数理解析研究所講究録
第871巻 1994年 161-167

Efficient Parallel Shortest Path Algorithms for Banded Matrices

Yijie Han* Yoshihide Igarashi†

韓 以捷 五十嵐 善英

Abstract. We present efficient parallel shortest path algorithms for an $n \times n$ banded matrix of bandwidth b. Our algorithm computes all pair shortest distances within the band in time $O(nb^2/p + I(b)\log b \log(n/b))$ on the PRAM using p processors, where $I(b)$ is $\log b$ on the *EREW PRAM*, loglog b on the *CRCW PRAM* and a constant on the randomized *CRCW PRAM*. It computes all pair shortest distances in time $O(n^2 b/p + I(b)\log b \log(n/b))$ using p processors.

1 Introduction

The best known sequential algorithm for the shortest path problem has time complexity slightly less than $O(n^3)$ [Fr]. It is known that all pair shortest paths in n-vertex directed graph can be computed in $O(n^3/p + I(n)\log n)$ time using p processors on *PRAM* [HPR], where $I(n)$ is the time for finding the minimum of n elements using n processors. $I(n)$ is $\log n$ on the *EREW PRAM* [FW][KR], $\log\log n$ on the *CRCW PRAM* [V] and a constant on the randamized *CRCW PRAM*. Seidel recently gave an algorithm for all pair shortest paths in unweighted graphs [S]. His algorithm can be implemented on a *PRAM* with time complexity $O(M(n)\log n/p + \log^2 n)$ using p processors, where $M(n)$ is the number of operations needed to multiply two $n \times n$ matrices (currently it is $n^{2.376}$ [CW]). Lingas studied path problems in planar graphs [L]. When a family of separators is available, the all shortest distances can be computed with substantial savings [PR].

In this paper we consider the problem of computing shortest paths for graphs whose underlying matrix is a banded matrix. The input is an $n \times n$ matrix A with bandwidth b, $\lfloor b/2 \rfloor$ diagonals on either side of the main diagonal. Each a_{ij} within the band of A is the weight of the arc from vertex i to vertex j. Entries outside the band are ∞'s. We consider two problems. One is the problem of computing all pair shortest distances within the band. The output is also a banded matrix B with bandwidth b, where each b_{ij} within the band of B represents the shortest distance from i to j. The other problem is the problem of computing all pair shortest distances. The output is a matrix giving all pair shortest distances.

*Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA.
†Department of Computer Science, Gunma University, Kiryu, 376 Japan.

Recently Allison *et al.* showed a fast sequential shortest path algorithm for banded matrices of bandwidth b [ADY]. Their algorithm computes all shortest distances within the band in $O(nb^2)$ time and computes all pair shortest distances in $O(n^2b)$ time. Unfortunately, the algorithm given by Allison *et al.* is highly sequential. Their algorithm is an incremental one which adds one vertex at a time to the graph. Therefore their algorithm requires $O(n)$ time when implemented on a *PRAM* even if the number of available processor is unlimited. Some of the known techniques for parallel shortest path algorithms [L][PR][S] cannot apply to our problem, while others [HPR][PK] do not yield savings in the number of operations when they are applied to a banded matrix.

In this paper we use a new technique to design shortest path algorithms for banded matrices. The technique allows us to smooth shortest paths in the process of contracting them. Our algorithms have time complexity $O(nb^2/p + I(b)\log b \log(n/b))$ for computing all pair shortest distances within the band, and time complexity $O(n^2b/p + I(b)\log b \log(n/b))$ for computing all pair shortest distances.

2 Path Contraction and Smoothing

The rows and columns of input matrix A are numbered from 1 to n. Let $k = \lceil b/2 \rceil$. Assume that n is a multiple of k. Let $A_{i,j}$, $1 \leq i,j \leq n/k$, be a $k \times k$ submatrix containing elements in rows $(i-1)k+1, \cdots, ik$ and columns $(j-1)k+1, \cdots, jk$ of A. We use $[i]$ to denote any vertex u such that $(i-1)k+1 \leq u \leq ik$. A path is a sequence of vertices, and it can be denoted by $[i_1][i_2]\cdots[i_j]$. A path $[i_1][i_2]\cdots[i_j]$ is a loop if $i_1 = i_j$. It is a simple loop anchored at i_1 if $i_t \neq i_1$, $2 \leq t < j$. The transitive closure of a matrix M is denoted by M^*. The transitive closure gives all pair shortest distances. For an $n \times n$ matrix M, the transitive closure gives all pair shortest distances. For an $n \times n$ matrix M, the transitive closure can be computed in time $O(n^3/p + I(n)\log n)$ using p processors [HPR].

