Optimizing Pheromone Modification for Dynamic Ant Algorithms

Ryan Ward
TJHSST Computer Systems Lab 2006/2007

Testing

To test the relative effectiveness of the reset and distance-based methods versus each other, multiple combinations for severity (denoted by *s*) and frequency (denoted by *f*) were used. All possible combinations of $s \in \{1, 5, 10\}$ and $f \in \{5, 10, 25\}$ were tested.

For one test run, 5 copies of the Ant Colony System using the reset method were created and 5 copies of the Ant Colony System using the distance-based method were created. All copies had the same set of changing nodes, but would store their pheromone and best-so-far tour values. Because of this, it is only possible to compare data received from within test runs and not to other runs, since it is possible for randomization to create a different set of nodes with a higher or lower best tour length each time.

All copies were allowed to run for 1000 iterations with no changes to the set of valid nodes. This was to establish a baseline pheromone matrix from which changes could occur; otherwise the distance-based method would behave similar to the reset method for small iteration values. After 1000 iterations, all copies were run for an additional 9000 iterations under dynamic conditions.

During the dynamic runtime, all Ant Colony System copies best-so-far tour length from a change in the problem was recorded. This was averaged with all other copies using the identical modification method and over 9000 iterations, such that the data received is an average tour length found after a certain number of steps from a change in the problem.

The different pheromone modification methods are compared based on how they find near-optimal routes quickly after a change in the problem. A good method will not sacrifice solution quality while still recovering from a change quickly.

For the completely dynamic situation, where *s* and *f* are random throughout the runtime, 3 copies of the Ant Colony System using the reset method, 3 copies of the Ant Colony System using the distance-based method, and 3 copies of the Ant Colony System using the combined method were created.

Having multiple copies of the same method and running them over 9000 iterations accounts for the inherent randomness in the tour construction of agents. Although doing multiple runs of the same *s* and *f* combination will yield different tour distances each time, the same trends arise when comparing the reset method with the distance-based method for multiple runs of the same *s* and *f* combination.

