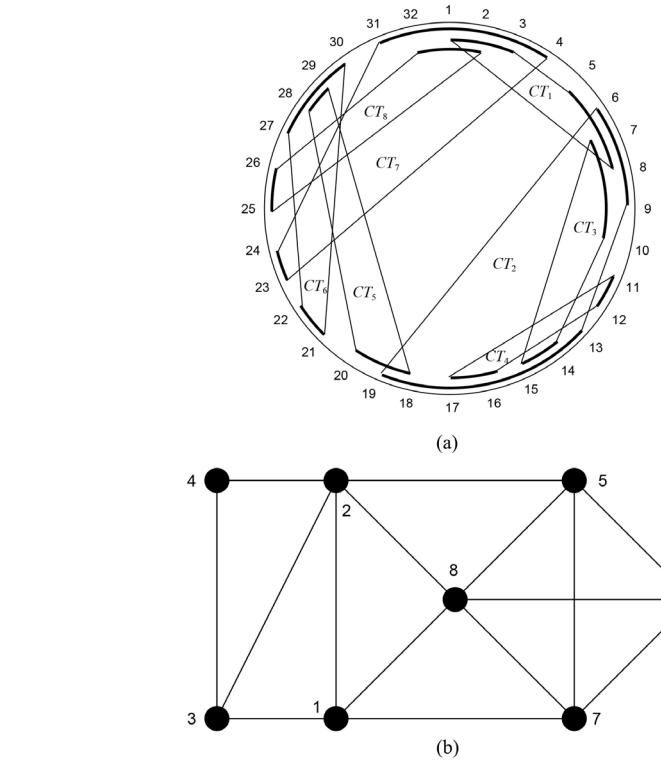

Figure 1. Circle trapezoid model and graph. (a) Circle trapezoid model *M*; (b) Circle trapezoid graph *G*.

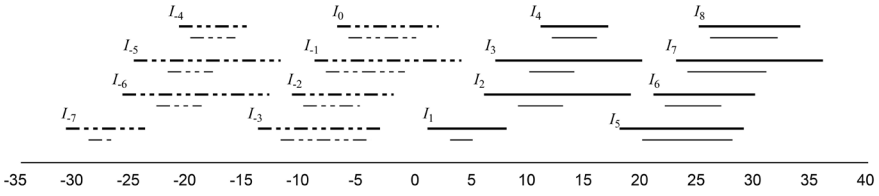

Figure 2. Extended circle trapezoid model *EM*.

Table 1. Details of circle trapezoid model *M*.

i	1	2	3	4	5	6	7	8
a_i	1	6	7	11	18	21	23	25
b_i	3	9	10	12	20	22	24	26
c_i	5	13	14	16	28	27	31	32
d_i	8	19	15	17	29	30	4	2

Table 2. Details of extended circle trapezoid model *EM*.

i	−7	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6	7	8
a_i	−31	−26	−25	−21	−14	−11	−9	−7	1	6	7	11	18	21	23	25
b_i	−29	−23	−22	−20	−12	−10	−8	−6	3	9	10	12	20	22	24	26
c_i	−27	−19	−18	−16	−4	−5	−1	0	5	13	14	16	28	27	31	32
d_i	−24	−13	−17	−15	−3	−2	4	2	8	19	15	17	29	30	36	34

circle trapezoid model *EM* constructed from *M* satisfies that $c_i < d_j$ for two interval pairs I_i and I_j $(i<j)$ in *EM*. The circle trapezoid model M_p is defined as *proper circle trapezoid model*. The graph corresponding to the proper circle trapezoid model is *proper circle trapezoid graph* G_p. In this study, we will develop a parallel algorithm for spanning tree problem on proper circle trapezoid graphs. The Figure 1 is also an example of M_p and G_p because $c_i < d_j$ for I_i and I_j $(i<j)$ in extended circle trapezoid model *EM*.

Here, some notations that form the basis of our algorithm are defined as follows.

The function $\mathrm{nor}(i)$ normalizes the interval pair number *i* in *EM* within the range 1 to *n*, which is expressed as

$$\mathrm{nor}(i)=\begin{cases} i & \text{if } i \geq 1, \\ i+n & \text{if } i<1. \end{cases}$$

For the example shown in Figure 2, for $i=4$ and $i=-5$, we have $\mathrm{nor}(4)=4$ and $v(-5)=3$, respectively.

The function $v_d(k)$ computes a vertex number *i* satisfying $d_i = k$ for a given number *k* on *EM*. For the example shown in Figure 2, for $k=29$ and $k=-13$, we have $v_d(29)=5$ and $v_d(-13)=-6$ by $d_5=29$ and $d_{-6}=-13$, respectively. Moreover, we use $nv_d(k)$ instead of $\mathrm{nor}(v_d(k))$ for simplicity. For the example shown in Figure 2, for $k=29$ and $k=-13$, we have $nv_d(29)=5$ and $nv_d(-13)=2$ by $v_d(29)=5$ and $v_d(-13)=-6$, respectively.

We next define $ld_i = \max\{d_{-n+1}, d_{-n+2}, \cdots, d_i\}$, for $-n+1 \leq i \leq n-1$, in EM_p. The details of ld_i, $v_d(d_i)$, and $nv_d(d_i)$ are shown in Table 3.

3.2. Property of Proper Circle Trapezoid Graph

We describe some properties on circle trapezoid graphs which are useful for constructing the algorithm for spanning tree problem on proper circle trapezoid graphs.

For two interval pairs I_i and I_j $(i<j)$ in *EM*, we say I_i and I_j are *disjoint* if $d_i < a_j$. Moreover, we say I_i *contain* I_j if $b_i < a_j$ and $d_j < c_i$. Figure 3 shows examples of the cases of disjoint and contain. The following Lemma 1 has been described in [9].

Table 3. Details of extended circle trapezoid model *EM*.

i	−7	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6	7	8
d_i	−24	−13	−17	−15	−3	−2	4	2	8	19	15	17	29	30	36	34
ld_i	−24	−13	−13	−13	−3	−2	4	4	8	19	19	19	29	30	36	-
$v_d(ld_i)$	−7	−6	−6	−6	−3	−2	−1	−1	1	2	2	2	5	6	7	-
$nv_d(ld_i)$	1	2	2	2	5	6	7	7	1	2	2	2	5	6	7	-

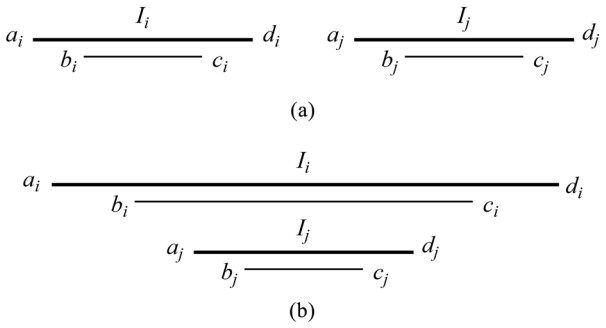

Figure 3. Examples of disjoint and contain. (a) I_i and I_j are disjoint ($d_i < a_j$); (b) I_i contains I_j ($b_i < a_j$ and $c_i > d_j$).

Lemma 1 *Let* CT_i *and* CT_j $(i < j)$ *be non-feedback circle trapezoids in circle trapezoid model M. Moreover, extended circle trapezoid model EM is constructed from M.* CT_i *and* CT_j *intersect if* I_i *and* I_j *are not disjoint and* I_i *does not contain* I_j *in EM.*

The following Lemma 2 generalizes Lemma 1. This is very useful to find the edges on circle trapezoid graph.

Lemma 2 G is a circle trapezoid graph corresponding to a circle trapezoid model M, and extended circle trapezoid model EM is constructed from M. For two interval pairs I_i, I_j $(i < j)$, an edge $(\mathrm{nor}(i), \mathrm{nor}(j))$ is in G if and only if at least one of the following conditions satisfies in EM;

1) $b_i > a_j$,

2) $d_i > a_j$ and $c_i < d_j$.

(Proof) By Lemma 1, for two non-feedback circle trapezoids CT_i and CT_j do not intersect if and only if ($d_i < a_j$) or ($b_i < a_j$ and $c_i > d_j$) in EM. By the contra position, for two non-feedback circle trapezoids CT_i and CT_j intersect if and only if ($d_i > a_j$) and ($b_i > a_j$ or $c_i < d_j$) in EM. Here, ($d_i > a_j$) and ($b_i > a_j$ or $c_i > d_j$) is logically equal to ($d_i > a_j$ and $b_i > a_j$) or ($d_i > a_j$ and $c_i < d_j$).

For the condition ($d_i > a_j$) and ($b_i > a_j$), we have $b_i > a_j$ whenever $d_i > a_j$. Thus, $(\mathrm{nor}(i), \mathrm{nor}(j))$ is an edge of CTG G if ($b_i > a_j$) or ($d_i > a_j$ and $c_i < d_j$) for two interval pairs I_i and I_j $(i < j)$ in EM. □

We obtain the following Lemma 3 for a proper circle trapezoid model M_p and graph G_p.

Lemma 3. G_p is a proper circle trapezoid graph corresponding to a proper circle trapezoid model M_p, and EM_p is an extended circle trapezoid model constructed from M_p. An edge $(\mathrm{nor}(i), \mathrm{nor}(j))$ is in G_p if and only if $d_i > a_j$ for $i < j$ satisfies in the EM_p.

(Proof) By Lemma 2, if either of conditions 1) ($b_i > a_j$) or 2) ($d_i > a_j$ and $c_i < d_j$) for two interval pairs I_i and I_j $(i < j)$ in EM_p, an edge $(\mathrm{nor}(i), \mathrm{nor}(j))$ is in G_p. By definition of proper circle trapezoid graph G_p, we have $c_i < d_j$ in EM_p. Hence, condition 2) satisfies when $d_i > a_j$ holds.

Moreover, condition 1) $b_i > a_j$ satisfies if $d_i > a_j$ holds because $a_i < b_i < c_i < d_i$ by the definition of interval pair. Therefore an edge $(\mathrm{nor}(i), \mathrm{nor}(j))$ is in G_p if and only if $d_i > a_j$ for $i < j$ satisfies in the EM_p. □

In Lemma 2, we have to test if $b_i > a_j$ or ($d_i > a_j$ and $c_i < d_j$) in EM to check whether (i, j) is an edge of normal circle trapezoid graph G. On the other hand, by Lemma 3, we only need to check $d_i > a_j$ holds for EM_p to determine whether an edge (i, j) is in proper circle trapezoid graph G_p.

The following Lemma 4 is core of solving this problem. An efficient algorithm can be constructed by using the following lemma.

Lemma 4. G_p is a proper circle trapezoid graph corresponding to an M_p, and EM_p is an extended circle trapezoid model constructed from M_p. For $1 \le i \le n$, an edge $(nv_d(ld_{i-1}), i)$ is in G_p if $ld_{i-1} > a_i$ satisfies in the EM_p.

(Proof) By the definition, $ld_{i-1} = \max\{d_{-n+1}, d_{-n+2}, \cdots, d_{i-1}\}$. Thus, we have $v_d(ld_{i-1}) \le i-1$. By Lemma 3, an edge $(\mathrm{nor}(i), \mathrm{nor}(j))$ is in G_p if and only if $d_i > a_j$ for $i < j$. Therefore, an edge $(nv_d(ld_{i-1}), i)$ is in G_p if $ld_{i-1} > a_i$ satisfies in the EM_p. □

4. Parallel Algorithm

In this section, we propose an algorithm for constructing a spanning tree of a connected proper circle trapezoid graph G_p. We assume that all trapezoids in the proper circle trapezoid model have been sorted by corner point *a* in ascending order, that is, Table 1 is given as an input of our algorithm. Algorithm CST returns a spanning tree if a given graph G_p is connected. Instead of using a sophisticated technique, we propose simple parallel algorithms using only the parallel prefix computation [11] and Brent's scheduling principle [12].

Algorithm CST

Input: a_i and b_i, $1 \le i \le n$.

Output: A spanning forest F of G. Initially F be an empty set.

Begin

(Step 1) % Initializing

$F := \varnothing$;

For *i*, $1 \le i \le n$ **pardo**

$M[i] := 0$;

For *i*, $1 \le i \le 2n$ **pardo**

$P[i] := 0$;

(Step 2) % Computing $M[i]$

For *i*, $1 \le i \le n$ **pardo**

$P[a_i] := i$;

$P[b_i] := i$;

For *i*, $1 \le i \le 2n$ **pardo**

$P[i] := \max\{P[1], P[2], \cdots, P[i]\}$;

For *i*, $1 \le i \le n$ **pardo**

$M[i] := P[b_i]$;

(Step 3) % Construct a spanning forest
For i, $1 \le i \le n$ **pardo**
If $i > M[i]$
then $F := F \cup (i, M[i])$;
End

Lemma 5. After executing Step 3 of Algorithm CST, graph T is a spanning tree of proper circle trapezoid graph G_p.

