Journal of Computer and Communications, 2017, 5, 48-55
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

A Parallel Algorithm for the Spanning Forest Problem on Proper Circle Graphs

Hirotoshi Honma, Yoko Nakajima, 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. and Sasaki, A. (2017) A Parallel Algorithm for the Spanning Forest Problem on Proper Circle Graphs. *Journal of Computer and Communications*, **5**, 48-55.
https://doi.org/10.4236/jcc.2017.514005

Received: October 6, 2017
Accepted: December 23, 2017
Published: December 26, 2017

Abstract

Given a simple graph G with n vertices, m edges and k connected components. The spanning forest problem is to find a spanning tree for each connected component of G. This problem has applications to the electrical power demand problem, computer network design, circuit analysis, etc. In this paper, we present an $O(\log n)$ time parallel algorithm with $O(n/\log n)$ processors for constructing a spanning forest on proper circle graph G on EREW PRAM.

Keywords

Design and Analysis of Parallel Algorithms, Proper Circle Graphs, Spanning Forest

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. Generally, there exist a number of different spanning trees in a connected graph. Let T be a tree with n vertices. Then the following statements are equivalent [1]:

1) T contains no cycles, and has $n-1$ edges;

2) T is connected, and has $n-1$ edges;

3) T is connected, and each edge is a bridge;

4) Any two vertices of T are connected by exactly one path;

5) T contains no cycles, but the addition of any new edge creates exactly one cycle.

Given a simple graph G with n vertices and k components, the spanning forest problem is to find a spanning tree for each component of G. The spanning forest F has $n-k$ edges. This 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 runs 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 algorithms for finding a spanning tree on *trapezoid graphs* that takes 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.

Let $\mathscr{F}$ be a family of nonempty sets. A simple graph G is the *intersection graph* of $\mathscr{F}$ if there exists a one-to-one correspondence between the vertices of G and the sets in $\mathscr{F}$, such that two vertices in G are adjacent if and only if their corresponding sets have a nonempty intersection. A circle graph is an undirected graph isomorphic to the intersection graph of a finite set of chords in a circle. It is not difficult to show that the class of circle graphs contains all the complete bipartite graphs. Circle graphs have been introduced by Even and Itai in [9] in connection with algorithms that sort permutations by using stacks. This aspect is detailed in the book by Golumbic [10]. Polynomial time algorithms for recognizing graphs in this class appear in [11] [12]. Applications of circle graphs are diverse, and without trying to be exhaustive, we can cite container ship stowage [13] and reconstruction of long DNA strings from short subsequences [14]. The best algorithms for recognizing circle graphs and constructing chord diagrams are by Gabor *et al.* [12] taking time $O(mn)$ where n is the number of vertices and m the number of edges, and by Spinrad [15] taking time $O(n^2)$. Gavril [16] presented a $O(n^3)$ time algorithm for finding a maximum independent set of a circle graph. $O(n^2)$ time algorithms were developed by other researchers [17] [18]. Valiente [19] solved the problem in $O(nd)$ time and only $O(n)$ space, where d is a parameter known as the density of the circle graph. Other problems that are NP-complete for general graphs can also be solved in polynomial time for circle graphs, for example Tiskin [20] has shown an $O(n\log^2 d)$ time algorithm for the maximum clique problem. There are also problems that are NP-complete for both circle graphs and general graphs, such as minimum dominating set [21].

In this study, we define proper circle graphs that are subclass of circle graphs. We propose a parallel algorithm for spanning forest problem on a proper circle graphs. It can run in $O(\log n)$ time with $O(n/\log n)$ processors on the EREW PRAM.

2. Preliminaries

2.1. Proper Circle Graphs

We first illustrate the chord diagram before defining the circle graph. There is a unit circle C such that the consecutive integer i, $1 \le i \le 2n$ are assigned clockwise on the circumference (n is the number of chords). A chord i is the line in a circle C that connects two points a_i and b_i $(a_i < b_i)$ on C and is denoted by $[a_i, b_i]$. Without loss of generality, each chord i has two terminal points a_i and b_i, and all terminal points are distinct. We assume that chords are labeled in increasing order of their corner points b_i's, *i.e.*, $i < j$ if $b_i < b_j$. The geometric representation described above is called the *chord diagram*. We next introduce the circle graphs. An undirected graph G_c is a *circle graph* if it can be represented by the following chord diagram; each vertex of the graph corresponds to a chord in the chord diagram, and two vertices are adjacent in G_c if and only if their chords intersect.

In the following, we define proper circle graphs. If $a_i < a_j$ for any two chords i and j $(i < j)$ in the chord diagram, such chord diagram is called "proper." The proper circle graph is a graph that is constructed by a proper chord diagram. Figure 1(a) illustrates an example of a proper chord diagram D with 12 chords. Figure 1(b) illustrates a proper circle graph G corresponding to D shown in Figure 1(a). Table 1 shows the details of the CD D of Figure 1.

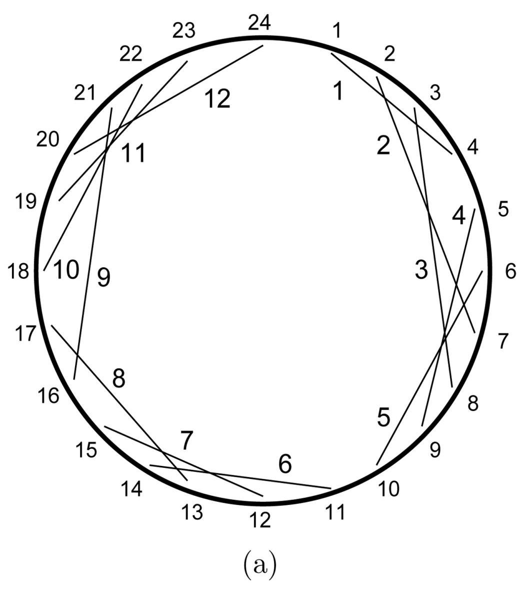
2.2. Expanded Chord Diagram

In the following, we introduce the *expanded chord diagram* constructed from a chord diagram for solving the problem easier. An expanded chord diagram can be constructed by transforming chord $[a_i, b_i]$ in chord diagram to a horizontal line segment such that left and right endpoints have values a_i and b_i, respectively. Figure 2 shows the expanded chord diagram ED constructed from the proper chord diagram D illustrated in Figure 1.

3. Property of Proper Circle Graph

We describe some properties on the proper circle graph which are useful for constructing the algorithm for spanning forest problem. For two chords i and j $(i < j)$ in the (normal) chord diagram, we say chords i and j are disjoint if $b_i < a_j$. Moreover, we say chord j contain i if $a_j < a_i$. Figure 3 shows examples of the cases of disjoint and contain.

The following Lemma 1 is well known [22].



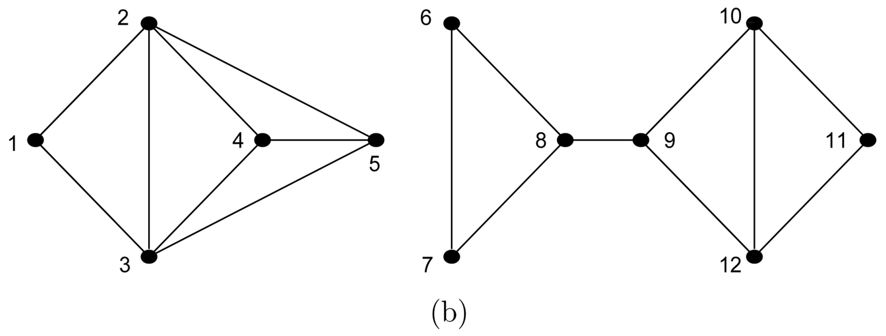


Figure 1. Proper chord diagram and circle graph: (a) Proper chord diagram *D*; (b) Proper circle graph *G*.

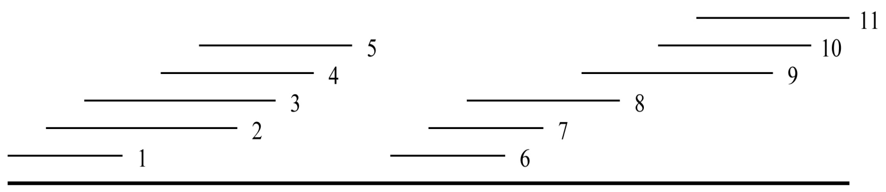


Figure 2. Expanded chord diagram *ED*.

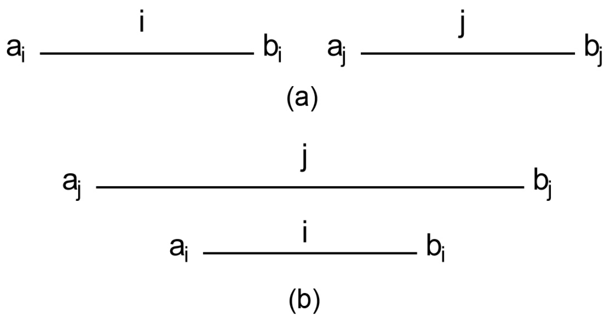


Figure 3. Examples of disjoint and contain: (a) Chords *i* and *j* are disjoint; (b) Chord *j* contains *i*.

Table 1. Details of proper chord diagram *D*.

i	1	2	3	4	5	6	7	8	9	10	11	12
a_i	1	2	3	5	6	11	12	13	16	18	19	20
b_i	4	7	8	9	10	14	15	17	21	22	23	24

Lemma 1. Let D be chord diagram and G_c be a circle graph corresponding to D. Moreover, let ED be an expanded chord diagram constructed from D. Then, vertices i and j $(i < j)$ are adjacent in G_c if and only if chord i and j are not disjoint and i does not contains j in ED.

We can establish the following lemma using Lemma 1.

Lemma 2. Let D be proper chord diagram and G be a proper circle graph corresponding to D. Moreover, let ED be an expanded chord diagram constructed from D. Then, vertices i and j $(i < j)$ are adjacent in G if $a_j < b_i$ for two chords i and j in ED.

(Proof) By Lemma 1, vertices v_i and v_j $(i < j)$ are adjacent in G_c if and only if chord i and j are not disjoint and i does not contains j in ED. By the definition of proper circle graph, $a_i < a_j$ for any two chords i and j $(i < j)$ in the chord diagram. Therefore, $a_j < a_i$ (that is, contain) never occur in proper chord diagram. Then, vertices i and j $(i < j)$ are adjacent in G if $a_j < b_i$ for two chords i and j in ED. □

4. Parallel Algorithm for Spanning Forest Problem

4.1. Algorithm CSF

In this section we propose a parallel algorithm for constructing a spanning forest F of a given proper circle graph G. The spanning forest F is constructed from proper chord diagram D corresponding to G. Input of the algorithm is each chord terminal points a_i and b_i of D. All chords have been sorted by corner point b_i in increasing order. Instead of using a sophisticated technique, we propose simple parallel algorithms using only the parallel prefix computation [23] and Brent's scheduling principle [24]. After executing Step 3, F consists of the spanning forest of proper circle graph G.

Algorithm CSF

```
Input: $a_i$ and $b_i$, $1 \le i \le n$.
Output: A spanning forest $F$ of $G$. Initially $F$ be an empty set.
(Step 1) % Initializing
$F := \emptyset$ ;
  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**
    $P[i] := \max\{P[1], P[2], \ldots, P[i]\}$ ;
  **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 of Algorithm CSF**
```

Lemma 3. After executing Step 3 of Algorithm CSF, F consists of the spanning forest of proper circle graph *G*.

(Proof) In Step 2 of Algorithm CSF, we compute an array $M(i)$ for $1 \le i \le n$. $M(i)$ is the largest chord number that intersects chord *i*. This means $(i, M(i))$ is an edge in proper circle graph G and $M(i)$ is the largest vertex adjacent to *i*.

In Step 3 of Algorithm CSF, we add an edge in G if $i > M[i]$. A graph that adds just one edge to a vertex larger than itself from each vertex is connected. Moreover, $M(i)$ of root vertices have same value as themselves. Then, after executing Step 3, each component in G is connected and has n' vertices and $n'-1$ edges. By the properties of tree, F is a spanning forest of *G*. □

Table 2 and Figure 4 show details of arrays executing Step 3 and constructed *F*.

4.2. Analysis of Algorithm CSF

We analyze the complexity of Algorithm CSF. In Step 1, we initialize F and $M[i]$. This step can be implemented in $O(\log n)$ time using $O(n/\log n)$ processors by applying Brent's scheduling principle [24]. In Steps 2, $M[i]$ is

Table 2. Details of $M(i)$.

i	1	2	3	4	5	6	7	8	9	10	11	12
a_i	1	2	3	5	6	11	12	13	16	18	19	20
b_i	4	7	8	9	10	14	15	17	21	22	23	24
M_i	3	5	5	5	0	8	8	9	12	12	12	0

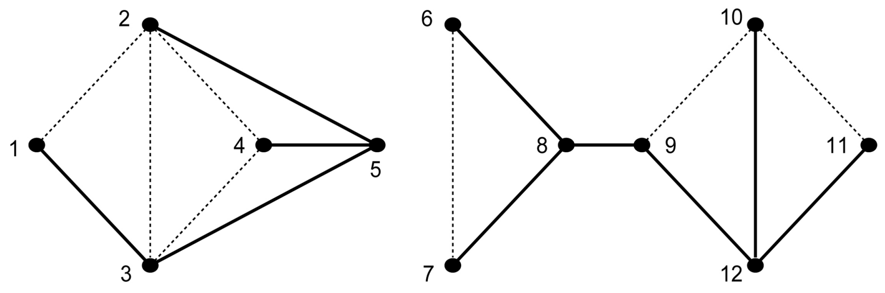


Figure 4. A spanning forest of *G*.

computed in $O(\log n)$ time using $O(n/\log n)$ processors by applying parallel prefix computation [23]. In Step 3, we obtain a spanning forest of G. This step can be done in $O(\log n)$ time using $O(n/\log n)$ processors by applying Brent's scheduling principle. These steps can be implemented in $O(\log n)$ time using $O(n/\log n)$ processors by applying Brent's scheduling principle. In addition, neither concurrent read nor concurrent write is caused anywhere, and the same edges are not written more than once. Hence, Algorithm CSF can be executed on EREW PRAM. We have the following theorem.

Theorem 4. Algorithm CSF constructs a spanning forest of trapezoid graph G in $O(\log n)$ time with $O(n/\log n)$ processors on EREW PRAM.

5. Concluding Remarks

In this paper, we presented a parallel algorithm to solve the spanning forest problem on proper circle graphs. 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 [23] and Brent's scheduling principle [24] without using a sophisticated technique. Solutions to the spanning problem have applications in electrical power provision, computer network design, 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 25330019 and 17K00324 both.

References

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

[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*, **58**, 83-87.

[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, Communications and Computer Sciences*, **E97.A**, 1365-1369. https://doi.org/10.1587/transfun.E97.A.1365

[9] Even, S. and Itai, A. (1971) Theory of Machines and Computations. Academic Press, New York.

[10] Golumbic, M.C. (2004) Algorithmic Graph Theory and Perfect Graphs, Volume 57. 2nd Edition, Elsevier, New York.

[11] Bouchet, A. (1987) Reducing Prime Graphs and Recognizing Circle Graphs. *Combinatorica*, **7**, 243-254. https://doi.org/10.1007/BF02579301

[12] Gabor, C.P., Supowit, K.J. and Hsu, W.L. (1989) Recognizing Circle Graphs in Polynomial Time. *Journal of the ACM*, **36**, 435-473. https://doi.org/10.1145/65950.65951

[13] Avriel, M., Penn, M. and Shpirer, N. (2000) Container Ship Stowage Problem: Complexity and Connection to the Coloring of Circle Graphs. *Discrete Applied Mathematics*, **103**, 271-279. https://doi.org/10.1016/S0166-218X(99)00245-0

[14] Arratia, R., Bollobas, B., Coppersmith, D. and Sorkin, G.B. (2000) Euler Circuits and DNA Sequencing by Hybridization. *Discrete Applied Mathematics*, **104**, 63-96. https://doi.org/10.1016/S0166-218X(00)00190-6

[15] Spinrad, J. (1994) Recognition of Circle Graphs. *Journal of Algorithms*, **16**, 264-282. https://doi.org/10.1006/jagm.1994.1012

[16] Gavril, F. (1973) Algorithms for a Maximum Clique and a Maximum Independent Set of a Circle Graph. *Networks*, **3**, 261-273. https://doi.org/10.1002/net.3230030305

[17] Asano, T., Imai, H. and Mukaiyama, A. (1991) Finding a Maximum Weight Independent Set of a Circle Graph. *IEICE Transactions on Fundamentals*, **E74A**, 681-683.

[18] O.Goldschmidt, A.T. (1994) An Efficient Algorithm for Finding a Maximum Weight Independent Set of a Circle Graph. *IEICE Transactions on Fundamentals*, **E77A**, 1672-1674.

[19] Valiente, G. (2003) A New Simple Algorithm for the Maximum-Weight Independent Set Problem on Circle Graphs, ISAAC. In: *Lecture Notes in Computer Science*, Vol. 2906, Springer, Berlin, 129-137.

[20] Tiskin, A. (2008) Semi-Local String Comparison: Algorithmic Techniques and Applications. *Mathematics in Computer Science*, **1**, 571-603. https://doi.org/10.1007/s11786-007-0033-3

[21] Keil, J.M. (1993) The Complexity of Domination Problems in Circle Graphs. *Discrete Applied Mathematics*, **42**, 51-63. https://doi.org/10.1016/0166-218X(93)90178-Q

[22] Ageev, A.A. (1999) Every Circle Graph of Girth at Least 5 Is 3-Colourable. *Discrete Mathematics*, **195**, 229-233. https://doi.org/10.1016/S0012-365X(98)00192-7

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

[24] 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