Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

An efficient parallel algorithm for shortest paths in planar layered digraphs

MISHRA P.K.

(*Dept. of Applied Mathematics, Birla Institute of Technology, Mesra, Ranchi 835215, India*)

E-mail: pkmishra@ieee.org; pkmishra@bitmesra.ac.in

Received July 21, 2003; revision accepted Nov. 4, 2003

Abstract: This paper presents an efficient parallel algorithm for the shortest path problem in planar layered digraphs that runs in $O(\log^3 n)$ time with n processors. The algorithms uses a divide and conquer approach and is based on the novel idea of a one-way separator, which has the property that any directed path can be crossed only once.

Key words: Parallel algorithms, Shortest paths, Planar layered digraphs
Document code: A **CLC number:** O29

INTRODUCTION

Computing shortest paths in directed graphs is a fundamental optimization problem with applications to many areas of computer sciences and operation research (Bellman, 1958; Coremen, 1990; Cohen, 1993). Given a digraph G with non-negative weights on its edges and two vertices s and t of G, the single-pair shortest path problem consists of determining a directed path from s to t with minimum total weight.

Among the well-known sequential algorithms for this problem is the classical Dijkstra's algorithm (Dijkastra, 1995) based on a dynamic programming approach. Its time complexity is $O((n+m)\log n)$ if elementary data structures are used, and $O(n\log n - m)$ when implemented with Fibonacci heaps (Fredman and Tarjan, 1987). For the important class of acyclic digraphs, a simple variation of Dijkstra's algorithm runs in $O(n+m)$. Here n and m denote the number of vertices and edges of G, respectively.

Developing a parallel shortest path algorithm that runs in polylogarithimic time with a linear number of processors is an outstanding open pro blem. Indeed, all the known polylog-time parallel techniques for this problem are based on matrix multiplication and are therefore far from optimal in terms of the total work done, especially when the number of digraphs is sparse. For general digraphs, the best algorithm runs in $O(\log^2 n)$ time with $n^3/\log n$ processor (using the naive algorithm for matrix multiplication). Alon and Galil (1991) gave an algorithm using fast matrix multiplication techniques to solve all the pairs shortest paths problem. The work done by their algorithm is $O[\frac{mn(3+\omega)}{2\log^3 n}]$ if the edges have integral weights bounded above. However, for calculating single-source shortest paths or for the single-pair shortest path problem, even this algorithm is far from optimal. For planar undirected graphs the number of processors for the single source problem can be reduced to $n^{1.5}/\log^3 n$ while keeping the parallel time down to $O(\log^3 n)$, by using the nested dissection technique of Mishra and Sharma (1997; 1999). Further improvements

have generalized these ideas to include directed planar graphs while keeping the time and processors bounds the same.

To our knowledge, efficient parallel algorithm for computing shortest paths has been devised only for two special classes of digraphs: series-parallel digraphs and grid digraphs (which are acyclic digraphs with exactly one source and exactly one sink recursively constructed by series and parallel compositions). In series-parallel digraphs, the weight of a shortest paths between the sources and sink is obtained by evaluating an arithmetic expression with operators + (associated with series compositions) and – (associated with parallel compositions), which can be done optimally in $O(\log n)$ time with $n/\log n$ processors.

A grid digraph has the vertices arranged in a rectangular grid and the edges directed from left to right and from bottom to top (Delche *et al*., 1992; Cohen, 1993). Apostolico *et al*.(1990) gave an algorithm to compute shortest paths in a grid in $O(\log^2 n)$ time with $O(n)$ processors. The shortest path problem of grid digraphs has applications in text processing, biological research, topography, and medical diagnosis (Levcopoulos and Lingas, 1992).

In this work we consider a class of digraphs that extend grid digraphs, namely, planar layered digraphs. In a planar layered digraph the vertices are arranged along parallel lines called layers and edges, that connect vertices of consecutive layers and do not intersect.

The rest of this paper is organized as follows.

In Section 2 we formally define planar layered digraphs and introduce the notion of one way separators.

In Section 3 we prove the existence of a one way separator for planar digraphs and show how to use such a separator to design an efficient divide and conquer shortest path algorithm.

In Section 4 we discuss how our result relate to the previous one for grid digraphs.

Finally, in Section 5 we present our conclusions and comments on the open problem of finding shortest paths in planar st-graphs.

