Parallel Algorithms for Connected Domination Problem on Interval and Circular-arc Graphs *

F.R. Hsu
Department of Information Technology
Taichung Healthcare and Management University
Taichung, Taiwan

M.K. Shan
Department of Computer Science
National Chengchi University
Taipei, Taiwan

Abstract: A connected domination set D of a graph is a set of vertices such that every vertex not in D is adjacent to D and the induced subgraph of D is connected. The minimum connected domination set of a graph is the connected domination set with the minimum number of vertices. In this paper, we propose parallel algorithms for finding the minimum connected domination set of interval graphs and circular-arc graphs. Our algorithms run in $O(\log n)$ time algorithm using $O(n/\log n)$ processors while the intervals and arcs are given in sorted order. Our algorithms are on the EREW PRAM model.

Key-Words: Connected Domination Set, Interval Graph, Circular-arc Graph.

1 Introduction

A graph $G = (V, E)$ is an *interval graph* if its vertices can be put in a one-to-one corresponded with a set F of intervals on a real line such that two vertices are adjacent in G if and only if their corresponding intervals (circular-arcs) have nonempty intersection. Such a set F is called an *interval* model of the interval graph G. See Figure 1. The definition of circular-arc graphs is the same as that of interval graphs, with the exception that the set of intervals on the real line is replaced by a set of circular-arcs on a unit circle C. Interval graphs and circular-arc graphs arise in many application areas, such as scheduling, traffic control, biology, and VLSI design. There is an extensive discussion on these graphs in [1].

A connected domination set D of a graph is a set of vertices such that every vertex not in D is adjacent to D and the induced subgraph of D is connected. The minimum connected domination set of a graph is the connected domination set with the minimum number of vertices. Once endpoints are given in sorted order, for the interval graph, Chang proposed a linear algorithm to compute its minimum connected domination set [2]. For the circular-arc graph, in [3], Hung and Chang proposed a linear time algorithm for the minimum connected domination set. In this paper, we consider the connected domination problem for both interval and circular-arc graphs. Once the interval and arcs are given in sorted order, our algorithms run in $O(\log n)$ time to find the center of interval and circular-arc graphs using $O(n/\log n)$ EREW PRAM processors.

The rest of this paper is organized as follows. Section 2 describes basic notations and some interesting properties and data structures on interval graphs. Sections 3 and 4 give algorithms for the minimum connected domination problem on interval and circular-arc graphs respectively. Finally, we conclude our results in Section 5.

2 Preliminaries

In this section, we propose how to compute some useful data structures on interval graphs which will be used in our algorithms. Assume that the interval graph is given by its interval model $F = \{I_1, I_2, \ldots, I_n\}$ with sorted order, where $I_i = [a_i, b_i]$. We can label intervals in F such a way that $b_i < b_j$ if and only if $i < j$. By doing parallel prefix computation [4], such labelling can be easily obtained from the sorted array of F in $O(\log n)$ time using $n/\log n$ processors. Since all endpoints are sorted, we can replace the real value of an endpoint by its rank in the sorted order. Therefore, we can assume all endpoints are distinct with coordinates of consecutive integer values $1, 2, \ldots, 2n$.

*Supported in part by the National Science Council, Taiwan, R.O.C, grant NSC-89-2213-E-126-017.

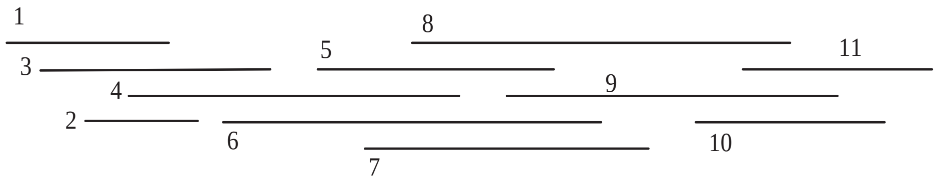

Figure 1: A set of intervals.

In [5], Chao et al. defined a successor function on intervals. For each interval I_i, among intervals intersect I_i, consider intervals with rightmost and leftmost endpoints respectively. Formally, let $RMOST(i) = \max\{j | I_j$ contains $b_i\}$ and $LMOST(i) = k$ where a_k is equal to $\min\{a_j | I_j$ contains $a_i\}$. For example, in Figure 1, the array $RMOST[1, \ldots, n]$ is equal to $(4, 4, 6, 8, 9, 9, 9, 11, 11, 11, 11)$.