Because the input matrix is a banded matrix, all entries outside the band are ∞'s. Therefore, a shortest path $[i_1][i_2]\cdots[i_j]$ has the property that $i_k - 1 \leq i_{k+1} \leq i_k + 1$.

The computation of shortest distances can be done by matrix multiplication over the semiring $(A, min, +)$, which may be viewed as a process of path contraction. For example, the shortest path from vertex $[i]$ to vertex $[i]$ may have the form $[i][i-1]\cdots[1][0][1]\cdots[i-1][i]$, the distance of this shortest path can be computed by matrix multiplication

$$A_{i,i-1}A_{i-1,i-2}\cdots A_{1,0}A_{0,1}\cdots A_{i-2,i-1}A_{i-1,i}.$$

This matrix multiplication can be viewed as path contraction. After $A_{i,i-1}A_{i-1,i-2}$ is computed, an arc $[i][i-2]$ labeled with the new weight obtained from the computation can be added to the input graph. Now the shortest path becomes $[i][i-2]\cdots[1][0][1]\cdots[i-1][i]$. The path is contracted and its length decremented by 1.

A common approach used in parallel computation is to put these matrices at the leaves of a binary tree and the matrix multiplications proceed as dictated by the tree. When we reach the root of the binary tree, the product of the matrices is obtained. The number of

steps needed is the height of the tree which is logarithmic in the number of leaves. This approach does not work for banded matrices because when matrices are multiplied together we are essentially filling the entries outside the band of the matrix. In order to take care of all possible shortest paths, we have fill all entries of the matrix, thus resulting in an $\Omega(n^2)$ time algorithm.

We use the following smoothing technique which is highly parallel. To smooth a loop of the form $[i]\cdots[i]$ with amplitude $d = 2^a - 1$ we first recursively smooth loops of the forms $[i-2^{a-1}]\cdots[i-2^{a-1}], [i]\cdots[i], [i+2^{a-1}]\cdots[i+2^{a-1}]$ with amplitude $2^{a-1}-1$, and then use matrix multiplications to finish smoothing. The next procedure smoothes a loop of amplitude $d = 2^a - 1$.

Procedure Smooth(i, a)
(* Smooth loops of the form $[i][i_1][i_2]\cdots[i]$ with amplitude $2^a - 1$. *)
begin
if $a = 0$ **then** $A_{i,i} := A^*_{i,i}$; (* Contract $[i][i]\cdots[i]$ to $[i][i]$. *)
else if $a = 1$ **then begin**
$A_{i,i} := min\{A_{i,i}, A_{i,i+1}A^*_{i+1,i+1}A_{i+1,i}, A_{i,i-1}A^*_{i-1,i-1}A_{i-1,i}\}$;
$A_{i,i} := A^*_{i,i}$
end
else begin
Smooth$(i-2^{a-1}, a-1)$;
Smooth$(i, a-1)$;
Smooth$(i+2^{a-1}, a-1)$;
$A_{i,i} := \min\ \{\ A_{i,i},$
$(\prod_{j=i}^{i+2^{a-1}-1}(A_{j,j+1}A^*_{j+1,j+1}))(\prod_{j=i+2^{a-1}}^{i+1}(A_{j,j-1}A^*_{j-1,j-1}))$,
$(\prod_{j=i}^{i-2^{a-1}+1}(A_{j,j-1}A^*_{j-1,j-1}))(\prod_{j=i-2^{a-1}}^{i-1}(A_{j,j+1}A^*_{j+1,j+1}))\ \}$;
$A_{i,i} := A^*_{i,i}$
end
end

Theorem 1 *Procedure Smooth smooths a loop* $[i][i_1]\cdots[i]$ *with amplitude* $2^a - 1$.

