Ant Colony System using MPI

Sachin Bhatt
CSE633 fall 2011
State University of New York at Buffalo

Outline

- About
- Problem to solve - tsp
- Types of algorithm
- Min-max
- RAM algo
- Parallel – coarse grained
- Fine-grained approach

Ant colony system

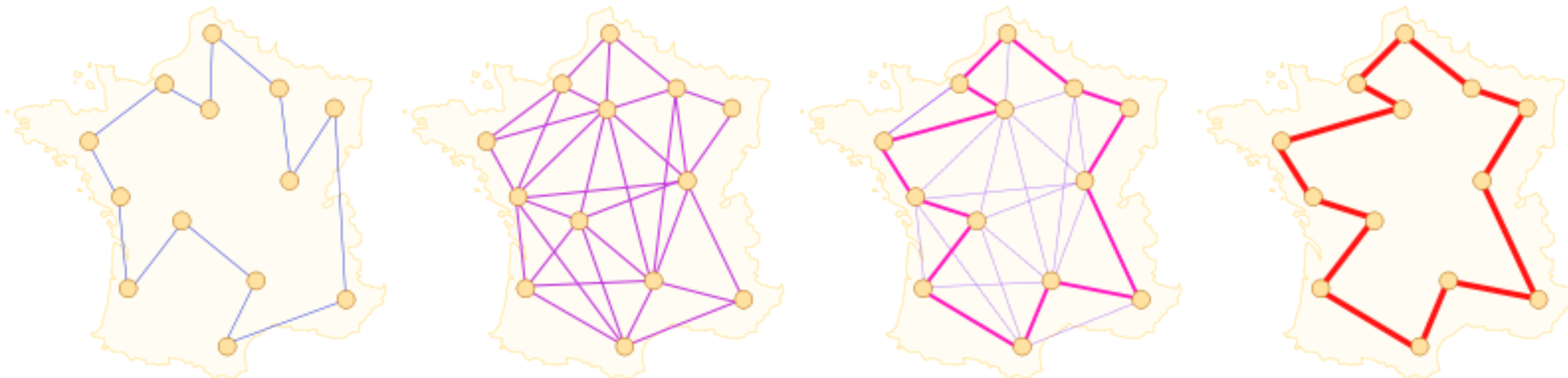
- Probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs.
- Meta heuristic optimizations

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}$$

- Trail update

$$\tau_{xy}^{k} = (1-\rho)\tau_{xy}^{k} + \Delta\tau_{xy}^{k}$$

$$\Delta\tau_{xy}^{k} = \begin{cases} Q/L_k & \text{if ant } k \text{ uses curve } xy \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}$$



1 2 3 4

Source: wikipedia

Travelling Salesman Problem

- Find the shortest possible tour that visits each city exactly once
- NP-hard problem
- Running time of exact algorithms $O(n!)$
- Applications
  - Planning
  - Logistics
  - Manufacture of microchips

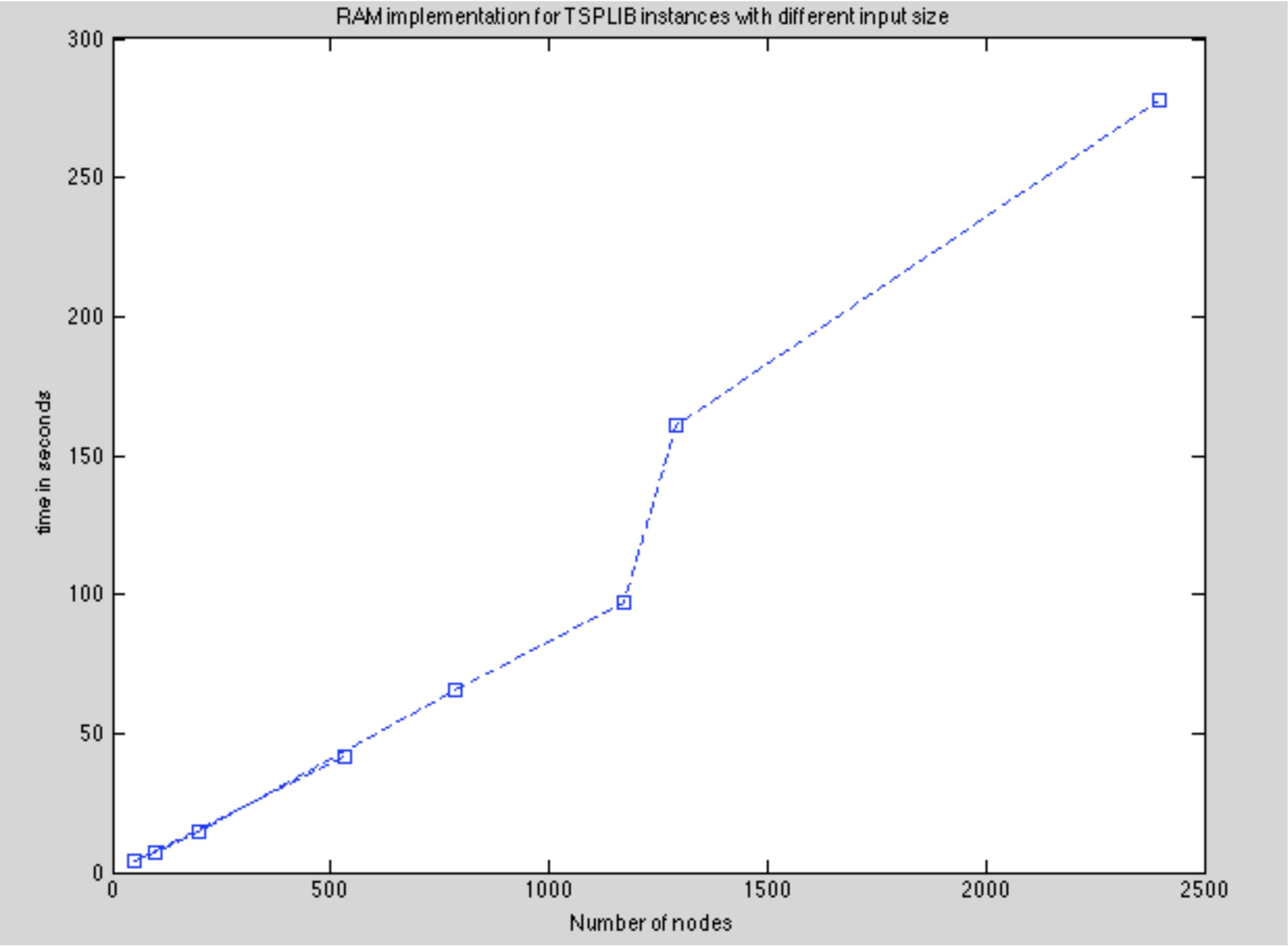
Types of ACS

- Ant System (AS)
- ACS
- Min-max (MMAS)

RAM algo

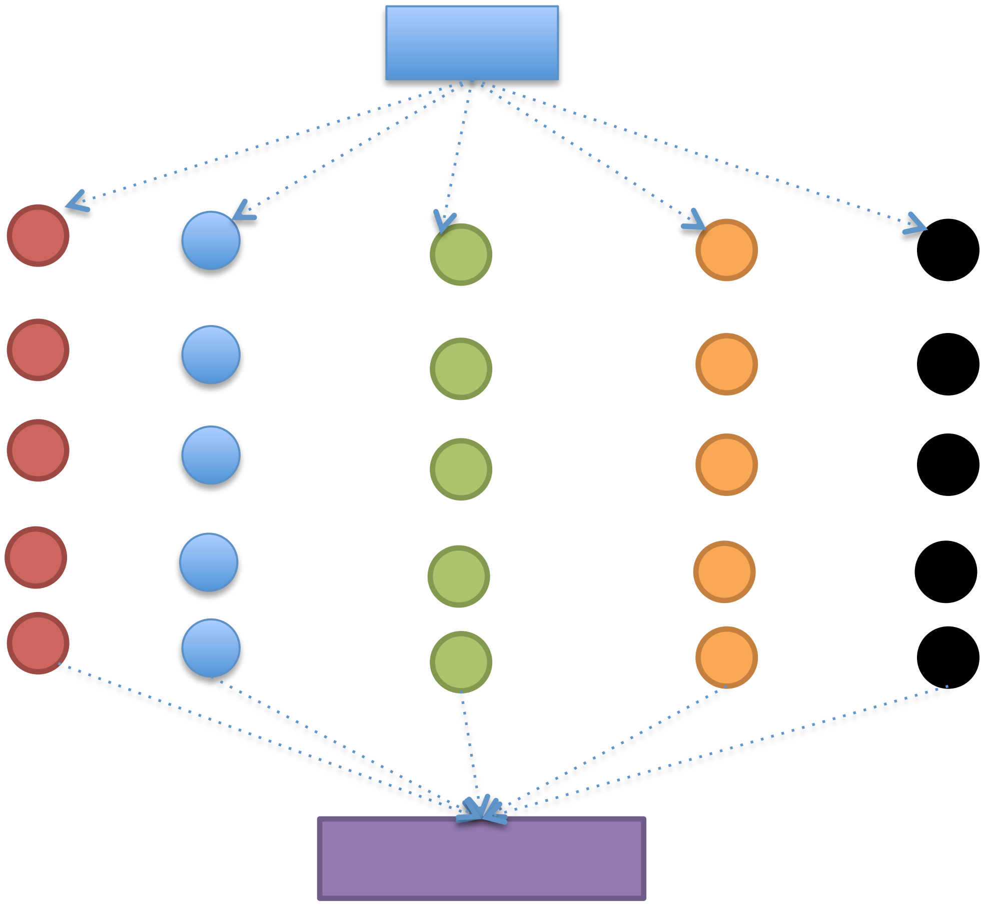
```
init(argc, argv);
for ( number of trials ) {
      initializeTry() //reset values

      while ( !termination_condition() ) {
         construct_solutions();

         update_statistics;
         pheromone_trail_update;
            search_control_and_statistics();
         iteration++;
      }
      exit_try()
     }
 exit_program();
```

RAM plot



RAM implementation for TSPLIB instances with different input size
time in seconds
Number of nodes
0
50
100
150
200
250
300
500
1000
1500
2000
2500