According to the $RMOST$ array, the successor tree T_{RMOST} is defined as follows: each interval I_i corresponds a node i in T_{RMOST} and its parent is $RMOST(i)$. For node i and its sibling j, i is on the left side of j if and only if $i < j$. Let $PO(i)$ and $LEV(i)$ denote the pre-order number and the level of interval i in tree T_{RMOST} respectively. In this example, the pre-order traversal of T_{RMOST} would be $(11, 8, 4, 1, 2, 9, 5, 6, 3, 7, 10)$ and $PO = (4, 5, 9, 3, 7, 8, 10, 2, 6, 11, 1)$. Chao et al. showed that these data structure can be found efficiently.

Lemma 1 *[5] For an interval graph, its corresponding arrays $RMOST$, $LMOST$, LEV and PO can be computed in $O(\log n)$ time using $O(n/\log n)$ processors on the EREW PRAM.* □

An interval is called *proper* if it is not contained by any other interval. Let $LEN_F(i, j)$ denote the shortest path length between I_i and I_j on F. The following lemmas show how to query the length between I_i and I_j.

Lemma 2 *[5] For any two intervals I_i and I_j, $i < j$, if $LEN_F(i, j) > 2$, then $LEN_F(i, j) = LEN_F(RMOST(i), LMOST(j)) + 2$.* □

Lemma 3 *[5]For any proper intervals I_i and I_j, $i < j$,*

$$LEN_F(i,j) = \begin{cases} LEV(i) - LEV(j) + 1, \\ \quad \text{if } PO(i) < PO(j), \\ LEV(i) - LEV(j), \\ \quad \text{otherwise.}\square \end{cases}$$

For each interval I_i, consider intervals on its right side. Let $RminB(i)$ denote the interval with the minimum right endpoint. If no such interval exists, let $RminB(i) = n + 1$. Formally, $RminB(i) = \min(\{j | a_j > b_i\} \bigcup \{n + 1\})$. For example, consider Figure 1. The array $RminB[1, \ldots, n]$ is equal to $(5, 5, 5, 9, 10, 10, 10, 12, 12, 12, 12)$. Using the list of all intervals in F sorted by the $a_i's$, we can apply the parallel prefix computations [4] to compute the array $RminB$ in $O(\log n)$ time using $O(n/\log n)$ processors on the EREW PRAM model. Therefore, we have the following lemma

Lemma 4 *For an interval graph, its corresponding array $RminB$ can be computed in $O(\log n)$ time using $O(n/\log n)$ processors on the EREW PRAM.* □

3 Connected Domination Problem on interval graphs

Now consider the minimum connected domination set problem on interval graphs. Let MD_F denote a minimum connected domination set of the model F. Note that if a connected set dominates the interval with the leftmost right endpoint and the interval with the rightmost left endpoint, then it is a connected domination set. Therefore we have the following lemma.

Lemma 5 *Given an interval model F, suppose I_k is the interval with largest left endpoint. Then the shortest path between $RMOST(1)$ and $LMOST(k)$ is a minimum connected domination set.* □

Since $RMOST(1)$ and $LMOST(k)$ are proper, by Lemma 3, $l = LEN_F(RMOST(1), LMOST(k))$ can be computed in constant time. Furthermore, $(RMOST(1), RMOST^2(1), RMOST^3(1), \ldots, RMOST^l(1), LMOST(k))$ is a minimum connected domination set. By the following lemma, it is not difficult to see that this path can be found in $O(\log n)$ time using $O(n/logn)$ EREW PRAM.

Lemma 6 *For any non-root interval I_i, $RMOST^t(i) = \max\{j | PO(j) < PO(i)$ and $LEV(j) = LEV(i) - t\}$.*

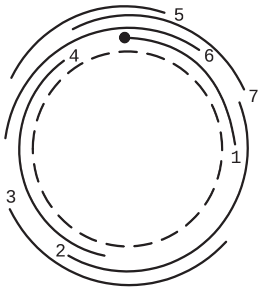

Figure 2: A circular-arc model on circle C.

Proof. For any non-root interval I_i, by definition, $RMOST^t(i)$ is at level $LEV(I_i) - t$ in T_{RMOST}. Since $RMOST^t(i)$ is an ancestor of I_i in T_{RMOST}, $PO(RMOST^t(i)) < PO(i)$. By definition, the pre-order numbers of $RMOST^t(i)$'s siblings on its left side are less than $PO(RMOST^t(i))$. Besides, the pre-order numbers of $RMOST^t(i)$'s siblings on its right side are greater than $PO(i)$. Therefore, $RMOST^t(i) = \max\{j|PO(j) < PO(i)$ and $LEV(j) = LEV(i) - t\}$. □

Therefore, we have the following corollary.