PRELIMINARIES

Let $G=(V, E)$ be a directed acyclic graph with n vertices and m edges. We say that G is layered digraph if V is partitioned into p subsets, called layers, $l_1, l_2, ..., l_p$ such that all edges of G are between consecutive layers. Given p parallel lines consecutively numbered in the plane, a p-line embedding of G is a drawing such that:

All vertices of layer l_i are drawn on line i. Edges are drawn as straight line. A planar p-line embedding is a p-line embedding without crossing. G is a planar layered digraphs if it admits a planar p-line embedding.

Layered graphs have been studied under the name of proper hierarchies. Di Battista and Nardelli (1987) gave efficient algorithms to test if a layered digraph with only one source is planar. Recently Lempel *et al*.(1986) developed an efficient parallel algorithm to evaluate planar layered circuits.

We can transform any planar layered digraph G into a planar layered st-digraph with exactly one source and one sink in the following manner:

We first introduce two new vertices s and t and put them in two new layers, one at the beginning and one at the end. Then, for each sink x in layer i, we find the closest node y to its left that is attached to a node in layer $i+1$. We then let z be the rightmost node in layer $i+1$ that is attached to y. If there is no such node y, then we set z to be the leftmost node in layer $i+1$.

To make x a non sink vertex we introduce an infinite weight edge from x to z. For example in a single-source single-sink planar layered digraph are the infinite weight edges (Generich and Lautenbach, 1973). It is not hard to see that these infinite weight edges do not alter any shortest paths or destroy planarity. This computation can be accomplished in $O(\log n)$ time with $n/\log n$ processors using standard techniques (Jàjà, 2001). A planar layered digraph with exactly one source s (on the first layer) and one sink t (on the last layer) is called a planar layered st-graph (Preparta and Shamons, 1985). In the rest of the paper we solve the problem of finding the shortest path between the source and the sink of a

planar layered st-graph. To find the shortest path between any two vertices u and v in G we perform a pre-processing step to remove all the vertices which are not on any path between u and v.

A brief outline of the algorithm is as follows.

Adding extra edges to create a planar layered st-graph:

Step 1. Let T_u be the set of vertices that are on some path from u to t in G.

Step 2. Let S_v be the set of vertices that are on some path from s to v in G.

Step 3. Discard all the vertices in G that are not in T_u or S_v.

Theorem 1 The algorithm outlined above can be implemented in $O(\log n)$ time with $n/\log n$ processors in an EREW PRAM.

Proof All the steps can be done in $O(\log n)$ time using the techniques of Tamassia and Preparta (1990). A planar layered st-graph is a special case of a planar st-graph which is defined as a planar acyclic digraph with exactly one source s, and exactly one sink t, embedded in the plane so that s and t are on the boundary of the external face.

These graphs were first introduced in the planarity testing algorithm of Lempel *et al.*(1986).

We now define the concept of a left ordering of the vertices in a planar st-graph, which proves useful in our algorithm. This ordering was introduced by Tamassia and Preparta (1990), Gondram and Minoux (1984). We do this by making use of the dual graph of G (labeled G^*) defined as follows:

(1) Every internal face f in G corresponds to a vertex in G^*.

(2) The dual edge e^* of an edge e is directed from the face to the left of e to the face to the right of e.

(3) The external face of G is dualized to two vertices of G^*, denoted s^* and t^* which are incident with the duals of the edges on the leftmost and rightmost paths from s to t, respectively.

Definition 1 For every $x \in V$ we denote by $left(x)$ and $right(x)$ the two faces that separate the incoming and outgoing edges of a vertex $x \neq s, t$: for $x=s$ or $x=t$, we conventionally define $left(x) = s^*$ and $right(x)=t^*$.

Definition 2 We say that x is below y, denoted $x\uparrow y$, if there is a path in G from x to y. Also, we say that x is to the left of y, denoted $x \rightarrow y$ if their is a path in G^* from $right(x)$ to $left(y)$.

Definition 3 The left ordering (denoted $<_l$) is defined on the basis of the relations "left" and "below".

A vertex x occurs before another vertex in the ordering if either $x \rightarrow y$ or $x\uparrow$.

Tamassia and Preparta (1990) gave optimal EREW algorithms to construct the left ordering of a planer st-Graph.

Definition 4 Given a graph $G=(V, E)$ with n vertices, we call a set $X \subset V$ and $f(n)$-separator that δ-splits G if $|X| \leq f(n)$ and the vertices in V-X can be partitioned in sets $\{A_1, A_2, A_3, ..., A_k\}$ such that their are no edges between any two sets A_i and A_j and for all A_i we have $|A_i|\ \delta_n$.