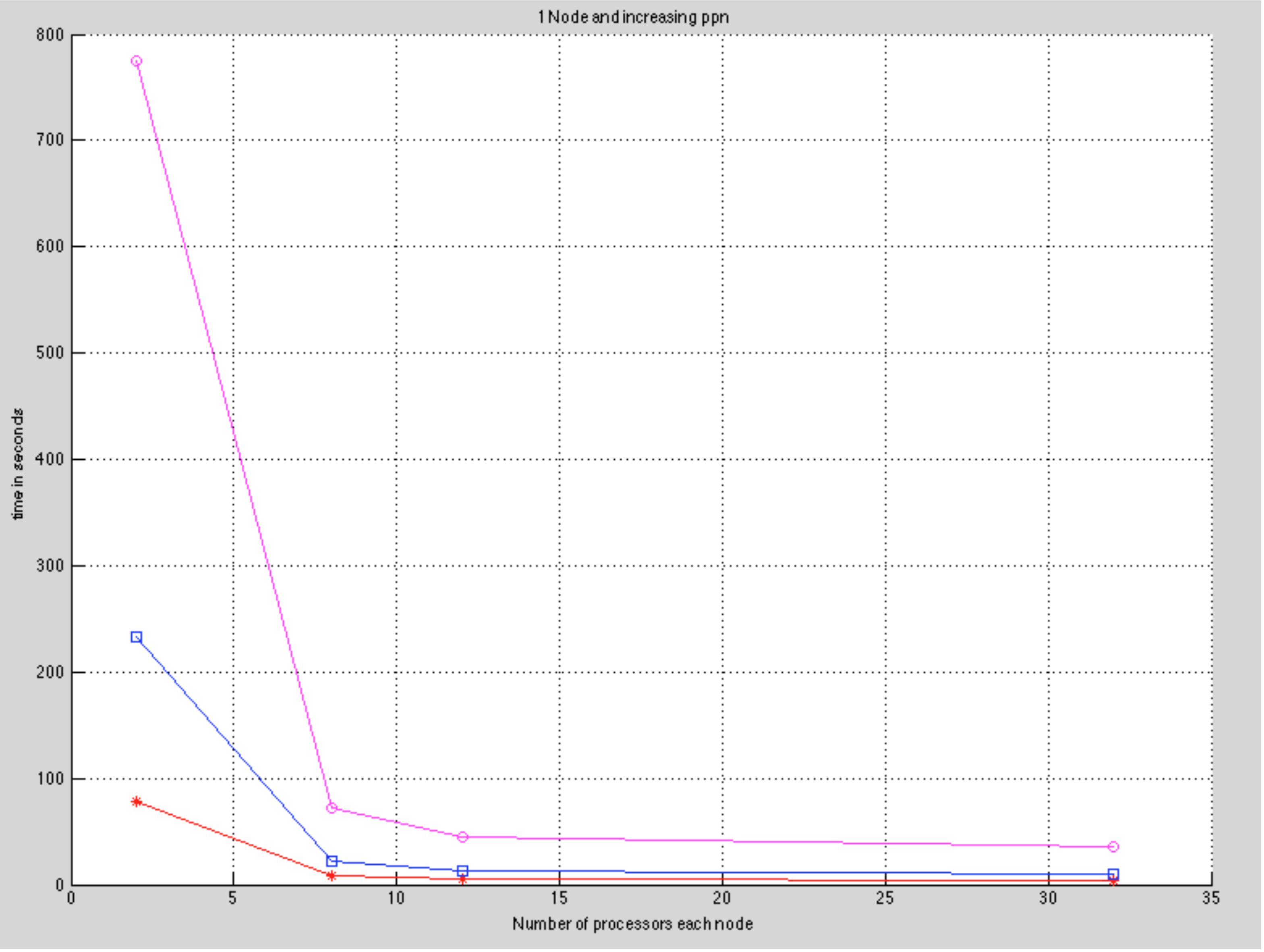
Coarse-grained



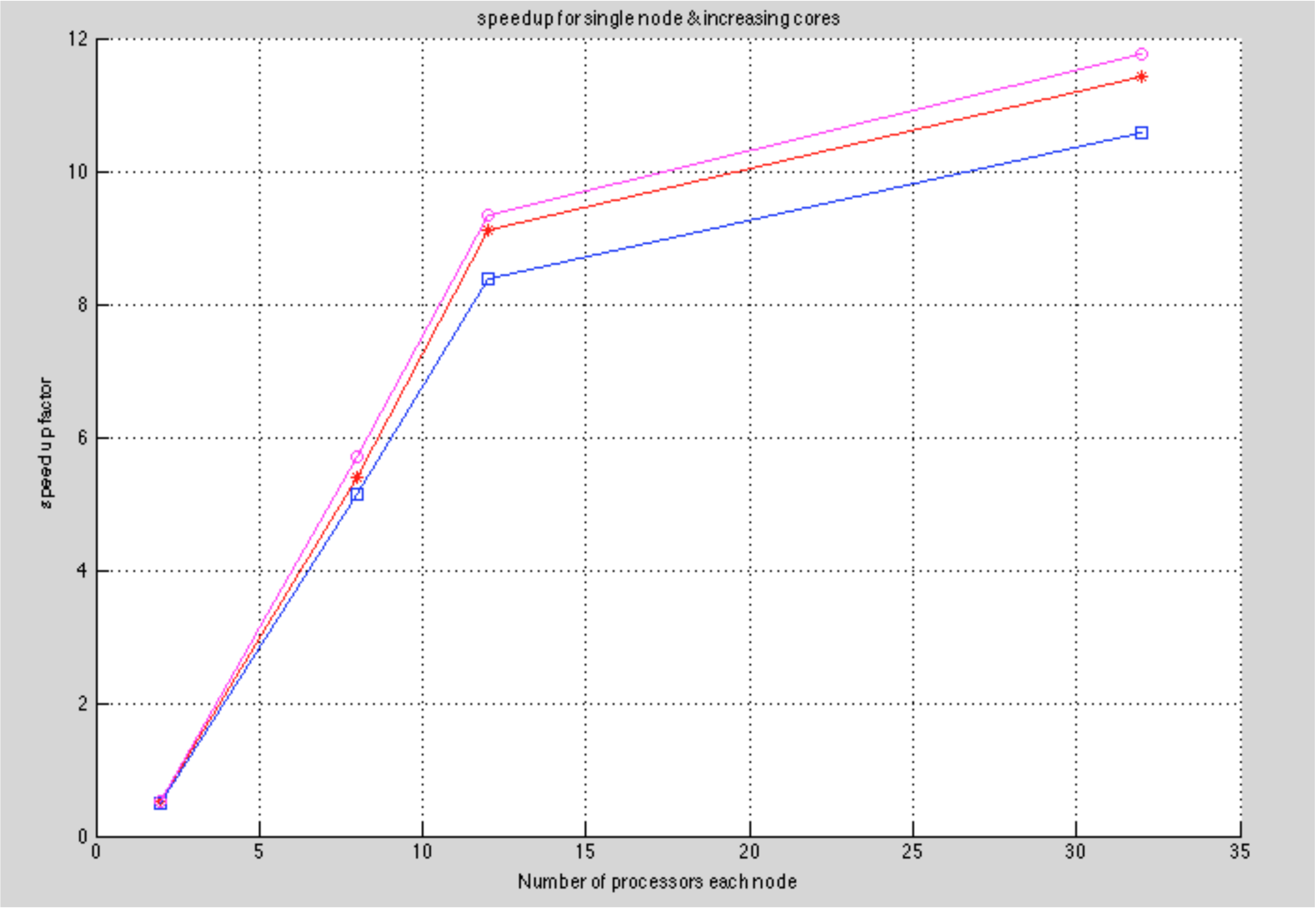
Parallel: Coarse grained implementation

```
init(argc, argv);

COMM_WORLD.Bcast(matrix,size,LONG,0);

        for ( number of trials/size ) {
                initializeTry() //reset values

                while ( !termination_condition() ) {
                    construct_solutions();

                    update_statistics;
                    pheromone_trail_update;
                        search_control_and_statistics();
                    iteration++;
                }
                exit_try()
              }
         exit_program();
COMM_WORLD.Gather(tour,sendcount,LONG,alltours,recvcount,LONG,0);
```

