A constant time parallel algorithm for the triangularization of a sparse matrix using CD-PARBS

Technical Report B-00-05
March 1, 2000

Rajeev Wankar[1]
N.S.Chaudhari*
Elfriede Fehr

Freie Universität Berlin
Institut für Informatik
{wankar, fehr}@inf.fu-berlin.de
*School of Computer Science, DAVV Indore, India

Abstract

An algorithm for the triangularization of a matrix whose graph is a directed acyclic graph, popularly known as *dag*, is presented. One of the algorithms for obtaining this special form has been given by Sargent and Westerberg. Their approach is practically good but sequential in nature and cannot be parallelised easily. In this work we present a parallel algorithm which is based on the observation that, if we find the transitive closure matrix of a directed acyclic graph, count the number of entries in each row, sort them in the ascending order of their values and rank them accordingly, we get a lower triangular matrix. We show that all these operations can be done using 3-d CD-PARBS(Complete Directed PARBS) in constant time. The same approach can be used for the block cases, producing the same relabelling as produced by Tarjan's algorithm, in constant time. To the best of our knowledge, it is the first approach to solve such problems using directed PARBS.

Keywords: PARBS, CD-PARBS, dag.

Introduction: Sparse matrix is one in which majority of the coefficients are zero. The sparse matrix solution is the core of many scientific and engineering problems, as many of the real problems include finding solution, which are sparse. Researchers have extensively studied the sparse matrix solution and presented many efficient algorithms. The process of obtaining the solution of a sparse linear system $Ax = b$, where A is a $n \times n$ sparse matrix, consists of four phases: ordering, symbolic factorization, numerical factorization and solving triangular systems.

---

1. This work is supported by the German Academic Exchange Services (DAAD) under the "Sandwich Model" fellowship with the author who is permanently working in the Department of Computer Science, North Maharashtra University, Jalgaon (MS), India.

There are many sparse matrix forms which are used for ordering strategy, i.e. triangular, diagonal, random, banded etc.. The advantage of using block triangular form is that the set of equations $Ax = b$ may be solved by the simple forward elimination and back substitution processes thus lot of computational saving. There are two approaches for permuting sparse matrices into the block triangular form- traversal technique and symmetric permutation technique. Based on the symmetric permutation technique, one of the algorithms for ordering the sparse matrix into triangular form was proposed by Sargent and Westerberg [DER86]. Their algorithm is based on the observation that, if A is a symmetric permutation of a triangular matrix then there must be a node in its digraph from where no path leaves. This node is ordered first in the relabelled digraph. Eliminating this node and all edges pointing to it leaves the remaining sub graph which again has a node from which no path leaves, continuing this way we get a lower triangular form. This is a quite straight forward idea but sequential in nature. Sargent and Westerberg then generalized the idea to the block case. They used the concept of the composite nodes, a group of nodes through which a closed path can be found and proposed the algorithm which required $O(n^2)$ relabelling. Tarjan [DER86, Tar72] followed the same basic idea as proposed by Sargent and Westerberg and eliminated all such strong components with the use of the stack.

In this work we deal with the matrix whose graph is a directed acyclic graph. A directed acyclic graph is one which does not contain a circuit. It is evident that it has no self loops also. Our algorithm is based on the known concept that, “a directed acyclic graph has at least one node whose in degree is zero and one node whose out degree is zero”. If we find the transitive closure of such a matrix, we find at least a row whose all the entries are zero. We rank this node (row number) as 1 and find the next row with ascending order of entries, rank it next and so on. One important point to mention is that, if sorting is done in the descending order and nodes are ranked accordingly, we get the upper triangular matrix. The algorithm proposed in the next section is not only applicable for the *dag,* but can be used for the block case and produces the same relabeling as the Tarjan’s algorithm. Our algorithm essentially consist of the following steps.

1. Find the transitive closure matrix A^+ of the *dag*.
2. Perform the sort on A^+, in the ascending order of the number of entries in the rows.

3. Relabel the digraph (perform row/column interchange).

The model of Computation: Ever since the introduction of the meshes with reconfigurable buses, the architecture gained a lot of popularity amongst the researchers and scientists for its high performance computing with general purpose processors used. It is a powerful model of computation which was proposed in the conference on advanced research in VLSI technology in March 1988 by Miller, Kumar *et al.* [MKRS88]. They described their architecture as a VLSI array of processors overlaid with a reconfigurable bus system. Though the different organizations of processors have been proposed in the past such as pyramid computer, mesh of tree and meshes with broadcast buses [Bok84], these organizations were static in nature and communication pattern between the processors could not be changed during the execution of the algorithm. The model proposed by Miller, Kumar *et al.* offers dynamic reconfigurability during the execution and allows to make different configurations to fulfil the different computational needs. In their paper they described algorithms with better parallel time complexities than the existing best ones. They have shown how reconfigurable meshes can act as a universal chip, simulating other VLSI organizations with equivalent chip area without a loss of time. They have also shown how reconfigurable mesh can be used to simulate certain fundamental techniques that have been developed for P-RAM and W-RAM model of computations.

Many problems in science and engineering can be formulated in terms of directed and undirected graphs. Designing a parallel graph algorithm is both theoretically and practically important. Many researchers are extensively engaged in finding graph formulations and parallelisation of important problems. Wang and Chen [WC90] proposed a constant time algorithm for computing transitive closure of an undirected graph. They designed two algorithms, one on a $n \times n \times n$ PARBS, and other on a 2-D $n^2 \times n^2$ PARBS. In their work, they presented constant time parallel algorithms for many problems including recognizing bipartite graph, finding connected components, articulation point, bi-connected components, bridges and minimum spanning tree, for undirected graphs.