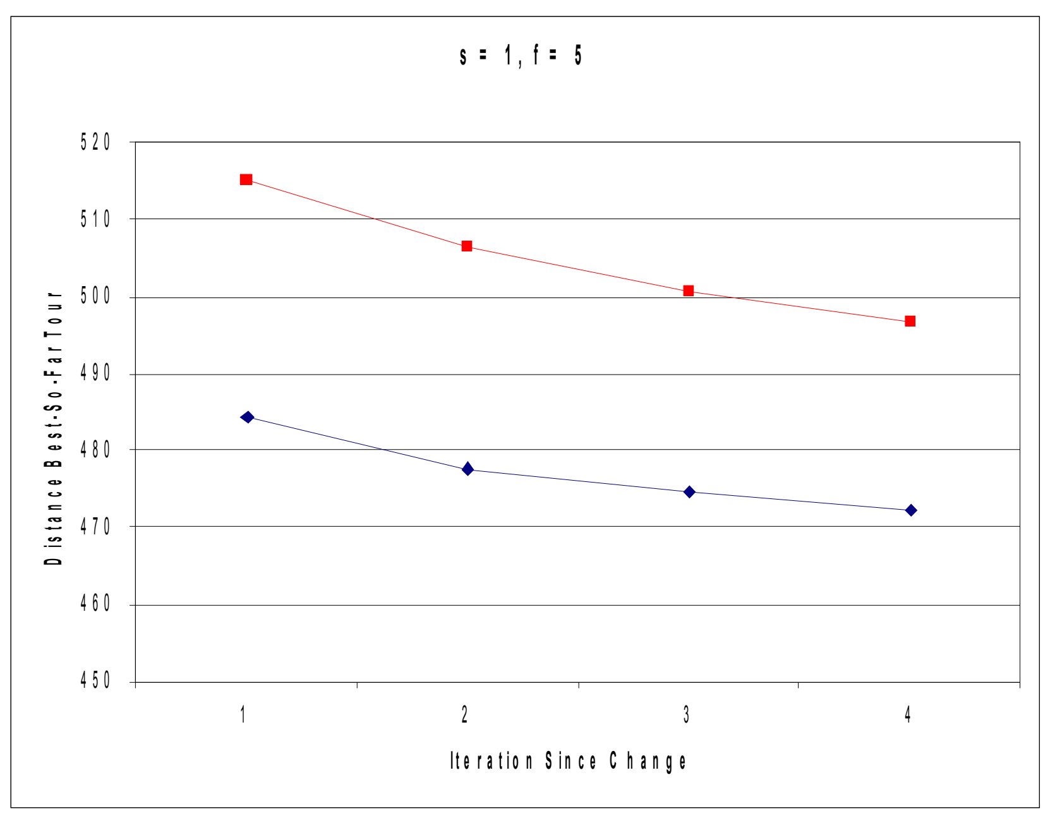
Data

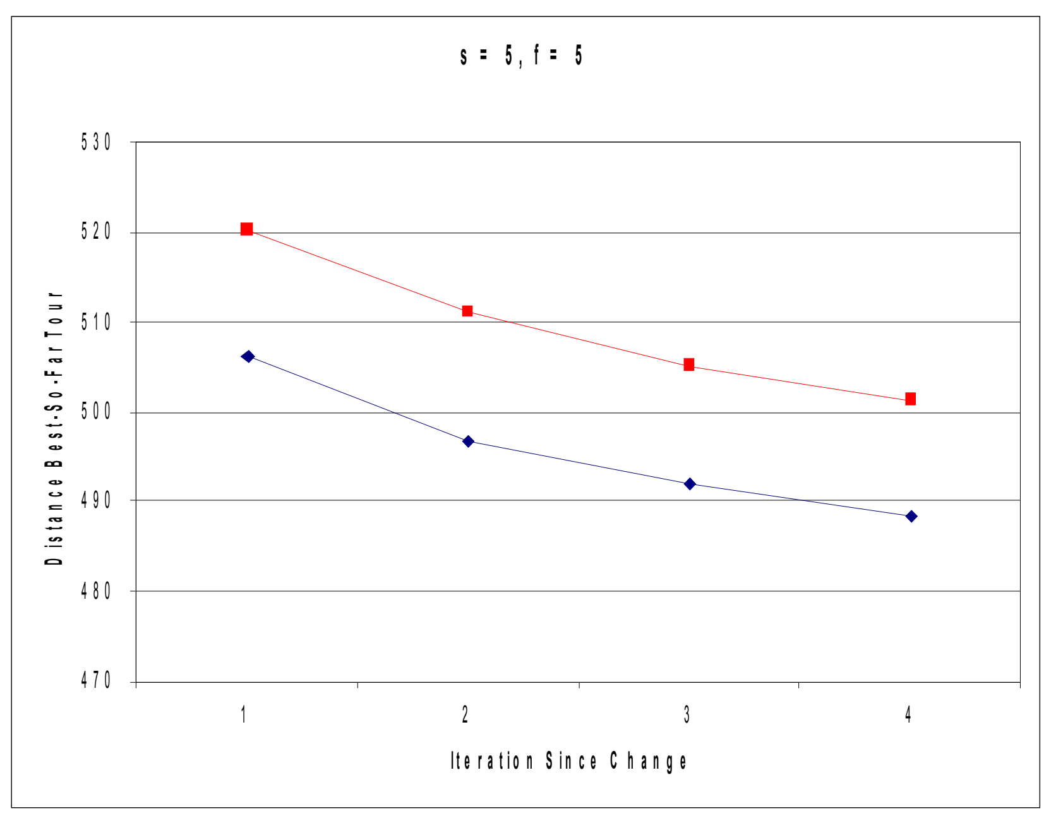
Comparison of Distance-Based and Reset Method Effectiveness for Different *f* and *s* Values

