ANT COLONY OPTIMIZATION BASED SIMULATION OF 3D AUTOMATIC HOSE/PIPE ROUTING

Gishantha I.F. Thantulage

A thesis submitted for the degree of Doctor of Philosophy

School of Engineering and Design
Brunel University, UK
March 2009

ABSTRACT

This thesis focuses on applying one of the rapidly growing non-deterministic optimization algorithms, the ant colony algorithm, for simulating automatic hose/pipe routing with several conflicting objectives. Within the thesis, methods have been developed and applied to single objective hose routing, multi-objective hose routing and multi-hose routing.

The use of simulation and optimization in engineering design has been widely applied in all fields of engineering as the computational capabilities of computers has increased and improved. As a result of this, the application of non-deterministic optimization techniques such as genetic algorithms, simulated annealing algorithms, ant colony algorithms, etc. has increased dramatically resulting in vast improvements in the design process.

Initially, two versions of ant colony algorithms have been developed based on, respectively, a random network and a grid network for a single objective (minimizing the length of the hoses) and avoiding obstacles in the CAD model.

While applying ant colony algorithms for the simulation of hose routing, two modifications have been proposed for reducing the size of the search space and avoiding the stagnation problem.

Hose routing problems often consist of several conflicting or trade-off objectives. In classical approaches, in many cases, multiple objectives are aggregated into one single objective function and optimization is then treated as a single-objective optimization problem. In this thesis two versions of ant colony algorithms are presented for multi-hose routing with two conflicting objectives: minimizing the total length of the hoses and maximizing the total shared length (bundle length). In this case the two objectives are aggregated into a single objective.

The current state-of-the-art approach for handling multi-objective design problems is to employ the concept of Pareto optimality. Within this thesis a new Pareto-based general-purpose ant colony algorithm (PSACO) is proposed and applied to a multi-objective hose routing problem that consists of the following objectives: total length of the hoses between the start and the end locations, number of bends, and angles of bends. The proposed method is capable of handling any number of objectives and uses a single pheromone matrix for all the objectives. The domination concept is used for updating the pheromone matrix. Among the currently available multi-objective ant colony optimization (MOACO) algorithms, P-ACO generates very good solutions in the central part of the Pareto front and hence the proposed algorithm is compared with P-ACO. A new term is added to the random proportional rule of both of the algorithms (PSACO and P-ACO) to attract ants towards edges that make angles close to the pre-specified angles of bends. A refinement algorithm is also suggested for searching an acceptable solution after the completion of searching the entire search space.

For all of the simulations, the STL format (tessellated format) for the obstacles is used in the algorithm instead of the original shapes of the obstacles. This STL format is passed to the C++ library RAPID for collision detection. As a result of using this format, the algorithms can handle freeform obstacles and the algorithms are not restricted to a particular software package.

Keywords: Multi-objective hose routing, Ant system, Tessellated format, Freeform CAD geometries, P-ACO, PSACO, MOACO, Multi-objective ant colony optimization, Pareto strength ant colony algorithms, Domination, Refining, Collision detection, RAPID, Multi-hose routing, Multi-colony ant system, Shared paths, Bundling, Foreign pheromone

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Dr. Tatiana Kalganova for the continuous support of my PhD study and research, for her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank internal and external examiners of my thesis: Prof. John Stonham, Prof. Malcolm Irving and Dr. Alex Freitas for their insightful comments, and hard questions.

I would also like to thank Department of Electronic & Computer Engineering, School of Engineering and Design of Brunel University and President's Fund of Sri Lanka for the award of scholarships, which have supported me during the period of my PhD research.

Further, I would like to express my sincere appreciation to my employer Vice Chancellor, University of Sri Jayewardenepura, Sri Lanka for granting me the required study leave during the period of this work.

Finally, I would like to thank my wife, Amanda whose constant encouragement and love I have relied throughout the period of my research work.

Gishantha IF Thantulage

TABLE OF CONTENTS

1 INTRODUCTION

This chapter introduces the thesis, emphasizing its objectives and its contributions to the fields of Automatic Hose/Pipe Routing and Ant Colony Optimization. The chapter also presents the outline of the thesis, and of published and planned papers based on the thesis.

1.1 Subject Matter

In the past few years nature-inspired techniques have been widely used for various optimization problems in design, planning, scheduling, communication, etc.. One field, which is receiving increasing interest from several researchers, is the automatic hose/pipe routing of electrical and hydraulic equipment.

A variety of deterministic and non-deterministic (probabilistic) algorithms have previously been applied to hose and pipe routing. The deterministic algorithms guarantee the same solution at different runs with the same parameter values, and the non-deterministic algorithms such as genetic algorithms and simulated annealing generate different solutions due to their randomness (see Chapter **2** for details of previously applied approaches).

Among the non-deterministic algorithms, ant colony algorithms are increasingly being used in various real-world applications such as the travelling salesman problem (TSP) [27, 28, 29], the quadratic assignment problem (QAP) [30], the Job Shop Scheduling Problem (JSP) [27], telecommunication routing and load balancing [58], etc. and it has been shown that they perform well compared to other non-deterministic algorithms such as genetic algorithms, simulated annealing, etc [26].

1.2 Scope of the Thesis and Motivations

Hose and harness routing is a significant research area in assembly design. Many CAD and solid-model manufacturers incorporate the ability to represent these components in their products. However, the programs available are not always able to produce efficient routing. Often, skilled personnel who understand the engineering requirements, the model representations and physical production issues fill this technical gap. This requires human intervention to create assemblies and as CAD design tools allow rapid design and redesign of products at speeds that exceed the current human capacity, hose and harness routing cannot be done efficiently. There is an unacceptable bottleneck in meeting the customer's demand when bringing products to the market. Hence, companies are required to create timely innovative products to satisfy their customers' demands and compete with other companies.

Hose routing is a technique of developing collision-free routes for hoses between two locations in a 3D environment that contains obstacles. It needs to be done under multiple objectives and constraints. For example, hose routing must take into account: selection from a pre-specified catalogue of angles for bends, avoiding collisions, minimizing the total length of the hoses, minimizing the total number of bends, etc. For a human, this type of work is very tedious and time consuming. Hose routing problems are highly non-linear and discontinuous. The problem can be resolved, if an automatic approach for suggesting the possible routing paths is adopted.

In this thesis, ant colony optimization based algorithms have been applied to simulate automatic hose/pipe routing. The search space is represented as a network of paths in the free space of the CAD model where the automatic hose/pipe routing needs to be done. It has been shown that when applied to routing in networks, ant colony algorithms give better results in terms of quality of results and computation time compared to other non-deterministic algorithms [58]. Further, graph-based deterministic algorithms (Breadth First Search, Depth First Search, Best First Search, A* algorithm, Dijstraka algorithm, etc.) are not suitable for multi-objective optimization problems when all the objectives are equally important. If these algorithms need to be run on a multi-objective optimization problem, the objectives need to be converted into a single objective (e.g. weighted sum of objectives) or priorities need to be assigned to the objectives (Goal programming) [63]. The major weakness of these algorithms is the inability to produce solutions that are Pareto efficient.