Corollary 7 *Given the interval model F of an interval graph G with sorted order, the minimum connected domination set can be found in $O(\log n)$ time using $O(n/\log n)$ processors on the EREW PRAM.* □

4 Connected Domination Problem on Circular-arc Graphs

The circular-arc model of a circular-arc graph consists of a set $S = \{I_1, I_2, \ldots, I_n\}$ of n circular-arcs on the unit circle C. For example, see Figure 2. Now consider the connected domination problem for circular-arc graphs. Without loss of generality, we assume that the union of all arcs is equal to C (otherwise, the problem becomes one on interval graphs). Besides, we assume that there is no arc equal to C (otherwise, the problem becomes trivial). We define $I_i = [a_i, b_i]$ is the arc on C from a_i clockwise to b_i. We also assume that the endpoints of the arcs in S are given in the order in which their b_i's points are visited during the clockwise traversal along C by starting at b_1. Without loss of generality, we assume all endpoints are distinct with coordinates of consecutive integer values $1, 2, \ldots, 2n$. Besides, for ease of reference, we assume the coordinate of a_1 is equal to 1. Such labelling can be easily obtained from the sorted array of S in $O(\log n)$ time using $n/\log n$ processors by doing parallel prefix [4].

Similar to the $RMOST$ function on an interval graph, for an arc I_i on a circular-arc graph, we define $CMOST(i)$ as follows. Let $N(i)$ denote the set $\{I_j|b_i$ in $I_j\}$. Starting from b_i, we visit right endpoints of arcs in $N(i)$ clockwise one by one. Let $CMOST(i)$ denote the last arc visited. For example, consider Figure 2. The arrays $CMOST[1, \ldots, n]$ is equal to $(2, 4, 4, 6, 1, 1, 1)$. Similar to the $LMOST$ function on an interval graph, for an arc I_i on a circular-arc graph, For ease of reference, let $CMOST^k(i)$ denote $CMOST(CMOST^{k-1}(i))$ and $CMOST^1(i) = CMOST(i)$. Besides, $CMOST^0(i) = i$.

In the following, we will show many problems on circular-arc graphs can be transformed into problems on interval graphs. We describe how to map S into an interval model F'. This mapping is done as if circle C is open at a_1 and unrolled onto the real line twice. Any arc I_k is mapped into two intervals J_k^1 and J_k^2 as follows. If the interior of I_k does not contain a_1, then $J_k^1 = [a_k, b_k]$ and $J_k^2 = [a_k + 2n, b_k + 2n]$. If the interior of I_k contains a_1, $J_k^1 = [a_k - 2n, b_k]$ and $J_k^2 = [a_k, b_k + 2n]$. Note that the mapping can be found by checking every endpoint in S to see whether it is an endpoint of an arc that contains a_1. This can be done in $O(\log n)$ time using $O(n/\log n)$ EREW PRAM processors.

Directly from the mapping, we have the following lemma. The following lemma shows how to compute array $CMOST$ of S through the help of F'. For an interval on F', its corresponding arc on S is the arc which mapped into the interval.

Lemma 8 *Given a circular-arc model S, I_i is an arc on it. Then, $CMOST(i)$ is equal to the corresponding arc of $RMOST(J_i^1)$ on F'.* □

Similar to the $RminB$ function on an interval graph, for an arc I_i on a circular-arc graph, we define $CminB(i)$ as follows. Starting from b_i clockwise, we visit right endpoints of arcs which do not intersect I_i. Let $CminB(i)$ denote the first arc visited. In Figure 2, array $CminB = (3, 5, 5, 7, 2, 2, 2)$.

Similar to Lemma 8, for the $CminB$ function, we have the following lemma.

Lemma 9 *Given a circular-arc model S, I_i is an arc on it. Then, $CminB(i)$ is equal to the corresponding arc of $RminB(J_k^1)$ on F'.* □

By Lemma 8 and 9, we can compute arrays $CMOST$ and $CminB$ on a circular-arc graph by computing its corresponding arrays $RMOST$,

$LMOST$ and $RminB$ on its corresponding interval graph. By Lemma 1 and 4, we have the following lemma.