Lipton and Tarjan (1979; 1980) proved that any planar graph with n vertices has a $\sqrt{\delta_n}$ -separator that (2/3)-splits. This result and other extensions to it proved the way to divide-and-conquer solutions for many problems in planar graphs. A parallel algorithm for finding a cycle separator for biconnected graphs which uses n-processors if the breadth-first search tree of the graph is already known was given by Miller and Thurston (1986). An improved version of the algorithm which uses randomization to find a cycle separator with $n^{1+\varepsilon}$ processors was given by Garit and Miller (1987).

Randomized parallel algorithms to find small separators for more general undirected graphs were given by Miller and Thurston (1986). However, these separators seem unsuitable for use in solving problems on directed planar graphs because they do not take into account the direction of edges, while separating the graph.

For example when considering the problem of determining the shortest path from s to t, the existence of a small separator which separates the underlying undirected graph does not guarantee an efficient recursive solution. We get into trouble because even though we can use the separator to divide the original problem into two roughly sub problems, the time taken to patch up the two re-

cursive solutions can be large, as the shortest path from s to t may cross the separator many times. For divide-and-conquer approach we use the following special kind of separator.

Definition 5 Let X be a separator of a digraph $G=(V, E)$ that divides V-X into sets $A_1, A_2, A_3, ..., A_k$ of separator and let p be a simple directed path in G.

We say that p crosses X_r times if there are disjoint subpaths $p_1, p_2, p_3, \ldots, p_r$ of p such that the end points of each p_i are in different sets A_j and A_i. We call X a one way separator if any directed path p between two vertices in G crosses X at most once.

A division of G into one or more pieces is a one-way division if the separator X used to divide G is a one-way separator.

Grid digraphs, for instance have one way separators of size $\sqrt{n}$ that split the graph in half. In fact the shortest-path algorithm by Apostolico *et al.* (1990) uses such separators to construct a divide-and-conquer solution. Our main result in this paper is that planar layered digraphs also admit small one way separators which we use to construct a recursive solution to the shortest-path problem.

SHORTEST PATHS IN A PLANAR LAYERED st-GRAPH

Let $G=(V, E)$ be a planar layered st-graph with source s and sink t. In this section we show how to determine the shortest path from s to t by a divide-and-conquer approach. The crux of the algorithm lies in finding one way separators. We show how to find one way separators X_1, X_2, and X_3 such that using three one way divisions we can partition G into at most four pieces A, B, C and D, none of which have more than two-thirds of the vertices in G. We then show that this division can be used to formulate a recursive solution to the single source shortest paths problem.

To show the existence of planar layered separators we need the following Lemma, which follows directly from the arguments of Lipton and Tarjan (1980).

Lemma 1 Let G be any n-vertex planar layered st-graph containing layers 1 through p. Let S_i denote the set of vertices in the ith layer, and n_i denotes the size of the set S_i. Also let $p+1$ be an additional layer containing no vertices. Given any layer j, a layer $i \le j$ exists such that $n_i+2(j-1) \le 2\sqrt{t_j}$ where t_j denotes the number of vertices in layer l through j. Similarly a layer j, and a layer $k \ge j$ exists such that

$$n_k+2(k-j-1) \le 2\sqrt{(n-t_j)}\ . \tag{1}$$

Theorem 2 Given an n-vertex planar layered st-graph G containing layers 1 through p, at most three one-way separators X_1, X_2, and X_3 exists such that $X=X_1 \cup X_2 \cup X_3$ 2/3-splits G into up to four pieces.

Furthermore,

$$|X| = O(\sqrt{n})\ . \tag{2}$$

Proof Given G, let j be the smallest layer such that the layers l through j have at least $n/2$ vertices. Let i and k be layers as in Lemma 1. We let X_1 be the set of vertices S_i in layer i, and let X_2 be the set of vertices S_k in layer k.

Let A be the part of G above k, let B be the part below I, and let M be the remaining part in the middle. By construction A and B each have less than $2n/3$ vertices. If the number of vertices in M is less than $2n/3$. We label M as C and we are done. Otherwise we form a new planar layered st-graph G' from the middle part M and two new vertices s', and t', adding edges from all the vertices in layer k to t' and from s' to all the vertices in layer k to t' and from s' to all the vertices in layer l. Let m be the middle vertex in the left ordering L_1 or G', and let q be a path constructed by taking the leftmost outgoing edges from M until reaching t' and the rightmost incoming edge from m until reaching. Let m be the middle vertex in the left ordering L_1 or G', and let q be a path constructed by taking the leftmost outgoing edges from M until reaching t', and the rightmost incoming edges from m until reaching s'.

The path q splits G' into two equal parts C and D. We let X_3 be the set of vertices on q.

We now claim that X_1, X_2 and X_3 are one-way separators and that $X=X_1\cup X_2\cup X_3$ is an $O(\sqrt{n})$ separator that (2/3)-splits G. It is easy to see that none of the pieces have more than (2n/3) vertices; therefore X2/3-splits G. By construction $|q|\le k-i+1$, therefore

$$X \le X_1 \cup X_2 \cup X_3 = n_i + n_k + |n_k + q| \quad (3)$$

$$\le n_i + n_k + (k-i+1) \le 2\left(\sqrt{t_j} + \sqrt{n} - t_j\right) \quad (4)$$

$$\le 2\left(\sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}}\right). \quad (5)$$

$$\le (2\sqrt{2n}) \quad (6)$$

To prove that X_1, X_2, and X_3 form one way separators we note the following:

By the definition of planar layered digraphs, no path in G intersects with vertices in S_i or S_k more than once; therefore X_1 and X_2 are one way separators. To look at the interactions of paths in G with the path q, we look at the two sub paths q_1 and q_2 of q, where q_1 is the portion below the middle vertex and q_2 is the portion above. Let C be the sub graph to the left of q and let D the sub graph to the right of q. By construction, we can see that no path can cross from D to C since no path can leave q_2 from the left. Therefore, X_3 is also a one way separator.

We now show how to obtain rapidly the separator described in Theorem 2 for planar layered st-graph G in parallel.

A brief sketch of the algorithm is outlined below.

Step 1. For every vertex n determine the level in which it is located.

Step 2. Determine the number of vertices in level i for all I.

Step 3. Determine the levels i, j, and k.

Step 4. Construct the middle graph M and determine its weight. If the weight is less than 2n/3 we are done.

Step 5. Construct the left ordering of M and find the middle vertex x.

Step 6. Construct the "leftmost outgoing and rightmost incoming" separator starting (as shown in Theorem 2) from X.

Theorem 3 Given an n-vertex planar layered st-graph G, the algorithm outlined above construct the separator X of Theorem 2 and can be implemented in time $O(\log n)$ using $n/\log n$ processors on an EREW PRAM.

Proof Steps 1–4 can be done using standard technique. Steps 5 and 6 can be completed in $O(\log n)$ time using the techniques of Tamassia and Preparta (1990).

We now use X to construct a divide-and-conquer solution for the shortest path problem. The basic idea is to solve the four all-pairs sub problems (among the boundary vertices) in the pieces A, B, C, and D and patch up efficiently the solutions along the separator X to obtain the shortest path from s to t. We construct planar layered st-graphs from the pieces A, B, C and D by introducing new sources and sink vertices and adding edges from the top and bottom layers. It is easy to see that we introduce only $O(n)$ new vertices b in this fashion. We still have to show that the number of all pairs sub problems at any stage is not too much to handle with just n processors since we are aiming for a linear processor solution. We use the following lemma which follows from the arguments of Lipton and Tarjan (1980) to show that the total number of source-link pairs at any level in the recursion is $O(n\log n)$.

Lemma 2 Let G be an n-vertex planar layered st-graph, and let $0\le\varepsilon\le1$. If the separator X defined in Theorem 2 is used to split the graph G recursively until we have no piece with more than εn vertices, then the number of boundary vertices (vertices which are a part of some separator in the recursive subdivision) is $O\left(\sqrt{\frac{n}{\varepsilon}}\right)$.

Before counting the number of source-sink pairs at some level in the recursion, let us analyze levels more precisely. Every split in our pieces, each of size at most 2k/3, where p has k vertices: However, we want to view the splits in terms of one-way separators X_1, X_2 and X_3, respectively.

Each such division partitions the graph into two pieces, and the four pieces we obtain after the

three divisions have no more than $2k/3$ vertices each. In this manner we build a recessive division tree (RDT), wherein the root corresponds to the graph G, and the levels to graphs of size at most 2: Each internal vertex has two children, which correspond to the two pieces resulting from one way division.

The ith level in the tree consists of all the pieces that are at distance i from the root. It is easy to see that the total number of levels in the RDT is $O(\log n)$ since once every three levels the number of vertices in any piece goes down by a factor of 2/3. We now claim that at each level the number of new source-sink pairs introduced is linear.

Lemma 3 The number of new source-sink sub problems introduced in any level of the recursive division tree is $O(n)$.

