Euro PVM/MPI 2003

Venezia, Italia

Efficient Parallel Implementation of Transitive Closure of Digraphs

C. E. R. Alves

Univsidade São Judas Tadeu

E. N. Cáceres

Universidade Federal de Mato Grosso do Sul

A. A. Castro Jr.

Universidade Católica Dom Bosco

S. W. Song

Universidade de São Paulo

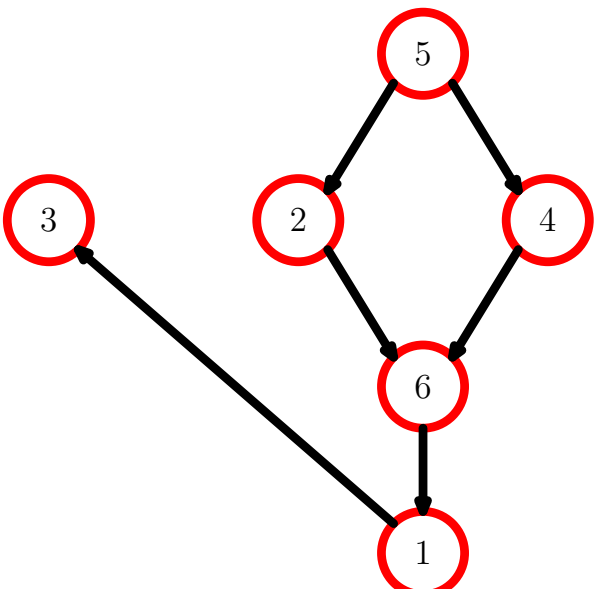
J. L. Szwarcfiter

Universidade Federal do Rio de Janeiro

The Transitive Closure Problem

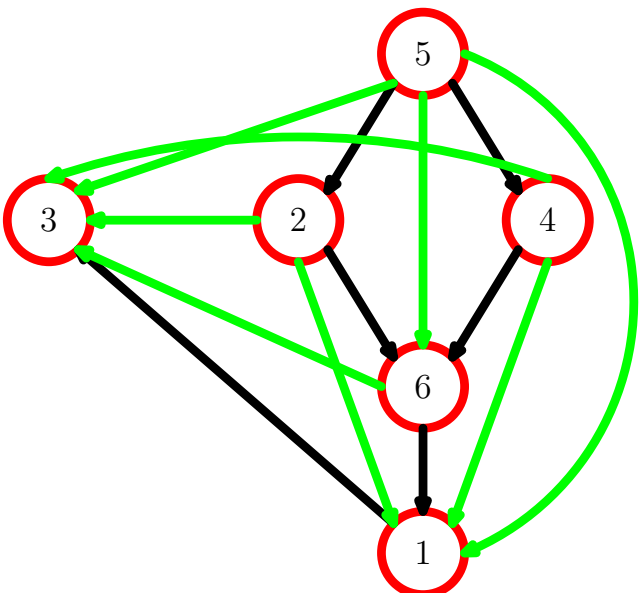
- Used in many areas such as
  - Network Planning
  - Distributed Systems Design
- Used in problems such as
  - All Shortest Paths in a Directed Graph
  - Breadth-First Spanning Trees
- Directed graph $D(V, E)$ with $|V| = n$, $|E| = m$
- We present a parallel algorithm to compute its transitive closure using
  - p processors
  - each with $O(\frac{n^2}{p})$ local memory

Example



A directed graph.

Example



Its transitive closure: green edges joining i to j if j can be reached from i.

BSP/CGM Model

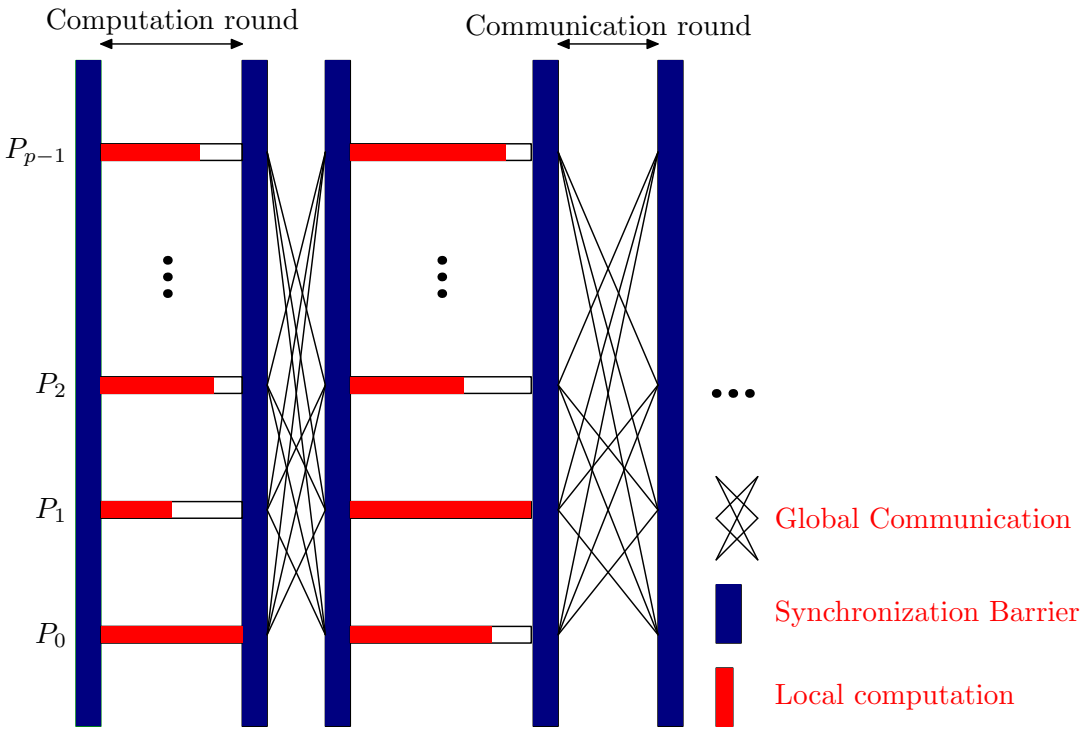
CGM (Coarse Grained Multicomputer) model: p of processors, each with its own local memory, communicating through a network.
The algorithm alternates between

- Computation round: each processor computes independently.
- Communication round: each processor sends/receives data to/from other processors.

Goals:

- Obtain a linear speed-up on p.
- Minimize the number of rounds.

The CGM Model



Previous Parallel Algorithms

1. **PRAM:**
   - **Karp et al.: CREW: $O(\log^2 n)$ time with $O(M(n))$[1] processors.**
   - **JáJá: CRCW: $O(\log n)$ time with $O(n^3)$ processors.**
2. **Cáceres et al.: Acyclic digraph with linear extension labeling $O(logp)$ rounds with $O(n^3/p)$ local time**
3. **Dependency Graph Approach:**
   - **Pagourtzis et al.: $O(p)$ rounds with $O(n^3/p)$ local time**

[1] $M(n)$ is the best known sequential bound for multiplying two $n \times n$ matrices over a ring

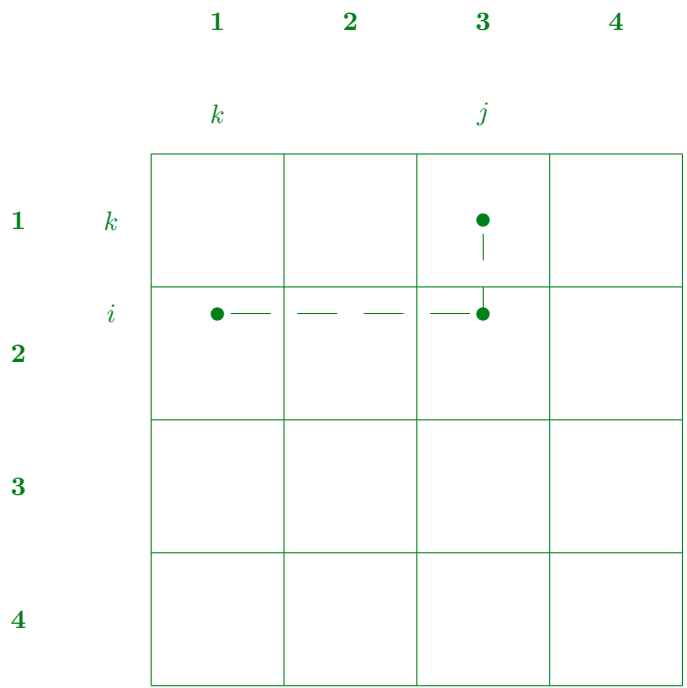
Warshall's Algorithm

Algorithm 1: Warshall's Algorithm

Input: Adjacency matrix $M_{n \times n}$ of graph G
Output: Transitive closure of graph G

```
1: for k ← 1 until n do
2:     for i ← 1 until n do
3:         for j ← 1 until n do
4:             M[i, j] ← M[i, j] or (M[i, k] and M[k, j])
5:         end for
6:     end for
7: end for
```

Partitioning the Adjacency Matrix



The Parallel Algorithm

Algorithm 2: Parallel Warshall

Input: Adjacency matrix M stored in the p processors: each processor q $(1 \leq q \leq p)$ stores submatrices $M[(q-1)\frac{n}{p}+1..q\frac{n}{p}][1..n]$ and $M[1..n][(q-1)\frac{n}{p}+1..q\frac{n}{p}]$.

Output: Transitive closure of graph G represented by the transformed matrix M.

Algorithm 3: Parallel Warshall

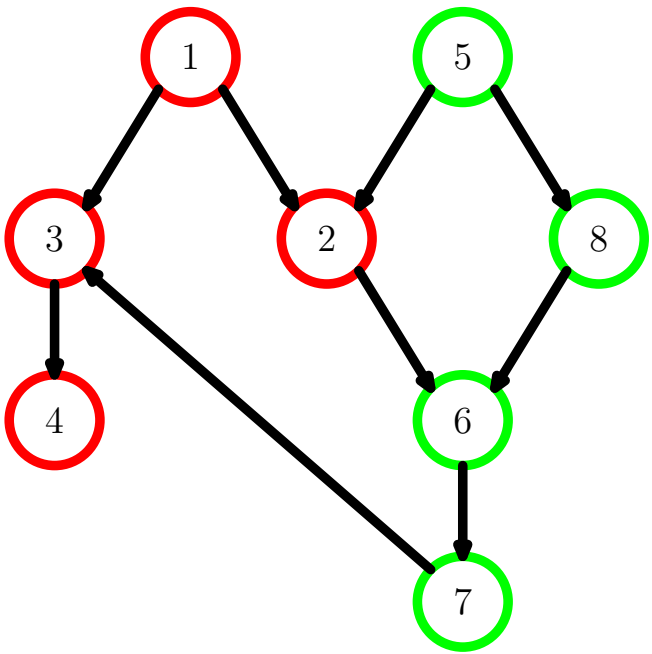
Each processor q $(1 \leq q \leq p)$ does the following.

1: **repeat**
2: **for** $k = (q-1)\frac{n}{p} + 1$ **until** $q\frac{n}{p}$ **do**
3: **for** $i = 0$ **until** $n - 1$ **do**
4: **for** $j = 0$ **until** $n - 1$ **do**
5: **if** $M[i][k] = 1$ **and** $M[k][j] = 1$ **then**
6: $M[i][j] = 1$ (if $M[i][j]$ belongs to processor different from q then store it for subsequent transmission to the corresponding processor.)
7: **end if**
8: Send stored data to the corresponding processors.
9: Receive data that belong to processor q from other processors.
10: **end for**
11: **end for**
12: **end for**
13: **until** no new matrix entry updates are done

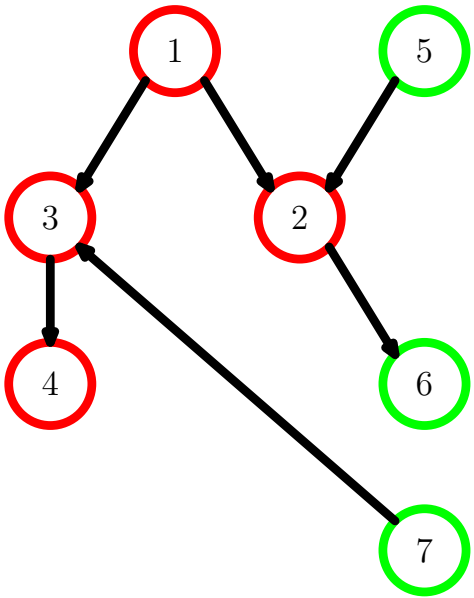
The Main Idea

- Make a partition of $V(D)$.
- In each partition, using the edges of D construct a digraph formed by the edges of D that have at least one of its extremes in the partition.
- Compute the Transitive Closure in each partition.
- Send the computed transitive edges to the proper partition.

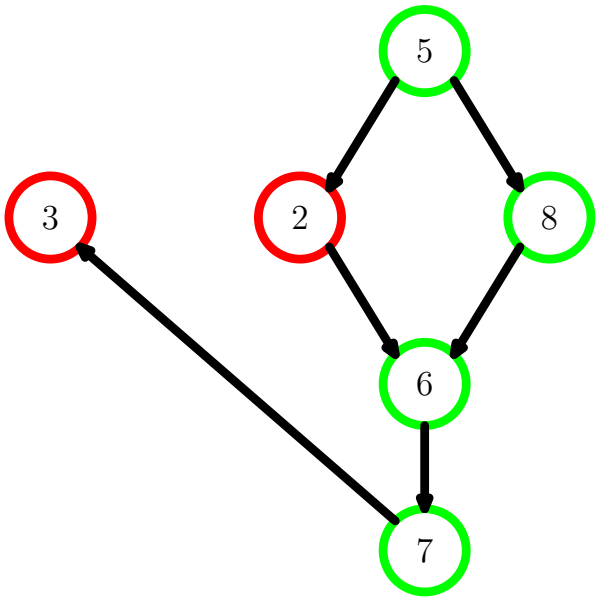
Example



Example

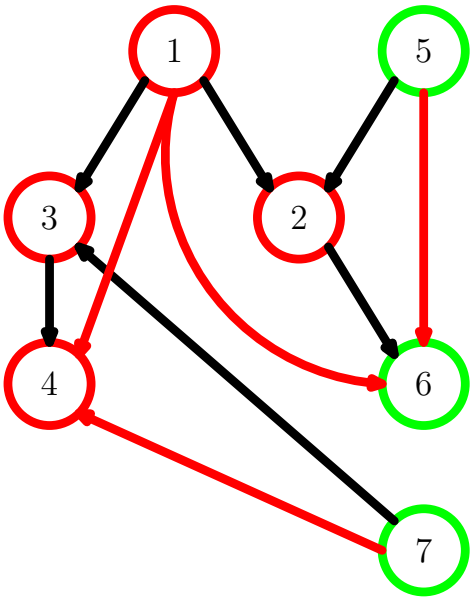


Processor 0

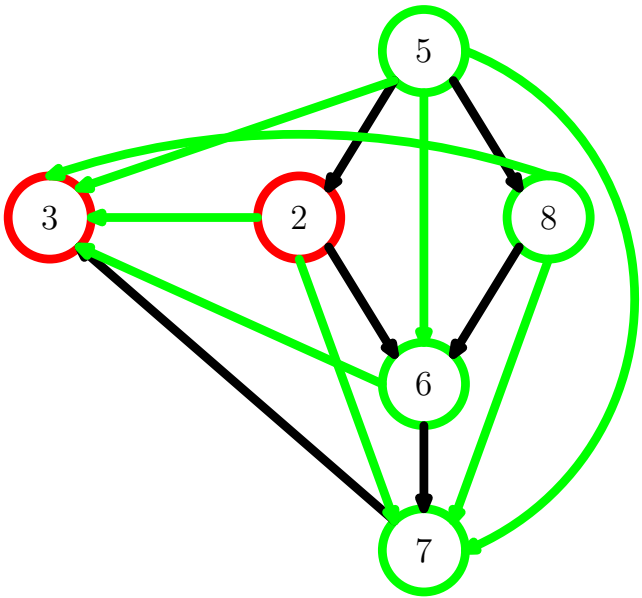


Processor 1

Example

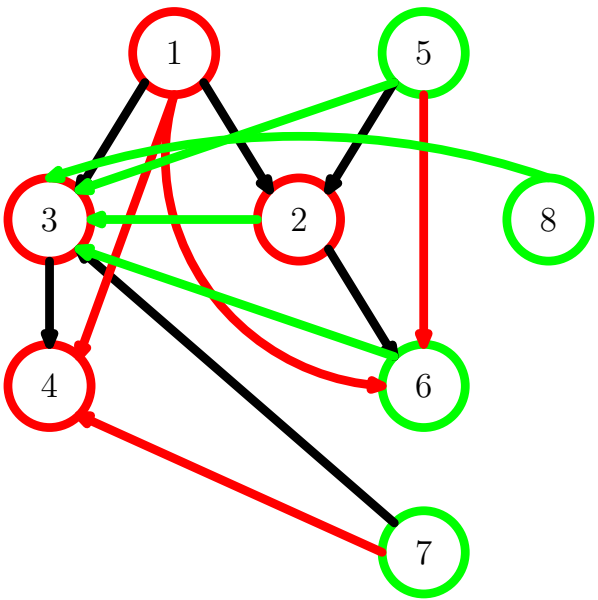


Processor 0

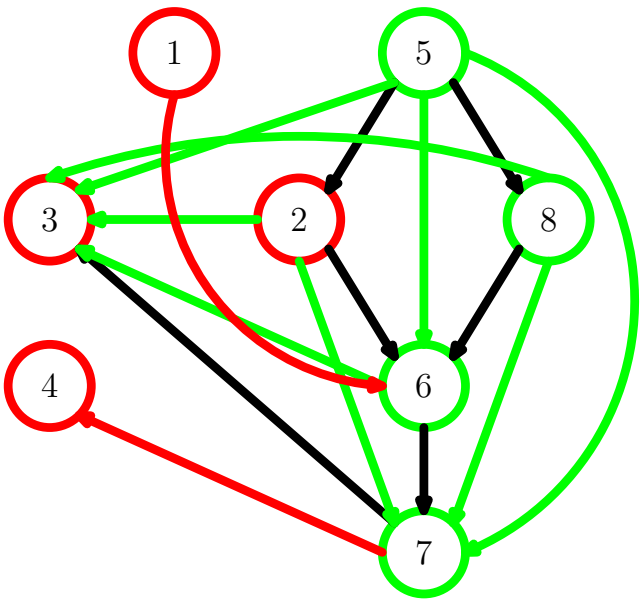


Processor 1

Example



Processor 0

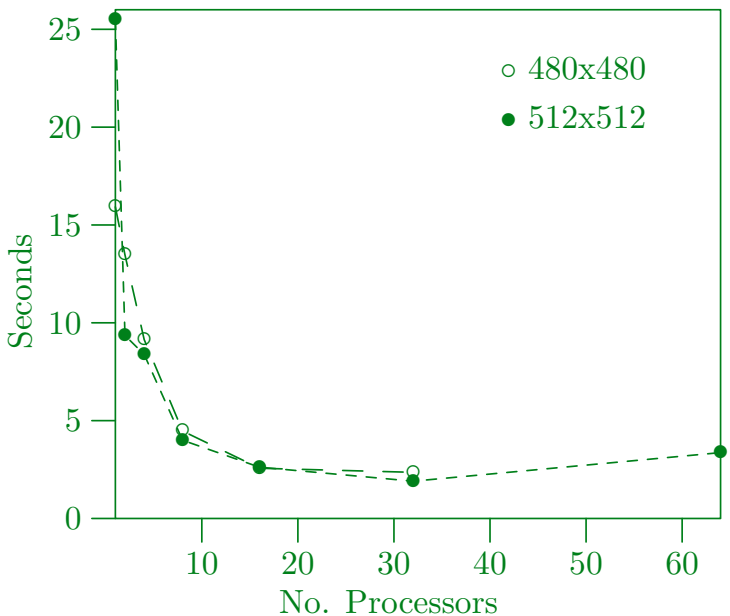


Processor 1

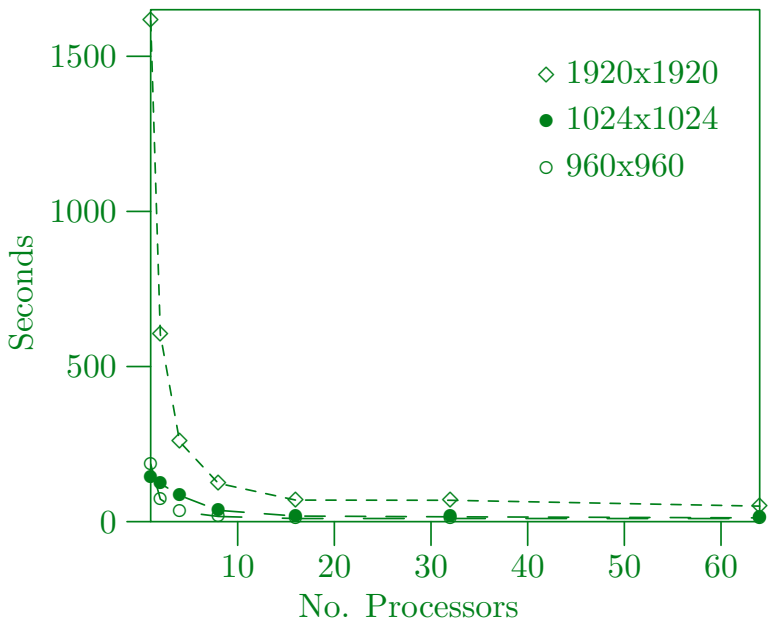
Implementation

- 64-node Beowulf cluster - low cost microcomputers with 256MB RAM, 256MB swap memory, CPU Intel Pentium III 448.956 MHz, 512KB cache.
- 100 Mb fast-Ethernet switch.
- Code in standard ANSI C and LAM-MPI Version 6.5.6.
- Tests on randomly generated digraphs with 20 % probability of an edge between two vertices.
- In all the tests, the number of communication rounds required are less than $\log p$.

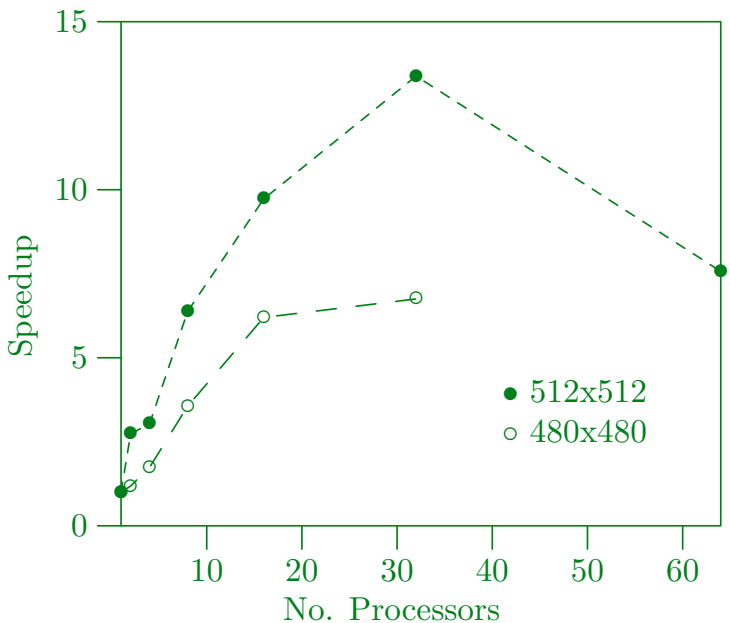
Implementation Results



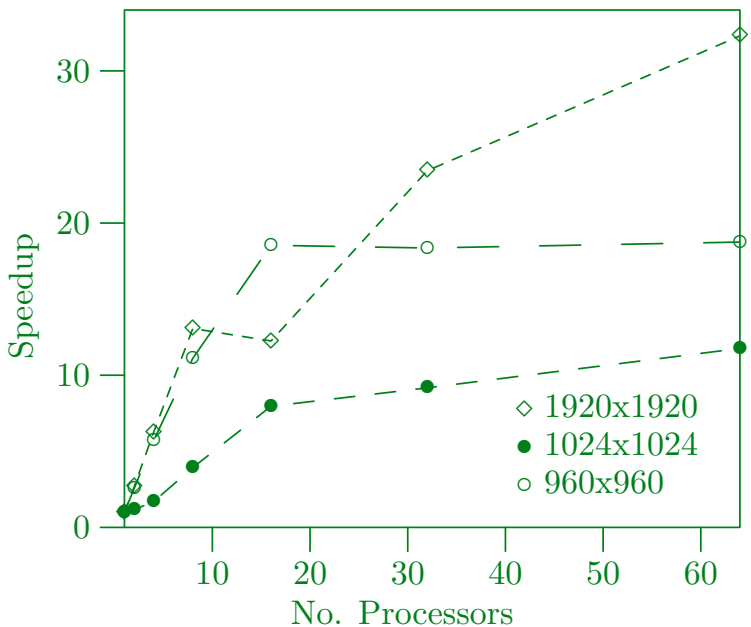
Implementation Results



Implementation Results



Implementation Results



Conclusion

A BSP/CGM algorithm for the Transitive Closure problem.

- **Digraph with n vertices and m edges.**
- **The number of communication rounds measured: $O(\log p)$.**
- **Local computation time: $O(mn/p)$.**