coarse plot node vs cores

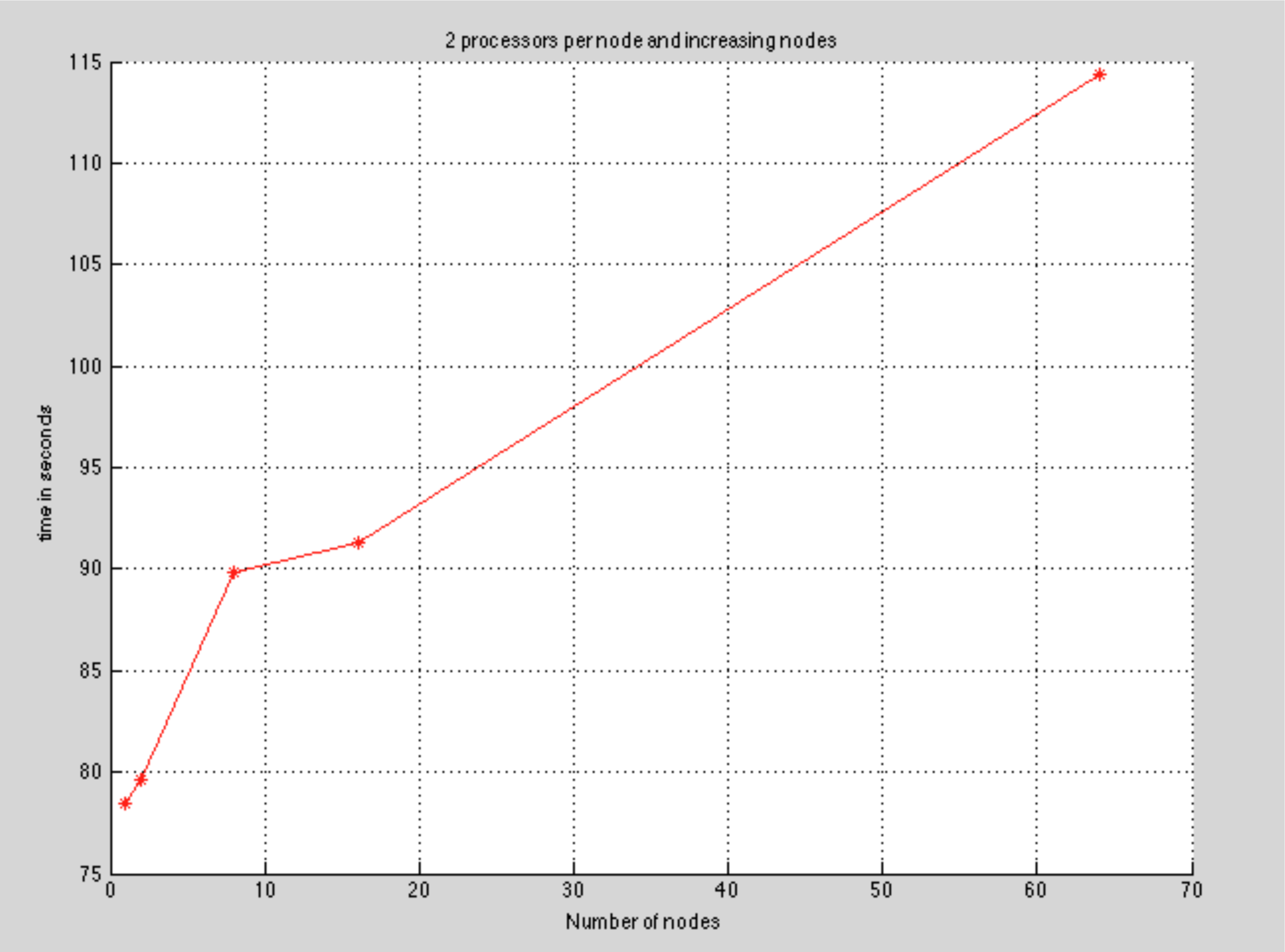


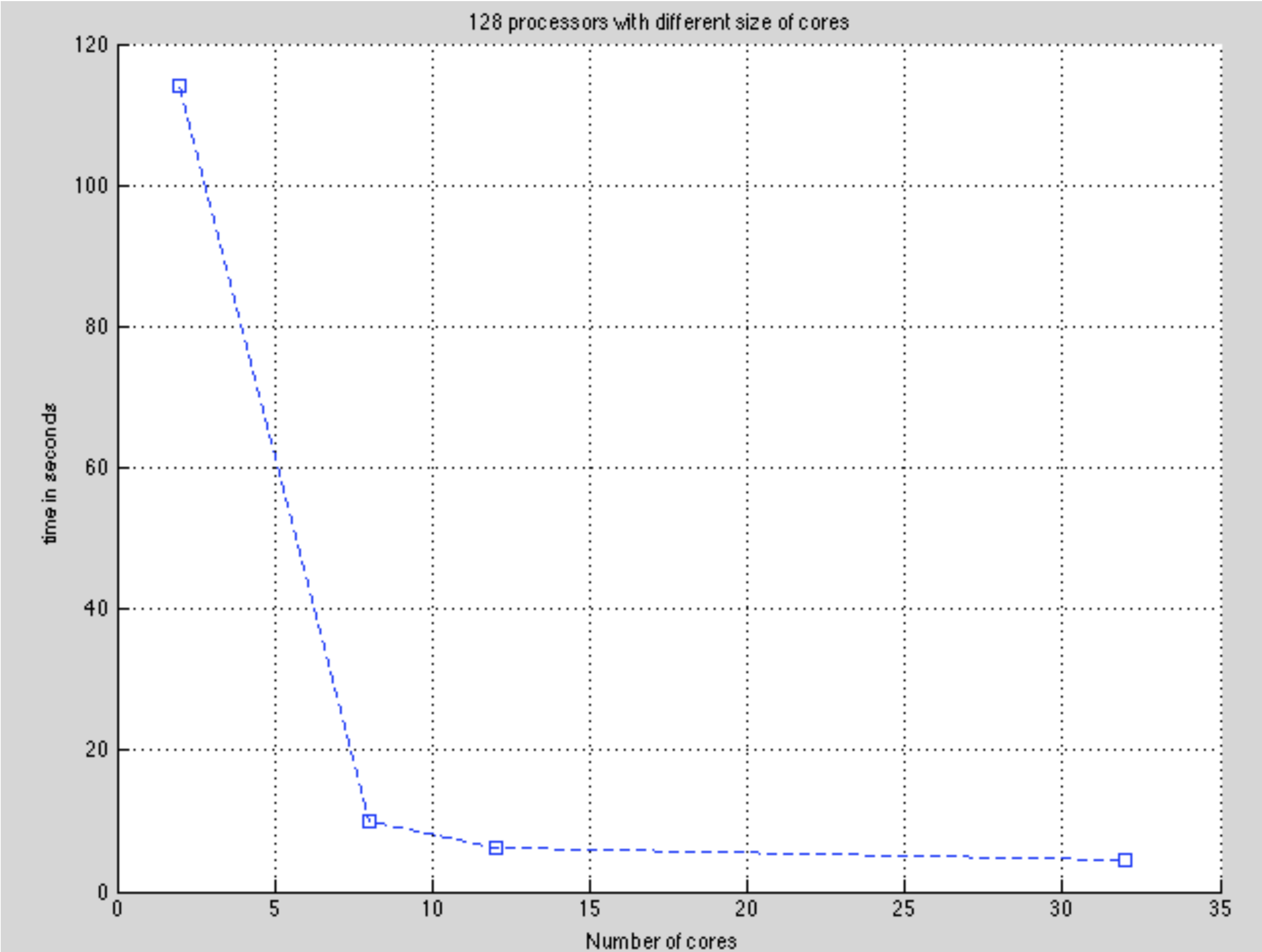
Speedup



speedup for single node & increasing cores
12
10
8
6
4
2
0
0
5
10
15
20
25
30
35
Number of processors each node
speed up factor