1.3 Thesis Achievements and Contributions

The significance of this research lies in the potential of the developed ant colony optimization algorithms for automatic hose/pipe routing, removing human intervention and the associated cost and time. Further, despite the recent advancements in the field of automatic routing algorithms, the following issues have not been addressed by other researchers (See Chapter **2** for details of previously applied approaches).

- None of the existing approaches have used Pareto-based ant colony optimization to obtain the optimum layout of hoses/pipes.
- Most of the algorithms are restricted to basic shaped obstacles (e.g., rectangles) for the routing domain.
- All of the algorithms use a classical multi-objective technique: weighted sum approach or single-objective function to obtain the optimum solution. Since there is more than one optimum solution for a multi-objective problem, previous algorithms are not capable of generating these conflicting optimum solutions (or trade-off solutions) in a single run. If all the trade-off solutions need to be found, these algorithms must be run several times with different parameter values. Normally these experiments are time consuming and if they need to be run several times, they are even more time consuming.
- Moreover, multi-hose routing is another important part of this research area and none of the algorithms are able to route multiple hoses/pipes in parallel.

In this thesis, the above issues are addressed. In addition to these issues, the thesis suggests some modifications to ant colony optimization and a new Pareto strength ant colony optimization algorithm for multi-objective optimization problems.

The main contributions of the thesis can be stated as follows:

First Contribution

Initially, an ant colony algorithm is developed for automatic hose routing with a single objective (minimizing the length of the pipes) (with the intention of extending it to multi-objectives) and avoiding free-form obstacles. For the avoidance of free-form obstacles, the current state of the art for collision detection, the STL (STereoLithography) format for the obstacles and the C++ collision detection library RAPID, have been used. The ant colony algorithm is based on networks generated using grid points and random points in the free space of the CAD model.

While developing the ant colony algorithms for automatic hose/pipe routing, two modifications have been proposed to improve the ant colony algorithms.

- The first modification reduces the size of the search space for the ant colony algorithms using the current best path of each generation.
- One of the major problems in non-deterministic algorithms is that these algorithms converge to local optimum solutions. This is called the stagnation problem. The second modification has been introduced to avoid the stagnation problem in ant colony algorithms.

Second Contribution

In a multi-objective optimization problem, there is more than one optimum solution that optimizes the given objectives. In Pareto strength optimization methods, different conflicting optimum solutions (or trade-off solutions) can be obtained within a single run. A new Pareto strength ant colony optimization algorithm (PSACO) is proposed and has been applied to multi-objective hose routing in 3D space. The algorithm updates the pheromones according to the domination concept introduced in SPEA2 [55]. A single pheromone matrix is used for all of the objectives in the problem.

This algorithm (PSACO) also uses a network to find the optimum path between two points. The network is created in the free space of the 3D space using randomly generated points. The STL format of the original obstacle shapes is used for collision detection between a hose pipe and the obstacles. The network is created before running the PSACO algorithm. Thus, the collision detection algorithm is not required while running each generation or cycle of PSACO as in the algorithm proposed in [11]. Further, as a result of using the STL format, PSACO can handle freeform obstacles and is not restricted to a particular CAD software package.

PSACO has been used to optimize three objectives: total length of the hoses between the start and end locations, number of bends, and angles of bends. Results are compared with the current best Pareto ant colony algorithm (P-ACO) [49].

Third Contribution

The ant colony algorithms introduced initially have also been extended to multi-path problems. Two versions of multi-colony ant systems are proposed for the multi-hose routing problem. In both versions, each colony of ants is required to search for an optimum path between two end points (or commodities). While each colony searches for optimum paths, it tries to maximize the use of other colonies' paths (sharing paths, or bundling) for easy handling of multiple paths. The first version uses a single pheromone matrix for all colonies, whilst the second version uses a pheromone matrix for each colony and a modified random proportional rule to attract ants towards foreign pheromones.

The multi-path problem has a wide area of applications such as hose/pipe harness, electrical and hydraulic wiring. When harnessing multiple hoses in the electrical circuitry of a motor vehicle or other equipment, it is also important to have the hoses bundled as much as possible.

1.4 Outline of the Thesis

Chapter 2 deals with the literature review of automatic hose routing. **Chapter 3** discusses multi-objective optimization in some detail. **Chapter 4** presents ant colony algorithms and multi-objective ant colony optimization. **Chapter 5** discusses the simulation of automatic hose/pipe routing, the STL format (tessellated format), the C++ collision detection algorithm RAPID and basic steps required in automatic hose routing. It then discusses modifications to the ant colony algorithms, namely, multi-objective ant colony optimization, multi-objective hose routing with multi-objective ant colony optimization and multi-hose routing with multi-colony ant colony optimization algorithms. **Chapter 6** reports the results of the study and discusses these results. Finally, **Chapter 7** presents the conclusions and recommendations for future work.

1.5 Publications

Some results of this research have already been published in the peer-reviewed publications below and are included in Appendix A.

Journal Publications

1. Thantulage, G., Kalganova, T. & Wilson, M. (2006). Grid Based and Random Based Ant Colony Algorithms for Automatic Hose Routing in 3D Space. Transactions on Engineering, Computing and Technology, Volume 14, International Journal of Applied Science, Engineering and Technology (IJASET), Enformatika, ISBN 1503-5313, ISBN 975-00803-3-5, Aug., 2006. pp. 144 – 150.

Conference Publications

1. Thantulage, G., Kalganova, T. & Fernando, W.A.C. (2006). A Grid-based Ant Colony Algorithm for Automatic 3D Hose Routing. IEEE Congress on Evolutionary Computation, CEC 2006, Vancouver, Canada, Jul., 2006. pp. 48 – 55.

2. Thantulage G., T. Kalganova and M. Wilson (2006) “Grid Based and Random Based Ant Colony Algorithms for Automatic Hose Routing in 3D Space” Proc. of International Conference on Machine Intelligence (ICMI’2006).

2 LITERATURE REVIEW

This chapter describes various approaches implemented by other researchers for the solution of the problem of automatic routing of hoses, pipes and cables. Further, advantages and disadvantages, dimensions (2D or 3D), domains, obstacles, etc. of these algorithms are discussed.

2.1 Automatic Hose/Pipe Routing: An Introduction

Automatic hose and harness routing includes selecting at least a pair of connection points, including a start point and an end point, and determining a desired path between the start and the end points. In real-world applications, the best possible path needs to be obtained under multiple objectives and constraints. Furthermore, the hose routing method must include a validity check on the desired path in order to decide whether the desired path is valid. An example showing the desired path is illustrated in red in Fig. 2.1.