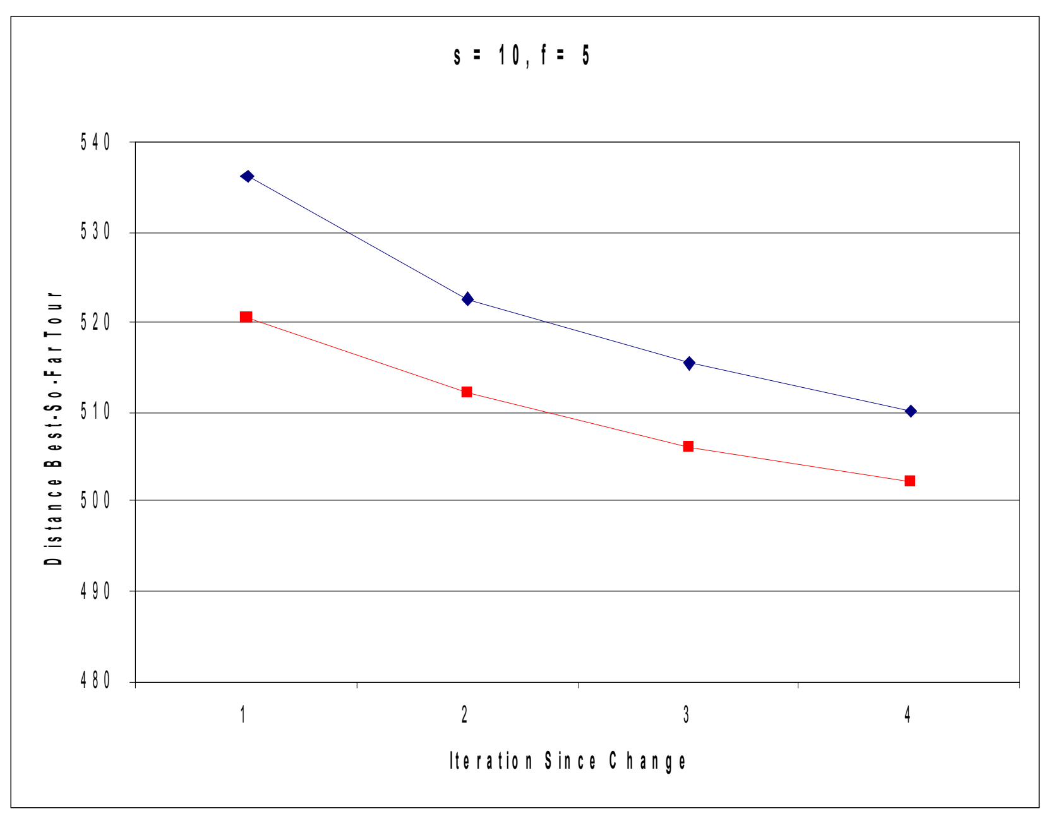
◆ Distance-Based
■ Reset

s = 1 | *s* = 5 | *s* = 10

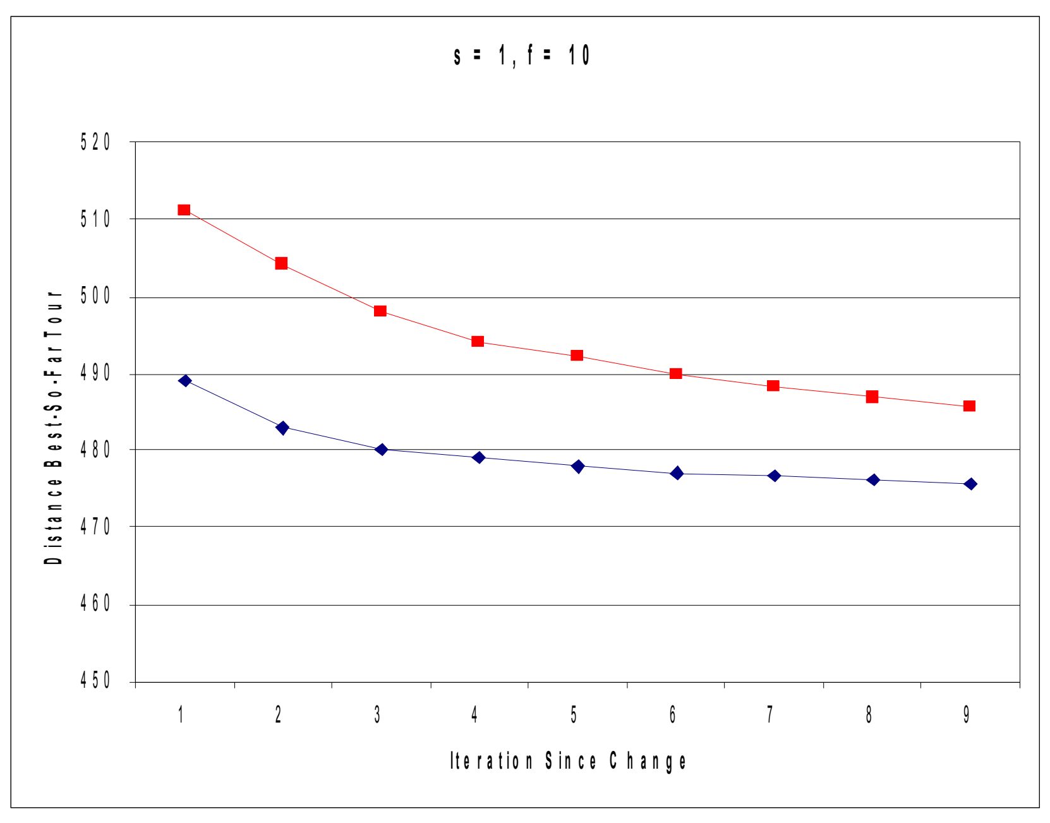
f = 5

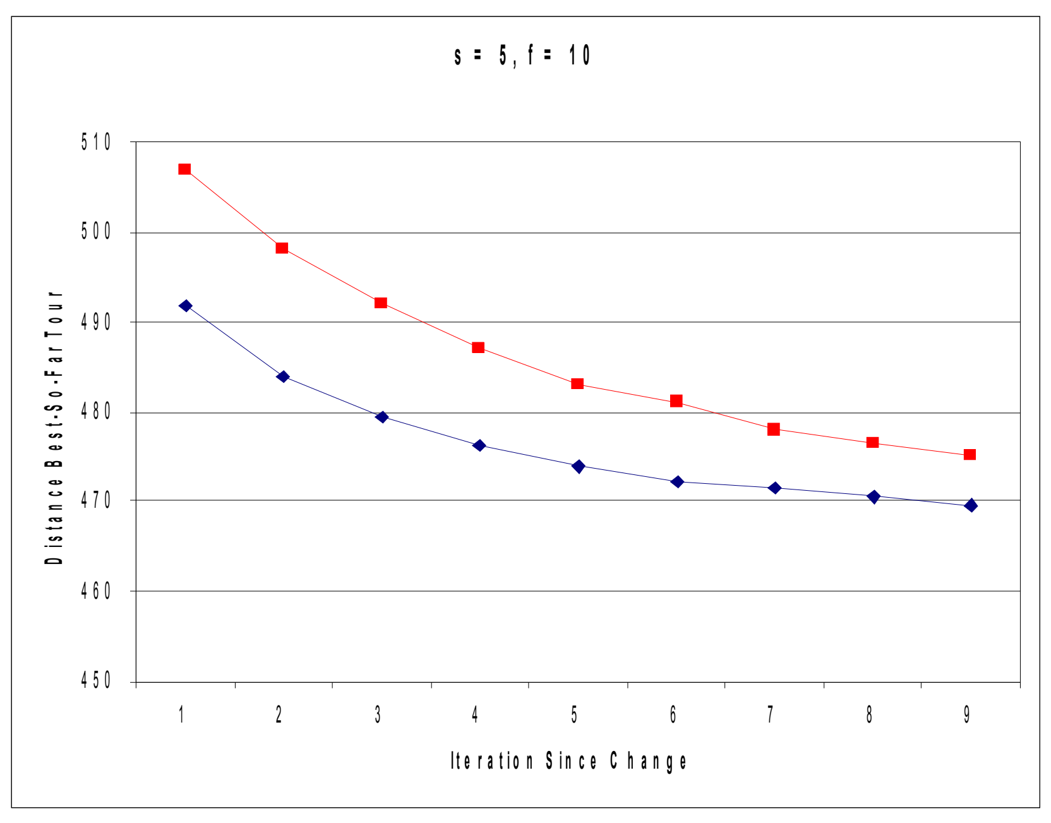


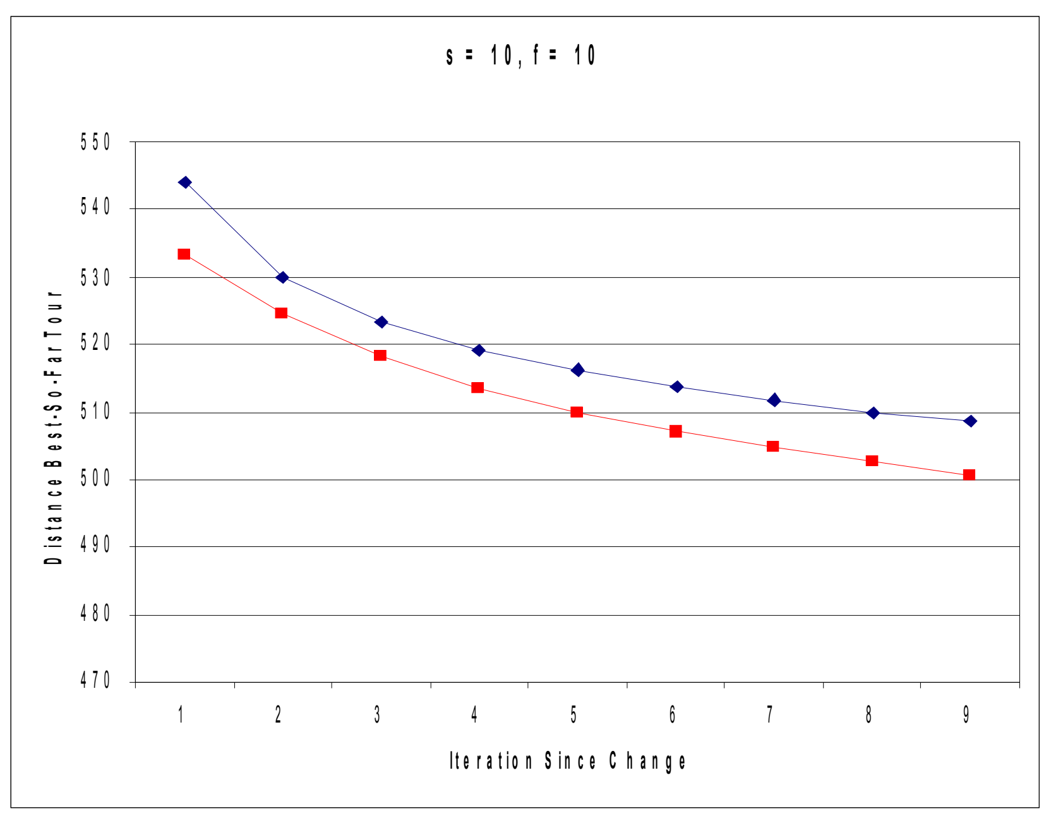




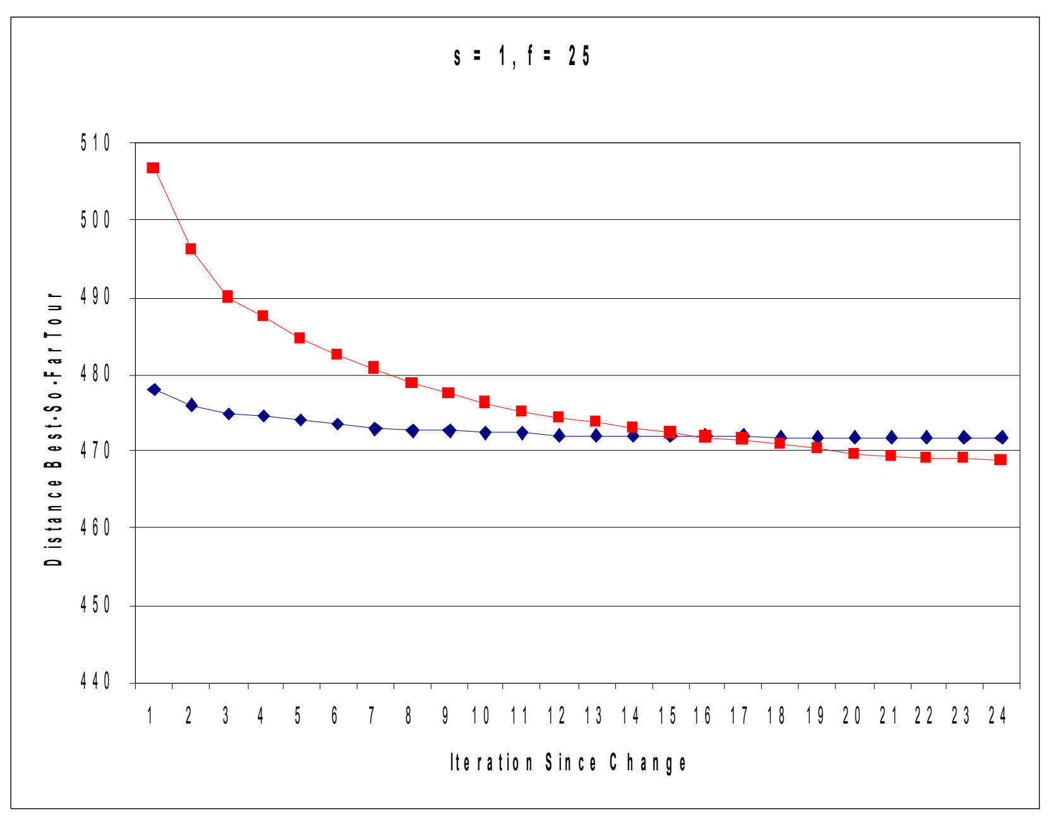
f = 10

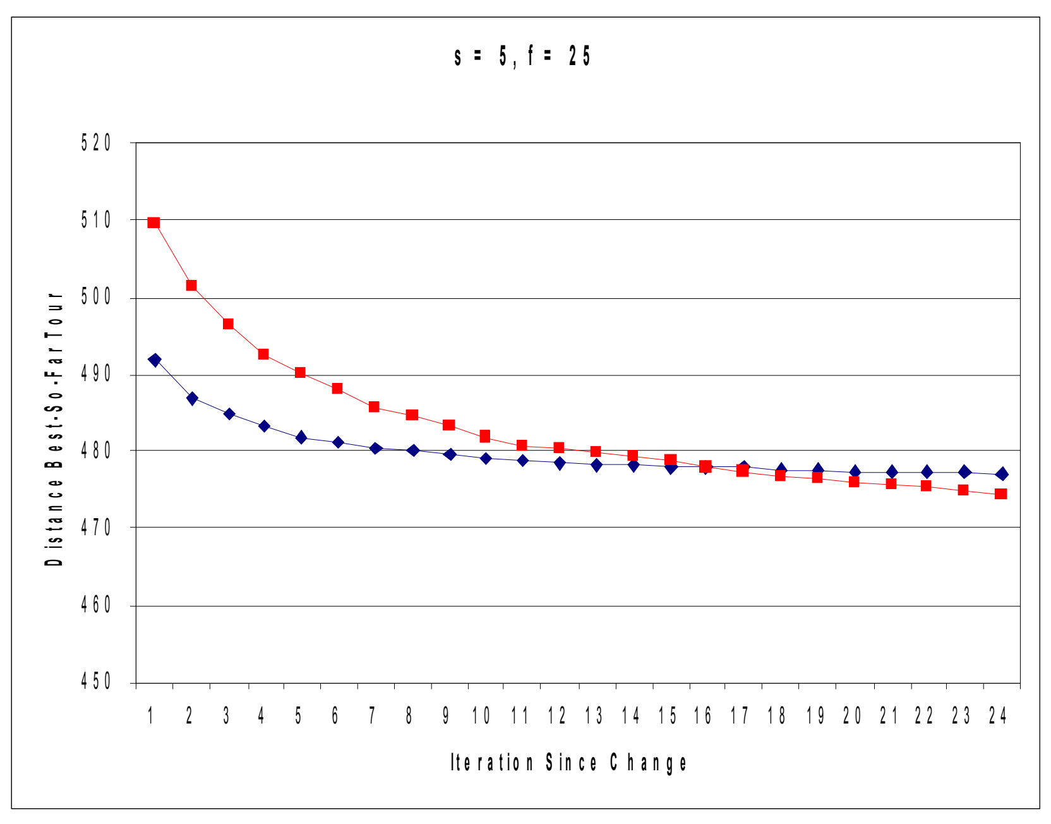


