Journal of Physical Sciences, Vol. 10, 2006, 178 – 187

Computation of Average Distance, Radius and Centre of a Circular-Arc Graph in Parallel

Anita Saha
Department of Mathematics, Bankura Unnayani Institute of Engineering, Subhankar Nagar, Pohabagan, Bankura-722142, West Bengal, India
email : anita.buie@gmail.com

Received 11 November, 2006 ; accepted 12 December, 2006

ABSTRACT

The determination of centre of a graph is very important task in facility location problem. Computation of centre depends on the computation of radius of the graph. In this paper, we have design some parallel algorithms to find the average distance, radius, diameter and centre of a circular-arc graph. The proposed parallel algorithms run in $O(n^2/p + \log n)$ time on a EREW PRAM, where p and n represent respectively the number of processors and the number of vertices of the circular-arc graph.

Keywords : Design of algorithms, Analysis of algorithms, Parallel algorithms, Shortest-paths, average distance, Centre, Diameter, Circular-arc graph.

1. Introduction

A graph $G = (V, E)$ is called an *intersection graph* for a finite family F of a non-empty set if there is a one-to-one correspondence between F and V such that tow sets in F have non-empty intersection if and only if their corresponding vertices in V are adjacent to each other. We call F an *intersection model* of G. For an intersection model F, we use $G(F)$ to dentoe the intersection graph for G. If F is a family of arcs on a circle, then G is called a *circular-arc* graph for F and F is called a *circular-arc model* of G. If F is a family of line segments on real line, then G is called an *interval graph* for F.

Circular-arc graphs have many applications in different fields such as genetic research, traffic control, computer complier design etc. Tucker [43], proposed an $O(n^3)$ time algorithm for recognizing a circular-arc graph. Deng *et al.* [11] presented an $O(n + m)$ time algorithm for presentation of circular-arc graph.

1.1. Our work

In this paper, parallel algorithms are presented to compute the average distance of a circular-arc graph, eccentricities of all vertices of a circular-arc graph, radius and diameter of a circular arc-graph and the centres of a circular-arc graph. The above mentioned algorithms are designed based on the parallel algorithm CAPSP [36] which is used to find all-pair shortest paths on circular-arc graphs. The parallel algorighms take $O\left(\frac{n^2}{p}+\log n\right)$ time using p processors on an EREW PRAM.

2. Definitions and Notations

Let $S = \{a_1, a_2,..., a_n\}$ be a family of n arcs on a circle C. Each endpoint of the arcs in assigned to a positive integer, called a *coordinate*. The endpoints of each arc are ocated on the circumference of C in the ascending order of the values of the coordinates in the clockwise direction. For convenience, each are a_i, $i = 1, 2,..., n$, is represented as (h_i, t_i), where h_i (the *head*) and t_i (the *tail*) denote, respectively that starting and ending points of the arc when it is traversed in counterclockwise manner, starting with an arbitrary chosen point on C which is not an endpoint of any arc in S.

Without loss of generality, we assume the following :

(i) no single arc in S covers the entire circle C by itself (otherwise, the shortest path problem becomes trivial and in this case the distance between any two arcs is either 1 or 2 unit),

(ii) no two arcs share a common endpoint,

(iii) $U_{i=1}^{n} C_i = C$ (otherwise, the problem becomes one on interval graph),

(iv) the endpoints of the arcs in S are already given and sorted, according to the order in which they are visited during the clockwise (anticlockwise, if necessary) traversal along C by starting at a_1, and

(v) the arcs are sorted in increasing values of h_i's i.e., $h_i > h_j$ for $i > j$.

The family of arcs S is said to be *cannonical* if

(i) h_i's and t_i's for $i = 1, 2, ..., n$ are all distinct integers between 1 and $2n$, and

(ii) point 1 is the head of the arc a_1.

A *path* of a graph G is an alternating sequence of distinct vertices and edges, beginning and ending with vertices. The length of a path is the number of edges in the path. A path from vertex i to j is a *shortest path* if there is no other path from i to j with lower length. The shortest distance (i.e., the length of the shortest path) between the vertices i and j is denoted by $\delta(i,j)$.