Proof By induction. The case i=2 is certainly true, since the separator X we get from Theorem 2 is of size at most $(2\sqrt{2n})$, and therefore the total number of source-sink sub problems introduced in the first level of recursion is at most $8n$. Let us consider some level i in the RDT, such that the induction hypotheses holds for all leaves up to $i-1$. We now prove that only $O(n)$ new source link pairs are introduced in constructing level i.

$$\text{Let } \varepsilon = \left(\frac{2}{3}\right)^{\left(\frac{i-1}{3}\right)}; \tag{7}$$

from the definition of RDT, we know that the size of any piece in level $i-1$ is at most εn. Therefore, by Lemma 2 the number of boundary vertices is $O\left(\sqrt{\frac{n}{\varepsilon}}\right)$. Let B be the set of boundary vertices in level $i-1$.

To construct the next level of the RDT we use separators $X^1, X^2, \ldots, X^l$ each of size at most $2\sqrt{2}\varepsilon n$, to divide the pieces in level $i-1$. Any new source-sink pair in level I will involve some separator vertex $a \in X^j$, and a boundary vertex $b \in B$. Hence, the total number of source-sink pairs introduced is at most

$$2\sqrt{2}\varepsilon n\, O\sqrt{\frac{n}{\varepsilon}} = O(n). \tag{8}$$

From Lemma 3, we know that the total number of source-sink sub problems in any level is $O(n\log n)$, since there are $O(\log n)$ levels and only $O(n)$ new sub problems are introduced in any level. We use the following strategy to solve the shortest path problem:

Step 1. Recursively subdivide the graph until there are only two vertices in any piece.

Step 2. Solve the problem in bottom-up manner, by dynamically assigning processors to each piece depending on the number of source-sink pairs in that piece.

In the remaining part we show how to patch up the result of any two pieces p_1 and p_2 to obtain the shortest-path information of the parent graph p at all levels of the RDT.

Let n_p denote the number of vertices in p. To patch up the pieces p_1 and p_2 we need to update the shortest path along the common boundary. Let us suppose without loss of generality that the source set S is in p_1 and the set T is in p_2, and the common boundary between then is Z. Let n_s, n_t and n_z be the number of vertices in S, T and Z respectively.

We know from Theorem 2 that Z is a one-way separator that is either a layer of vertices or a special "rightmost-edge-in, leftmost-edge-out" path constructed from a middle vertex m. We have already solved the sub problems associated with pieces p_1 and p_2; therefore, we know the all pairs shortest paths from vertices in S to those in Z and from vertices in Z to those in T. On the face of it this seems an easy enough job, since for each pair of vertices (a, b) for which $a \in S$, $b \in T$ all we have to do is check for every vertex in Z whether it is in the shortest path from a to b or not. This can be done in constant time with $|S|$ processors in a CRCW PRAM. However, a little thought reveals that such a naive approach would require too many processors.

Assume without loss of generality that $n_s \le n_t$.

We first use an idea of Apostolico *et al.*(1990) to show that the all pairs shortest paths from ver-

tices in T can be computed in $\log^2 n$ time with $(n_s n_t + n_s n_z)$ processors. We make use of the following definition and lemma.

Definition 6 Let a and b be two vertices separated by a one-way separator $Z \subset V$ in the graph G. We denote by $l(a, b)$ that it is the lowest point in the left ordering of the vertices in X in the graph G such that a shortest path from a to b in G passes through it.

Lemma 4 Let a be a vertex in S and let b_1 and b_2 be vertices in T. We claim that if b_2 occurs after b_1 in the left ordering of the parent graph p, then $l(a, b_2)$ does not occur before $l(a, b_1)$ in the left ordering.

Proof We give a proof by contradiction for the case when Z is a path; the proof when Z is a layer is similar.

Let $m = l(a, b_2)$ and $m_1 = l(a, b_1)$. Let us suppose m occurs before m_1 in the left ordering of P.

By construction of the separator we know that both b_1 and b_2 are on the boundaries of the parent graph P (either on the right boundary or on the layer just below the sink. Therefore, the paths from a to b_1 and b_2 must cross each other at some point. Let the point of crossing be Z. Consider the two pieces of each path; let the sub path from a to z be p_1, and the one from z to b_1 be p_2. Similarly, let the two parts of the path from a to b_2 be q_1 and q_2. Without loss of generality assume that the weight of p_1 is less than that of q_1. We now have a contradiction because the path composed of p_1 and q_2 has weight less than the one made up of q_1 and q_2.

A similar lemma can be proved about paths from two source vertices to a single sink vertex. We now show how to use Lemma 4 to do the patching up across boundary Z with $(n_s n_t + n_s n_z)$ processors. For every vertex $a \in S$ we use $(n_t + n_z)$ processors to determine simultaneously all the shortest paths P_b^a from a to each $b \in T$.