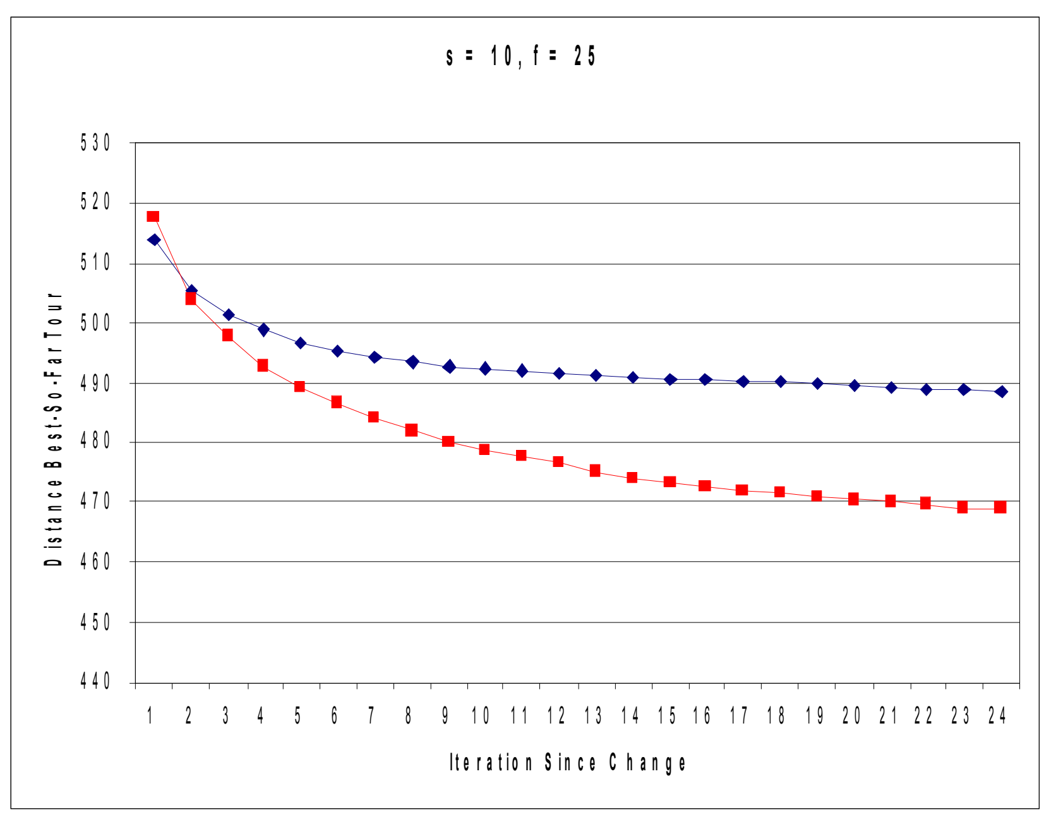




f = 25

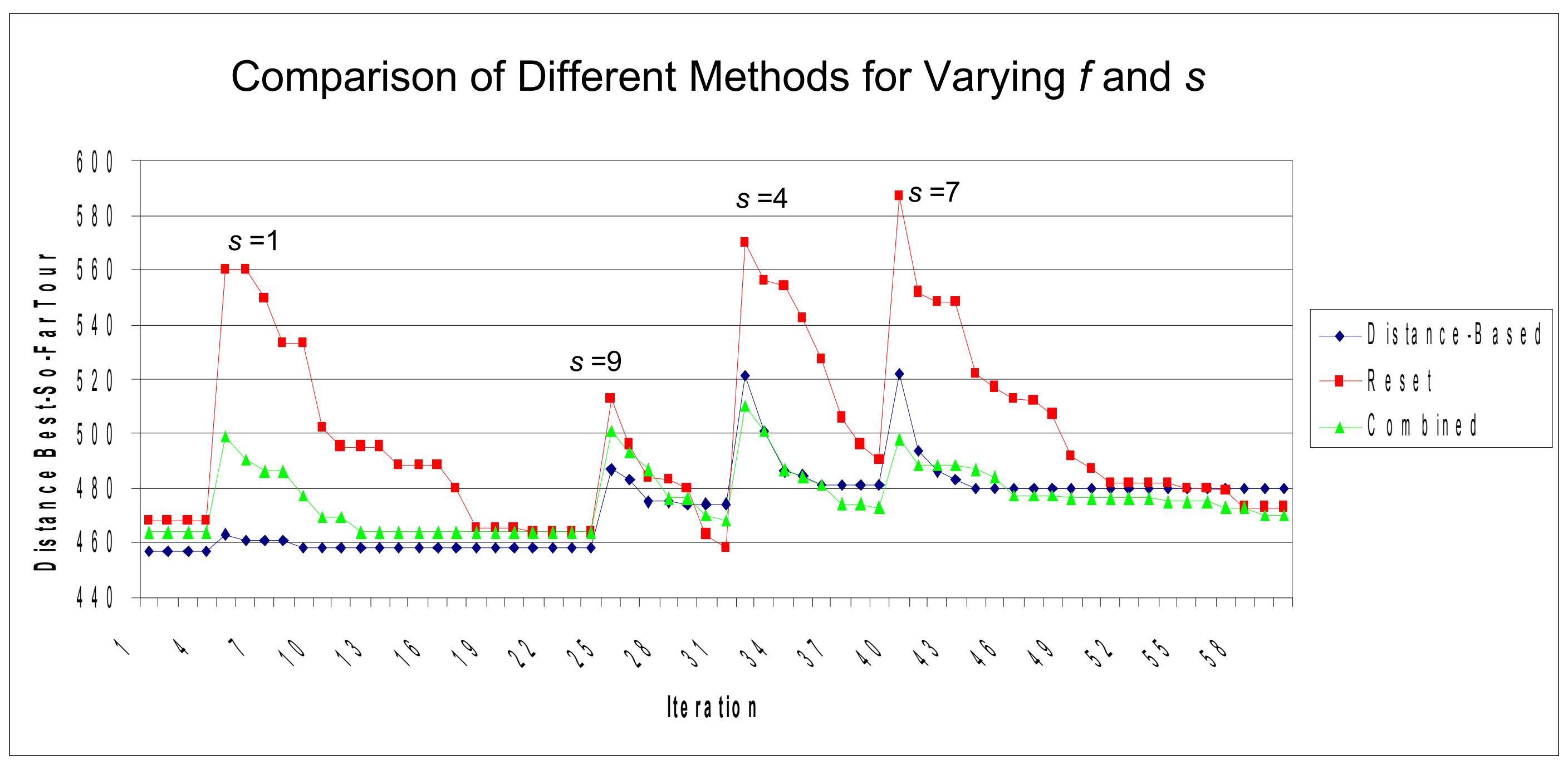






The above graphs pertain to the length of the best-so-far found tour after a certain number of iterations after a change for different combinations of *s* and *f*. In general, the reset method will outperform the distance-based method when the severity of the