For illustration we consider circular-arc graphs shown in Figure 1.

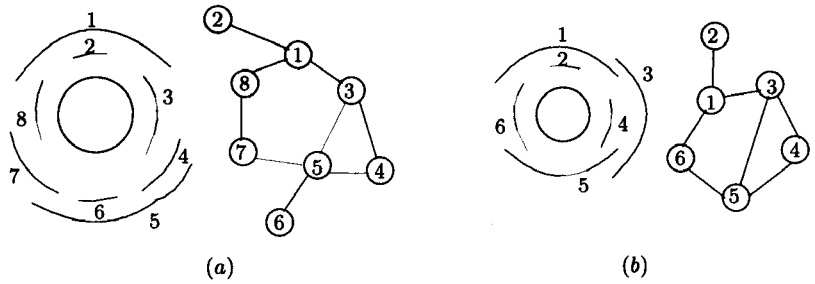

Figure 1 : Circular-arc graphs and their circular-arc representation.

Alternatively, a circular-arc graph can be defined as follows :
An undirected graph $G = (V, E)$ is a *circular-arc* graph if and only if
(i) its vertices circularly indexed as $v_1, v_2,...,v_n$, and
(ii) $(v_i, v_j) \in$ E, provided a_i and a_j intersect with each other, where v_i and v_j are the vertices in the graph G corresponding to the arcs a_i and a_j is S respectively.
It may be noted that the arc a_i and the vertex v_i or i are one and the same thing.

3. All Pairs Shortest Distances

Shortest-paths problems are among the most fundamental and also the most commonly encountered graph problems, both in themselves and as subproblems in more complex setting [3]. Besides obvious applications like preparing travel time and distance charts [21], shortest paths computation is frequently needed in telecommunications and transportation industries [38], where message or vehicles must be sent between two geographical locations as quickly or as cheaply as possible. Other examples are complex traffic flow simulatyions and planning tools [21], which rely on a large number of individual shortest paths problems. further applications include many practical integer programming problems. Shortest paths computations are used as subroutines in solution procedure for computational biology (DNA sequence alignment [44], VLSI design [8], knapsack packing problems [18], travelling salesman problems [24] and for many other problems. A diverse set of shortest path models and algorithms have been developed to accommodate these various applications.

The sequential algorithm of $O(n^3)$ time to solve all-pairs shortest paths (APSPs) problem on arbitrary graph with n vertices due to Floyd [16] is well known. Ahuja *et al.* [1] have given a faster sequential algorithm using Radix heap and Fibonacci heap for the single source shortest path problem; the algorithm runs in $O\left(m + n\sqrt{\log C}\right)$ time for a newtork with n vertices and m edges and non-negative integer arc costs bounded by C. In [39], Seidel has given an $O(M(n) \log n)$ time sequential algorithm for APSP problem for an undirected and unweighted arbitrary graphs with n vertices, $M(n)$ being the time necessary to multiply two

$n \times n$ matrices of small integers; the best known time for $M(n)$ is of $O(n^{2.376})$. Alon *et al.* [2] have reported a sub-cubic algorithm for computing all-pairs shortest distances on directed graphs with integer edge-lengths. The algorithm requires $O((Wn)^n)$ time, where $n = (3 + \omega)/2$, $\omega < 3$, and W is the largest edge-length. Galil *et al.* [19] have improved the dependence on W and have also given an $O(W^{(\omega + 1)/2} n^{\omega} \log n)$ algorighm for undirected graphs. Fredman [17] given the first subcubic algorithm for all-pairs shortest paths. His algorithm runs in $O\left(n^3 \left(\log\log n / \log n \right)^{1/3} \right)$ time. Later Takaoka [41] improved the upper bounds for all-pairs shortest paths to $O\left(n^3 \left(\log\log n / \log n \right)^{1/2} \right)$. Dobosiewicz (6) given an upper bound of $O\left(n^3 / \left(\log n \right)^{1/2} \right)$ with extended operations such as normalization capability of floating point numbers in $O(1)$ time. In [22], Han presented an improve algorithm to find all-pairs shortest paths. The algorithm runs in $O\left(n^3 \left(\log\log n / \log n \right)^{5/7} \right)$ time. An algorithm is presented in [25] to find the next-to-shortest paths in a general graph. When the size of the graph is very large, i.e., the entire graph cannot be stored into the main memory then a new type of the algorighm is required to solve all-pairs shortest path problem. In [40], an algorithm is designed to find external matrix multiplication and this result is used to solve all-pairs shortest path problem. Ravi *et al.* [35] and Mirchandani [27] have given sequential algorithms to solve APSP problem on an interval graph in $O(n^2)$ time. Pal and Bhattacharjee [34] have designed a parallel algorithm to solve APSP problem on interval graphs using $O(n^2/p + \log n)$ time and p processors on a EREW PRAM. Mondal *et al.* have solved APSP problem on permutation graphs [30] and on trapezoid graphs [29] in $O(n^2)$ time.