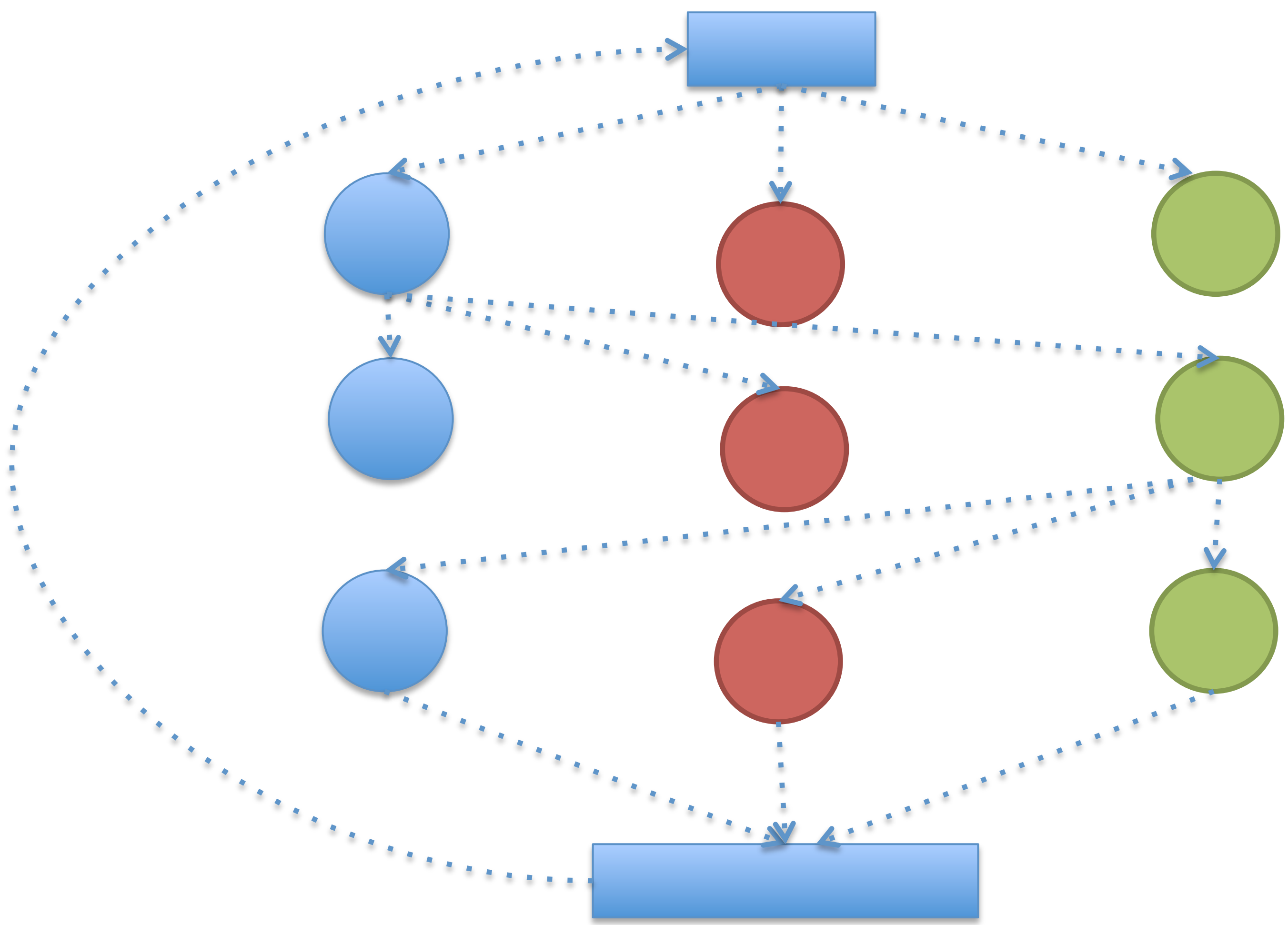
Increasing nodes





128 processors with different size of cores
time in seconds
Number of cores
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35