changes is high or when the frequency of the changes is low.

The graph to the right is data from 60 iterations of a completely dynamic situation, where the values of *s* and *f* change during the runtime of the algorithm. Even though the distance-based and the reset methods perform better for certain values of *f* and *s*, the developed combined method performs better on average over 9000 iterations of random *f* and *s* values.



Purpose

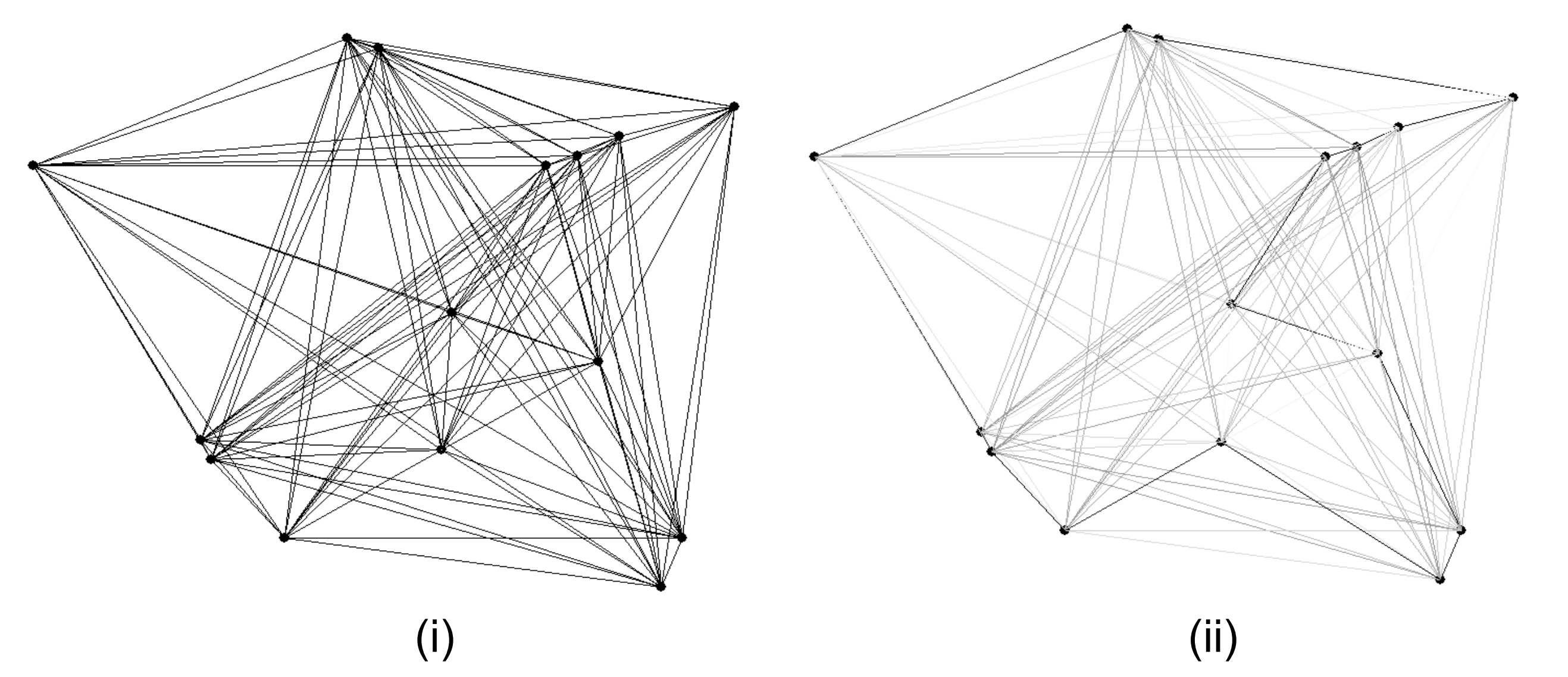
How can pheromone modification during the dynamic Traveling Salesman Problem be optimized so that Ant Algorithms find near-optimal solutions quickly after changes in the problem have taken place?