Lemma 10 *For a circular-arc graph, its corresponding array $CMOST$ and $CminB$ can be computed in $O(\log n)$ time using $O(n/\log n)$ processors on the EREW PRAM model.* $\square$

For arc I_i in S, let $R(i)$ denote the shortest path length walking from I_i clockwise and visiting I_i again. For example, in Figure 2, $(I_1, I_2, I_4, I_6, I_1)$ is a path and $R(1) = 4$.

For two arcs I_i and I_j, let $cpath(i,j)$ denote a shortest path from I_i to I_j clockwise. Let $union(i,j)$ denote the union of arcs in $cpath(i,j)$. By definition, we have the following lemma.

Lemma 11 *For any arc I_i and positive integer k and $k < R(i)$, $union(i, CMOST^k(i))$ is equal to $[a_i, b_{CMOST^k(i)}]$.* $\square$

Consider the connected domination sets walking from arc i clockwise. Let $cmd(i)$ denote the one with minimum arcs. Let MD_S denote a minimum connected domination set of the model S. It follows $|MD_S| = \min\{|cmd(i)| | i \in S\}$.

Now consider how to find $cmd(i)$. We have the following lemma.

Lemma 12 *For any proper arc I_i in the circular-arc model S, let $|cmd(i)| = k$. Then $cpath(i, CMOST^{k-1}(i))$ is a $cmd(i)$ and $R(i)-2 \leq k \leq R(i)$.*

Proof. Suppose I_i is a proper arc in the circular-arc model S. For ease of reference, let $|cmd(i)| = k$. Suppose $A = \{I_{i_1}, I_{i_2}, \ldots, I_{i_k}\}$ is a $cmd(i)$. Let B be the set of arcs in $cpath(i, CMOST^{k-1}(i))$. Note that $|B| = k$. By definition of $CMOST$, union of arcs in B is contained by union of arcs in B. Therefore arcs in $cpath(i, CMOST^{k-1}(i))$ form a $cmd(i)$.

By definition of $R(i)$, arc $I_{CMOST^{R(i)-1}(i)}$ connects arc I_i. Therefore, $union(i, CMOST^{R(i)-1}(i))$ is equal to a circle. It follows $cpath(i, CMOST^{R(i)-1}(i))$ is a connected domination set and $k \leq R(i)$.

Suppose $j \leq R(i) - 4$. Consider $cpath(i, CMOST^j(i))$. Arc $CMOST^{R(i)-2}(i)$ dose not intersect $union(i, CMOST^j(i))$. Therefore, $cpath(i, CMOST^j(i))$ is not a connected domination set. Note that there are $j+1$ arcs in $cpath(i, CMOST^j(i))$. It follows $j+2 \leq k$. Hence, $R(i) - 2 \leq k \leq R(i)$. $\square$

By Lemma 12, $|cmd(i)|$ has only three possibility: $R(i)-2$, $R(i)-1$ and $R(i)$. Note that by definition, $R(i) \geq 2$. If $R(i) = 2$, then I_i is an unit circle and I_i itself is a minimum connected domination set. Now consider that $R(i) \geq 3$. According to the above lemma, for any proper arc I_i, we can find $cmd(i)$ as follows. First, we test if $|cmd(i)|$ is equal to $R(i) - 2$. We test if there exists any arc not connected with $union(i, CMOST^{R(i)-3}(i))$. Note that $union(i, CMOST^{R(i)-3}(i))$ is equal to $[a_i, b_{CMOST^{R(i)-3}(i)}]$. We can perform this test by testing if there exists any arc contained in $[b_{CMOST^{R(i)-3}(i)}, a_i]$. We denote this area as $Da1(i)$. If no such arc exists, $|cmd(i)| = R(i) - 2$ and arcs in $cpath(i, CMOST^{R(i)-3}(i))$ form a $cmd(i)$. Otherwise, we test if $|cmd(i)|$ is equal to $R(i) - 1$ similarly. That is to test if there exists any arc contained in $[b_{CMOST^{R(i)-2}(i)}, a_i]$. We denote this area as $Da2(i)$. If $|cmd(i)|$ is neither equal to $R(i)-2$ nor to $R(i)-1$, then $|cmd(i)|$ is equal to $R(i)$ and arcs in $cpath(i, CMOST^{R(i)-1})$ form a $cmd(i)$.

Suppose $t = \min_{i \in S} R(i)$. Let the set of arcs $SA = \{I_i | R(i) = t$ or $R(i) = t+1\}$. Since there are only three possibility for $|cmd(i)|$, in order to find I_k with minimum number of $|cmd(k)|$, we can only find $cmd(i)$ in SA.

Now, we list steps for finding the minimum connected domination sets of circular-arc graphs.

Step 1. For every arc I_i in S, compute $R(i)$.
Step 2. Find $\min_{i \in S} R(i)$. Let $t = \min_{i \in S} R(i)$. Let the set of arcs $SA = \{I_i | R(i) = t$ or $R(i) = t+1\}$.
Step 3. For every arc I_j in SA, compute $Da1(j)$ and $Da2(j)$.
Step 4. For every arc I_j in SA, find $|cmd(j)|$ by testing if there exists any arc in $Da1(j)$ and $Da2(j)$.
Step 5. Find the minimum connected domination set $cmd(i)$ where $|cmd(i)| = \min\{|cmd(j)| | I_j \in SA\}$.

Now, we consider Step 1. Recall that when we map the circular-arc model S into its corresponding interval model F', we map each arc I_i into two intervals J_i^1 and J_i^2. We have the following lemma.

Lemma 13 *Given a circular-arc model S and its corresponding interval model F', $R(i)$ is equal to $LEN_{F'}(J_i^1, J_i^2)$ in F'.* $\square$

For an interval model F, for interval I_i and I_j, we describe how to query $LEN_F(i,j)$. Without loss of generality, suppose $i < j$. There are only three cases: 1)$LEN_F(i,j) = 1$, 2)$LEN_F(i,j) = 2$ and 3)$LEN_F(i,j) > 2$. First, we test whether $LEN_F(i,j) = 1$ or not in constant time. Second, if

$LEN_F(i,j) \neq 1$, we try to test whether $LEN_F(i,j)$ is equal to 2 or not. It is not difficult to see that if $LEN_F(i,j) = 2$, then $I_{RMOST(i)}$ intersects I_j. Therefore, we can test whether $LEN_F(i,j) \leq 2$ in constant time. Now consider the case that $LEN_F(i,j) > 2$. By Lemma 2, $LEN_F(i,j) = LEN_F(RMOST(i), LMOST(j)) + 2$. Note that $RMOST(i)$ and $LMOST(j)$ are proper. By Lemma 3, we can find $LEN_F(RMOST(i), LMOST(j))$ in constant time. Therefore, we can use one processor to query $LEN_{F'}(J_i^1, J_i^2)$ in constant time. Note that in order to avoid read conflict, for every interval I_i in F', we need to store $PO(RMOST(i))$, $PO(LMOST(i))$, $LEV(RMOST(i))$ and $LEV(LMOST(i))$ for future query during the preprocessing phase. Therefore, Step 1 can be performed in $O(\log n)$ time using $O(n/\log n)$ EREW PRAM processors. Obviously, Step 2 can be done in the same time and processor complexity.

Regarding Step 3, we need to find $Da1(j)$ and $Da2(j)$. That is, we need to query $CMOST^{R(j)-3}(j)$ and $CMOST^{R(j)-2}(j)$. Note that $CMOST^k(j)$ is equal to $RMOST^k(j)$ in its corresponding T_{RMOST} tree. We can use the technique of the level-ancestor query in trees introduced by Berkman and Vishkin [6] to solve these queries. However, it is a fairly hard implemented algorithm and run on the CREW PRAM. In stead of answering these queries individually, we perform these queries in batch. With the help of T_{RMOST}, the following lemma shows how to find $CMOST^k(j)$ for all I_j in S for some fixed k.

Lemma 14 *Given a circular-arc model S and a positive integer k, $CMOST^k(i)$ for all I_i in S can be found in $O(\log n)$ time using $O(n/\log n)$ EREW PRAM processors.*

Proof. First, we map the circular-arc model S into corresponding interval model F'. Given arc I_i on S, by Lemma 8, $CMOST^k(i)$ is equal to the corresponding arc of $RMOST^k(J_i^1)$ on F'.

Now, consider how to compute $RMOST^k(t)$ on F' for all t. Note that $RMOST^k(t)$ is the ancestor of t on level $LEV(t) - k$ in T_{RMOST}. By Lemma 6, we can compute $RMOST^k(t)$ on F' for all node t at level j as follows. We merge the nodes on level $j-k$ and j according to their pre-order number in T_{RMOST}. For nodes on level $j-k$ and on level j, define $JK(t)$ such that $JK(t) = t$ if node t is on level $j-k$, otherwise $JK(t) = 0$. Then, the prefix maximum of JK on the merged list is equal to $RMOST^k(t)$ for node t on level j. The merging process for two sorted lists and prefix computation can be performed in $O(\log h)$ time using $O(h/\log h)$ EREW PRAM processors [4, 7], where h is the size of lists. The total size for all levels is at most $2n$. Then, $RMOST^k(t)$ on F' for all t can be computed in $O(\log n)$ time using $O(n/\log n)$ EREW PRAM processors.

It follows $CMOST^k(i)$ for all arc I_i on S can be found in the same time and processor complexity. □

Suppose $t = \min_{i\in S} R(i)$ and $SA = \{I_i | R(i) = t$ or $R(i) = t+1\}$. We can find $Da1(j)$ and $Da2(j)$ for every arc I_j in SA, by performing batch query (as described in Lemma 14) *three* times. Therefore, Step 3 can be done in $O(\log n)$ time using $O(n/\log n)$ EREW PRAM processors.

Now, consider Step 4. We need to test if there exists any arc in $Da1(j)$ or $Da2(j)$ for arc I_j. The following lemma shows how to perform this test efficiently.

Lemma 15 *For arc I_j in S, there exists any arc contained in $[b_k, a_i]$ if and only if arc $CminB(k)$ is contained in $[b_k, a_i]$.*

Proof. The 'if' part is trivial and its proof is omitted. Now consider the 'only if' part. By definition of $CminB(k)$, starting from b_k, visiting the right endpoints of arcs which do not intersect I_k clockwise, $CminB(k)$ is the first arc visited. Clearly, $a_{CminB(k)}$ is contained in $[b_k, a_i]$.

Assume that there exists arc I_l which is contained in $[b_k, a_i]$. It follows I_l does not intersect b_k. Therefore, the clockwise order should be $(b_k, a_{CminB(k)}, b_{CminB(k)}, b_l, a_i)$. Therefore, arc $CminB(k)$ is also contained in $[b_k, a_i]$. □

Recall that when we compute $Da1(j)$ and $Da2(j)$, we store $CMOST^{R(j)-3}(j)$ and $CMOST^{R(j)-2}(j)$ for node j. To avoid read conflict, in Step 3, when we compute $Da1(j)$ and $Da2(j)$, we can also store $CminB(CMOST^{R(j)-3}(j))$ and $CminB(CMOST^{R(j)-2}(j))$ for node j. Therefore, Step 4 be done in $O(\log n)$ time using $O(n/\log n)$ EREW PRAM processors. Obviously, Step 5 can be performed in the same time and processor complexity. Therefore, we have the following corollary.

Corollary 16 *Given the circular-arc model S of an interval graph G with sorted order, the center can be found in $O(\log n)$ time using $O(n/\log n)$ processors on the EREW PRAM.* □

5 Conclusion

In this paper, we propose parallel algorithms for center problems on interval and circular-arc graphs. Our parallel algorithms lead to new linear time algorithms. We define some useful data structures on interval graphs. These data structures may be useful for other problems, like the median problem, on interval and circular-arc graphs. The extension to trapezoid graphs [8] is left for future study.

References

[1] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[2] M. S. Chang. Efficient algorithms for the domination problems on interal and circular-arc graphs. *SIAM J. on Computing*, 27:1671–1694, 1998.

[3] R. W. Hung and M. S. Chang. A linear algorithm for the connected domination problem on circular-arc graphs. *Proc. of the 19th Workshop on Combinatorial Mathematics and Computation Theory*, pages 70–78, Mar. 2002.

[4] S.G. Akl. *Parallel computation: models and methods*. Prentice Hall, Upper Saddle River, New Jersey, 1997.

[5] H. S. Chao, F. R. Hsu, and R. C. T. Lee. On the shortest length queries for interval and circular-arc graphs. *Proc. of the Fifth World Multiconference on Systemics, Cybernetics and Informatics, Orlando, USA*, VII:331–336, 2001.

[6] O. Berkamn and U. Vishkin. Finding level-ancestors in trees. *J. Comput. System Sci.*, 48:214–230, 1994.

[7] R. Cole. Parallel merge sort. *SIAM J. on Computing*, 17:770–785, 1988.

[8] I. Dagan, M.C. Golumbic, and R.Y. Pinter. Trapezoid graphs and their coloring. *Discr. Applied Math.*, 21:35–46, 1988.