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

\[ p_{xy}^k = \frac{\left( \tau_{xy}^\alpha \right) \left( \eta_{xy}^\beta \right)}{\sum \left( \tau_{xy}^\alpha \right) \left( \eta_{xy}^\beta \right)} \]

• Trail update

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

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

```c
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

Graph shows the relationship between the number of nodes and the time taken for the RAM implementation.
Coarse-grained
Parallel: Coarse grained implementation

`init(argc, argv);`

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

```c
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();
```
coarse plot node vs cores
Speedup

speedup for single node & increasing cores

Number of processors each node

speedup factor

0  5  10  15  20  25  30  35
Increasing nodes

2 processors per node and increasing nodes

time in seconds

Number of nodes
Fine grained implementation
Fine grained implementation

```c
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

# node = 1 ppn={2,8,12,32}
Running on heterogeneous machines

# node = \{1, 2, 4, 8, 16, 32, 64\} ppn=2
Running on homogenous machines. 19x speedup over running on heterogeneous machines.
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


• http://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms