Information Processing Letters 40 (1991) 295-302
North-Holland

30 December 1991

Reducing conflict resolution time for solving graph problems in broadcast communications *

Chang-Biau Yang

Department of Applied Mathematics, National Sun Yat-sen University, Kaohsiung, Taiwan 80424, ROC

Communicated by F. Dehne
Received 11 March 1991
Revised 18 October 1991

Abstract

Yang, C.-B., Reducing conflict resolution time for solving graph problems in broadcast communications, Information Processing Letters 40 (1991) 295-302.

In this paper, we propose some algorithms to solve the topological ordering problem, the breadth-first search problem and the connected component problem under the broadcast communication model. The basic idea of our algorithms is to divide a graph into several layers. Only after all vertices in one layer are processed, we begin to process the vertices in another layer. Thus, the number of broadcast conflicts is reduced. We also propose a randomized conflict resolution scheme to resolve conflicts. We show that the average time complexity of our algorithms is $\Theta(n)$, where n is the number of available processors and also the number of vertices in the graph.

Keywords: Parallel algorithms, broadcast communication, graph, topological ordering, breadth-first search, connected component

1. Introduction

The *broadcast communication model* consists of some processors sharing one common channel for communications [1,3,8-12,14-18]. In this model, all processors communicate with one another through the shared communication channel. In addition, all processors are connected to a global clock which is used for synchronizing among processors. The channel time is divided into many segments of equal length, called *time slots*. It is assumed that each slot occupies constant time. Each processor can broadcast one packet of data within one time slot. After a processor broadcasts messages, each processor can hear the channel. It can determine whether zero, one or multiple packets exist in the channel. If more than one processor attempts to broadcast simultaneously, i.e. attempts to broadcast in the

* This research work was partially supported by the National Science Council of the Republic of China under contract NSC80-0480-E-110-04.

same time slot, there will be multiple packets in the channel within that time slot, and then a *broadcast conflict* occurs. Thus, it is assumed that a broadcast conflict can be detected in constant time by each processor if it occurs. When a conflict occurs, we have to use a *conflict resolution scheme* to resolve the conflict. This resolution scheme will enable one of the broadcasting processors to broadcast successfully.

The time required for an algorithm to solve a problem under the broadcast communication model includes three parts: (1) resolution time: spent to resolve conflicts, (2) transmission time: spent to transmit data, (3) computation time: spent to solve the problem. We hope that the sum of resolution time, transmission time and computation time, which is the time complexity of the algorithm, is minimized.

Several researchers have proposed some algorithms to solve some problems under the broadcast communication model. Levitan and Foster [8,9], Dechter and Kleinrock [3], Ramarao [12], Marberg and Gafni [10] and Tang and Chiu [14] proposed sorting algorithms under the broadcast communication model. Tang and Wu [15] used this model to solve some graph problems. Levitan [8] and Yang, Lee and Chen [16] proposed algorithms for the minimum spanning tree problem under this model.

Using the broadcast communication model, Yang, Lee and Chen [17] also proposed some algorithms for some graph problems, including the topological ordering problem, breadth-first search and the connected component problem. They used Capetanakis' tree algorithm [1] as their conflict resolution scheme. The time complexities of all of these algorithms are $O(n \log n)$.

In this paper, we propose a new algorithm for solving the topological ordering problem. In our new algorithm, we divide the vertices in a graph into several "layers". We assign the topological order of each vertex layer by layer. Only after the topological order for all vertices in one layer is determined, we begin to process the vertices in another layer. By the concept of layers, only the vertices in the same layer may induce broadcast conflicts. Besides, we proposed a randomized conflict resolution scheme to resolve conflicts. Thus, the number of broadcast conflicts is reduced. The average time complexity of our algorithm is reduced to $\Theta(n)$ if n processors are used to solve an n-vertex graph. We also apply this technique to the breadth-first search problem and the connected component problem, and obtain the same time complexity.

2. Previous works

In this section, we shall review the algorithm for the topological ordering problem proposed by Yang et al. [17]. At first, we shall define the topological ordering problem [2,4–7,13,17]. In an acyclic digraph $G=(V, E)$, all vertices are put into a linear list such that for $v_i, v_j \in V$, v_i is before v_j in the list if $(v_i, v_j) \in E$, or if there is a directed path starting at v_i and ending at v_j. If $(v_i, v_j) \in E$, v_i is a *predecessor* of v_j. The sequence of this linear list forms a *topological order* for this digraph. That is, there are some partial orders among the vertices in an acyclic digraph and we want to find a total order to fit all original partial orders. For example, one of the feasible solutions for the digraph in Fig. 1 is (d, h, b, a, g, e, c, f).

Based upon the definition of the topological order, Yang et al. [17] proposed an algorithm to obtain one of the answers under the broadcast communication model. They put one of the vertices which have no predecessor into the first position of the linear list. Then, they remove this vertex with all of its outgoing edges from the graph. At the next step, in the subgraph with the

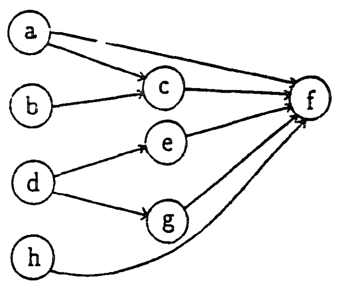

Fig. 1. An acyclic digraph.

selected vertex and all of its outgoing edges deleted, they again choose one of the vertices which have no predecessor and put it into the second position of the list. This chosen vertex and all of its outgoing edges are then removed. Repeatedly, each vertex will be put into the linear list eventually. This linear list forms a topological order for the graph.

In the algorithm proposed by Yang et al., it is assumed that n processors are used to solve the topological ordering problem with n vertices. Each processor stores a vertex with its adjacency list. Each processor locally and independently decides whether the vertex stored in itself fulfills the condition to be put into the linear list. Those processors which fulfill the condition broadcast their vertex labels. It is possible that more than one processor can broadcast. When broadcast conflicts occur, Capetanakis' tree algorithm [1], which will be briefly introduced later, is used to resolve the broadcast conflicts. The processor which broadcasts successfully after resolving conflicts will put its vertex into the linear list. The algorithm requires n successful broadcasts. And, one successful broadcast needs $O(\log n)$ time since $O(\log n)$ time is required to resolve a broadcast conflict in the tree algorithm. Therefore, it takes $O(n \log n)$ time to solve the topological ordering problem.

The tree algorithm [1] is a deterministic algorithm for resolving conflicts. In the broadcast communication model with n processors, if more than one processor attempt to broadcast at time slot t simultaneously, the tree algorithm will force one half of the n processors to be silent and give chance to another half of the n processors to broadcast at time slot $t+1$. If a broadcast conflict occurs again at time slot $t+1$, only one quarter of these n processors have chance to continue to broadcast at time slot $t+2$. In general, the number of processors being able to broadcast at time slot $t+i$ is one half of that at time slot $t+i-1$. At time slot $t+\lceil\log_2 n\rceil$, at most one processor can broadcast, thus there is no conflict. In the worst case, the number of time slots required to resolve a broadcast conflict is $O(\log n)$. Hence, the time required to resolve a conflict is $O(\log n)$.

3. A new algorithm for topological ordering

In this section, we shall propose a new algorithm to solve the topological ordering problem under the broadcast communication model. The data structure used in our algorithm is the same as that used by Yang et al. [17]. It is assumed that there are n processors in the broadcasting system and there are also n vertices in the given digraph. Each processor stores a vertex with its adjacency vector. An *adjacency vector* is an n-bit vector. For a graph $G=(V, E)$, if $v_i, v_j \in V$ and $(v_j, v_i) \in E$, then the jth bit of the adjacency vector of v_i is 1; otherwise the jth bit is 0. In fact, the adjacency vector of v_i is the column of the adjacency matrix corresponding to v_i. For example, in Fig. 1, the adjacency vector of vertex c is (11000000). The processor storing v_i can check if there is an edge starting at other vertex v_j and ending at v_i in constant time by using its adjacency vector.

The basic idea of our new algorithm is that the vertices in a digraph are divided into several "layers". We give the topological order of each vertex layer by layer. Only after all vertices in one layer are processed, we begin to process the vertices in another layer. Even if broadcast conflicts still exist, the conflicts may occur only among the vertices in the same layer. The vertices in different layers will not fight for the right of broadcasting. Thus, the number of broadcast conflicts is reduced.

For shortening description in our algorithm, we use the term "vertex" to represent "the processor storing the vertex" sometimes and to represent the vertex itself at different times if there is no ambiguity.

We take the digraph in Fig. 1 as an example to illustrate our idea. Vertices a, b, d and h are marked as layer 1 since they have no predecessor. Now, only the vertices in layer 1 can broadcast. There will be broadcast conflicts. After broadcast conflicts are resolved, suppose that vertex d is the successful one. In the next time slot, vertices e and g can compete with vertices a, b and h for the second position of the topological order. However, to reduce the number of conflicts, we mark vertices e and g as layer 2. Only after all vertices in layer 1 finish their broadcasts, the vertices in layer 2 begin to broadcast. Suppose that vertex h is the second successful one. No vertex will be marked as layer 3 since there is no additional vertex all of whose predecessors have broadcast. Thus, layer 3 is empty. Suppose that we have a complete broadcasting sequence: d, h, b, a, g, e, c, f. Then, the corresponding sets in the layers are $\{a, b, d, h\}$, $\{e, g\}$, empty, empty, $\{c\}$, empty, empty and $\{f\}$. As we have seen, after one vertex broadcasts successfully, it will create one layer. Therefore, if there are n vertices in the graph, there will be n layers in which some are empty.

For implementation under the broadcast communication model, the layer information can be maintained in a queue-like structure. Each layer corresponds to one position of the queue. Thus, each position of the queue can have zero, one or more than one vertex. Each vertex simply keeps the index of the position where it is in the queue.

Besides, we propose a randomized resolution scheme to resolve the conflicts of the vertices in one layer. For doing this, we use a stack-like structure to perform resolution. Initially, all vertices in the same layer are put into the bottom of the stack. Now, all vertices on the bottom of the stack can broadcast. If a conflict occurs, each vertex has the probability $\frac{1}{2}$ to continue to broadcast at the next step. Those vertices who decide to continue to broadcast put themselves onto the upper level of the stack. At the next step, only the vertices on the top level of the stack can broadcast. This broadcasting rule will be followed for the subsequent steps. Similar to the queue, each level of the stack can accommodate zero, one or more than one vertex. And, each vertex simply records which level it occupies.

Algorithm Topological-Ordering

Step 1. Each vertex which has no predecessor puts itself in the first position of the queue.

Step 2. Each vertex in the front of the queue deletes itself from the queue and puts itself onto the bottom of the stack.

Step 3. Invoke Procedure One-Layer-Resolution to process the vertices in the stack.

Step 4. If the queue is not empty, then go to step 2; otherwise terminate.

Procedure One-Layer-Resolution

Step 1. Each vertex sets the active level as the bottom of the stack.

Step 2. Each vertex on the active level broadcasts its vertex label.

Step 3. If a broadcast conflict occurs, in the next time slot, all vertices which cause the conflict randomly choose to continue to broadcast or to abandon broadcasting with equal probabilities. That is, the probability of continuing to broadcast is $\frac{1}{2}$. Each vertex which chooses to continue to broadcast brings itself into the upper level of the stack. The active level is also changed to the upper level. Go to step 2.

Step 4. If there is only one vertex that broadcasts successfully, it deletes itself from the stack and it will never broadcast hereafter. Each vertex which has not been put into the queue and all of whose predecessors have broadcast puts itself into the rear of the queue.

Step 5. (It is possible that no vertex broadcasts.) Change the active level to the lower level. If the stack is not empty (the active level is not below the bottom), then go to step 2; otherwise, return to the main algorithm.

We shall show that the sequence of successful broadcasts forms a legal topological order. Let $G=(V, E)$ be an acyclic digraph and $v_1, v_2 \in V$. In Algorithm Topological-Ordering, suppose that v_1 and v_2 are put into the queue at step t_1 and t_2, respectively. If $t_1 < t_2$, v_1 will be broadcast earlier than v_2 due to the property of queues. By definition, "v_1 is before v_2" is the case in at least one of the legal topological orders. Thus, it is correct for $t_1 < t_2$. If $t_1 = t_2$, v_1 may be broadcast before or after v_2 by executing Procedure One-Layer-Resolution. By the definition of the topological ordering, it is correct. The case for $t_1 > t_2$ is similar to that for $t_1 < t_2$.

4. Analysis of Algorithm Topological-Ordering

In this section, we shall show that the average time complexity of Algorithm Topological-Ordering is $\Theta(n)$. To obtain this result, we need to prove the following lemma first:

Lemma 1. *Suppose that there are k vertices passed to Procedure One-Layer-Resolution when Algorithm Topological-Ordering invokes the procedure. Let g_k denote the average number of time slots required (including empty slots, successful broadcasts and conflicts) when the procedure is executed. Then*:

$$3k - 1 \geqslant g_k \geqslant \tfrac{8}{3}k - \tfrac{1}{3}, \quad \text{for } k \geqslant 2.$$

Proof. We shall prove this lemma by induction on k. When zero or one vertex is passed to the procedure, it will require one time slot to perform the work. Thus,

$$g_0 = 1 \quad \text{and} \quad g_1 = 1.$$

g_2 can be obtained by the following formula:

$$g_2 = 1 + \tfrac{1}{4}(g_0 + g_2) + \tfrac{1}{2}(g_1 + g_1) + \tfrac{1}{4}(g_2 + g_0). \tag{1}$$

Since two processors want to broadcast in the first time slot, a conflict occurs. It will take one time slot. In the second time slot, there may be zero, one or two vertices to choose to broadcast again. The number of vertices remaining to be processed in the future are two, one and zero, respectively. The probabilities for these three cases are $\frac{1}{4}$, $\frac{1}{2}$ and $\frac{1}{4}$, respectively. Thus,

$$g_2 = 1 + \tfrac{1}{2}g_0 + g_1 + \tfrac{1}{2}g_2 = \tfrac{5}{2} + \tfrac{1}{2}g_2.$$

We have

$$g_2 = 5.$$

Now we prove the first part, $g_k \leqslant 3k - 1$, for $k \geqslant 2$. When $k = 2$, it is trivially true since $g_2 = 5 \leqslant 3 \cdot 2 - 1$. By hypothesis, assume that $g_2, g_3, \ldots, g_{k-1}$, $k \geqslant 3$, satisfy the inequality. We shall prove that g_k, $k \geqslant 3$, also satisfies the inequality. g_k can be obtained by the following formula:

$$g_k = 1 + \sum_{i=0}^{k} C_i^k \left(\tfrac{1}{2}\right)^i \left(\tfrac{1}{2}\right)^{k-i} (g_i + g_{k-i}). \tag{2}$$

The meaning of the terms in (2) are similar to those in (1). We can derive (2) as follows.

$$\begin{aligned}
g_k &= 1 + \left(\tfrac{1}{2}\right)^k \Big\{ C_0^k (g_0 + g_k) + C_k^k (g_k + g_0) \\
&\qquad + C_1^k (g_1 + g_{k-1}) + C_{k-1}^k (g_{k-1} + g_1) \\
&\qquad + \sum_{i=2}^{k-2} C_i^k (g_i + g_{k-i}) \Big\} \\
&\leqslant 1 + \left(\tfrac{1}{2}\right)^k \Big\{ 2 + 2g_k + C_1^k (1 + 3(k-1) - 1) \\
&\qquad + C_{k-1}^k (3(k-1) - 1 + 1) \\
&\qquad + \sum_{i=2}^{k-2} C_i^k (3i - 1 + 3(k-i) - 1) \Big\} \\
&= 1 + \left(\tfrac{1}{2}\right)^k \Big\{ 2 + 2g_k - 2k + \sum_{i=1}^{k-1} C_i^k (3k-2) \Big\} \\
&= 1 + \left(\tfrac{1}{2}\right)^k \{ 2 + 2g_k - 2k + (2^k - 2)(3k-2) \} \\
&= \left(\tfrac{1}{2}\right)^{k-1} g_k + 3k - 1 - \left(\tfrac{1}{2}\right)^{k-1} (4k-3).
\end{aligned} \tag{3}$$

We obtain

$$\begin{aligned}
g_k &\leqslant \frac{3k - 1 - \left(\frac{1}{2}\right)^{k-1}(4k-3)}{1 - \left(\frac{1}{2}\right)^{k-1}} \\
&= 3k - 1 + \frac{\left(\frac{1}{2}\right)^{k-1}(2-k)}{1 + \left(\frac{1}{2}\right)^{k-1}} \\
&\leqslant 3k - 1, \quad \text{for } k \geqslant 3.
\end{aligned}$$

We conclude that $g_k \leqslant 3k - 1$, for $k \geqslant 2$.

In the following, we shall prove the second part, $g_k \geqslant \frac{8}{3}k - \frac{1}{3}$, for $k \geqslant 2$. When $k = 2$, it is trivially true since $g_2 = 5 \geqslant \frac{8}{3} \cdot 2 - \frac{1}{3}$. By hypothesis, assume that $g_2, g_3, \ldots, g_{k-1}$, $k \geqslant 3$, satisfy the inequality. We shall prove that g_k, $k \geqslant 3$, also satisfies the inequality. From (3),

$$\begin{aligned} g_k \geqslant 1 + \left(\tfrac{1}{2}\right)^k \Big\{ & 2 + 2g_k + C_1^k\left(1 + \tfrac{8}{3}(k-1) - \tfrac{1}{3}\right) \\ & + C_{k-1}^k\left(\tfrac{8}{3}(k-1) - \tfrac{1}{3} + 1\right) \\ & + \sum_{i=2}^{k-2} C_i^k\Big(\tfrac{8}{3}i - \tfrac{1}{3} \\ & \qquad + \tfrac{8}{3}(k-i) - \tfrac{1}{3}\Big)\Big\} \\ = 1 + \left(\tfrac{1}{2}\right)^k \Big\{ & 2 + 2g_k - \tfrac{4}{3}\cdot 2k \\ & + \sum_{i=1}^{k-1} C_i^k\left(\tfrac{8}{3}k - \tfrac{2}{3}\right)\Big\} \\ = 1 + \left(\tfrac{1}{2}\right)^k & \left\{2 + 2g_k - \tfrac{8}{3}k + (2^k - 2)\left(\tfrac{8}{3}k - \tfrac{2}{3}\right)\right\} \\ = \left(\tfrac{1}{2}\right)^{k-1} & g_k + \tfrac{8}{3}k + \tfrac{1}{3} - \left(\tfrac{1}{2}\right)^{k-1}\left(4k - \tfrac{5}{3}\right). \end{aligned}$$

We get

$$\begin{aligned} g_k &\geqslant \frac{\tfrac{8}{3}k + \tfrac{1}{3} - \left(\tfrac{1}{2}\right)^{k-1}\left(4k - \tfrac{5}{3}\right)}{1 - \left(\tfrac{1}{2}\right)^{k-1}} \\ &= \tfrac{8}{3}k - \tfrac{1}{3} + \frac{\tfrac{2}{3} - \left(\tfrac{1}{2}\right)^{k-1}\left(\tfrac{4}{3}k - \tfrac{4}{3}\right)}{1 - \left(\tfrac{1}{2}\right)^{k-1}} \\ &\geqslant \tfrac{8}{3}k - \tfrac{1}{3}, \quad \text{for } k \geqslant 3. \end{aligned}$$

We conclude that $g_k \geqslant \frac{8}{3}k - \frac{1}{3}$, for $k \geqslant 2$. This completes the proof. □.

Now, we are able to prove the main theorem.

Theorem 2. *The average time complexity of Algorithm Topological-Ordering is* $\Theta(n)$.

Proof. Since the number of elements of the queue increases one only after a successful broadcast (stated in step 4 of Procedure One-Layer-Resolution) or initialization (stated in step 1 of Algorithm Topological-Ordering), there are n positions which have ever existed in the queue. Let k_i denote the number of vertices occupying position i of the queue. Then $\sum_{i=1}^{n} k_i = n$ since there are n vertices. Note that there may be no vertex residing in some positions, that is, $k_i = 0$ for some i's. Let T denote the average number of time slots required in Algorithm Topological-Ordering. T can be calculated as follows.

$$T = \sum_{i=1}^{n} g_{k_i} \tag{4}$$

To calculate t, we have to define the following three sets:

$$A = \{i \mid 1 \leqslant i \leqslant n,\ k_i = 1\},$$
$$B = \{i \mid 1 \leqslant i \leqslant n,\ k_i = 0\},$$
$$C = \{i \mid 1 \leqslant i \leqslant n,\ k_i > 1\}.$$