	1	2	3	4	5	6	7	8
1	0	1	1	2	2	3	2	1
2	1	0	2	3	3	4	3	2
3	1	2	0	1	1	2	2	3
4	2	3	1	0	1	2	2	3
5	2	3	1	1	0	1	1	2
6	3	4	2	2	4	0	2	3
7	2	3	2	2	1	2	0	1
8	1	2	2	3	2	3	1	0

Table 1 : The all pairs shortest distances of the circular-arc graph of Figure 1(a).

Saha *et al.* [36] have designed a parallel algorithm to find all pairs shortest distances of a circular-arc graph.

The all pairs shortest distances, using the algorithm CAPSP of Saha *et al.* [36], of the graph of Figure 1(a) are shown in Table 1.

The complexity of the algorithm CAPSP [36] to compute the all-pair shortest distances is stated in the following theorem.

Theorem 1. *The all-pairs shortest distances of a circular-arc graph with n vertices can be computed in* $O(n^2/p + \log n)$ *time using p processors on an EREW PRAM.*

The algorithm is cost optimal when p is equal to $O(n^2/log\ \mathrm{n})$.

From the above theorem one can conclude the following result.

Theorem 2. *The all-pair shortest distances of a circular-arc graph with n vertices can be computed in* $O(n^2)$ time.

4. Average Distance

Many works on average distance in graphs are available in literature [4, 5, 12, 13, 15, 23, 28, 42, 45, 46]. Chung [7] give a bound of average distance of a graph in terms of independent number. He shown that $\mu(G) \leq \alpha(G)$, where $\mu(G)$ and $\alpha(G)$ denote respectively the average distance and the independent number of the graph G.

Also in [10], the average distance of an interval graph with edges of unit length can be computed in $O(m)$ time where m is the number of edges. In this section, we discuss about the computation of average distance of a circular-arc graph.

The *average distance* $\mu(G)$ of a connected circular-arc graph is defined to be the average of all distances in G

$$\mu(G) = \frac{1}{n(n-1)} \sum_{\substack{x,y \in V(G) \\ x \neq y}} \delta(x,y),$$

where $\delta(x,y)$ denotes the length of a shortest path joining the vertices x and y. The average distance can be used as a tool in analytic networks where the performance time is proportionsl to the distance between any two nodes. It is a measure of the time needed in the average case, as opposed to the diameter, which indicates the maximum performance time.

4.1. Algorithm to compute average distance and its computation

At first we compute $\delta(x,y)$ for every pair $x, y (x \neq y)$ using Algorith **CAPSP** then we compute the sum of distance between all pairs of vertices, and finally we multiply it by the factor $\frac{1}{n(n-1)}$ to get the average distance. From above procedure it follows that the time to

compute the average distance is same as the time to compute all pairs shortest distances. Hence, we can draw the following conclusion.

Theorem 3. *The average distance of a circular-arc graph can be computed in sequential using $O(n^2)$ time, where n is the number of vertices of the graph.*

Theorem 4. *The average distance of a circular-arc graph with n vertices can be computed in parallel using $O(n^2/p + \log n)$ time and p processors on an EREW PRAM. The algorith is cost optimal when p is equal to $O(n^2/p + \log n)$.*

vertex	1	2	3	4	5	6	7	8	Total
distance	12	18	11	14	11	17	13	14	110

Table 2 : The sum of the distance from a vertex to all other vertices of the graph of Figure 1(a).