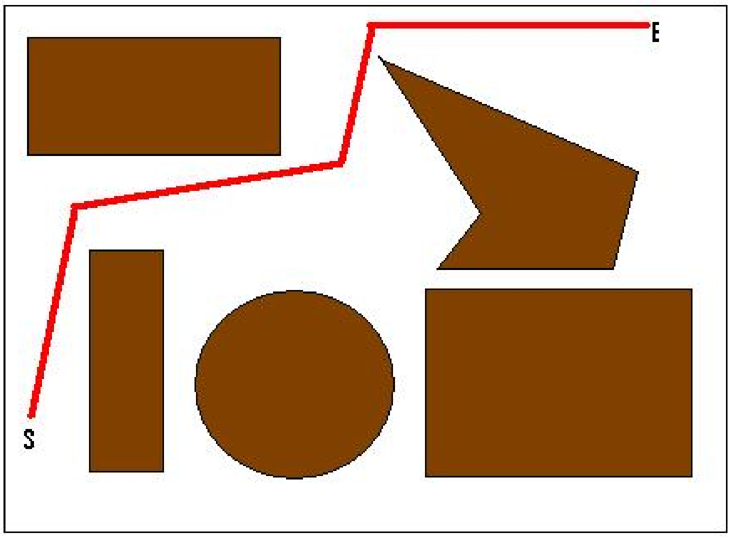


Fig. 2.1 Example: hose/pipe routing, where **S** is the start point and **E** is the end point

Previously, hose/pipe/cable routing was addressed by various approaches. These algorithms have been developed from a stationary 2D workspace and simple objects to a more complex 3D environment involving dynamic, multi-constraint and multi-objective problems. Methods for pipe routing can be traced back to techniques for robot path planning that have been traditionally classified into four major categories: the Skeleton Search (roadmap) [11], the Cell Decomposition approach [11, 20], the Potential Field method [11] and the Mathematical Programming method [11, 20].

The Skeleton approach involves capturing the set of feasible motions in a network of one dimensional lines and conducting a graph search of this network [21, 11]. The Cell Decomposition approach consists of decomposing the free space into cells and connecting the start and end configurations by a sequence of connected cells. In the Potential Field method [11], a scalar mathematical function is constructed whose value is a minimum when the robot is at the end configuration, and a maximum near the obstacles [22]. The path from the start to the end is determined by putting a small marble at the start and following its movement. The Mathematical Programming approach involves computing the path as a mathematical objective function and trying to minimize it while satisfying constraints (obstacle avoidance).

Mathematical Programming [11] techniques can be further classified into deterministic and non-deterministic (probabilistic) methods based on the search algorithms employed. Deterministic techniques guarantee the same solution for a problem when run at different times with the same starting solution, while non-deterministic techniques generate different solutions to the same problem at different runs due to the randomness involved in the solution process. Deterministic methods such as linear and nonlinear programming methods usually, in theory, find the optimal solution, but behave inefficiently with highly nonlinear and possibly discontinuous problems like pipe routing and often result in a local optimum. In contrast, non-deterministic algorithms [64], such as genetic algorithms, simulated annealing and ant colony algorithms, cannot guarantee to find the optimal solution, but are aimed at generating a set of globally good solutions (hopefully near-optimal). This feature is of practical relevance in engineering applications.

2.2 Previous Approaches in Hose/Pipe Routing

This section describes in some detail the various algorithms previously used in hose/pipe routing.

Zhu et al. [1] described a system for designing pipe layouts automatically using robot path planning techniques. This system, pipe routes are treated as paths left behind by rigid objects or robots (disc in the 2D case and ball in the 3D case). The Cell Decomposition approach described in [2, 3] is used to define the paths. As a result of this, the algorithm generates only orthogonal (Manhattan-style) routes. Initially the algorithm was developed for minimizing intrinsic factors such as pipe lengths and number of turns. Later, the algorithm was extended in order to make it capable of dealing with a variety of extrinsic factors such as location constraints and shape constraints. Location constraints specify preferred locations, undesirable locations, and forbidden locations for a pipe. For examples, a heat sensitive pipe should be kept sufficiently away from high-temperature equipment and a pipe should go as much as possible through existing pipe racks. Shape constraints apply to the shape of the pipe routes. For example, a drainage pipe should be non-ascending, a pipe should not have a vertical drop of more than d_{max} feet to avoid being over-stressed. The pipes are considered in sequence.

After having decomposed the free space, the algorithm generates the connectivity graph (*G*) representing the adjacency relation among the generated cells. A channel is constructed by searching the connectivity graph (*G*) for a path connecting the start and the goal nodes. The search of the connectivity graph (*G*) is performed by an A* algorithm guided by an admissible evaluation function $f(N) = g(N) + h(N)$ defined over the set of nodes in the connectivity graph (*G*). Here $g(N)$ is defined as the weighted sum of the length $l(N)$ of the path constructed so far and its number of turns $n(N)$. The function $h(N)$ is simply computed as the Manhattan distance between the centre of the current cell and of the terminal cell. Here N is the number of nodes in the connectivity graph.

A back-tracking algorithm is also implemented in the case where the search fails to produce a channel for the *k-th* route. The back-tracking algorithm adjusts previous routes generated to make room for the *k-th* route.

The location constraints are conceptualized as *virtual obstacles* and *virtual sinks*. A virtual obstacle can be *hard* or *soft*. Forbidden regions are protected by hard virtual obstacles which then act as real obstacles, while undesirable regions are protected by soft virtual obstacles which can be traversed by pipes, but at some additional cost. Virtual sinks are treated much in the same way as soft virtual obstacles, but with a bonus associated with the corresponding cells. The bonus only applies if these cells are traversed in some pre-specified directions.

In order to produce a channel containing paths satisfying the shape constraints, the A* search algorithm of the pipe router is modified so that the cost of including a cell in a channel depends on the channel generated so far.

This algorithm, however, was focused on piping layouts for power plants, chemical plants, etc., and dealt only with orthogonal (Manhattan-style) routes. Such an approach is unsuitable for the subtle, detailed, and highly optimized environment involving, for example, heavy-weight equipments such as bulldozers, cranes. Thantulage et al. [12, 13] have shown empirically, that the resolution of the cell decomposition plays an important part in the determination of the optimal route and it affects the computational time. If none of the cells falls on the optimal route during decomposition, the algorithm fails to obtain the optimal route. Thus, selecting the right resolution is important in the cell decomposition method.

Lee [15] proposed the Maze algorithm which generates the optimal path between two locations with no interference with obstacles. Mitsuta et al. [16] applied Lee's Maze algorithm to generate the optimal route for pipes. This algorithm also uses the cell decomposition approach to generate the paths as in [1]. However, this algorithm may require considerable computation time according to the amount of equipment and the number of pipes when applied to real-world applications.

Kim et al. [17] explored the possibility of automated industrial pipe-route design on three test problems defined in 2D space using stochastic hill-climbing, simulated annealing, and genetic algorithms. The algorithm takes into consideration the minimization of the total length of pipes and avoidance of obstacles. The problem is defined in terms of choices of Steiner points [18] and rectilinear connections. Results demonstrated that genetic algorithms are superior to either hill-climbing or simulated annealing for the problems tested. Obstacles were restricted to basic shapes such as rectangles.

A genetic algorithm approach to support interactive planning of a piping route path in plant layout design was presented by Ito [19]. The objective function is defined considering the total length of the pipes, how close they are to the obstacles, whether the route goes along the walls, and avoiding diagonal paths. The concept of spatial potential is used to quantify the degree of access to the wall or the obstacles. The algorithm was tested on 2D space and primitive shapes. The workspace is defined by using the cell decomposition approach. In [25], Ito et al. added a rule-based inference (RBI) engine for selecting a path from the results obtained by GA-based inspiration (GBI) [19, 25] as well as associative information. The RBI contains both outputs from the GBI and rules derived from experts' knowledge.

Kang et al. [23] proposed a method for generating the optimal route for pipes using a knowledge-based expert system called 'NEXPERT". The system is defined using experts' knowledge of the piping design of a ship. The knowledge consists of regulations of classification societies and port authorities, design practices and experience of experts. Three different knowledge-bases were constructed in this research storing 167 rules and 106 supporting methods, and were applied to the upper deck of a ship. However, this research is not of practicable interest since it is hard to define quantitatively all design knowledge and to maintain it when the practice of the designer for routing pipes changes.

Park, J.H. et al. [24] proposed the “cell generation” method for generating the optimal route for pipes which has the minimum length and least amount of bending and no interference between pipes, and applied the method to the engine room of a ship. To make interference checking easier, each obstacle is laid in a cubic box which is large enough to contain it. The objective function is defined taking into account the material and installation cost of the pipes, the maintenance cost according to the position of valves, etc. and then found the best solution from a tree of combinations of possible pipe routes.

Counru [4] solved the pipe routing problem from a cable harness routing perspective. He discusses the cable harness domain, defines the harness routing problem and presents an adaptive solution methodology based on a genetic algorithm. Part of the cable harness routing problem is to determine how well various candidate configurations map to the available routing space. As the bundle lengths and the cost of the harness cannot be determined until the routing has been done, it is necessary to try many different configurations to find the best. While it is possible to test all the configurations when the number of transitions is small, it becomes impractical as the number of configurations increases with the number of transitions.

The cable harness problem has been divided into three parts: ports, transitions and bundles. Ports connect the cable harness to the electrical assemblies. Each port is mapped to a graph node in the environment representation. Transitions (or junctions) form branching points among the bundles. A transition is represented as a graph node and three pointers to its adjacent nodes in the configuration. Bundles are defined as segments of the cable harness that connect any two adjacent nodes (ports or transitions).

The free space in which the harness is routed is abstracted by a sparse 3D graph. This graph shares the basic topology of the free-space using a small number of nodes. A graph with approximately 450 nodes was used for the experiments described in [4].

The cable harness problem is defined as follows:

Given a bi-directional graph G of N nodes ($N_1, N_2, \ldots, N_N$), P ports ($P_1, P_2, \ldots, P_P$) and a list of wires ($W_1, W_2, \ldots, W_W$), each connecting two ports (P_i, P_j), choose a subset, G' of the edges of G which minimizes an objective function based on the number of wires passing through each edge while ensuring that there is a connected path in G' for each of the wires. This problem is further constrained by precluding cycles in G' and requiring G' to be fully connected.

The objective function (representing the cost for wiring) is defined as follows:

$$Min \quad Cost = \sum_{i=1}^{\#Bundles} f_i(nWb_i) * g_i(Lb_i) \qquad 2.1$$

where nWb_i is the number of wires in bundle i
f_i is the unit cost of bundle i
Lb_i is the length of bundle i
g_i is the adjusted length of bundle i

Thus, finding the best cable harness routing boils down to finding the configuration that has the lowest cost when its transitions are placed in G. G' is then the union of all the links in the shortest paths between the end nodes of all the bundles.

In [4], the problem is divided into two procedures: *transition locator* and *configuration generator*. The first takes a configuration and returns the transition locations that produce the lowest cost routing. The second procedure intelligently uses the first procedure to prune through the possible configurations to find low cost routing.

The transition locator starts by routing each wire in the wiring list W in the configuration using Dijkstra’s shortest path algorithm [5] to determine the number of wires going through each bundle and, hence, its cost per unit length. Once these costs are defined, a standard genetic procedure is applied in the transition locator to locate the transitions.

The goal of the configuration generator is to intelligently create test configurations to find the one which maps well to the routing environment (minimizes the objective function shown in eq. 2.1). This procedure has two tasks: creating an initial population of test configurations and using fitness information of these test configurations to populate each subsequent generation. A valid configuration must connect all the ports to one another while having each transition connecting only three configuration nodes. At the end of each generation, the transition locator is applied to the individuals of the new population.

However, this algorithm assumes that the cable is composed of rigid segments and the number of segments can be adjusted by the user. Further, this algorithm is restricted to minimization of a single objective: the bundle length of the wires.

Cable harness design is also addressed in [6, 7, 8, 9]. Park et al. [6] proposed the use of agents to produce different cable configurations that satisfy the pin-to-pin connections of a typical harness circuit layout and automate routine operations such as moving a section of bundles from one position to another. The harness design system “Next-Link” proposed by Petrie et al. [7] creates different harness layouts concurrently. It is a management tool that uses software agents to coordinate, update and keep track of the work of individual designers, evaluating all the routings developed by each designer based on satisfying global constraints. Cerezuela et al.’s [8] study on cable harness design was carried out at a helicopter manufacturing company. From the case study, it is understood that harness design is an iterative process involving schematic, routing and component design and it is not possible to automate it completely by computers. Ng et al. [9] describe the effectiveness of immersive virtual reality for designing and routing cable harnesses by enhancing the expertise of the cable harness designer rather than replacing the individual by an automated system. The software tool described in this work assists users in performing cable routing in a virtual environment. The system was successfully tested in pilot trials. Recently, a route planning algorithm for cable and wire layouts in complex environments has been presented by Kabul et al. [10]. This algorithm pre-computes a global roadmap of the environment by using a variant of the probabilistic roadmap method (PRM) and performs constraint sampling near the contact space. The algorithm computes approximate paths using the initial roadmap generated on the contact space, given the initial and final configurations. The approximate paths are refined by performing constrained sampling and using adaptive forward dynamics. The algorithm takes into account geometric constraints like non-penetration and physical constraints like multi-body dynamics and joint limits. However, this algorithm does not guarantee physically accurate motion at all times.

Sandurkar et al. [11] proposed a non-deterministic optimization approach based on Genetic Algorithms (GAPRUS) to generate pipe routing solution sets. Objects are represented in tessellated format and it offers huge benefits in computation as well as usage. This approach can handle 3D free-form obstacles as the algorithm uses the tessellated format. This approach is applicable to any geometry that can be generated using commercial CAD packages.