Let $|A| = a$, $|B| = b$ and $|C| = c$. We have $a + b + c = n$. (4) can be rewritten as

$$T = \sum_{i \in A} g_{k_i} + \sum_{i \in B} g_{k_i} + \sum_{i \in C} g_{k_i}. \tag{5}$$

By definition,

$$\sum_{i \in A} g_{k_i} = a \quad \text{and} \quad \sum_{i \in B} g_{k_i} = b. \tag{6}$$

We also have

$$\sum_{i \in C} k_i = n - a.$$

From Lemma 1, it follows

$$\tfrac{8}{3}(n-a) - \tfrac{1}{3}c \leqslant \sum_{i \in C} g_{k_i} \leqslant 3(n-a) - c. \tag{7}$$

Combining (5), (6) and (7), we get

$$\tfrac{8}{3}n + \frac{3b - 5a - c}{3} \leqslant T \leqslant 3n - 2a + b - c.$$

Replacing c by $n - a - b$, this inequality becomes

$$\tfrac{7}{3}n + \frac{4b - 4a}{3} \leqslant T \leqslant 2n - a + 2b.$$

By the definitions of A and B, we have

$$b - a \geqslant -n \quad \text{and} \quad 2b - a \leqslant 2(n-1).$$

Now, we obtain the final result

$$n \leqslant T \leqslant 4n - 2.$$

Since each processor can perform its task in constant time after one successful broadcast, the total time required in the algorithm is $h \cdot T$, where h is a constant. Thus, the average time complexity of the algorithm is $\Theta(n)$. □.

5. Breadth-first search and the connected component problem

In this section, we shall apply the technique for solving the topological ordering problem to breadth-first search [4,6,13,17] in a connected undirected graph. Then, we use the technique of breadth-first search to solve the connected component problem [5,13,17].

The problem of breadth-first search for us is: Given a connected undirected graph $G = (V, E)$ and a root (starting) vertex $r \in V$, find a legal sequence for breadth-first search in G.

The algorithm for solving the breadth-first search problem can be obtained from Algorithm Topological-Ordering by slight modification as follows. Step 1 in Algorithm Topological-Ordering is modified to:

> The root vertex is put into the first position of the queue.

And in Procedure One-Layer-Resolution, a part of step 4 is modified to:

> When only one vertex broadcasts successfully, all of its neighboring vertices which have not been put into the queue are put into the queue.

After doing the above modification, we can solve the breadth-first search problem. The sequence of successful broadcasts forms a legal sequence for breadth-first search. The average time complexity is the same as that of Algorithm Topological-Ordering, which is $\Theta(n)$.

Now, we shall solve the connected component problem. Our connected component problem is: For a given undirected graph $G = (V, E)$, we assign a component number to each vertex such that this component number is the smallest vertex label among the vertices in the same connected component.

We use the technique of breadth-first search to solve this problem. Initially, one vertex in G is randomly chosen as the root vertex. Then, breadth-first search in G is performed. After the search terminates, an entire connected component is found. We can use a straightforward algorithm with linear time to find the smallest vertex label and to assign it to each vertex in the component. Meanwhile, a grand iteration for finding a component ends. After a grand iteration finishes, a grand iteration for finding another component begins. Again, one of the remaining vertices in G is selected as the root vertex randomly. Repeatedly, perform breadth-first search in the remaining vertices in G. All connected components will be found out eventually.

To avoid all remaining vertices to compete for the root vertex at the beginning of a grand iteration, we can maintain a stack-like structure in each processor similar to that used in Algorithm Topological-Ordering to overcome this problem. Thus, the average time complexity of the algorithm for the connected component problem is also $\Theta(n)$ if n processors are used and there are n vertices in the given graph.

6. Concluding remarks

The time required for an algorithm executed under the broadcast communication model is divided into three parts: resolution time, transmission time and computation time. If we want to reduce the required time, we may reduce one of them. However, in many situations, it is hard to reduce either transmission time or computation time. The remaining thing we can do is to make every effort to reduce the resolution time. To minimize broadcast conflicts is one of the ways to achieve this goal.

In this paper, we propose an algorithm, under the broadcast communication model, to solve the topological ordering problem, which is superior to the previous one [17]. In our algorithm, a graph is viewed to consist of many layers. Only after processing all vertices in one layer, we process the vertices in another layer. In such a situation, only the vertices in the same layer may induce broadcast conflicts. Hence, the total number of broadcast conflicts occurring in the algorithms is reduced. The average time complexity of the algorithm is proved to be $\Theta(n)$ if n processors are used and there are n vertices in the graph. We also apply this technique to the breadth-first search and the connected component problems. We obtain the same complexity for these two problems.

As mentioned by Dechter and Kleinrock [1], the broadcast communication model is very close to the CREW PRAM model with only one shared word. If k words are shared in the CREW PRAM model, this model corresponds to k channels available in the broadcast communication model. We surely can simulate the algorithms for the PRAM model in the broadcast communication model. However, most algorithms under the PRAM model use many memory spaces, even unbounded memory spaces. Thus, they are not applicable to the broadcast communication model because the bottle neck becomes the shared memory or shared channels. We have to design new algorithms for the broadcast communication model. In [10,18], the authors discussed the broadcast communication model with multiple channels. It may be a direction for future research.

References

[1] J.I. Capetanakis, Tree algorithms for packet broadcast channels, *IEEE Trans. Information Theory* **25** (5) (1979) 505–515.

[2] P. Chaudhuri and R.K. Ghosh, Parallel algorithms for analyzing activity networks, *BIT* **26** (1986) 418–429.

[3] R. Dechter and L. Kleinrock, Broadcast communications and distributed algorithms, *IEEE Trans. Comput.* **35** (3) (1986) 210–219.

[4] E. Dekel, D. Nassimi and S. Sahni, Parallel matrix and graph algorithms, *SIAM J. Comput.* **10** (4) (1981) 657–675.

[5] N. Deo, *Graph Theory with Applications to Engineering and Computer Science* (Prentice-Hall, Englewood Cliffs, NJ, 1974).

[6] S. Even, *Graph Algorithms* (Computer Science Press, Potomac, MD, 1979).

[7] L. Kucera, Parallel computation and conflicts in memory access, *Inform. Process. Lett.* **14** (2) (1982) 93–96.

[8] S. Levitan, Algorithms for broadcast protocol multiprocessor, in: *Proc. 3rd Internat Conf. on Distributed Computing Systems* (1982) 666–671.

[9] S.P. Levitan and C.C. Foster, Finding an extremum in a network, in: *Proc. 1982 Internat. Symp. on Computer Architecture*, Austin (1982) 321–325.

[10] J.M. Marberg and E. Gafni, Sorting and selection in multi-channel broadcast networks, in: *Proc. 1985 Internat. Conf. on Parallel Processing* (1985) 846–850.

[11] R. Metcalfe and D. Boggs, Ethernet: Distributed packet switching for local computer networks, *Comm. ACM* **19** (7) (1976) 395–404.

[12] K.V.S Ramarao, Distributed sorting on local area networks, *IEEE Trans. Comput.* **37** (2) (1988) 239–243.

[13] E.M. Reingold, J. Nievergelt and N. Deo, *Combinatorial Algorithms: Theory and Practice* (Prentice-Hall, Englewood Cliffs, NJ, 1977).

[14] C.Y. Tang and M.J. Chiu, Distributed sorting on the serially connected local area networks, in: *Proc. 1989 Singapore Internat. Conf. on Networks*, Singapore, pp. 458–462.

[15] C.Y. Tang and S.C. Wu, Parallel graph algorithms under broadcast communication model, in: *Proc. Internat. Computer Symp. 1988*, Taipei, Taiwan, ROC, pp. 759–763.

[16] C.B. Yang, R.C.T. Lee and W.T. Chen, Finding minimum spanning trees based upon single-channel broadcast communications, in: *Proc. Internat. Computer Symp. 1988*, Taipei, Taiwan, ROC, pp. 1451–1456.

[17] C.B. Yang, R.C.T. Lee and W.T. Chen, Parallel graph algorithms based upon broadcast communications, *IEEE Trans. Comput.* **39** (12) (1990) 1468–1472.

[18] C.B. Yang, R.C.T. Lee and W.T. Chen, Conflict-free sorting algorithms under single-channel and multi-channel broadcast communication models, in: *Proc. Internat. Conf. on Computing and Information*, Ottawa, Canada, May 1991; also in: *Lecture Notes in Computer Science* **497** (Springer, Berlin, 1991) 350–359.