3 Computing Shortest Paths

Assume that $n/k = 2^a - 1$ for an integer a. There are a total of $2^a - 1$ submatrices $A_{i,i}$ $(1 \le i \le 2^a - 1)$ on the main diagonal of the input matrix A. The idear of path smoothing can be used for computing the shortest distance. If the three recursive calls in procedure Smooth are executed in parallel we obtain a parallel algorithm for computing shortest distances of vertices $([i],[i])$. For the input matrix A we may execute Smooth$(2^{a-1}, a-1)$, which returns

shortest distances for pairs of vertices of the form $([2^{a-1}],[2^{a-1}])$. In order to compute all shortest distances for all entries within the band, we use two stages.

First stage: For each i, let j be the largest integer such that $i/2^j$ is odd. Smooth loops of the form $[i]\cdots[i]$ with amplitude 2^j-1.

Second stage: Smooth all loops and compute all pair shortest distances within the band.

The first stage can be accomplished by the principal of procedure Smooth. We now a bottom-up description of the algorithm.

Procedure Band-1.1
  for all i, $1\le i\le 2^a-1$, **do** in parallel
    $A_{i,i} := A^*_{i,i}$
  for $t := 1$ to $a-1$ **do begin** $(*\ 2^a-1=n/k.\ *)$
    for all i, $1\le i\le 2^a-1$, $i \bmod 2^t = 0$, **do** in parallel **begin**
      $A_{i,i} := \min\{\ A_{i,i},$
        $(\prod_{j=i}^{i+2^{a-1}-1}(A_{j,j+1}A_{j+1,j+1}))(\prod_{j=i+2^{a-1}}^{i+1}(A_{j,j-1}A_{j-1,j-1}))$,
        $(\prod_{j=i}^{i-2^{a-1}+1}(A_{j,j-1}A_{j-1,j-1}))(\prod_{j=i-2^{a-1}}^{i-1}(A_{j,j+1}A_{j+1,j+1}))\ \}$;
      $A_{i,i} := A^*_{i,i}$
    end
  end

In the h-th iteration of the loop indexed by t in Band-1.1, if 2^h divides i, then loops of the form $[i]\cdots[i]$ with amplitude 2^h-1 are smoothed. Therefore, procedure Band-1.1 accomplished the tasks of the first stage.

Theorem 2 *Procedure Band-1.1 smoothes loops of the form* $[i]\cdots[i]$ *with amplitude* 2^h-1, *where* h *is the largest integer such that* $i/2^h$ *is odd.*

Proof: By induction on h. Before the execution of the loop indexed by t in procedure Band-1.1, instruction $A_{i,i} := A^*_{i,i}$ is executed. Therefore, loops of the form $[i]\cdots[i]$ with amplitude 0 have been contracted to a single arc. Assume that after the h-th iteration, loops of the form $[i]\cdots[i]$ with amplitude 2^h-1, where 2^h divides i, have been smoothed. Consider the case of $h+1$. After the h-th iteration, a simple loop of the form $[i]\cdots[i]$ with amplitude $2^{h+1}-1$, where 2^{h+1} divides i, has been contracted to a loop of the forms $[i][i+1][\delta_{i+1}][i+2][\delta_{i+2}]\cdots[i+2^{j-1}]\cdots[i+1][\delta_{i+1}][i]$ and $[i][i-1][\delta_{i-1}][i-2][\delta_{i-2}]\cdots[i-2^{j-1}]\cdots[i-1][\delta_{i-1}][i]$ by the principal of path smoothing, where δ_s is either s or empty ϵ. Instruction

$$A_{i,i} := \min\{\ A_{i,i},\ (\prod_{j=i}^{i+2^{a-1}-1}(A_{j,j+1}A^*_{j+1,j+1}))(\prod_{j=i+2^{a-1}}^{i+1}(A_{j,j-1}A^*_{j-1,j-1})),\ (\prod_{j=i}^{i-2^{a-1}+1}(A_{j,j-1}A^*_{j-1,j-1}))(\prod_{j=i-2^{a-1}}^{i-1}(A_{j,j+1}A^*_{j+1,j+1}))\ \}$$

in the $h+1$-st iteration smoothes such a simple loop. And then instruction $A_{i,i} := A^*_{i,i}$ smoothes non-simple loops. □