The total distances from a vertex to all other vertices of the graph of Figure 1(a) are shown in Table 2.

Finally, the average distance of the graph of Figure 1(a) is

$$\mu(G) = \frac{55}{28}.$$

5. Centre and Diameter

Location problem is a topic of great important in the fields such as transporation, communication, service areas and computer sciences. The criteria for the locating problem in the literature are minmax criteria in which the distance to the furthest vertex from the site is minimized and minsm criteria in which the total distance to the vertices from the site is minimized.

The *eccentricity* $e(i)$ of a vertex i in a graph is the distance from vertex i to a vertex furthest from i. Vertex j is said to be a *furthest neighbour* of the vertex i if $\delta(i, j) = e(i)$. The *diameter* of a graph G is the maximum among all eccentricities. The *radius* of a graph is the minimum among all eccentricities. A *centre* of a graph is a vertex whose eccentricity equal to radius.

The eccentricity, diameter (diam (G)), radius ($\rho(G)$) and centre ($C(G)$) of a graph are defined as follows.

$$\begin{aligned} e(i) &= \max \{\delta(i, j) : j \in V\} \\ \text{diam } (G) &= \max \{e(i) : i \in V\} \\ \rho(G) &= \min \{e(i) : i \in V\} \\ C(G) &= \{u \ V : e(u) = \rho(G)\}. \end{aligned}$$

The cemtre of a graph may be a single vertex or more than one vertices.

Determining the diameter of a graph is a fundamental and seemingly quite time-consuming operation. For arbitrary graphs, the current algorith runs in $O(nm)$ time, which is too slow to be practical for very large graphs [9].

For some particular types of graph such as tree [20], outerplanar graph [14] etc. linear time algorithms can be devised to compute the centre. For the interval graph $O(n)$ time sequential algorithm is presented for computing diameter and centre of an interval graph with n vertices by Pal and Bhattacharjee [32]. In [31], Olariu has presented an $O(n + m)$ time sequential algorith where input is an adjacency list that takes $O(n + m)$ space, where n and m are the number of vertices and edges respectively. A linear time algorith for solving the centre problem is presented by Lan *et al.* [26] for cactus graph.

vertex	1	2	3	4	5	6	7	8
eccentricities	3	4	2	3	3	4	3	3

Table 3 : The eccentricities of the vertices of the graph of Figure 1(a).

5.1. Procedure to compute radius, diameter and centre

In this section, a method is presented to find the eccentricities of all vertices. Using the eccentricities of the vertices the diameter, radius and centre of a circular-arc graph is computed.

For a given circular arc graph, $G = (V, E)$ the all pairs shortest distances $\delta(x, y)$ for all $x, y \in V$ can be determined by using the algorith of Saha *et al.* [36]. Then the eccentricity of the vertex x can be compted using the formla

$$e(u) = \max \{\delta(u, v) : v \in V\} \text{ for all } u \in V.$$

The radius $\rho(G)$ and diameter diam (G) are determined by the formula

$$\rho(G) = \min \{e(u) : u \in V\}$$

and

$$\text{diam } (G) = \max \{e(u) : u \in V\}$$

After determination of the radius one can test whether a vertex is a member of the cener or not using the following technique :

If $e(u) = \rho(G)$ then u is a member of $C(G)$ for all $u \in V$.

The eccentricities of all vertices of the graph of Figure 1(a) are shown in Table 3.

The radius and the diameter of the graph of Figure 1 are

$$\rho(G) = \min \{e(u) : u \in V\} = 2$$

and

$$\text{diam } (G) = \max \{e(u) : u \in V\} = 4.$$

Since the radis of the graph of Figure 1(a) is 2 and which is attained at the vertex 3, the center of this graph is 3, i.e., $C(G) = \{3\}$. While the center of the graph of Figure 1(b) is $C(G) = \{1, 3, 6\}$.

Since the radius of the graph of Figure 1(a) is 2 and which is attained at the vertex 3, the centre of this graph is 3, i.e., $C(G) = \{3\}$. While the centre of the graph of Figure 1(b) is $C(G) = \{1, 3, 6\}$.

Since the all-pairs shortest distances are known, the eccentricities of all vertices can be computed using the same time and processors, used to solve all-pairs shortest distances.

The following results follow from the definitions of radius, diameter, centre and theorems 1 and 2.

Theorem 5. *The eccentricities of all vertices, diameter, radius and centre of a cricular-arc graph with n vertices can be computed in sequential using* $O(n^2)$ *time.*

Theorem 6. *The eccentricities of all vertices, diameter, radius and centre of a circular-arc graph with n vetices can be computed in parallel using* $O(n^2/p + \log n)$ *times and p processors.*

REFERENCES

1. Ahuja, R. K., Mehlhorn, K., Orlin, J. B. and Tarjan, R.E., Faster algorithm for the shortest path problem, J. ACM, 37(2) (1990) 213-223.
2. Alon, N., Galil, Z. and Margalit, O., *On the exponent of the all pairs shortest path problem,* in Proc. : *32th IEEE FOCS, IEEE* (1991) 569-575.
3. Ahuja, R. K., Magnanti, T. L. and Orlin, J. B., *Network Flows : Theory, Algoriths, and Applications,* (Prentice Hall, 1993).
4. Althofer, I., Average distance in undirected graphs and the removal of vertices, *J. Combin. Theory Ser. B,* 48 (1990) 140-142.
5. Bienstock, D., and Gyon, E., Average distance in graphs with removed elements, *J. Graph Thorey,* 12 (1988) 375-390.
6. Dobosiewicz, W., A more efficient algorithm for min-plus multiplication, *Intern. J. Comput. Math.,* 32 (1990) 49-60.
7. Chung, F. R. K., The average distance and independence number, *J. Graph Theory,* 12 (1988) 229-135.
8. Cong, J., Kahng, A. B. and Leung, K. S., Efficient algorithms for the minimum shortest path Steiner arborescence problem with applications to VLSI physical design, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* 17 (1998) 24-39.
9. Corneil, D. G., Dragan, F. F. and Köhler, E., On the power of BFS to determine a graph's diameter, *Network,* 42 (4) (2003) 209-222.
10. Dankelmann, P., Computing the average distance of an interval graphs, *Information Processing Letters,* 48 (1993) 311-314.

11. Deng, X., Hell, P. and Huang, J., Linear time representation algorithms for proper circular-arc graphs and proper interval graphs, *SIAM J. Comput.,* 25 (1996) 390-403.

12. Erdos, P., Pach, J. and Spencer, J., On the mean distance between points of a graph, *Congr. Numer.,* 64 (1988) 121-124.

13. Fajthlowich, S. and Waller, W. A., On two conjectres of GRAFFITI, *Congr. Numer.,* 55 (1986) 51-56.

14. Farlay, A. and Proskurowski, A., Computation of the center and diameter of outerplanar graphs, *Discrete Applied Math.,* 2, (1980) 185-191.

15. Favaron, O., Kouider, M, and Maheo, M., Edge-vulnerability and mean distance, *Networks,* 19 (1989) 493-504.

16. Floyd, R., Algorithm 97 : shortest path, *Comm. ACM,* **5** (1962) 345.

17. Fredman, M. L., New bounds on the complexity of the shortest path problem, *SIAM J. Computing,* 5 (1976) 83-89.

18. Frieze, A., Shortest path algorithms for knapsack type problems, *Math. Pro-gramming,* 11 (1976) 150-157.

19. Galil, Z. and Margalit, O. All pairs shortest distances for graphs with small integer length edges, *Information and Computing,* 134 (1997) 103-139.

20. Goldman, A. J., Minimax location of a facility in a network, *Transpotation Science,* 6 (1972) 407-418.

21. Golden, B. and Magnanti, T., *Transportation planning : Network models and their implementation,* in : Studies in Operations Management, (1978) 365-518.