Let us denote the problem of patching across the separator, for a given source vertex a, as $P(a, Z, T)$ where we are given the shortest paths a to every vertex in Z and between all pairs of vertices (z, b) where z is in Z and b is in T. Our goal is to determine the shortest paths from a to every vertex $b \in T$.

The algorithm to solve $P(a, Z, T)$ is as follows:

Step 1. Determine the vertex z_m such that $z_m = l(a, b_m)$.

Step 2. If $n_z = 1$ use nt processors to determine for all $a \in T$ the shortest path from a to b. Otherwise go to Step 3.

Step 3. Divide T into two sets T_1 and T_2 such that $T_1 = \{b \in T \mid b < b_m\}$ and $T_2 - T_1$. Similarly divide Z into two corresponding sets Z_1 and Z_2 such that

$$Z_1 = \{z \in Z \mid z_m\} \text{ and } Z_2 = Z - Z_1.$$

Step 4. Recursively solve the two subproblems $P(a, Z_1, T_1)$ and $P(a, Z_2, T_2)$.

Step 5. For all vertices $b \in T_1$ compare the shortest path from a to b through Z_1, with the one through z_m, and pick the smaller one.

Theorem 4 Starting with an n-vertex planar layered st-graph, it is possible at any stage of the recursion to obtain the shortest-path results of the parent graph P from those of its children p_1 and p_2 by running S copies of the algorithm outlined above.

Moreover, this can be done in time $O(\log^2 n)$ in CREW PRAM, using $\left(\frac{n_s n_t + n_s n_z}{\log n}\right) + \left(\frac{n_p}{\log n}\right)$ processors.

Proof Steps 1 and 2 are obviously correct. To prove the correctness of Step 4 we make use of Lemma 4 that for any $b \in T_2$ the shortest path from a to b does not use any vertex in Z_1, and for all $b \in T_1$ the shortest path from a to b either passes through z_1 or through z_m. We can therefore recursively divide the problem into two sub-problems $P(a, Z_2, T_2)$ and do the correction in Step 6 to determine the shortest paths to vertices in T_1.

To determine the processor and time complexity we note that the left ordering of any planar st-graph with n-vertices can be determine in $O(\log n)$ time with $n/\log n$ processors. Thus we can determine the orderings of S and T in $O(\log n_p + \log n)$ time with $\sqrt{\frac{n_p}{\log n}}$ processors. The time complexity of the various steps are as follows:

Step 1 can be performed in $O(\log n_t + \log n)$ time with $\sqrt{\frac{n_t}{\log n}}$ processors in a EREW PRAM.

Step 2 can be performed in time $O(\log n_z+\log n)$ time with $\sqrt{\frac{n_z}{\log n}}$ processors in an EREW-PRAM.

Step 3 takes $O(\log n)$ time with $n_t/\log n$ processors.

Step 4 can be done in $O(\log n_t+\log n_z+\log n)$ time with $\left(\frac{(n_t+n_z)}{\log n}\right)$ processors.

The depth of recursion is $O(\log n_t)$, since the size of the sink set T goes down by a factor of 2 every time Step 4 is executed. Therefore, the total time taken for the patchup is

$$O(\log n_p+\log^2 n_t+\log n_z+\log n)=O(\log^2 n). \quad (9)$$

Step 6 can be done in $O(\log n)$ time with $\left(\frac{n_t}{\log n}\right)$ processors in a CREW PRAM.

The processors complexity of the recursive sub problems are solved in parallel partitioning the processors between them. Sub problem $P(a, Z_1, T_1)$ needs $\left(\frac{|z_1|+|T_1|}{\log n}\right)$ processors while sub problem $P(a, Z_2, T_2)$ needs $\left(\frac{|z_2|+|T_1|}{\log n}\right)$ processors. The total number of processors needed is $n_s\left(\frac{n_t+n_z}{\log n}\right)+\left(\frac{n_p}{\log n}\right)$.

Theorem 5 Given an n-vertex planar layers st-graph G with none negative edge weights, we can determine the shortest path from s to t in time $O(\log^3 n)$ with $n/\log n$ processors in a CREW PRAM.

Proof The proof immediately follows from Lemma 2 and Theorems 3 and 4.

THE TUBE MINIMA PROBLEM AND A MORE EFFICIENT SOLUTION

Consider that at any stage of the computation the array $\boldsymbol{A}$, where $\boldsymbol{A}[i, k, j]$ is the value of the shortest path from vertex $i\in S$ to vertex $j\in T$ is constrained to pass through the vertex $k\in Z$. Let $\Theta_A(i, j)$ be the smallest index that minimizes $\boldsymbol{A}[i, k, j]$.

We say that $\boldsymbol{A}$ is totally monotone if Θ_A satisfies the following properties:

$$\Theta_A(i, j)\le\Theta_A(i+1, j), \quad (10)$$

$$\Theta_A(i, j)\le\Theta_A(i, j+1), \quad (11)$$

Furthermore $\Theta_{A'}$ for every sub matrix $\boldsymbol{A}'$ of $\boldsymbol{A}$ should also satisfy those properties. It can be shown that at every stage of the computation the matrix $\boldsymbol{A}$ constructed as above is totally monotone. It is not hard to see that to patch up two sub problems p_1 and p_2, we only need to determine the value of $\Theta_A(i, j)$ for all i, j. The problem of determining the value of Θ was formalized as the tube minima problem and has many applications in computational geometry (Aggarwal and Park, 1988). They gave the following bounds for calculating the tube minima of an $\sqrt{n}\times\sqrt{n}\times\sqrt{n}$ array:

Theorem 6 The tube minima of an $\sqrt{n}\times\sqrt{n}\times\sqrt{n}$ array can be found in time $O(\log n)$ and work $O(n)$ in a CREW PRAM.

Similarly for the CRCW model of computation Atallah (1990) gave an algorithm with the following bounds:

Theorem 7 The tube minima of an $\sqrt{n}\times\sqrt{n}\times\sqrt{n}$ array can be found in time $O(\log\log n)$ and work $O(n)$ in a CRCW PRAM.

We can use those algorithms in the patchup stage to get improved algorithms. To do that we need to make a slight change to our recursive decomposition. In our current decomposition tree we only worry about the total number of source-sink pairs at any level, and not about the number of boundary vertices in any one particular region. Therefore, our division does not guarantee that the number of recursive sub problems in each pieces is small. It is possible to do the division so that the number of recursive sub problems in each pieces is small. It is possible to do the division so that the number of boundary vertices in any piece p (at any level) is $O(\sqrt{n_p})$, where the number of vertices in p

is n_p. This can be accomplished by using the following idea due to Fredericckson (1987): Mark the boundary vertices and use the separator algorithm from Theorem 3 once to divide the marked vertices (into small pieces), at every stage. Our separator algorithm can be easily modified to divide a collection of marked vertices. This change in our division strategy guarantees that at any stage each piece has a small number of boundary vertices. In particular, we have the following lemma, which follows from the arguments of Fredericckson (1987).

Lemma 5 Let G be an n-vertex planar layered st-graph, and let $0 \le \varepsilon \le 1$. Using the separator X defined in Theorem 2 we can divide the graph into $\Theta\left(\frac{n}{r}\right)$ regions, such that each region has at most r vertices, and there are only $O(\frac{n}{\sqrt{r}})$ boundary vertices in all. Furthermore, none of the regions contain more than $O(\sqrt{r})$ boundary vertices.

Using this division strategy the patchup problem for any parent piece P at any stage can be solved as a tube-minima problem on an $O(\sqrt{n_p} \times \sqrt{n_p} \times \sqrt{n_p})$ array. We therefore get the following bounds for the shortest-path problem:

Theorem 8 Given an n-vertex planar layered st-graph G with non-negative edge-weights, we can determine the shortest path from s to t in time $O(\log n)$ with $n/\log n$ processors in a CRCW PRAM.

CONCLUSIONS

We have given a linear processor CREW algorithm for determining the shortest paths in a planar layered digraph which runs in $O(\log^3 n)$. Our motivation for considering layered graphs was to see if they could be used to solve the shortest-path problem for planar st-graphs; however, since they extend the class of grid digraphs they may have other applications. As far as planar st-graphs are concerned, it seems unlikely that a similar approach would give a linear processor solution.

As evidence we present the graph that cannot be decomposed into smaller pieces (that are a constant fraction smaller in weight) by using one way separators of size $O(\sqrt{n})$. Klein and Subramanian (1993) to some extent resolved this problem by presenting a linear processor poly-log time algorithm for finding single source shortest-path problem in planar digraphs. However their algorithm had a poly-logarithmic running time with a large exponent.

It may be possible to use a combination of the ideas presented in this paper along with the one of Klein and Subramanian (1993) to get a linear-processor algorithm that has a better running time.

ACKNOWLEDGEMENTS

The author thanks an anonymous referee for several helpful comments and questions. The author is also thankful to Prof. H.C. Pande, Vice-Chancellor Emeritus, Prof. S.K. Mukhrerjee, Vice-Chancellor and Prof. N.C. Mahnati, Professor & Head of Dept. of Applied Mathematics, Birla Institute of Technology, Mesra for encouragement and support.

References

Aggarwal, A., Park, J., 1988. Notes on Searching Dimensional Monotone Arrays. Proc. 29th Annual IEEE Symposium on Foundations of Computer Science, p.496-512.

Alon, N., Galil, Z., 1991. On the Exponent of the All Pairs Shortest Path Problem. Proc. 32th Annual IEEE Symposium on Foundations of Computer Sciences, **32**:569-575.

Apostolico, A., Atallah, M.J., Larmore, L., Mc Faddin, H.S., 1990. Efficient parallel algorithms for editing and related problems. *SIAM Journal on Computing*, **19**:968-988.

Atallah, M.J., 1990. A Faster Parallel Algorithm for A Matrix Searching Problem. Proc. 2th Scandinaviam Workshop on Algorithm Theory, p.193-200.

Bellman, R., 1958. On a routing problem. *Quarterly Journal of Applied Mathematics*, **16**:87-90.

Cohen, E., 1993. Efficient Parallel Shortest Paths in Digraphs with A Separator Decomposition. Proc. 5th Annual Symposium on Parallel Algorithms and Architectures, p.57-67.

Coremen, T.H., 1990. Introduction to Algorithms. MIT Press, Cambridge, MA.

Delche, A.L., Kosarju, S.R., 1992. An N.C. Algorithm for Evaluating Monotone Planar Circuits.

Di Battisa, G., Nardelli, E., 1987. An Algorithm for Testing Planarity of Hierarchical Graphs. Proc. Workshop WG 86, Bernierd, p.277-289.

Dijkastra, E.W., 1995. A note on two problems in connection with graphs. *Numer. Math.*, **1**:269-271.

Fredericckson, G., 1987. Fast algorithms for shortest paths in planar graphs with Applications. *SIAM J. Comput.*, **16**:1004-1022.

Fredman, M.L., Tarjan, R.E. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, **34**(3):596-615.

Garit, H., Miller, G.L., 1987. A Parallel Algorithm for finding Separator in Planar Graphs. Proc. 28th Annual IEEE Symposium on Foundation of Computer Science, p.238-248.

Generich, H.J., Lautenbach, K., 1973. Synchronisations graphics. *Acta Inform*, **2**:513-520.

Gondram, M., Minoux, M., 1984. In Graphs and Algorithms. Willey Interscience, New York, p.143-161.

Jàjà, J., 2001. Introduction to Parallel Algorithms. Addision-wesley, Reading MA.

Klein, P.N., Subramanian, S., 1993. A Linear Processor Polylog-time Algorithm for Shortest Path in Planar Graphs. Proc. IEEE Symposium on Foundation of Computer Science, p.259-270.

Levcopoulos, C., Lingas, A., 1992. There are planar graphs almost as good as the complete graphs and almost as cheap as minimum spanning trees. *Algorithmica*, **8**(3): 251-256.

Lempel, A., Even, Cederbaum, S., 1986. An Algorithm for Planitry Testing of Graphs. Proc. Internet. Symposium, p.215-232.

Lipton, R.J., Tarjan, R.E., 1979. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, **36**:177-189.

Lipton, R.J., Tarjan, R.E., 1980. Application of a planar separator theorem. *SIAM J. Appl. Math.*, **9**:615-627.

Miller, G.L., Thurston, W., 1986. Separator in Two and Three Dimensions. Proc. 22nd Annual ACM Symposium on Theory of Computing, p.300-309.

Mishra, P.K., Sharma, C.K., 1997. A Computational Study of An Efficient Shortest Path Algorithms in C-programming Language. Proc. of 5th International Conference on Application of HPC in Engineering, Santiago de Compstela, Spain.

Mishra, P.K., Sharma, C.K., 1999. An Efficient Parallel Algorithm for Shortest Paths in Planar Layered Digraphs. Proc. of 65th Annual Conference of Indian Mathematical society, India, **2**:635-654.

Preparta, P.P., Shamons, M.I., 1985. Computational Geometry. Sringer Verlag, New-York.

Tamassia, R., Preparta, P.P., 1990. Dynamic maintenance of planar digraphs with applications. *Algorithmica*, **5**:509-527.

Welcome visiting our journal website: ***http://www.zju.edu.cn/jzus***
Welcome contributions & subscription from all over the world
The editor would welcome your view or comments on any item in the journal, or related matters
Please write to: Helen Zhang, Managing Editor of JZUS
E-mail: **jzus@zju.edu.cn** Tel/Fax: 86-571-87952276