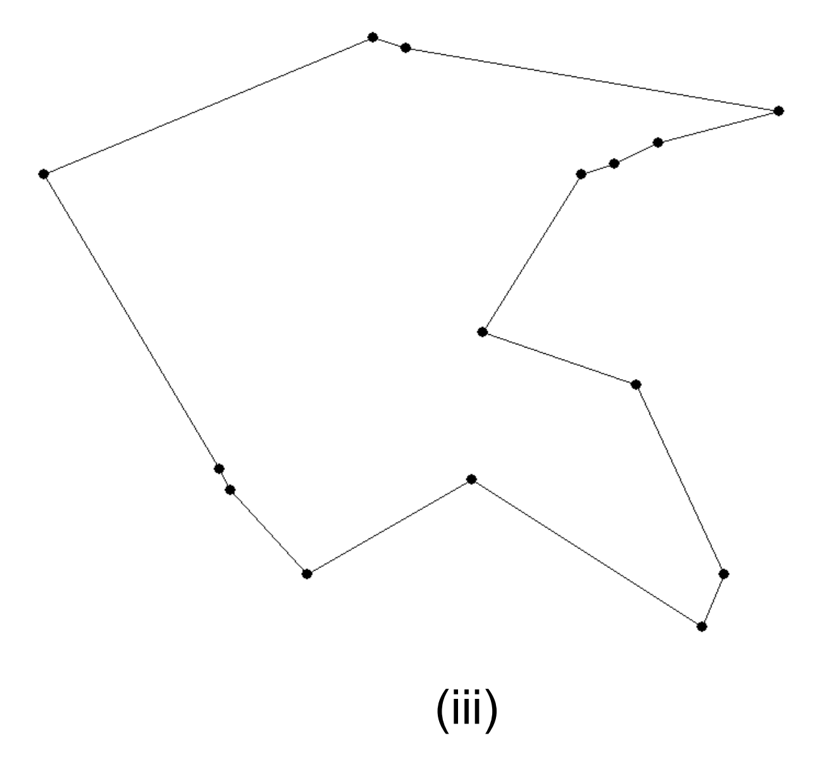
Background

Ant Colony Optimization (ACO) is a metaheuristic that is used to find near-optimal solutions to combinatorial optimization problems. It was originally inspired by the natural system that ants use to develop a short path between their colony and a food source. When ants find a food source and travel back to their colony, they leave behind pheromone, a hormone that other ants follow. ACO algorithms, also known as Ant Algorithms, mimic this system, using agents that independently solve a problem and then modify pheromone values in the problem to increase or decrease the probability of a solution area being searched again.

The Traveling Salesman Problem (TSP) is, formally: given a graph, find the tour that visits all the nodes with the least amount of cost. This research will be using Euclidean 2D dynamic TSP (dTSP), where the cost from moving to node i to node j is related to the distance between the nodes, and where the set of nodes in the graph changes over time.

The two main factors that agents use when constructing a tour are the cost of moving across an edge (in this case, the distance between nodes) and the pheromone value associated with an edge (this changes during runtime). The cost of an edge represents known information about the problem, while the pheromone represents learned information about the problem. In a dTSP, the set of nodes in the problem changes, so learned information, pheromone, may become obsolete based on the location, frequency, and severity of the changes. It is important that pheromone is altered during a change in such a way that useful pheromone information is kept while obsolete pheromone information is reset.





Various images depicting pheromone in a 15 node TSP problem. At the beginning of the problem, the algorithm assigns an equal pheromone value to all moves (i). During runtime, the algorithm either increases or decreases pheromone on certain moves based on whether or not agents who use those moves generate good solutions (ii). The resulting solution is a near optimal tour of all the nodes (iii). It is important to note that while the moves leading to the solution are favored by a high pheromone amount, the pheromone amount of other moves does not approach zero. This is to insure continuous exploration of the solution space.

Algorithm

This research used a modified version of the Ant Colony System for dynamic problems. For a problem with n nodes, n agents are initialized to random starting locations. A cost matrix is initialized, where the cost associated with a move from node i to node j, $\eta_{ij,}$ is the inverse of the distance between the node i and j, $1/d_{ij}$. A pheromone matrix is also initialized, where the pheromone value for a move from i to j, $\tau_{ij,}$ is initially a constant $1/[n-1]$.

At each iteration, agents are asked to construct a tour. Tour construction is dictated by a tabu list that stores all the nodes already visited. The attractiveness of a move is given by $a_{ij} = [\tau_{ij}]^{\alpha}[d_{ij}]^{\beta}$, where α and β are constants giving the weight of cost and pheromone values. With probability q, the agent will select the move with the highest attractiveness, otherwise the agent will select a move m_{ij} based on the probability $a_{ij}/[\Sigma a_{ih}]$, where $h \in N$, the set of valid nodes. After a move m_{ij}, τ_{ij} is decreased by a constant percentage (4% of τ_{ij}).

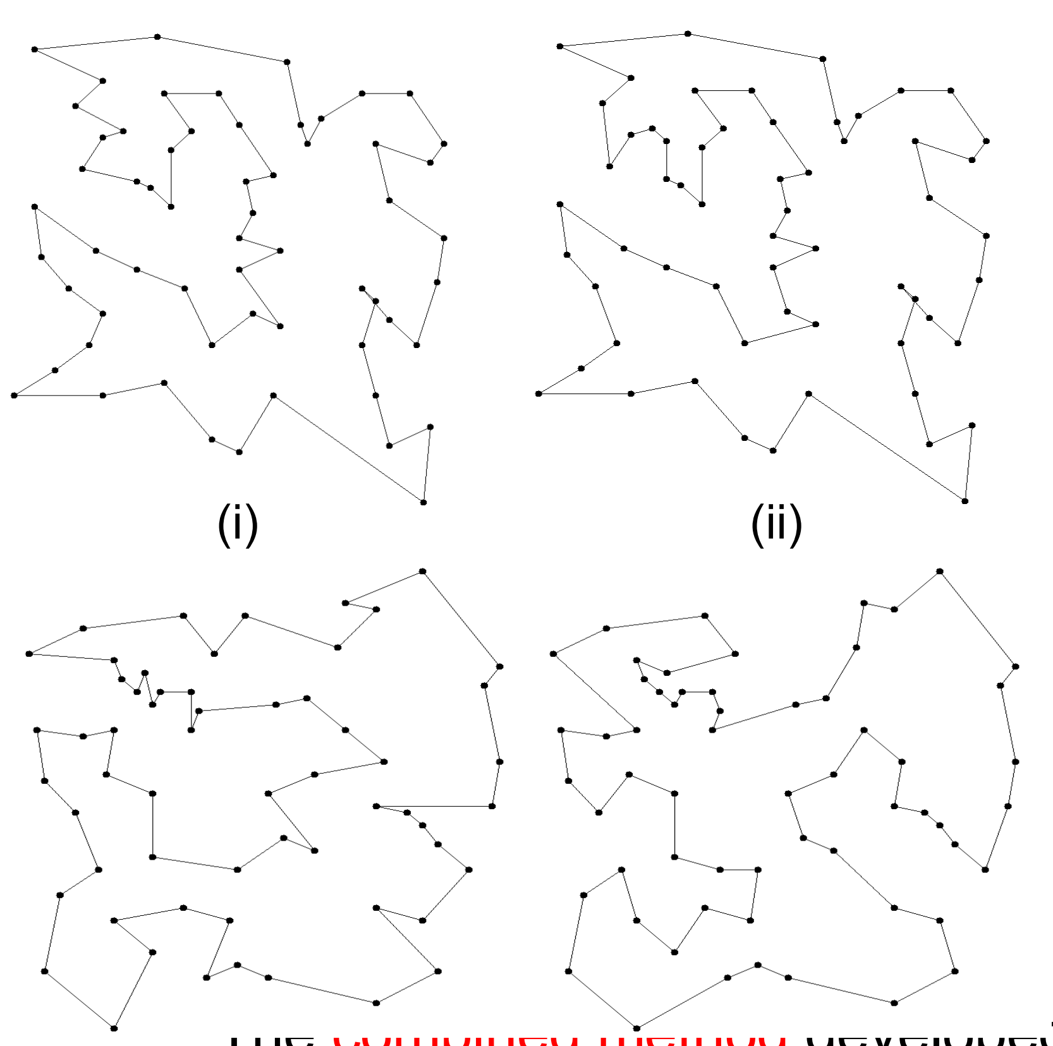
After an iteration, pheromone evaporates on all moves given by $\tau_{ij} \leftarrow \tau_{ij}(1-\rho)$, where ρ is a constant denoting pheromone evaporation. Pheromone along the best-so-far tour found is increased by $\tau_{ij} \leftarrow \tau_{ij} + .25\rho$.

Problem Construction

To construct a dTSP, a standard TSP data set was used. This research primarily uses the eil101 problem, a standard TSP, for testing. To make the problem dynamic, half the nodes are removed from the problem set during initialization, to create a set of valid nodes and a pool of invalid nodes. After f iterations, s nodes are removed from the valid nodes, and s nodes from the invalid pool are added to the problem.

The reset method assigns all pheromone values the same initial constant given by $\tau \leftarrow 1/[n-1]$. This effectively drops all learned information about the problem.

The distance-based method resets pheromone values from a node according to that node's distance to the closest changed node. Each node is assigned a reset value, r, such that $r_i = [\eta_{max} - \eta_{io}]/\eta_{max}$, where o is the closest changed node. Pheromone values are then reset such that $\tau_{ij} \leftarrow (1-r_{avg})\tau_{ij} + r_{avg} * 1/[n-1]$, where $r_{avg} = [r_i + r_j]/2$.



Solutions found 10 iterations after a 1 node change depicting the difference between the distance-based and reset methods. The best tour found before the change (i) is very similar to the best tour found after the change for the distance-based method(ii). This is because pheromone information is preserved throughout a change based on a particular nodes distance from the changes, increasing the likelihood that the new solution will contain elements of the old one.

On the other hand, the best tour found before the change for the reset method (iii) is radically different from the best tour found after the change (iv). Pheromone information is completely reset, so the algorithm searches the entire solution space equally after the change, often arriving at a different solution area.

The combined method developed for dynamic f and s values uses f and s to determine the relative weight, w, of how much each method should be applied. A high w value indicates a situation where the reset method would result in a better solution over the distance-based method, such that $w = [f / f_{max}] * [s / s_{max}]$, where f_{max} and s_{max} are both the maximum attainable values of f and s for the problem. The effect of the pheromone modification using the reset method is multiplied by w, while the effect of the distance-based method is multiplied by $[1-w]$.

Conclusion

After testing, it was determined that the distance-based method performed better than the reset method for small values of f and s, while the reset method outperformed the distance-based method for large values of f and s. When changes to the problem are small, preserving pheromone information is useful, since the solution to the new set is likely to share moves with the old solution. Large changes in the problem cause the solution to change radically, so preserving pheromone information is harmful, since it can mislead the algorithm initially and cause the algorithm to be stuck in a sub-optimal local minimum.

When changes to the problem are frequent, the reset method does not have enough time to converge on an answer. But, since the distance-based method saves pheromone information, it converges on a solution area faster, therefore being well suited for problems with frequent changes. In general, s is a more important variable for deciding which method performs better than f, unless f is a very small value (f<10 for eil101).

For the completely dynamic situation, the developed combined method outperformed either method separately. This is because it uses a different ratio of the reset and distance-based methods to utilize their strengths for particular s values.

Application

Dynamic Traveling Salesman Problems can be used to model many industrial problems, such finding optimal delivery routes, where customers are added and removed from the route, and computer network data flow, where servers go on and off line and ping time changes between servers depending on traffic.

TSP is a particular problem in the set of combinatorial optimization problems. Other combinatorial optimization problems include the dynamic Job-Shop Problem, where the amount of jobs done in a factory must be maximized while machines are either active or down for maintenance, and the Quadratic Assignment Problem, where factory locations must be optimized to minimize the transportation of goods between factories.