Theorem 3 *The time complexity of Band-1.1 is*

$$O(\frac{nb^2\log\frac{n}{b}}{p} + I(b)\log n\log\frac{n}{b}).$$

Proof: $O(\log(n/b))$ iterations of the loop indexed by t in Band-1.1 are executed. In the h-th iteration, $\lfloor n/(2^h k) \rfloor$ parallel matrix products and matrix transitive closures are computed, where each product contains $O(2^h)$ matrix multiplications. Each matrix multiplication takes $O(b^3/p + I(b))$ time, and each matrix transitive closure takes $O(b^3/p + I(b)\log b)$ time. Therefore, the h-th iteration takes $O((nb^2)/p + I(b)\log b + I(b)h)$ time. The time complexity of the algorithm is

$$O(\frac{nb^2 \log \frac{n}{b}}{p} + I(b)\log b \log \frac{n}{b} + I(b)\log^2 \frac{n}{b}) = O(\frac{nb^2 \log \frac{n}{b}}{p} + I(b)\log n \log \frac{n}{b}).$$

□

The computation of matrix products in procedure Band-1.1 can be improved. We note that after the first iteration of computing the matrix transitive closure is executed, matrices $A_{i,i}$, $i \bmod 2 \neq 0$, are fixed and will be not modified in the remaining execution of Band-1.1. After the first iteration of the loop indexed by t, matrices $A_{i,i}$, $i \bmod 4 \neq 0$, are fixed and will not be modified, and so on. The modified procedure Band-1.2 is as follows:

Procedure Band-1.2
  for all i, $1 \leq i \leq 2^a - 1$, **do** in parallel
    $A_{i,i} := A^*_{i,i}$;
  for $t := 1$ to $a - 1$ **do begin** $(* \ 2^a - 1 = n/k. \ *)$
    for all i, $1 \leq i \leq 2^a - 1$, $i \bmod 2^t = 0$, **do** in parallel **begin**
      if $t \neq 1$ **then begin**
        $A_{i,i+2^{t-1}} := A_{i,i+2^{t-2}} A_{i+2^{t-2},i+2^{t-2}} A_{i+2^{t-2},i+2^{t-1}}$;
        $A_{i+2^{t-1},i} := A_{i+2^{t-1},i+2^{t-2}} A_{i+2^{t-2},i+2^{t-2}} A_{i+2^{t-2},i}$;
        $A_{i,i-2^{t-1}} := A_{i,i-2^{t-2}} A_{i-2^{t-2},i-2^{t-2}} A_{i-2^{t-2},i-2^{t-1}}$;
        $A_{i-2^{t-1},i} := A_{i-2^{t-1},i-2^{t-2}} A_{i-2^{t-2},i-2^{t-2}} A_{i-2^{t-2},i}$
      end;
      $A_{i,i} := \min\{A_{i,i}, A_{i,i+2^{t-1}} A_{i+2^{t-1},i+2^{t-1}} A_{i+2^{t-1},i}, A_{i,i-2^{t-1}} A_{i-2^{t-1},i-2^{t-1}} A_{i-2^{t-1},i}\}$;
      $A_{i,i} := A^*_{i,i}$
    end
  end

Theorem 4 *The time complexity of Band-1.2 is*

$$O(\frac{nb^2}{p} + I(b)\log b \log \frac{n}{b}).$$

The second stage is to smooth loops $[i]\cdots[i]$ with any amplitude and to compute all pair shortest distances within the band. When the first stage finishes, loops of the form $[2^{a-1}]\cdots[2^{a-1}]$ with any amplitude are smoothed, where $2^a - 1 = n/k$. This results in two

graphs G_1, G_2 from the input graph, where G_1 contains vertices in $[1], [2], \cdots, [2^{a-1}]$ and G_2 contains vertices in $[2^{a-1}], \cdots, [2^a - 1]$. Each original loop of the form $[i] \cdots [i]$ is now a loop in one of the resulting graphs. Let us consider G_1. A simple loop of the form $[2^{a-2}] \cdots [2^{a-2}]$ should have already been smoothed in the first stage if it does not contain a vertex in $[2^{a-1}]$. If it does, then the loop is now in the form $[i][\delta_i][i+1][\delta_{i+1}] \cdots [2^{a-1}] \cdots [i][\delta_i]$, where $i = 2^{a-2}$ and δ_j is either j or ϵ. We can use matrix multiplication to smooth such a loop. After all loops of the form $[2^{a-2}] \cdots [2^{a-2}]$ are smoothed, G_1 is divided into two graphs, one contains vertices in $[1], [2], \cdots [2^{a-2}]$ and the other contains vertices in $[2^{a-2}], \cdots, [2^{a-1}]$. G_2 can be processed in a similar way. Thus we can continue the dividing process recursively.

After we smooth all loops, we have to compute shortest distances for all other entries within the band. The lower left triangle of $A_{i,i+1}$ is updated with $A_{i,i}A_{i,i+1}$ and upper right triangle of $A_{i,i-1}$ is updated with $A_{i,i}A_{i,i-1}$.

We now give procedure Band-2 for the second stage.

Procedure Band-2
  for $t := a - 2$ **downto** 0 **do begin** ($* \ 2^a - 1 = n/k. \ *$)
    for all i, $1 \leq i \leq 2^a - 1$, $i \bmod 2^t = 0$ and $i \bmod 2^{t+1} \neq 0$, **do** in parallel
      begin
        if $i \neq 2^a - 2^t$ **then** $A_{i,i} := \min\{A_{i,i}, A_{i,i+2^t}A_{i+2^t,i+2^t}A_{i+2^t,i}\}$;
        if $i \neq 2^t$ **then** $A_{i,i} := \min\{A_{i,i}, A_{i,i-2^t}A_{i-2^t,i-2^t}A_{i-2^t,i}\}$;
        $A_{i,i} := A^*_{i,i}$
      end
  end
  for all i, $1 \leq i \leq 2^a - 2$, **do** in parallel **begin** ($*$ fill all entries within the band. $*$)
    begin
      $A_{i,i+1} := A_{i,i}A_{i,i+1}$;
      $A_{i+1,i} := A_{i+1,i}A_{i,i}$
  end

Theorem 5 *Procedure Band-2 computes all pair shortest distances within the band in time*

$$O(\frac{nb^2}{p} + I(b) \log b \log \frac{n}{b}).$$

After all pair shortest distances are computed within the band, the all pair shortest distances outside the band can be computed progressively from the main diagonal toward the top right corner and the bottom left corner. For convenience we add n/k dummy vertices and number them from $n+1$ to $n + n/k$. Arcs incident with dummy vertices are labeled with weight ∞. There are now 2^a submatrices on the main diagonal. The algorithm for computing all pair shortest distances is given below.

```
Procedure Band-3
  for t := 1 to a − 1 do begin (* 2^a − 1 = n/k. *)
    for all i, 1 ≤ i ≤ 2^a, i mod 2^t = 0 and i mod 2^{t+1} ≠ 0, do in parallel
      begin
        for all j, k, i − 2^t + 1 ≤ j ≤ i, i + 1 ≤ k ≤ i + 2^t, do in parallel
          begin
            A_{j,k} := A_{j,i} A_{i,i+1} A_{i+1,k};
            A_{k,j} := A_{k,i+1} A_{i+1,i} A_{i,j}
          end
      end
  end
```

Theorem 6 *Procedure Band-3 computes all pair shortest distances, and its time complexity is* $O(n^2 b/p + I(b)\log(n/b))$.

References

[ADY] L. Allison, T. I. Dix and C. N. Yee. Shortest path and closure algorithms for banded matrices. Information Processing Letters 40 (1991), 317-322.

[CW] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. J. of Symbolic Computation 9 (1990), 251-280.

[FW] S. Fortune and J. Wyllie. Parallelism in random access machines. Proc. 10th ACM Symposium on the Theory of Computing, San Diego, California (1978), 114-118.

[Fr] M. L. Fredman. New bounds on the complexity of the shortest path problem. SIAM J. Comput. 5 (1976), 83-89.

[HPR] Y. Han, V. Pan and J. Reif. Efficient parallel algorithms for computing all pair shortest paths in directed graphs. In Proc. 4th ACM Symposium on Parallel Algorithms and Architectures, San Diego (1992), 353-362.

[KR] R. M. Karp and V. Ramachandran. A survey of parallel algorithms for shared memory machines. in "Handbook of Theoretical Computer Science", Amsterdam (1990), 869-941.

[L] A. Lingas. Efficient parallel algorithms for path problems in planar directed graphs. Proc. SIGAL'90, Lecture Notes in Computer Science, 450 (1990), 447-457.

[PR] V. Pan and J. H. Reif. Fast and efficient solution of path algebra problems. J. Computer and System Sciences 38 (1989), 494-510.

[S] R. Seidel. On the all-pair-shortest path problem. Proc. of the 24th ACM Symposium on the Theory of Computing, Victoria (1992), 745-749.