Since PARBS cannot control the direction of the signal flow, for many directed graph problems it not possible to find correct connections on PARBS. Hence a new modified model called D-

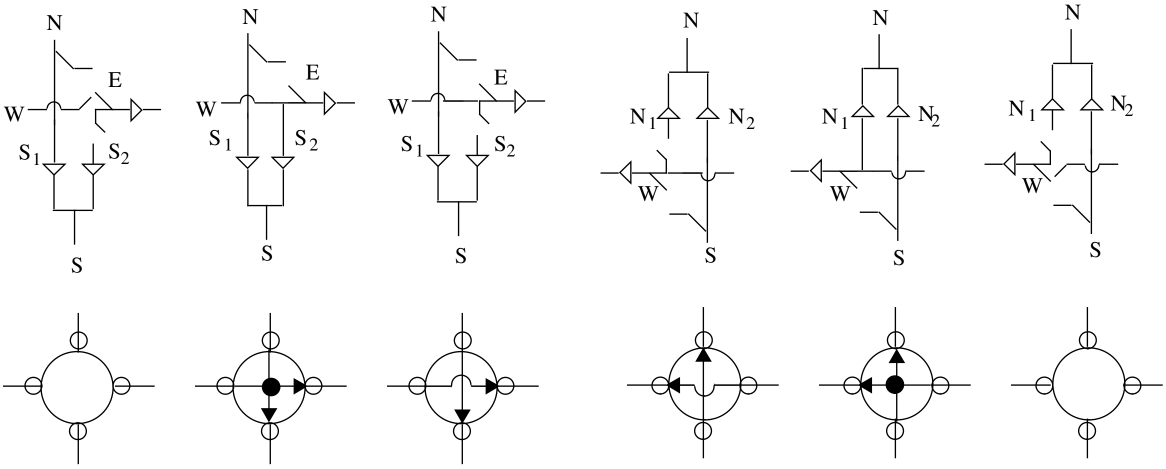
PARBS (directed PARBS) has been proposed to solve these directed graph problems by Kuo, Hsu *et al.* [KHF99]. They proposed constant time algorithms with $O(N^3)$ CD-PARBS to solve topological sort, transitive closure, cyclic graph checking and strongly connected problem, on directed graphs.

In order to get the correct direction control in D-PARBS, the vertices in the 2-d D-PARBS are numbered in a slightly different way. The lower/upper triangle of D-PARBS is obtained by rotating upper/lower D-PARBS by 180 degree. Figure 2 shows that a Complete Directed PARBS (CD-PARBS) can be obtained by combining these two to get a $(n+1) \times n$ D-PARBS. On a 2-d CD-PARBS, a matrix $A_{i,j}$, for $i \le j$, is stored in processors $P_{i,j}$ of the upper triangular part, and $A_{i,j}$, for $i \ge j$, is stored in processors P_{i^*, j^*} of the lower triangular part. Processors on 2-d CD-PARBS are connected according to the following basic connection rules (Figure 1 a & b).

1. On the upper triangle of the CD-PARBS, the diagonal processor $P_{i,i}$, for $1 \le i \le n$, establishes connection $\langle N, E, S_1 \rangle$, called *fork connection*. Processor $P_{i,j}$, for $1 \le i < j \le n$, establishes connection $\{ \langle N, S_1 \rangle, \langle W, E, S_2 \rangle \}$ if $A_{i,j} = 1$, called *join connection*, otherwise $P_{i,j}$ establishes $\{ \langle N, S_1 \rangle, \langle W, E \rangle \}$, called *cross connection*.

2. On the lower triangle of the CD-PARBS, the diagonal processor P_{i^*, i^*}, for $1 \le i \le n$, establishes connection $\langle S, W, N_1 \rangle$, called *fork connection*. Processor P_{i^*, j^*}, for $1 \le j^* < i^* \le n$, establishes connection $\{ \langle S, N_1 \rangle, \langle E, W, N_2 \rangle \}$ if $A_{i^*, j^*} = 1$, called *join connection*, otherwise P_{i^*, j^*} establishes $\{ \langle S, N_1 \rangle, \langle E, W \rangle \}$, called *cross connection*.

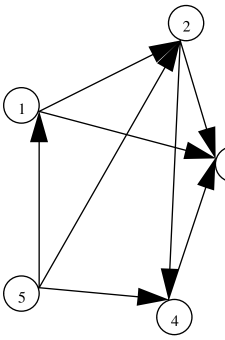
Three dimensional CD-PARBS can be constructed similarly by adding U(upper) and D(down) ports. Only two kinds of connections are allowed in a 3-d CD-PARBS, {U, D} connected or disconnected, which is used to send data to the other planes.



(a) Upper triangular D-PARBS

(b) Lower triangular D-PARBS

Figure 1 Processors and switches in D-PARBS

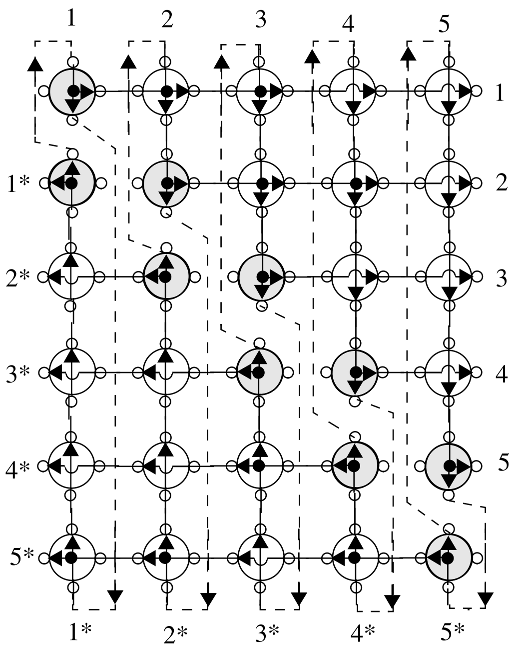


(a) Directed acyclic graph G

A=

1	0	1	1	0	0
2	0	0	1	1	0
3	0	0	0	0	0
4	0	0	1	0	0
5	1	1	0	1	0

(b) Adjacency matrix A for G



(c) Corresponding 6 x 5 CD-PARBS for G

Figure 2 Embedding the directed acyclic graph in to a 6 x 5 CD-PARBS

Algorithm for triangularization:

The algorithm for triangularization of a sparse matrix can be built using these theorems:

Theorem 1: The transitive closure matrix of a directed graph with n vertices can be computed in O(1) time on a 3-d $n \times (n+1) \times n$ CD-PARBS [KHF99].

Theorem 2: The topological sort of a directed graph with n vertices can be solved in O(1) time on a 3-d $n \times (n+1) \times n$ CD-PARBS [KHF99].

The transitive closure matrix $A^+$of a directed graph G is a matrix where $A^+_{i,j} = 0$, if there is a path of length > 0 from i to j, otherwise $A^+_{i,j} = 1$. In this work we will reproduce the portion of the algorithms for the transitive closure matrix and modify topological sort algorithm, given in [KHF99], according to our need. Due to the directional flow of the CD-PARBS, the architecture proposed by Kuo, Hsu and Fang does not support the operation like column interchange. They proposed this architecture to perform some directed graph problems. We here propose few modifications in to the architecture so that the special column interchange operations can be performed and it does not destroy the directional flow of the original model. Initially, we assume that the adjacency matrix A are stored in a $(n+1) \times n$ CD-PARBS. The entry of a boolean matrix A is set to 0, if sparse matrix $S_{i,j} = 0$ or 1, otherwise.

.

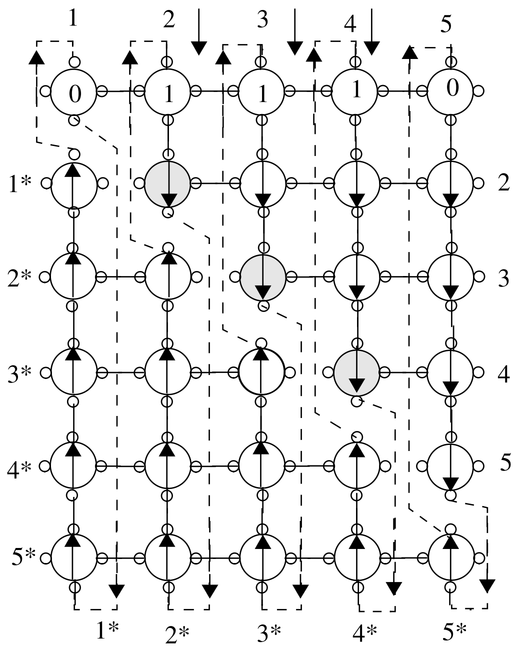


Figure 3 The first row of the transitive closure matrix A^+ on the first plane

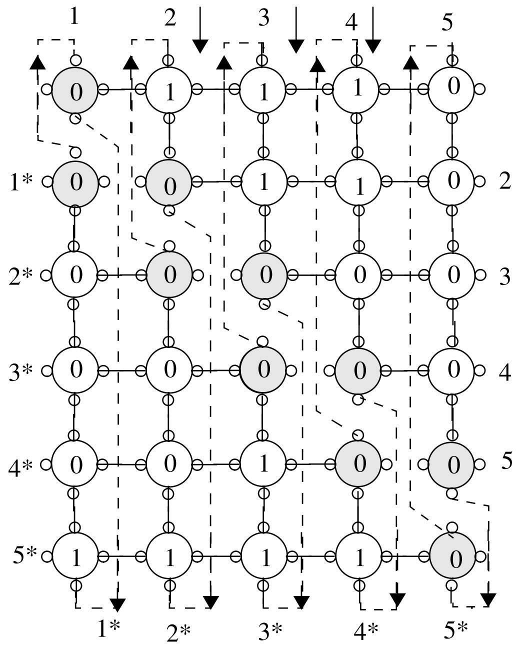


Figure 4 Each plane sends it transive closure value to the first plane

Algorithm:

I. Obtain transitive closure matrix of a sparse matrix whose graph is a *dag*.

Step 1: Input the adjacency matrix A into the first 2-d CD-PARBS plane.

Step 2: Transmit *A* from the first plane to other 2-d CD-PARBS planes.

Step 3: for all k CD-PARBS planes, $1 \le k \le n$, **in parallel do**

begin

3.1: Construct CD-PARBS planes according to the basic connection rules (Figure 2).

3.2: $P_{k,k,k}$ sends signal to its east and south port.

3.3: Every processor on the upper triangle constructs $\langle N, S_1 \rangle$ and every processor on the lower triangle constructs $\langle S, N_1 \rangle$ (Figure 3).

3.4: If the diagonal processor on the upper triangle receives the signal, sent by $P_{k,k,k}$, then it sends signal to S port (Figure 3).

3.5: If the processor $P_{k, j, k}$ on the kth row on the upper triangle receives the signal then sets $A^{+}{}_{k,j} = 1$, otherwise $A^{+}{}_{k,j} = 0$ (Figure 3).

3.6: If the processor $P_{k*, j*, k}$ on the kth row of the lower triangle receives the signal then sets $A^{+}{}_{k*,j*} = 1$, otherwise $A^{+}{}_{k*,j*} = 0$ (Figure 3).

end.

Step 4: Each 2-d CD-PARBS sends A to the first plane (Figure 4).

II. Obtain the number of entries in each row of the transitive closure matrix and rank them in the ascending order of their values.

Once the transitive closure matrix has been obtained and available at the first plane of CD-PARBS, we count number of entries in each row and sort them in ascending order of their values. Here we make few changes in the topological sort algorithm proposed in [KHF99].

Each processor establishes the $\langle U, D\rangle$ connection, processor $P_{i, j, 1}$ if $i \le j$ and $P_{i*, j*, 1}$ if $i \ge j$, on the first plane, transmits the values of A^{+} to the other planes. Next, except the kth row on the kth plane, all processors make cross connection and diagonal processors make fork connection. Processors on the kth row and kth plane set the join or cross connections depending on the value of A^{+}.

The algorithm for sorting and labeling consists of two steps: Counting and Ranking.

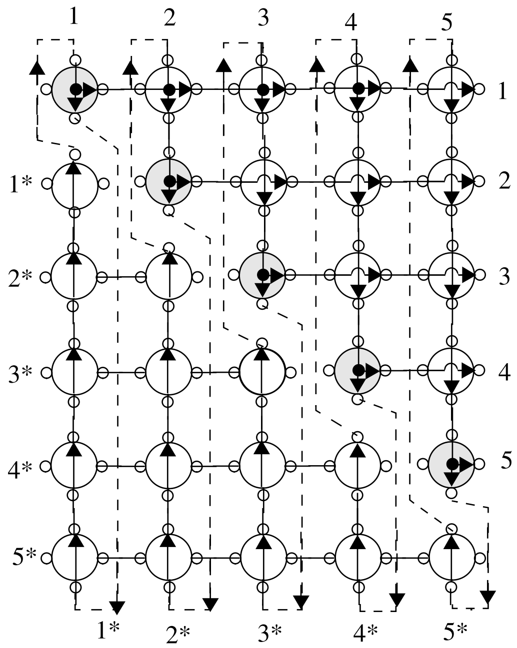


Figure 5 For the first plane $P_{2,2}$, $P_{3,3}$, $P_{4,4}$ sends signal to S port

Counting Phase: In this phase the number of entries of a particular row are counted. This phase of the algorithm is similar to one proposed in [KHF99].

Step 1: $P_{k,k,k}$ sends signal to E and S ports (Figure 2 (c)).

Step 2: Every processor on the lower triangle constructs $\langle S, N_1 \rangle$ (Figure 5).

Step 3: If the diagonal processors on the upper triangle have received signal sent from $P_{k,k,k}$ then send signal to S port (Figure 5).

Step 4: $P_{n^*,n^*,k}$ constructs $\langle E, W \rangle$ connection. Except $P_{n^*,n^*,k}$, if the processor on the lower triangle receives signal then constructs $\{\langle E, N_2 \rangle, \langle S, W \rangle\}$, otherwise construct $\langle E, W \rangle$ (Figure 6).

Step 5: If the last diagonal processor on the lower triangle $P_{n^*,n^*,k}$, has received signal then it sends 1 to W port, otherwise, sends 0 to W port (Figure 6).

Step 6: If the processor in the first column of the lower triangle, $P_{i^*,1^*,k}$, receives a value then compute the *count* = n - i + value (Sent in the step 5) (Figure 6).

Step 7: $P_{i*,\,1*,\,k}$ sends *count* to $P_{k,\,k,\,k}$ (Figure 7).

Step 8: $P_{k,\,k,\,k}$ sends *count* to the other planes along the U and D directions (Figure 8).

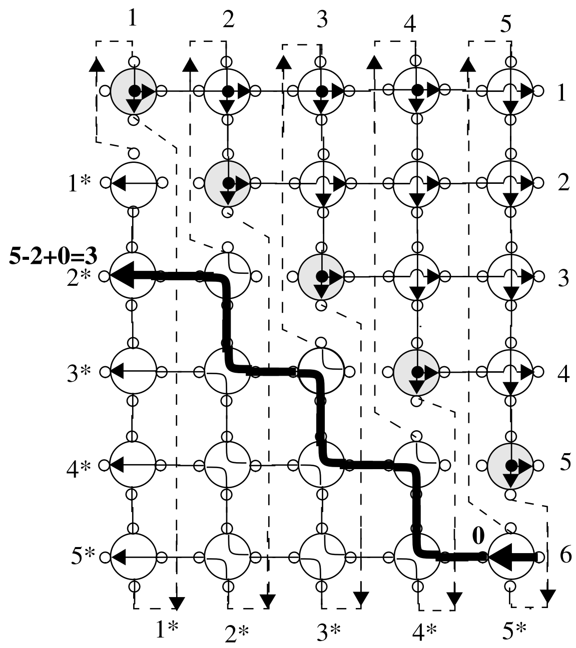


Figure 6 Counting the number of entries in a row

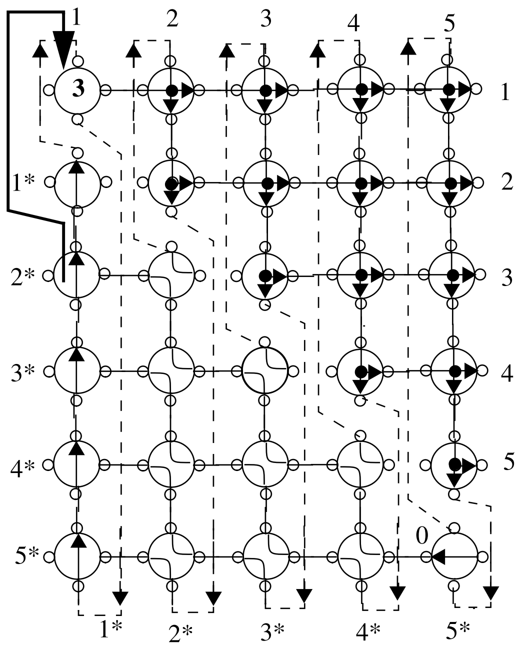


Figure 7 *Count* result is sent to $P_{1,1}$

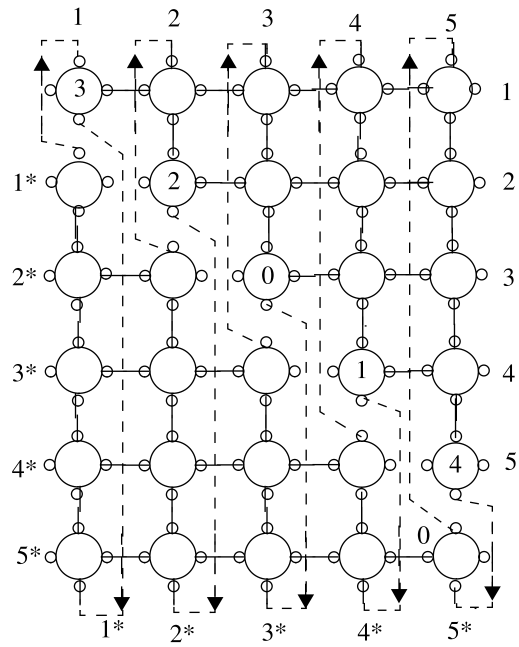


Figure 8 Each 2-d CD-PARBS send *count* result to first plane

Ranking Phase: This phase of the algorithm is different from one proposed in [KHF99], we make few modifications according to our need.

Step 9: Each processor $P_{i, i, k}$ and $P_{i^*, i^*, k}$, $1 \le i \le n$ establishes fork connection and others join connection.

Step 10: Diagonal processor $P_{k, k, k}$ broadcasts *count* to the other diagonal processors on the upper triangle (Figure 9).

Step 11: Every processor on the lower triangle constructs $\langle S, N_1 \rangle$ connection.

Step 12: For the diagonal processor of the upper triangle, $P_{i, i, k}$ (except $P_{k, k, k}$) "if (*count* < *count* at $P_{i, i, k}$) or ((*count* = *count* at $P_{i, i, k}$) and $(k > i)$)" then send signal to S port (Figure 10).

Step 13: If the processor on the lower triangle receives a signal then construct $\{\langle E, N_2 \rangle, \langle S, W \rangle\}$, otherwise construct $\langle E, W \rangle$.

Step 14: If the last diagonal processor on the lower triangle $P_{n^*, n^*, k}$ has received a signal then send 1 to W port, else send 0 to W port (Figure 11).

Step 15: For the processor in the first column of the lower triangle, if $P_{i^*, 1^*, k}$ receives a value 1, then $P_{i^*, 1^*, k}$ sends k to N port, which is received and stored at $P_{i^* - 1, 1^*, k}$, otherwise k is stored at $P_{i^*, 1^*, k}$ (Figure 11).

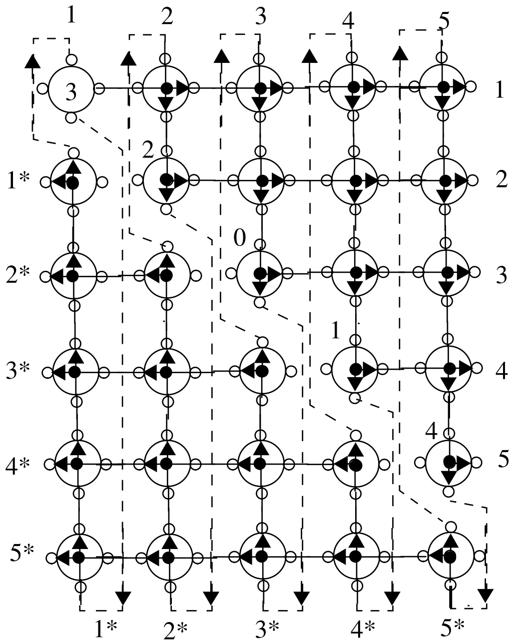


Figure 9 $P_{1,1}$ broadcast the *count*

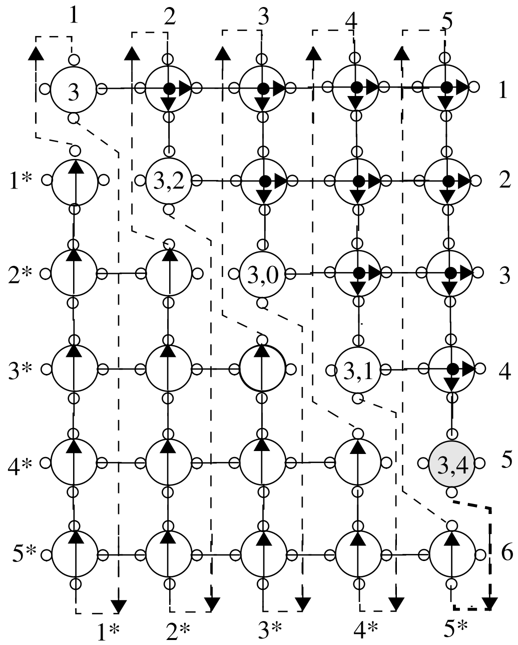


Figure 10 $P_{5,5}$ send signal to S port.