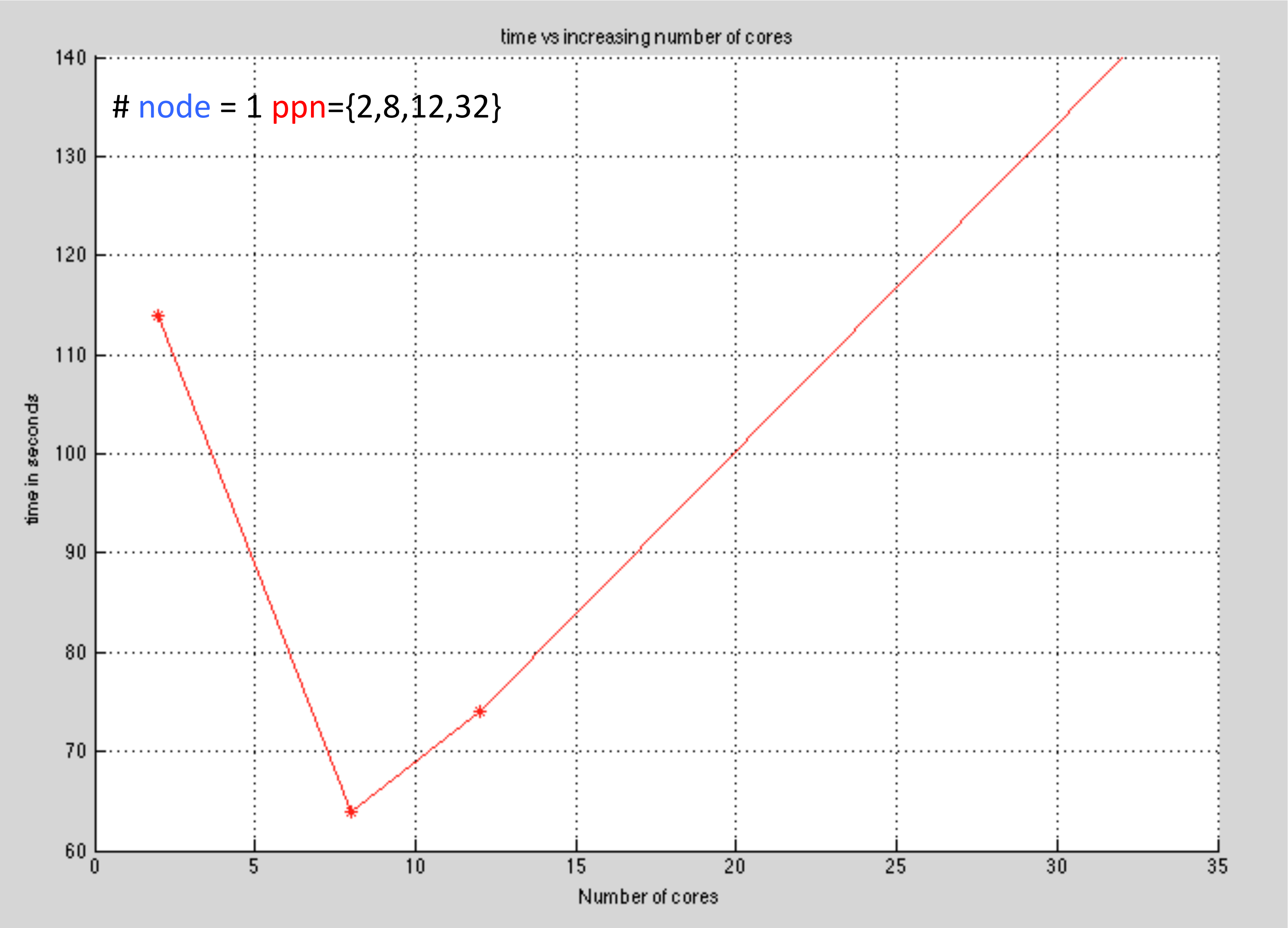
Fine grained implementation

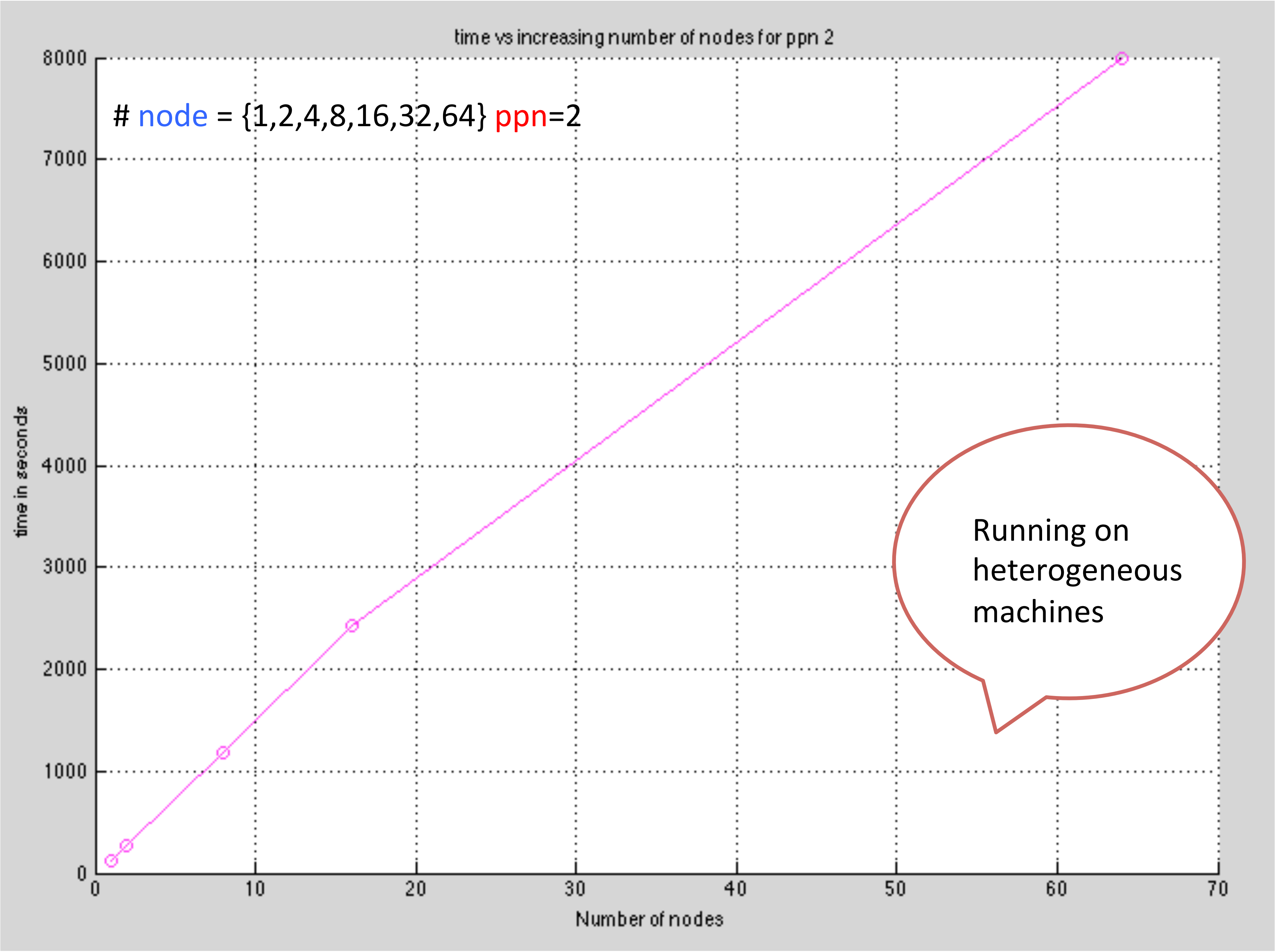


Fine grained implementation

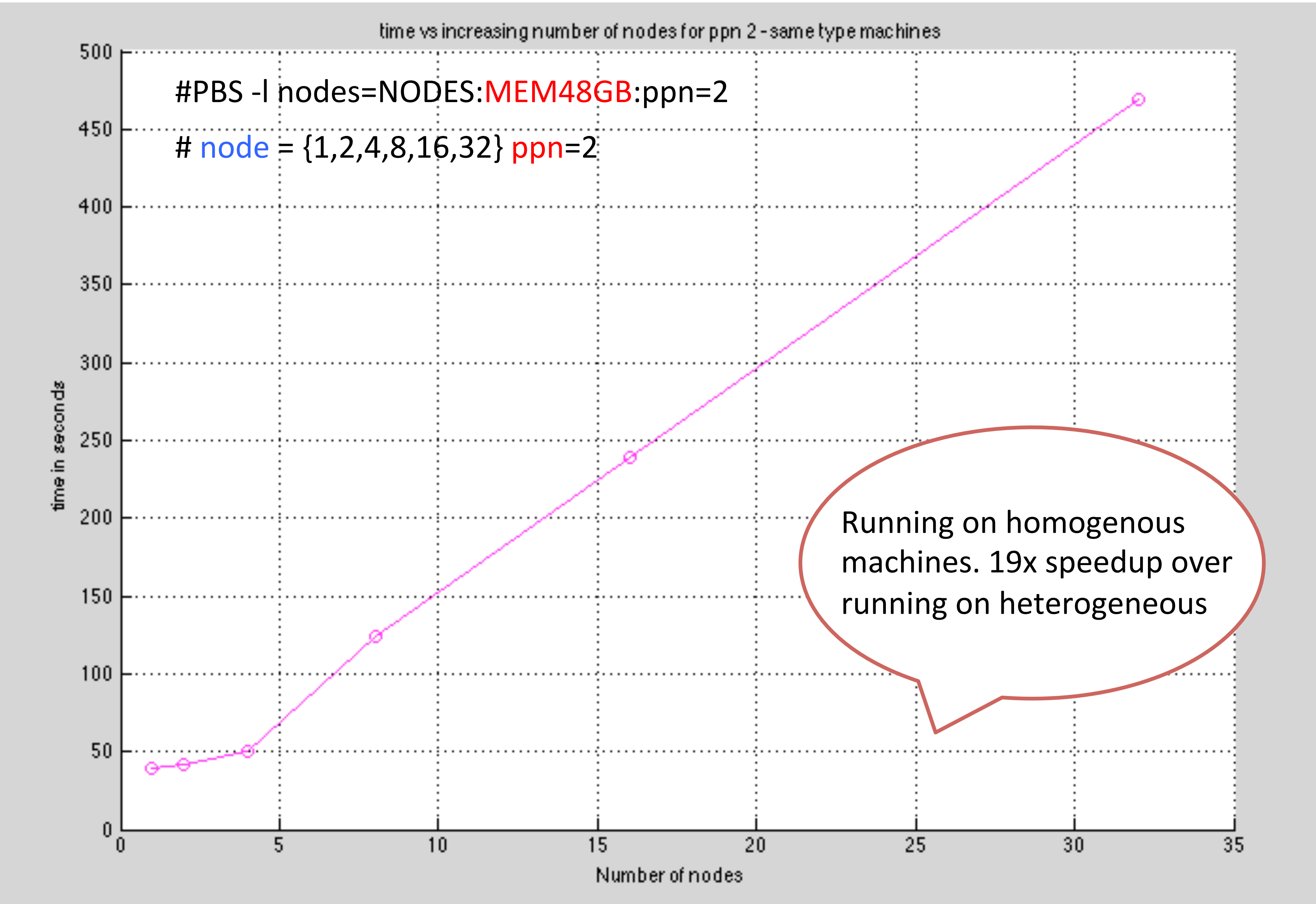
```
init(argc, argv);
COMM_WORLD.Bcast(distance_matrix,size,LONG,0);
        for ( number of trials ) {
                initializeTry() //reset values
                while ( !termination_condition() ) {
                    construct_solutions(ants/size);

                    update_statistics;
                    pheromone_trail_update;
                        search_control_and_statistics();
                    iteration++;
COMM_WORLD.Barrier();
COMM_WORLD.Gather(tour_value,sendCount,LONG,best_tour,recvCount,LONG,0);
if(rank==0)
    COMM_WORLD.Bcast(bestRank[0],1,INT,0);

if(bestRank[0]==rank)
        COMM_WORLD.Bcast(matrix,size,DOUBLE,bestRank[0]);
                }
                exit_try()
              }
         exit_program();
```

Fine grained plot





time vs increasing number of nodes for ppn 2
node = {1,2,4,8,16,32,64} ppn=2
Running on heterogeneous machines
time in seconds
0
1000
2000
3000
4000
5000
6000
7000
8000
Number of nodes
0
10
20
30
40
50
60
70


time vs increasing number of nodes for ppn 2 - same type machines
#PBS -l nodes=NODES:MEM48GB:ppn=2
node = {1,2,4,8,16,32} ppn=2
Running on homogenous machines. 19x speedup over running on heterogeneous
time in seconds
Number of nodes
0
50
100
150
200
250
300
350
400
450
500
0
5
10
15
20
25
30
35

Observations

- Increasing number of cores gave a factor of 14X speed up for coarse-grained approach
- Fine grained approach gave best results with 8 cores for one node execution
- Increasing number of nodes keeping cores constant increased execution time with number of nodes & was dominated by I/O
- For fixed number of processor execution, more number of cores performed better than small cores size.
- Running fine grained approach on cluster with homogenous machines performed 20X times better running same with heterogeneous systems.
- Fine grained approach suffers from communication overhead & not suited to run in parallel

Future work

- Implement OpenMP for fine-grained approach and compare performance with single node, multiple core run of MPI implementation.
- Modifying problem to include convergence factor into termination condition.

References

- M. Dorigo & T. Stützle, 2004. Ant Colony Optimization, MIT Press. ISBN 0-262-04219-3
- http://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms