Parallel Algorithm for Constructing a Spanning Tree on a Certain Class of Circle Trapezoid Graphs

Hirotoshi HONMA$^1$ Yoko NAKAJIMA$^1$

Abstract: Given a simple graph $G$ with $n$ vertices and $m$ edges. 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]:

(i) $T$ contains no cycles, and has $n - 1$ edges,
(ii) $T$ is connected, and has $n - 1$ edges,
(iii) $T$ is connected, and each edge is a bridge,
(iv) any two vertices of $T$ are connected by exactly one path,
(v) $T$ contains no cycles, but the addition of any new edge creates exactly one cycle.

The spanning tree problem have 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 algorithms for finding a spanning tree on interval graphs that can be executed in $O(n)$ time with $O(n^2 \log n)$ processors on the EREW PRAM [2]. Wang et al. proposed optimal parallel algorithms for finding a spanning tree on trapezoid graphs that takes in $O(n^2 \log n)$ time using $O(n/\log n)$ processors on the EREW PRAM [3]. Bera et al. presented an optimal parallel algorithm for finding a spanning tree on circular permutation graphs [7] and circular trapezoid graphs [8]. Both of them take in $O(n^2 \log n)$ 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 finding a spanning tree on a certain class of circle trapezoid graphs. It can run in $O(n/\log n)$ time with $O(n/\log n)$ processors on the EREW PRAM.

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 $i, 1 \leq i \leq 4n$ 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

---

$^1$Department of Creative Engineering, National Institute of Technology, Kushiro College
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 $(a_i, d_i)$ and $(b_i, c_i)$. 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 (CTM). Figure 1-(a) illustrates an example of CTM $M$ 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 ($CT_i, CT_j$) in CTM $M$.

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

<table>
<thead>
<tr>
<th>$i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_i$</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>11</td>
<td>18</td>
<td>21</td>
<td>23</td>
<td>25</td>
</tr>
<tr>
<td>$b_i$</td>
<td>3</td>
<td>9</td>
<td>10</td>
<td>12</td>
<td>20</td>
<td>22</td>
<td>24</td>
<td>26</td>
</tr>
<tr>
<td>$c_i$</td>
<td>5</td>
<td>13</td>
<td>14</td>
<td>16</td>
<td>28</td>
<td>27</td>
<td>31</td>
<td>32</td>
</tr>
<tr>
<td>$d_i$</td>
<td>8</td>
<td>19</td>
<td>15</td>
<td>17</td>
<td>29</td>
<td>30</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

### 2.2 Extended circle trapezoid model

In the following, we introduce the extended circle trapezoid model (ECTM) constructed from a CTM for making the problem easier. We first cut CTM 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 CTM is also changed to a pair of line segment $I_i = ([a_i, d_i], [b_i, c_i])$ called interval pair by executing the above process. Here, feedback circle trapezoid $CT_i = [a_i, b_i, c_i, d_i]$ in CTM 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 $I_i, 1 \leq i \leq n$. Note that both interval pairs $I_i$ and $I_{i-n}$ in ECTM are corresponding to $CT_i$ in CTM.

The following Algorithm 1 constructs an ECTM from a CTM. Figure 2 shows the ECTM $EM$ constructed from the CTM $M$ illustrated in Fig. 1. Table 2 shows the details of ECTM $EM$ of Figure 2.

### 2.3 Restricted circle trapezoid model and graph

In this study, we focus and treat a certain class of circle trapezoid graphs. Graph $G$ is a CTG corresponding to a CTM $M$ and an ECTM $EM$ is constructed from $M$ by executing Algorithm 1. We consider CTM $M$ such that the ECTM $EM$ constructed from $M$ satisfies that $c_i < d_j$ for two interval pairs $I_i$ and $I_j$ ($i < j$) in $EM$. The CTM $M$ is defined as restricted circle trapezoid model (rCTM). The graph corresponding to the rCTM is restricted circle trapezoid graph (rCTG). In this study, we will develop a parallel algorithm for spanning tree problem on rCTGs. The Figure 1 is also an example of rCTM and rCTG because $c_i < d_j$ for $I_i$ and $I_j$ ($i < j$) in ECTM $EM$.

### 2.4 Other definitions

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

The function nor($i$) normalizes the interval pair number $i$ in $ECTM$ within the range 1 to $n$, which is expressed as
Construct Extended Model (CEM)

**Input**: Corner points \([a_i, b_i, c_i, d_i]\) of \(CT_i\) in CTM.

**for each non feedback circle trapezoid** \(CT_i\) **in pardo**
- Create a interval pair \(I_i = ([a_i, d_i], [b_i, c_i])\);

**end**

**for each feedback circle trapezoid** \(CT_i\) **in pardo**
- **for each** \(b_i, c_i, d_i < a_i\) **do**
  - Create a interval pair \(I_i = ([a_i, d_i + 4n], [b_i + 4n, d_i + 4n])\);
- **end**

**for** \(1 \leq i \leq n\) **in pardo**
- Create copies \(I_{-n}\) by shifting \(4n\) to the left for \(I_i\)

**end**

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

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

The function \(v_d(k)\) computes a vertex number \(i\) satisfying \(d_i = k\) for a given number \(k\) on \(E C T M\). For the example shown in Fig. 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 \(\text{nor}(v_d(k))\) for simplicity. For the example shown in Fig. 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 \(ld_i\), \(v_d(d_i)\), and \(nv_d(d_i)\) are shown in Table 3.

![Figure 2: Extended circle trapezoid model EM](image)

#### Table 2: Details of extended circle trapezoid model EM

<table>
<thead>
<tr>
<th>(a_i)</th>
<th>(-7)</th>
<th>(-6)</th>
<th>(-5)</th>
<th>(-4)</th>
<th>(-3)</th>
<th>(-2)</th>
<th>(-1)</th>
<th>(0)</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
<th>(6)</th>
<th>(7)</th>
<th>(8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b_i)</td>
<td>(-31)</td>
<td>(-26)</td>
<td>(-25)</td>
<td>(-21)</td>
<td>(-14)</td>
<td>(-9)</td>
<td>(-7)</td>
<td>(1)</td>
<td>(6)</td>
<td>(7)</td>
<td>(11)</td>
<td>(18)</td>
<td>(21)</td>
<td>(23)</td>
<td>(25)</td>
<td></td>
</tr>
<tr>
<td>(c_i)</td>
<td>(-29)</td>
<td>(-23)</td>
<td>(-22)</td>
<td>(-20)</td>
<td>(-12)</td>
<td>(-10)</td>
<td>(-8)</td>
<td>(-6)</td>
<td>(3)</td>
<td>(9)</td>
<td>(10)</td>
<td>(12)</td>
<td>(20)</td>
<td>(22)</td>
<td>(24)</td>
<td>(26)</td>
</tr>
<tr>
<td>(d_i)</td>
<td>(-27)</td>
<td>(-19)</td>
<td>(-18)</td>
<td>(-16)</td>
<td>(-4)</td>
<td>(-5)</td>
<td>(-1)</td>
<td>(0)</td>
<td>(5)</td>
<td>(13)</td>
<td>(14)</td>
<td>(16)</td>
<td>(28)</td>
<td>(27)</td>
<td>(31)</td>
<td>(32)</td>
</tr>
</tbody>
</table>

**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 \(c_i > d_j)\)

### 3 Property of circle trapezoid graph

We describe some properties on CTGs which are useful for constructing the algorithm for spanning tree problem on rCTGs.

For two interval pairs \(I_i\) and \(I_j\) \((i < j)\) in ECTM, 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].

**Lemma 1** Let \(CT_i\) and \(CT_j\) \((i < j)\) be non-feedback circle trapezoids in CTM \(M\). Moreover, ECTM \(EM\) 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 CTG.
Lemma 2 G is a CTG corresponding to a CTM M, and ECTM EM is constructed from M. For two interval pairs I_i, I_j (i < j), an edge (nor(i), 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 (b_i > a_j) and (b_i > a_j), we have b_i > a_j whenever d_i > a_j. Thus, (nor(i), 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 restricted circle trapezoid model M_r and graph G_r.

Lemma 3 G_r is a rCTG corresponding to a rCTM M_r, and EM_r is an extended circle trapezoid model constructed from M_r. An edge (nor(i), nor(j)) is in G_r if and only if d_i > a_j for i < j satisfies in the EM_r.

(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_r, an edge (nor(i), nor(j)) is in G_r. By definition of rCTG G_r, we have c_i < d_j in EM_r. 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 (nor(i), nor(j)) is in G_r if and only if d_i > a_j for i < j satisfies in the EM_r.

Table 3: Details of extended circle trapezoid model EM

<table>
<thead>
<tr>
<th>i</th>
<th>-7</th>
<th>-6</th>
<th>-5</th>
<th>-4</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>d_i</td>
<td>-24</td>
<td>-13</td>
<td>-17</td>
<td>-15</td>
<td>-3</td>
<td>-2</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>19</td>
<td>15</td>
<td>17</td>
<td>29</td>
<td>30</td>
<td>36</td>
<td>34</td>
</tr>
<tr>
<td>ld_i</td>
<td>-24</td>
<td>-13</td>
<td>-13</td>
<td>-13</td>
<td>-3</td>
<td>-2</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>29</td>
<td>30</td>
<td>36</td>
<td>-</td>
</tr>
<tr>
<td>v(d_i)</td>
<td>-7</td>
<td>-6</td>
<td>-6</td>
<td>-6</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>-</td>
</tr>
<tr>
<td>v(d_i)</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

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

Lemma 4 G_r is a rCTG corresponding to a rCTM M_r, and EM_r is an extended circle trapezoid model constructed from M_r. For 1 ≤ i ≤ n, an edge (v(d_i), i) is in G_r if ld_i > a_i satisfies in the EM_r.

(Proof) By the definition, ld_i = max{d_{i-1}, d_{i+2}, ..., d_{n-1}}. Thus, we have v(d_i, i) satisfying is in G_r if ld_i > a_i for i < j. Therefore, an edge (v(d_i), i) is in G_r if ld_i > a_i satisfies in the EM_r.

4 Parallel Algorithm

In this section, we propose an algorithm for constructing a spanning tree of a connected rCTG G_r. We assume that all trapezoids in the rCTM 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_r 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].

Lemma 5 After executing Step 3 of Algorithm CST, graph T is a spanning tree of CTG G_r.

(Proof) Step 1 is a process for initialization. T is empty set and all ck_i are set to ‘∅’. For all 0 ≤ i ≤ n − 1, compute ld_i := max{d_{i+1}, d_{i+2}, d_{i+3}, ..., d_{n-1}} using parallel prefix computation [11].

In Step 2, we set ck_i = 1, 1 ≤ i ≤ n, if ld_i > a_i. In addition, we compute s := max{ v(d_i), i) satisﬁes in G_r if ld_i > a_i. Thus, a vertex i that ck_i = 1 can have least one edge from i to other vertex v(d_i). Vertex s is the largest v(d_i) satisfying v(d_i) > i. For the example shown in Fig. 4, we set ck_i = 1 because ld_i > a_i,
for $1 \leq i \leq n$. For the only case of $i = 1$, we have $nv_d(ld_{i-1}) > i$ then $s = nv_d(d_b) = 7$.

We consider that $T$ is added an edge form $i$ to $v_d(ld_{i-1})$ smaller for $1 \leq i \leq n$. 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 $\sum_{i=1}^n c_i = n$, $T$ constructed in above way is connected graph that has $n$ vertices. Thus, $T$ is not a tree that have 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 $G_r$ is connected. In Step 2, we obtained $s = \max\{ nv_d(ld_{i-1}) | nv_d(ld_{i-1}) > i, c_i = 1\}$. In not adding $(nv_d(ld_{i-1}), s)$ to $T$, we can remove a cycle $C$.

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

Figure 4 shows a spanning tree $T$ constructed from CTG $G_r$ by executing Algorithm CST. In the following, we analyze the complexity of Algorithm CST.

In Step 1, an ECTM is constructed from a CTM 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 1 Algorithm CST constructs a spanning tree of a restricted 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 restricted 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, 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.
Construct Spanning Tree (CST)

**Input**: Corner points \([a_i, b_i, c_i, d_i] \) of \(I_i \) in \(M_r \).

**Output**: Spanning tree \(T \) of \(G \).

(Step 1) /* Initialization */
Construct \(EM \) from \(M_r \) using Algorithm CEM;
\(T := \emptyset \);
for \(1 \leq i \leq n \) in pardo \(ck_i := 0 \);
for \(-n + 1 \leq i \leq n - 1 \) in pardo 
\(ld_i := \max\{d_{n+1}, d_{n+2}, \ldots, d_i\} \);
end

(Step 2) /* Set Flag \(ck_i \) and Compute \(s \)*/
for \(1 \leq i \leq n \) in pardo 
\[ \text{if } ld_{i-1} > a_i \text{ then } ck_i := 1 \; \]
\[ s := \max\{nv_d(ld_{i-1}) \mid nv_d(ld_{i-1}) > i, ck_i = 1\} \];
end

(Step 3) /* Construction Spanning Tree */
\[ \text{if } \sum_{i=1}^{n} ck_i = n \text{ then} \]
\[ \text{for } 1 \leq i \leq n, i \neq s \text{ in pardo} \]
\[ T := T \cup \text{edge}(nv_d(ld_{i-1}), i) \; \]
end
\[ \text{return } T \; \]
else
\[ \text{for } 1 \leq i \leq n, ck_i = 0 \text{ in pardo} \]
\[ T := T \cup \text{edge}(nv_d(ld_{i-1}), i) \; \]
end
\[ \text{return } T \; \]
end

References