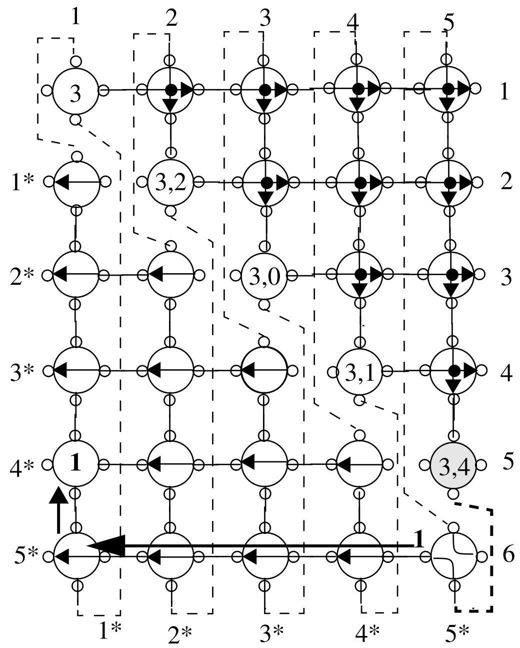


Figure 11 Calulating the *rank* of node 1

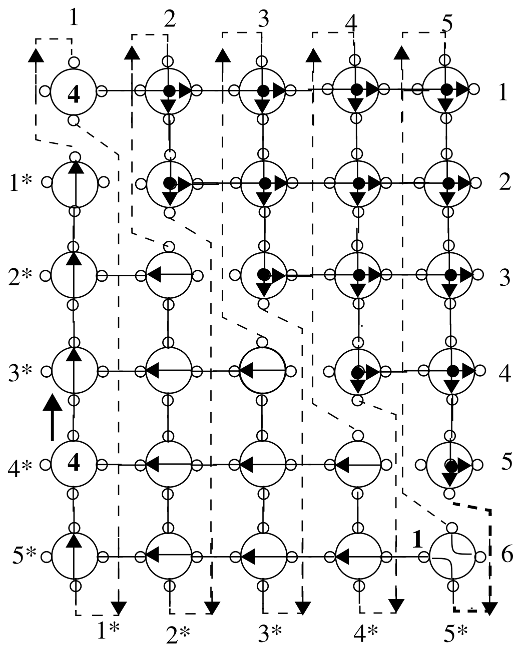


Figure 12 Sending the *rank* of node 1 to $P_{1,1}$

Step 16: On all the planes, all processors on the first column of the lower triangle constructs $\langle S, N_1 \rangle$ connection.

Step 17: The processor on the first column, containing k, broadcasts the value of the ***rank = row index of the processor at the lower triangle,*** to the diagonal processor $P_{1, 1, k}$ (Figure 12).

Step 18: Each diagonal processor $P_{i, i, k}$ and $P_{i^*, i^*, k}$ establishes fork connection, other processors establish join connection, and *rank* is broadcasted from $P_{1, 1, k}$ (Figure 13).

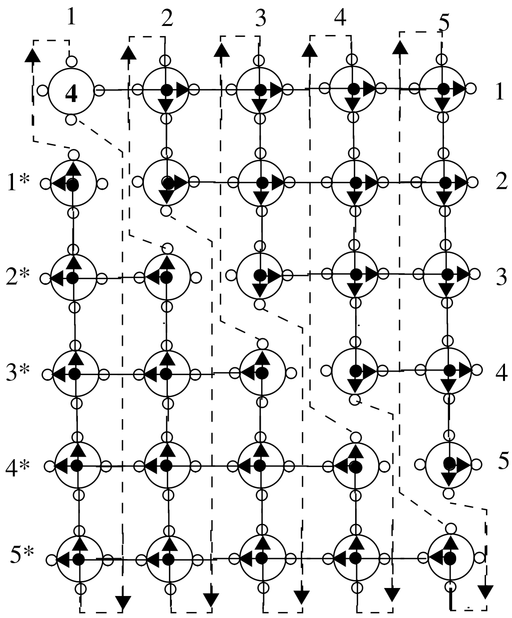


Figure 13 Broadcasting the *rank* of the node 1

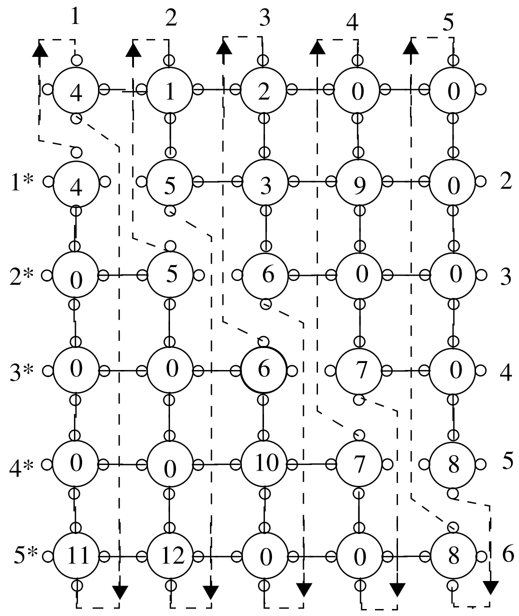


Figure 14 Sparse Matrix A is available at the first plane

III. Perform the row/column interchange operation:

In this phase of the algorithm, depending on the result of the Step II, rows and columns of the sparse matrix *S* are interchanged. For this phase of the algorithm we make a modification in the existing model of the CD-PARBS. We allow horizontal wrap around connections.

We assume that the sparse matrix *S* is available at the first plane (Figure 14). It is broadcasted from the first plane to all other planes, using $\langle U, D \rangle$ bus.

Row interchange:

Step 1: On every plane, all processors on the lower triangle construct $\langle S, N_1 \rangle$ and all processors on the upper triangle construct $\langle N, S_1 \rangle$.

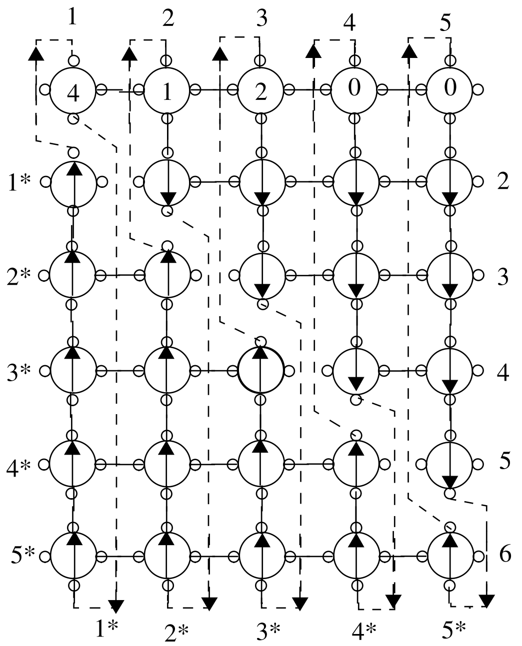


Figure 15 On the first plane, first row is broadcasted

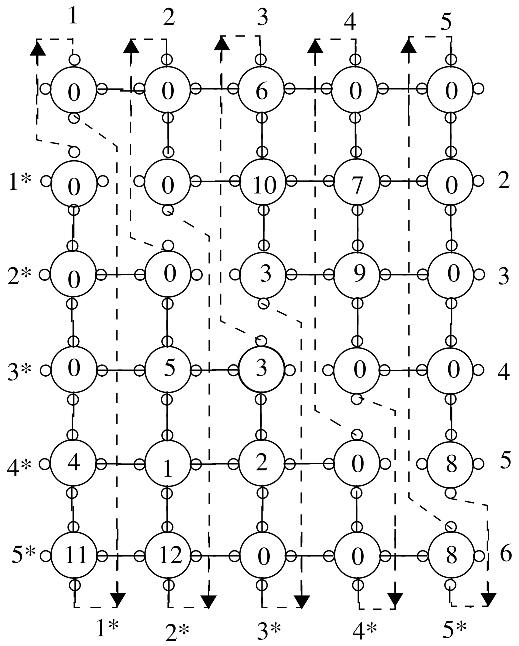


Figure 16 After row interchange

Step 2: Except the diagonal processor $P_{i^*, i^*, k}$, all processors $P_{k, j, k}$ and $P_{k^*, j^*, k}$, $1 \le j \le n$ broadcast row values, which are received by all processors on the kth plane and stored in *S* (Figure 15).

Step 3: Depending on the value of *rank (here denoted by index r)*, the processor $P_{r, j, k}$, in the upper triangular part and $P_{r^*, j^*, k}$, $1 \le j \le n$ in the lower triangular part broadcasts value on $\langle U, D \rangle$ bus which overwrites the value of *S*. This makes the row interchange and the modified value *S* is available at every processor (Figure 16).

Column interchange:

Processor makes wrap around connections in a way that the E port of the right most column of the upper triangle is connected to the E port of the diagonal processor of the lower triangle i.e., the processor $P_{i,n,k}$ is connected to $P_{i^*,i^*,k}$. In the same way, the W port of the left most column of the lower triangle is connected to the diagonal processor of the upper triangle i.e., processor $P_{i^*,1^*,k}$ is connected to the $P_{i,i,k}$. We assume that the vertical connections are disabled when the horizontal connections are established (Figure 17).

Step 4: Every processor on the upper and lower triangle construct $\langle W, E\rangle$ connection.

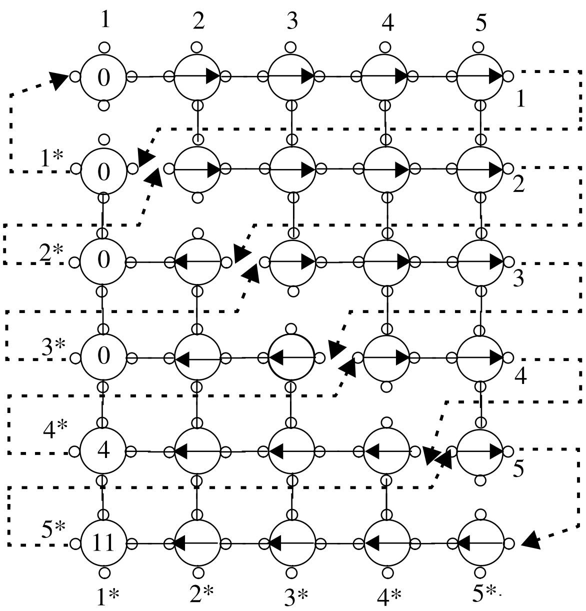


Figure 17 First column on the first plane broadcast values

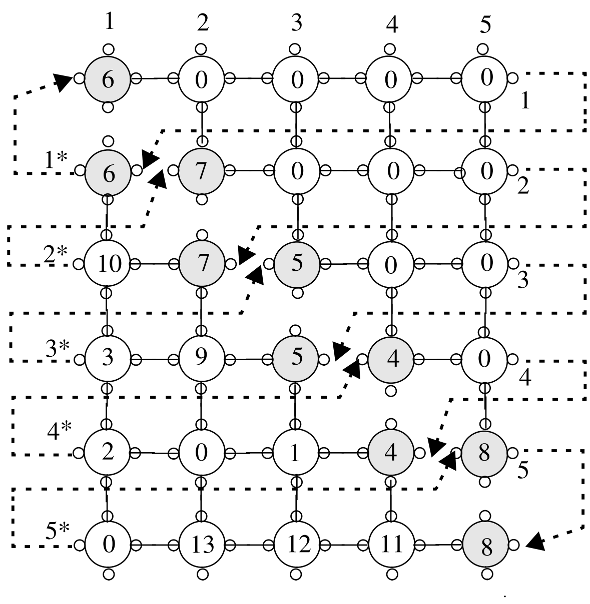


Figure 18 Lower triangularization

Step 5: Except the diagonal processor $P_{i,i,k}$, all processors $P_{i,k,k}$ and $P_{i^*,k^*,k}$, $1 \le i \le n$, broadcast column values of *S*, which are received by all processors on the kth plane and stored in *S* (Figure 17).

Step 6: Depending on the value of the *rank*, processors $P_{i,r,k}$, in the upper triangular part and $P_{i^*,r^*,k}$ in the lower triangular part broadcast values on $\langle U, D\rangle$ bus which overwrites the value of *S*. This makes the column interchange and lower triangularize the matrix *S* (Figure 18).