22. Han, Y., Improved algorithm for all pairs shortest paths, *Information Processing Letters,* 91 (2004) 245-250.

23. Hendry, G. R. T., On mean distance in certain classes of graphs, *Networks,* 19 (1989) 451-557.

24. Houck, D., Picard, J., Queyranne, M. and Vemuganti, R., The travelling salesman problem as a constrained shortest path problem : Theory and computation experience, *Opsearch,* 17 (1980) 94-109.

25. Krasikov, I. and Noble, S. D., Finding next-to-shortest paths in a graph, *Information Processing Letters,* 92 (2004) 117-119.

26. Lan, Y. -F., Wang, Y. -L. and Suzuki, H., A linear-time algorithm for solving the center problem on weighted cactus graphs, *Information Processing Letters,* 71 (1999) 205-212.

27. Mirchandani, P., A simple $O(n^2)$ algorithm for all pairs shortest path problem on an interval graph, *Networks,* 27 (1996) 215-217.

28. Mohar, B., Eigenvalues, diameter, and mean distance in graphs, *Graphs and Combinatories,* 7 (1991) 53-64.

29. Mondal, S., Pal, M. and Pal, T. K., An optimal algorithm for solving all-pairs shortest paths on trapezoid graphs, *Intern. J. Computational Engineering Science,* 3(2) (2002) 103-116.

30. Mondal, S., Pal, M. and Pal, T. K., An optimal algorithm to solve all-pairs shortest paths problem on permutation graphs, *Journal of Mathematical Modelling and Algorithms,* **2** (2003) 57-65.

31. Olariu, S., A simple linear-time algorithm for computing the center of an interval graph, *Intern. J. Computer Maths.,* 34 (1990) 121-128.

32. Pal, M. and Bhattacharjee, G. P., An optimal parallel algorighm for computing all maximal cliques of an interval graph and its applications, *J. of Institution of Engineers (India),* 76 (1995) 29-33.

33. Pal, M. and Bhattacharjee, G. P., A data structure on interval graphs and its applications, *J. Circuits, Systems, and Computer,* 7 (1997) 165-175.

34. Pal, M. and Bhattacharjee, G. P., An optimal parallel algorithm for all-pairs shortest paths on unweighted interval graphs, *Nordic J. Computing,* 4 (1997) 342-356.

35. Ravi, R., Marathe, M. V. and Pand Rangan, C., An optimal algorithm to solve the all-pair shortest path problem an interval graphs, *Networks,* 22 (1992) 21-35.

36. Saha, A., Pal. M. and Pal, T. K., An optimal parallel algorithm for solving all-pairs shortest paths problem on circular-arc graphs, *Journal of Applied Mathematics and Computing,* 17(1+2) (2005) 1-23.

37. Saha, A., *Sequential and Parallel Algorithms on Some Problems of Intersection Graphs,* Ph. D. Thesis, Vidyasagar University, Midnapore, India, 2005.

38. Schwartz, M. and Stern, T.E., Routing techniques used in computer communication networks, in Proc. : *IEEE Transactions on Communications,* 1980, pp. 539-552.

39. Seidel, R., On the all pairs shortest path problem, in Proc. : *24th ACM STOC. ACM Press,* (1992) 745-749.

40. Sibeyn, J. F., External matrix multiplication and all-pairs shortest path, *Information Processing Letters,* 91 (2004) 99-106.

41. Takaoka, T., A new upper bound on the complexity of the all pairs shortest path problem, *Information Processing Letters,* 43 (1992) 195-199.

42. Tomescu, I. and Malter, R. A., On distances in chromatic graphs, *Quart. J. Math. Oxford (2),* 40 (1989) 475-480.

43. Tucker, A., An efficient test for circular-arc graphs, *SIAM J. Comput.,* 9 (1980) 1-24.

44. Waterman, M. S., *Mathematical Methods for DNA Sequence,* (CRC Press, Boca Raton, FL, 1988).

45. Winkler, P., Mean distance and the 'four-thirds conjecture', *Congr. Numer.,* 54 (1986) 63-72.

46. Winkler, P., Mean distance in a tree, Discrete Applied Math., 27 (1990) 179-185.