(Proof) Step 1 is a process for initialization. T is empty set and all ck_i are set to "0". For all $-n+1 \le i \le n-1$, compute $ld_i := \max\{d_{-n+1}, d-n+2, \cdots, d_i\}$ using parallel prefix computation [11].

In Step 2, we set $ck_i = 1$, $1 \le i \le n$, if $ld_{i-1} > a_i$. In addition, we compute $s := \max\{nv_d(ld_{i-1}) \mid nv_d(ld_{i-1}) > i, ck_i = 1\}$. By Lemma 4, an edge $(nv_d(ld_{i-1}), i)$ is in G_p if $ld_{i-1} > a_i$. Thus, a vertex i that $ck_i = 1$ can have least one edge from i to other vertex $nv_d(ld_{i-1})$. Vertex s is the largest $nv_d(ld_{i-1})$ satisfying $nv_d(ld_{i-1}) > i$. For the example shown in **Figure 4**, we set $ck_i = 1$ because $ld_{i-1} > a_i$, for $1 \le i \le n$. For the only case of $i = 1$, we have $nv_d(ld_{i-1}) > i$ then $s = nv_d(ld_0) = 7$.

We consider that T is added an edge form i to $v_d(ld_{i-1})$ smaller for i, $1 \le i \le n$. By definitions of v_d and ld, if $v_d(ld_{i-1})$ corresponds a copy of a non-feedback circle trapezoid, we have $nv_d(ld_{i-1}) < i$. On the other hand, if $v_d(ld_{i-1})$ corresponds a copy of a feedback circle trapezoid, we have $nv_d(ld_{i-1}) > i$. In the case of $\sum_{i=1}^{n} ck_i = n$, T constructed in above way is connected graph that has n vertices. Thus, T is not a tree that has exactly one cycle C. There exist some edge $(nv_d(ld_{i-1}), i)$ in C such that $nv_d(ld_{i-1})$ is feedback circle trapezoid, because a given G_p is connected. In Step 2, we obtained $s = \max\{nv_d(ld_{i-1}) \mid nv_d(ld_{i-1}) > i, ck_i = 1\}$. In not adding $(nv_d(ld_{s-1}), s)$ to T, we can remove a cycle C.

i	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
a_i	-31	-26	-25	-21	-14	-11	-9	-7	1	6	7	11	18	21	23	25
dl_i	-24	-13	-13	-13	-3	-2	4	4	8	19	19	19	29	30	36	
ck_i									1	1	1	1	1	1	1	1

i	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
$nv_d(dl_i)$								7	1	2	2	2	5	6	7	

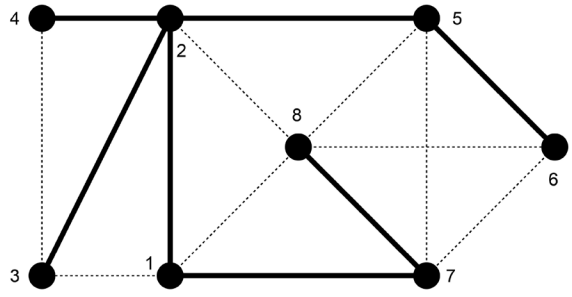

Figure 4. Example of constructed spanning tree.

In Step 3, in the case of $\sum_{i=1}^{n} ck_i = n$, we add an edge $\left(nv_d\left(ld_{i-1}\right), i\right)$ to T for vertex i, $1 \le i \le n$, $i \ne s$. T is connected with $n-1$ vertices, that is, T is a tree. In Step 3, we consider the case of $\sum_{i=1}^{n} ck_i \ne n$. This implies $\sum_{i=1}^{n} ck_i = n-1$. If $\sum_{i=1}^{n} ck_i < n-1$, this means that a given G_p is disconnected, which is a contradiction to our hypothesis. Therefore, in the case of $\sum_{i=1}^{n} ck_i = n$, we add an edge $\left(nv_d\left(ld_{i-1}\right), i\right)$ to T for vertex i, $1 \le i \le n$, $ck_i = 1$. T is connected with $n-1$ vertices, that is, T is a tree.

Therefore, after executing Step 3 of Algorithm CST, graph T is a spanning tree of proper circle trapezoid graph G_p. □

Figure 4 shows a spanning tree T constructed from CTG G_p by executing Algorithm CST.

In the following, we analyze the complexity of Algorithm CST.

In Step 1, an extended circle trapezoid model is constructed from a circle trapezoid model in $O(1)$ time using $O(n)$ processors, which can be implemented in $O(\log n)$ time using $O(n/\log n)$ processors by applying Brent's scheduling principle [12]. Moreover, all rd_i are obtained in $O(\log n)$ time using $O(n/\log n)$ processors by applying parallel prefix computation [11]. In Step 2, ck_i and s are computed in $O(\log n)$ time using $O(n/\log n)$ processors by applying Brent's scheduling principle. Step 3 can also be implemented in $O(\log n)$ time using $O(n/\log n)$ processors by applying Brent's scheduling principle. In addition, Algorithm CST can be executed on an EREW PRAM because neither concurrent read nor concurrent write are necessary. Thus, we have the subsequent theorem.

Theorem 6. Algorithm CST constructs a spanning tree of a proper circle trapezoid graph in $O(\log n)$ time using $O(n/\log n)$ processors on EREW PRAM. □

5. Concluding Remarks

In this paper, we presented a parallel algorithm to solve the spanning tree problem on a proper circle trapezoid graph. This algorithm can be implemented in $O(\log n)$ time with $O(n/\log n)$ processors on an EREW PRAM computation model using only parallel prefix computation [11] and Brent's scheduling principle [12] without using a sophisticated technique. Solutions to the spanning problem have applications in electrical power provision, computer network design, and circuit analysis, among others. For this reason, we think this paper is also worthy from both a theoretical and algorithmic point of view. In the future, we will continue this research by extending the results to other classes of graphs.

Acknowledgements

We express many thanks to anonymous referees for their valuable advices on the theory of our attacks and their helpful editorial comments. This work was supported by JSPS KAKENHI Grant Number 17K00324 and 17K04965.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Wilson, R.J. (1996) Introduction to Graph Theory. Prentice Hall, Upper Saddle River.

[2] Chin, F.Y., Lam, J. and Chen, I. (1982) Efficient Parallel Algorithms for Some Graph Problems. *Communications of the ACM*, **25**, 659-665. https://doi.org/10.1145/358628.358650

[3] Klein, P. and Stein, C. (1990) A Parallel Algorithm for Eliminating Cycle in Undirected Graphs. *Information Processing Letters*, **34**, 307-312. https://doi.org/10.1016/0020-0190(90)90015-P

[4] Wang, Y. L., Chen, H. C. and Lee, C. Y. (1995) An $O(\log n)$ Parallel Algorithm for Constructing a Spanning Tree on Permutation Graphs. *Information Processing Letters*, **56**, 83-87. https://doi.org/10.1016/0020-0190(95)00125-V

[5] Wang, Y. L., Chiang, K. C. and Yu, M. S. (1998) Optimal Algorithms for Interval Graphs. *Journal of Information Science and Engineering*, **14**, 449-459.

[6] Bera, D., Pal, M. and Pal, T. K. (2003) An Optimal PRAM Algorithm for a Spanning Tree on Trapezoid Graphs. *Journal of Applied Mathematics and Computing*, **1-2**, 21-29. https://doi.org/10.1007/BF02936178

[7] Honma, H., Honma, S. and Masuyama, S. (2009) An Optimal Parallel Algorithm for Constructing a Spanning Tree on Circular Permutation Graphs. *IEICE Transactions on Information and Systems*, **E92-D**, 141-148. https://doi.org/10.1587/transinf.E92.D.141

[8] Honma, H., Nakajima, Y., Igarashi, Y. and Masuyama, S. (2014) Algorithm for Finding Maximum Detour Hinge Vertices of Interval Graphs. *IEICE Transactions on Fundamentals of Electronics*, **E97-A**, 1365-1369. https://doi.org/10.1587/transfun.E97.A.1365

[9] Felsner, S., Müller, R. and Wernisch, L. (1997) Trapezoid Graphs and Generalization, Geometry and Algorithms. *Discrete Applied Mathematics*, **74**, 13-32. https://doi.org/10.1016/S0166-218X(96)00013-3

[10] Lin, W. L. (2006) Circular and Circle Trapezoid Graphs. *Journal of Science and Engineering Technology*, **2**, 11-17.

[11] Gibbons, A. and Rytter, W. (1988) Efficient Parallel Algorithms. Cambridge University Press, Cambridge.

[12] Brent, R. P. (1974) The Parallel Evaluation of General Arithmetic Expressions. *Journal of the ACM*, **21**, 201-206. https://doi.org/10.1145/321812.321815