Conclusion: In this technical report we have presented a constant time parallel algorithm for triangularization of a linear system of equations, where the system is sparse and the graph of it is a directed acyclic graph, using the complete directed PARBS. The same algorithm can be used for producing the block form. It can be observed that the relabelling of the graph produces the same numbering as produced by Tarjan's algorithm [Ste73, Tar72]. To the best of our knowledge, this is the first approach for the design of these algorithms using the CD-PARBS model.

References:

[Agg86] Alok Aggarwal, "Optimal Bounds for Finding Maximum on Array of Processors with k Global Buses", *IEEE Transaction on Computers*, Vol. c-35, No. 1, January 1986, pp. 62-64.

[AHU74] A. Aho, J. Hopcroft and J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[APRS91] Y. Ben-Asher, D. Peleg, R. Ramaswami and A. Schuster, "The Power of Reconfiguration", *Journal of Parallel and Distributed Computing*, 13 (1991), pp.139-153.

[BGOS95] V. Bokka, H. Gurla, S. Olaru and J.L. Schwing, "Constant Time Convexity Problems on Reconfigurable Meshes", *Journal of Parallel and Distributed Computing*, 27 (1995), pp.86-99.

[BK97] K. Bondalapati and V.K. Prasanna Kumar, "Reconfigurable Meshes: Theory and Practice", Reconfigurable Architecture Workshop, *International Parallel Processing Symposium*, April 1997.

[Bok84] S.H. Bokhari, "Finding Maximum on an Array Processor with a Global Bus", *IEEE Transaction on Computers*, Vol. c-33, No. 2, February 1984, pp. 133-139.

[CC94] Yen-Cheng Chen and Wen-Tsuen Chen, "Constant Time Sorting on Reconfigurable Meshes", *IEEE Transaction on Computers*, Vol. 43, No. 6, June 1994, pp. 749-751.

[Che90] Yen-Cheng Chen et al., "Designing Efficient Parallel Algorithms on Mesh-Connected Computers with Multiple Broadcasting", *IEEE Transaction on Parallel and Distributed Systems*, Vol. 1, No. 2, April 1990, pp. 241-245.

[Chen71] Wai-Kai Chen, *Applied Graph Theory*, North Holland Publishing Company, 1971.

[CL96] Gary Chartrand and Linda Lesniak, *Graphs and Digraphs*, Chapman & Hall London, 1996.

[CLR90] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest, *Introduction to Algorithms*, McGraw-Hill Book Company, 1990.

[COS+96] G-H Chen, S. Olariu, J.L. Schwing, B-F. Wang and J. Zhang, "Constant Time Tree Algorithms on Reconfigurable Meshes on Size n x n", *Journal of Parallel and Distributed Computing*, 26 (1995), pp. 137-150.

[CWL92] G-H Chen, B-F Wang and C.J. Lu, "On the Parallel Computation of the Algebraic Path Problem", *IEEE Transaction on Parallel and Distributed Systems*, Vol. 3, No. 2, March 92, pp. 251-256.

[DER86] I.S. Duff, A.M. Erisman and J.K. Ried, *Direct Methods for Sparse Matrices*, Clarendon Press Oxford, 1986.

[GR88b] Alan Gibbons and Wojciech Rytter, *Efficient Parallel Algorithms*, Cambridge University Press, Cambridge, May 1988.

[HS78] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, Addison-Wesley, Computer Science Press, NY, 1978.

[KHF99] Chi-Jung KUO, Chiun-Chieh HSU and Wei-Chen FANG, "Parallel Directed Graph Algorithms on Directional Processor Arrays with Reconfigurable Bus Systems", *New generation Computing, 17(1999)*, pp.175-200.

[MKRS88] R. Miller, V.K. Prasanna Kumar, Dionisios Reisis and Quentin F. Stout, "Meshes with Reconfigurable Buses", *Proc. 15th MIT Conference on Advance Research in VLSI*, March 1988, pp. 163-178.

[MKRS93] R. Miller, V.K. Prasanna Kumar, Dionisios Reisis and Quentin F. Stout, "Parallel Computations on Reconfigurable Meshes", *IEEE Transactions on Computers*, Vol. 42, No. 6, June 1993.

[NW95] Koji Nakano and Koichi Wada, "Integer Summing Algorithms on Reconfigurable Meshes", *IEEE First International Conference on Algorithms and Architectures for Parallel Processing*, Brisbane, Australia, April 1995, pp. 19-21.

[ST81] M.N.S. Swamy and K. Thulasiraman, *Graphs, Networks and Algorithms*, A Wiley Interscience Publication, 1981.

[Ste73] G.W. Stewart, *Introduction to Matrix Computations*, Academic Press Inc., San Diego California 1973.

[Tar72] Robert Tarjan, "Depth-First Search and Linear Graph Algorithms", *SIAM Journal of Computing,* Vol. 1, No.2, 1972, pp. 146-160.

[THT+97] Horng-Ren Tsai, Shi-Jinn Horng, Shun-Shan Tsai, Tzong-Wann Kao and Shung-Shing Lee, "Solving an Algebraic Path Problem and Some Related problems on a Hyper Bus Broadcast Network", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 12, No. 5, Dec. 1997.

[WC90] Biing-Feng Wang and Gen-Huey Chen, "Constant Time Algorithms for the Transitive Closure and Some Related Problems on Processor Arrays with Reconfigurable Bus Systems", *IEEE Transaction on Parallel and Distributed Systems*, Vol. 1, No. 4, October 1990, pp. 500-507.

[WCL90] Biing-Feng Wang, Gen-Huey Chen and Ferng-Ching Lin, "Constant Time Sorting on a Processor Array with Reconfigurable Bus System", *Information Processing Letters*, 34(1990), pp. 187-192.