The system accepts the tessellated file of the obstacles and the coordinates of the start and end locations. In addition to this, other supporting parameters related to the genetic algorithm such as the number of generations, population size, rate of mutation and cross-over, etc. can be set according to the problem.

When generating a solution, GAPRUS follows three steps:

1. An .STL file (tessellated format) of the CAD model is generated using a CAD package.
2. The .STL file is formatted into columns of vertices of facets suitable for the collision checking software and introduced into the iterative process of optimization.
3. Pipe routes generated by the optimizer are checked for collisions with the model and the results are fed back to the optimizer for the iterative search (of the genetic algorithm).

At the end of the iterative process, the system generates an .STL file containing the obstacle assembly and a set of pipe routes achieving objectives such as minimizing the total length between the specified start and goal locations and satisfying constraints on collisions. An optimal number of bends may also be generated as an output of this process.

A pipe route is modelled by the lengths of the pipes, their direction cosines and the angles of pipe bends. The design variables consist of the length of the pipes (d_i), one of the direction cosines of each pipe (m_i), the angles of bends between successive pipes (θ_i) and the number of bends (N).

The objective function plays a major role in controlling the problem and in emphasizing the objectives and constraints of the problem with respect to their relative importance. It consists of two parts: the objective part and the penalty part (see eq. 2.2).

$$Min \quad Z = \underbrace{W_1 * \left[1 - \frac{SE}{\sum_{i=1}^{N+1} d_i} \right] + W_2 * \left[\frac{N - N_{\min}}{N_{\max} - N_{\min}} \right]}_{Objective} + \underbrace{R * \left[\frac{N_c}{N_T} \right]}_{Penalty} \qquad 2.2$$

The objective part minimizes the length of the pipes and the number of bends. *SE* is the length of the straight line joining the start and end points. This is the absolute minimum length without considering the obstacles and is used to normalize the total length of the pipes. The number of bends (N) is normalized using N_{max} and N_{min} such that the ratio reaches unity when the number of bends is maximized and tends to zero when the number of bends is minimum.

The second part (penalty) indicates the constraints on collisions that are treated as a penalty to the objective function. This part is included in the objective function due to the fact that the GA determines the quality of the solution according to a fitness function which includes the degree of violation of constraints. The number of colliding triangles is taken to represent the degree of constraint violation. The penalty is defined as a ratio of the number of colliding triangles (N_C) to the total number of triangles in the model (N_T).

W_1 and W_2 are the weighting factors associated with the objectives. Since they indicate the relative importance being given to one objective over the other, their sum equals unity. The assigning of values to W_1 and W_2 expresses the designer's willingness in making tradeoffs between multiple objectives. R denotes the coefficient of penalty for the constraint violation term. A high value of R guides the solution away from the infeasible design space. The values are set by the user according to the application.

However, this approach was applied only to one model and took 18-19 hours to obtain the best layout of the pipes for a single run. Since stochastic algorithms such as GA usually need to be run several times to obtain a better solution, the computation time would be several days for running the algorithm several times. Furthermore, this algorithm needs to call the collision detection algorithm for each path found in each generation. As a result of this, the algorithm takes an additional computation time in each generation, in addition to searching the paths in the search space. Selection of the right weights (W_1 and W_2) and the penalty coefficient R is a critical factor of this algorithm.

Drumheller [14] proposed a method for generating the optimal route of pipes considering constraints relating to the pipe bending. This research focuses on the generation of a route of pipes having the minimum installation space in an aeroplane. This algorithm uses a weighted sum approach to evaluate the combined effect of the various constraints and objectives. Hence, a user can explore trade-offs between multiple optimal routes by running the algorithm with different sets of weights. The designer should provide an initial path from start and goal configurations, but it is not necessary that this path be a feasible solution. A heuristic approach is used to find the node distribution of a route.

Fan et al. [76] proposed an automatic ship pipe route design (SPRD) using ACO. The objective function is formed using the weighted sum of the objectives: avoiding obstacles, shortest length of path and number of bends. This research is similar to the proposed "Grid-based Ant Colony Algorithm for Automatic 3D Hose Routing” in this thesis (published in [12]). Each obstacle of the model [76] is simplified into a cuboid that is large enough to contain the obstacle. The grid-based algorithm proposed in this thesis, however, can handle free-form obstacles. Furthermore, the research in [76] has not used Pareto-based ACO to obtain the various trade-off solutions in a single run.

Ma et al. [77] propose a structural method for a genetic algorithm (GA) for the optimization problem of cable routing in which cables have to be laid optimally. The cable routing problem is defined as follows: there exists a network of cable trays that link equipment and facilities already set up. Cables are laid out in the tray network with facilities and equipment as points of origin and destination. Each tray has an allowed capacity for cables, and no more cables than that capacity can be laid in a tray. Given these constraint conditions, a route from the point of origin to the destination for each cable is selected, and a routing plan that minimizes the total routing length for all cables is found. Here, for the sake of simplicity, the types of cable and tray ratings are unified, and the constraints are deemed to be conditions related to the number of routes. A tray network is represented using a tray graph as a discrete graph, with the junctions (branching) and end points of the trays and the connecting points of equipment and devices as nodes, and the trays as edges. The proposed GA is a two-level hierarchical GA that uses chromosome coding involving two levels. In the first level GA, several good routes (shortest or approximately shortest routes) are found for each cable, and then in the second-level GA, the optimum combination of the good routes for each cable is found.

Liu et al. [78] proposed a method for pipe route design based on the grid method and particle swarm optimization (PSO). This paper adopts a fixed-length encoding mechanism based on grids and the following objectives and constraints are taken into consideration in defining the evaluation function: avoiding obstacles, the shortest length of path, the least number of bends and most pipes must go as much as possible through existing equipment.

The routing of pipes should meet various requirements [24], including a large number of rules of physical constrains, economic constraints, safety constraints, production constrains, flexibility constraints, and so on. The traditional design approaches pay more attention to physical constraints, and none or few other constraints can be satisfied. In free space methods, some non-physical constraints can be described by virtual solids and can be easely handled. But this brings another problem. The current AI algorithms for pipe-routing arrangements do not adapt to free spaces. Because all of the existing algorithms consider obstacles as the forbidden zone for the pipes, the routing must avoid such cells or area. When free space is used, the terminals of the routings may be in the free spaces. A routing must connect terminals of given locations and avoid obstacles. Huibiao et al. [79] proposed the Hanging bridge algorithm to generate a bridge cell to translate an inside connect terminal outside the free space, and found a solution for pipe-routing arrangements using free space models.

2.3 Summary of Chapter 2

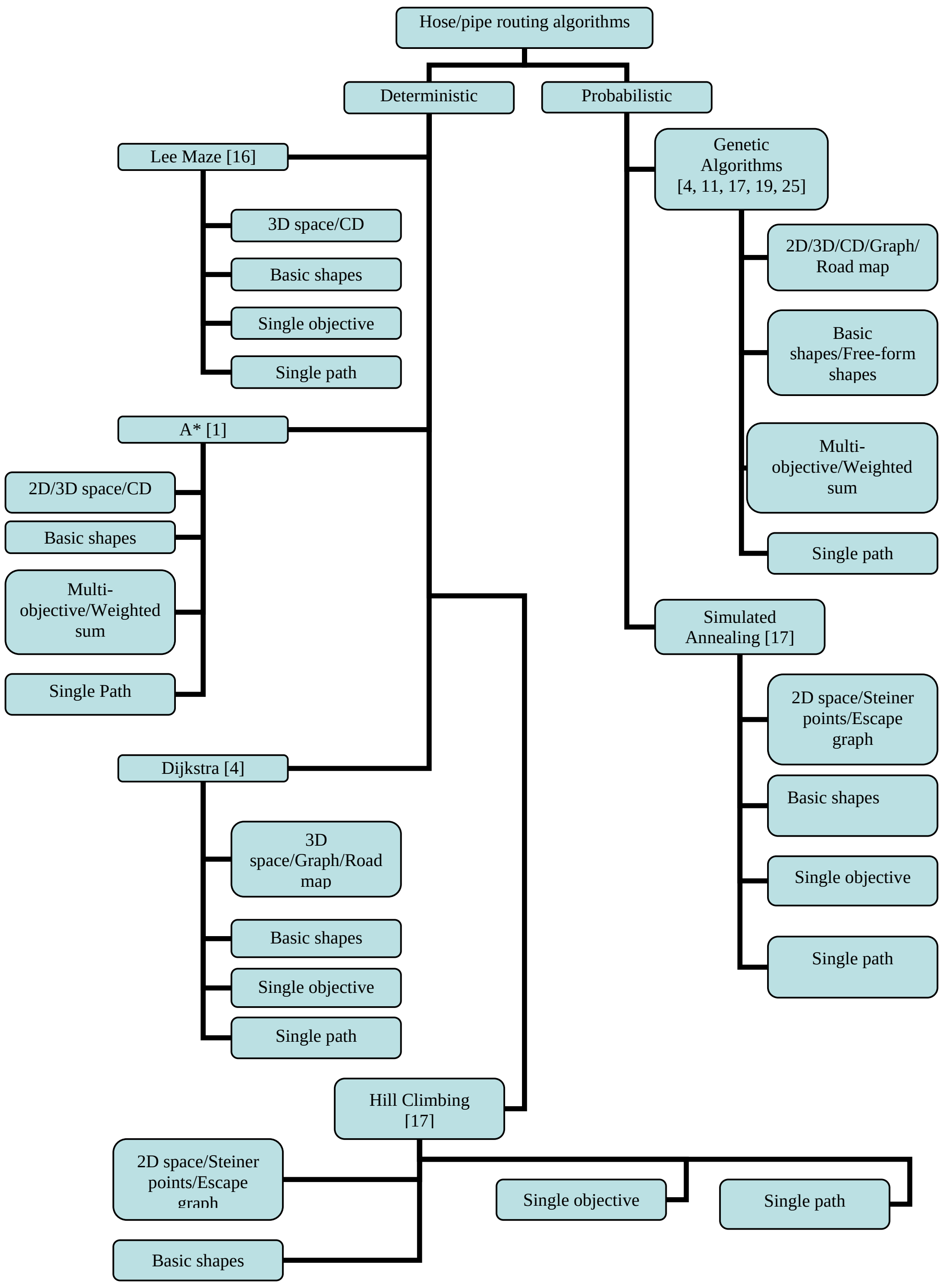
This chapter has reviewed the previous approaches to solving automatic hose routing and similar problems and Table 2.1 briefly summarises previous work done on hose/pipe/cable routing. The “knowledge-based” column states whether the algorithm is embedded or not with a knowledge-based (e.g. rule-based, etc.) system for selecting a path from the solution set. The last column shows the capability of the algorithm of producing multiple paths in parallel.

Fig. 2.2 shows the classification of previous hose/pipe routing algorithms.

According to earlier algorithms developed for hose/pipe routing, one may notice that, although the developed algorithms can route pipes according to multi-objective criteria and with minimum use of expert knowledge (genetic algorithms, etc.), the computational effectiveness of these algorithms is very low and in the real world they require many hours of computational time to achieve a satisfactory solution.

Previously it has been shown that ant colony algorithms outperform genetic algorithms in terms of computational time and quality of the solutions produced for a number of applications [58]. This thesis will look into the improvement of hose/pipe routing in terms of computational effort and the quality of the solutions by development of ant colony algorithms.

Further, all previous algorithms used a classical multi-objective technique: the weighted sum approach or a single-objective function to obtain the optimum solution. Since there is more than one optimum solution for a multi-objective problem, previous algorithms are not capable of generating these conflicting optimum solutions (or trade-off solutions) in a single run. If all the trade-off solutions need to be found, these algorithms must be run several times with different parameter values. Normally these experiments are time consuming and if they need to be run several times, they are even more time consuming. Moreover, multi-hose routing is another important part of this research area and none of these algorithms are able to route hoses/pipes in parallel.



CD – Cell decomposition

Fig. 2.2 The taxonomy of hose/pipe routing algorithms

TABLE 2.1 Previous work on hose/pipe/cable routing

Author(s)	**Year**	**Dimensions**	**Category**	**Domain**	**Obstacles**	**Algorithm/Objective function**	**Objectives/Constraints**	**Knowledge-based**	**Multi-path**
Mitsuta et al. [16]	1986	3D	CD	General	Basic shapes	Lee Maze/Single objective	Pipe lengths/Obstacle avoidance	Yes	No
Zhu et al. [1]	1991	2D/3D	CD	Power plants, Chemical plants, etc.	Basic shapes (Rectangles)	A*/Weighted Sum	Pipe lengths, No of turns/Location constraints, Shape constraints	No	No
Conru [4]	1994	3D	Graph/ Roadmap	Cable harness	Basic shapes	Dijkstra, GA/Single objective	Cable lengths/Obstacle avoidance	No	No
Kim et al. [17]	1996	2D	Steiner points/Escape graphs	Industrial plants	Basic shapes	HC, SA, GA/Single objective	Pipe lengths/Obstacle avoidance	No	No
Sandurkar et al. [11]	1998	3D	ND	General	Free-form	GA/Weighted Sum	Pipe lengths, No of bends/Select angles from catalogue angles, obstacle avoidance	No	No
Ito et al. [19, 25]	1998	2D	CD	General	Basic	GA/Weighted Sum	Pipe lengths/Closer to the obstacles, Go along the walls	Yes	No
Kang et al. [23]	1996	3D	NA	Ship building	NA	NA	Pipe lengths/Placing pipes away from onboard equipments	Yes	No
Drumheller [14]	2002	3D	Graph/ Roadmap	Aeroplane building	Free-form	Heuristic/Weighted Sum	Pipe lengths/Intrinsic and extrinsic factors	No	No
Park, J.H. et al. [24]	2004	3D	CG	Ship building	Free-form	TC	Material and installation cost of pipes, maintenance cost/No interference between pipes, Obstacle avoidance	No	No
Proposed Grid-based ACO [12]	2006	3D	CD	General	Free-form	ACO/Single	Pipe lengths/Obstacle avoidance	No	No
Proposed Random-based ACO [13]	2006	3D	Network	General	Free-form	ACO/Single	Pipe lengths/Obstacle avoidance	No	No
Fan et al. [76]	2006	3D	CD	Ship building	Cuboid	ACO/Weighted sum	Pipe lengths, no of bends/Obstacle avoidance	No	No

TABLE 2.1 Previous work on hose/pipe/cable routing (Contd.)

Author(s)	**Year**	**Dimensions**	**Category**	**Domain**	**Obstacles**	**Algorithm/Objective function**	**Objectives/Constraints**	**Knowledge-based**	**Multi-path**
Ma et al. [77]	2006	2D	Graph/ Roadmap	Cable Harness	NA	GA/Single	Cable lengths/capacity of trays	No	No
Liu et al. [78]	2008	2D	CD	General	Basic	PSO/Weighted sum	Pipe lengths, no of bends, most pipes go through existing elements/Obstacle avoidance	No	No
Huibiao et al. [79]	2008	2D/3D	CD	Ship building	Basic	Hanging Bridge Algorithm	Pipe length/Physical, economic, safety, etc.	No	No
Proposed MOACO	2009	3D	Network	General	Free-form	ACO/Pareto optimization	Pipe lengths, No of bends, Bend angles/Obstacle avoidance	No	No
Proposed MCAS-MHR-1 and MCAS-MHR-2	2009	3D	Network	General	Free-form	ACO/Weighted sum	Pipe lengths, shared path length (No limitations on no of objectives)/Obstacle avoidance	No	Yes

CD – Cell Decomposition, ND – Non Deterministic, GA – Genetic Algorithm, HC – Hill-Climbing, SA – Simulated Annealing, CG – Cell Generation, TC – Tree of Combination, ACO – Ant Colony Optimization, PSO – Particle Swarm Optimization. NA – Not Applicable

3 MULTI-OBJECTIVE OPTIMIZATION

The work reported in this thesis has extensively used multi-objective optimization for automatic hose/pipe routing. This chapter describes the multi-objective optimization in detail. It describes the multi-objective optimization problem (section 3.1), the domination concept (section 3.1.1), the non-dominated set (section 3.1.2) and the illustrative representation of the non-dominated set (section 3.2), how to compare the performances of two multi-objective optimization methods (section 3.3) and how to select a preferred solution from the non-dominated solutions (section 0).

A multi-objective optimization problem (MOOP) deals with more than one objective function. Many real-world problems require the simultaneous optimization of a number of objective functions. Some of these objectives may be in conflict with one another. In other words optimizing one of these objective functions makes it more difficult to optimize the other objective functions. For example, consider finding optimal routes in data communication networks, where the objectives may include minimizing routing cost, minimizing route length, minimizing congestion, and maximizing the use of physical infrastructure. There is an important trade-off between the last two objectives: minimization of congestion is achieved by reducing the utilization of links; a reduction in utilization, on the other hand, means that infrastructure, for which high installation and maintenance costs are incurred, is under-utilized [37].

There exist many classical algorithms and application case studies involving multiple objectives. The majority of these methods avoid the complexities involved in a true multi-objective optimization problem and transform multiple objectives into a single objective function by using some user-defined parameters (e.g., weighed sum method). Finally, these algorithms treat the multi-objective optimization as a single-objective optimization.

However, there is a fundamental difference between single and multi-objective optimization which is ignored when using a transformation method [38]. In a single-objective optimization problem, there exists only one optimal solution. In contrast, in problems with more than one conflicting objectives, there is no single optimum solution. There exists a number of solutions which are all optimal. The classical algorithms that convert a multi-objective optimization problem into a single-objective optimization problem converge to one of these optimal solutions. Thus, a classical approach needs to be run (with different parameter values) several times to obtain the set of all optimal solutions of a multi-objective optimization problem. A multi-objective optimization algorithm produces a set of solutions closer to the optimal solutions of the MOOP in a single run.

Although the fundamental difference between these two optimizations lies in the cardinality of the optimal set, from a practical standpoint a user needs only one solution, no matter whether the associated problem is a single- or a multi-objective problem [38]. In the case of multi-objective optimization, the higher-level of information (non-technical, qualitative and experience-driven) that is not incorporated into the model is taken into consideration for selecting a solution.

The principles of an ideal multi-objective optimization procedure are shown in Fig. 3.1. In step 1, multiple trade-off solutions are found. Thereafter, in step 2, a higher-level of information is used to choose one of the trade-off solutions.

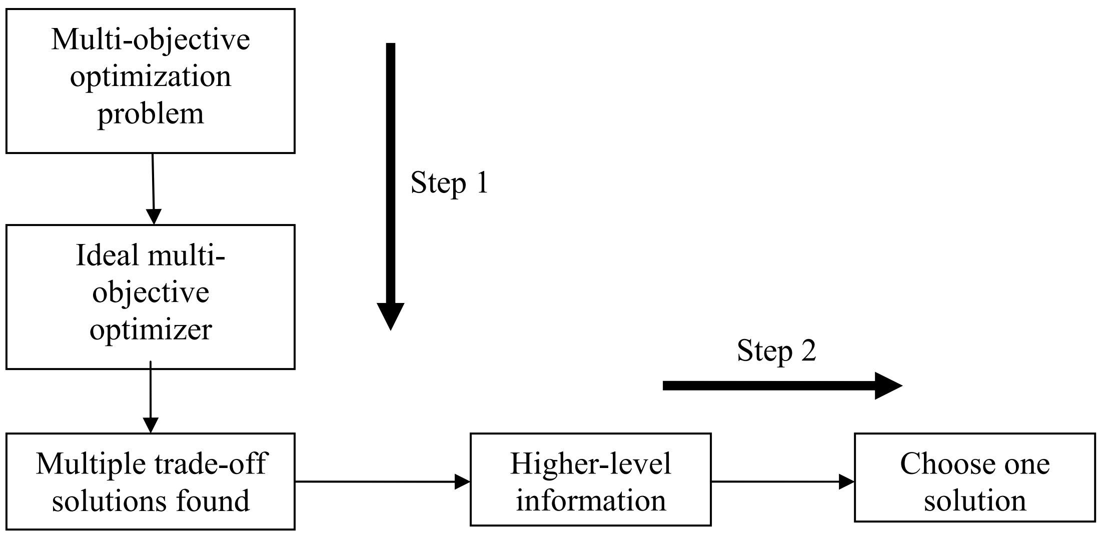

Multi-objective optimization problem
Step 1
Ideal multi-objective optimizer
Step 2
Multiple trade-off solutions found
Higher-level information
Choose one solution

Fig. 3.1 Flowchart of a multi-objective optimization procedure [38]

3.1 The Multi-Objective Optimization Problem

A multi-objective minimization problem with n parameters (also called decision variables) and M objectives can be stated as follows:

$$Min\ f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})),\ \mathbf{x} = (x_1, x_2, \ldots x_n) \in X \qquad 3.1$$

where $\mathbf{x}$ is a decision vector and X is the decision space.

One of the striking differences between single-objective and multi-objective optimization is that in multi-objective optimization the objective functions constitute a multi-dimensional space, in addition to the usual decision space. This additional space is called the *objective space*, Z [38]. For each solution $\mathbf{x}$ in the decision space, there exists a point in the objective space, denoted by $\mathbf{z} = \mathrm{f}(\mathbf{x}) = (z_1, z_2, \ldots, z_M)$, where $z_k = \mathrm{f}_k(\mathbf{x})$ and $k = 1, 2, \ldots, M$.

3.1.1 Domination Concept

Most multi-objective optimization algorithms use the concept of Pareto domination, hereafter called domination for short. Two solutions are compared on the basis of whether one dominates the other solution or not. A decision vector $\mathbf{x} \in \mathrm{X}$ dominates another $\mathbf{y} \in \mathrm{X}$ (denoted by $\mathbf{x} \succ \mathbf{y}$) if, and only if:

1. $\forall\ k = 1, 2, \ldots, M$, $\mathrm{f}_k(\mathbf{x}) \le \mathrm{f}_k(\mathbf{y})$ and
2. $\exists\ m = 1, 2, \ldots, M$ s.t. $\mathrm{f}_m(\mathbf{x}) < \mathrm{f}_m(\mathbf{y})$

3.1.2 Non-dominated Set

Among a set of solutions *P*, the non-dominated set of solutions *P′* are those that are not dominated by any member of the set *P*.

When the set *P* is the entire search space, the resulting non-dominated set *P′* is called the *Pareto-optimal set*. The Pareto-optimal set therefore contains the set of solutions, or balance trade-offs, for the MOOP. The corresponding objective vectors are referred to as the Pareto-optimal front:

$$PF = \{ f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})) | \mathbf{x} \in P' \} \qquad 3.2$$

3.2 Illustrative Representation of Non-Dominated Solutions

In multi-objective optimization, there exists more than one objective and in the most interesting cases they behave in a conflicting manner. In the case of two objectives, the performance of the algorithm can be shown by illustrating the obtained non-dominated solutions on a two-dimensional objective space plot. When the number of objective functions is larger than two, such an illustration is difficult. There are number of ways to illustrate the non-dominated solutions in such situations [39].

3.2.1 Scatter-Plot Matrix Method

In [40, 41], it was suggested plotting all $\binom{M}{2}$ pairs of plots among the *M* objective functions. Fig. 3.2 shows a typical example of such a plot with $M = 3$ objective functions. With $M = 3$ objectives, there are a total of $\binom{M}{2} \times 2 = 3 \times 2$ or 6 plots. The arrangement of the sub-plots is important. The diagonal sub-plots mark the axis for the corresponding off-diagonal sub-plots. For example, the sub-plot in position (1, 2) has its horizontal axis marked with f_2 and the vertical axis marked with f_1. If a user is not comfortable in viewing a plot with f_1 in the vertical axis, the sub-plot in position (2, 1) shows the same plot with f_1 marked in the horizontal axis.

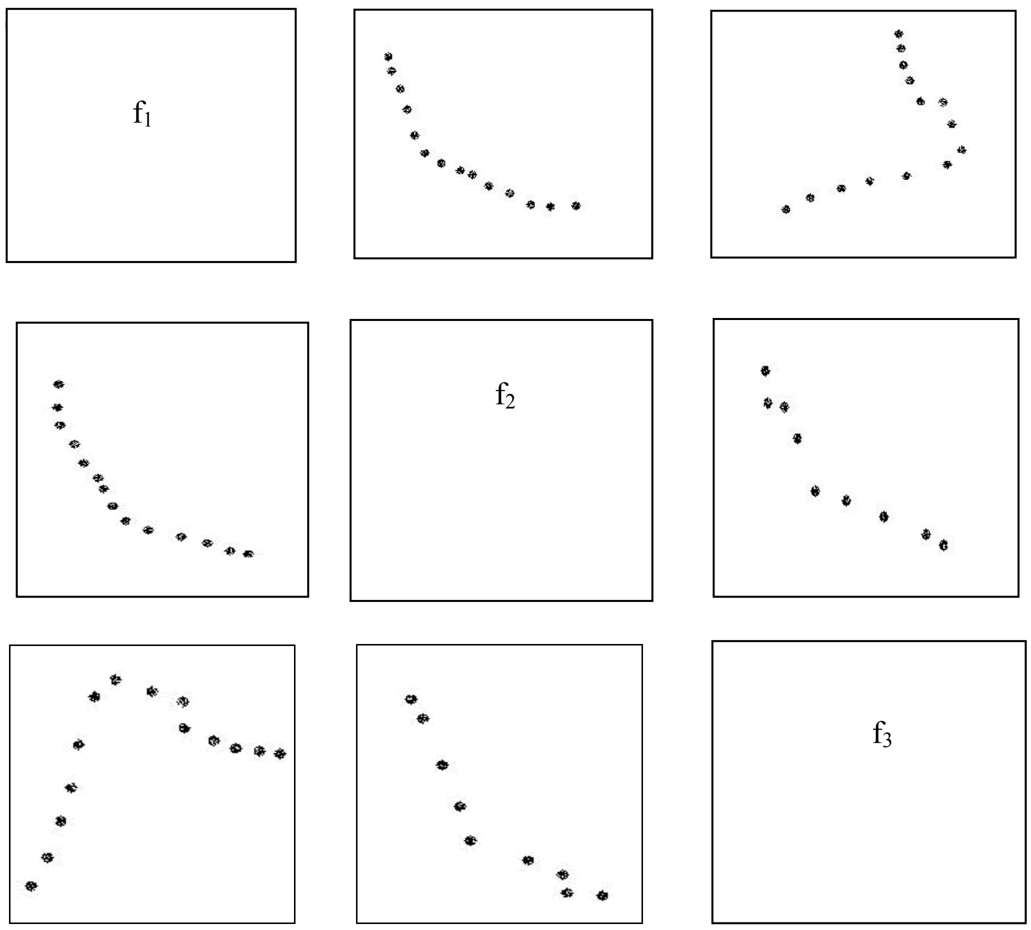


Fig. 3.2 The scatter-plot matrix method [38]

3.2.2 Value Path Method

In [42], the authors proposed this method for representing a set of non-dominated solutions (See Fig. 3.3). The horizontal axis contains each of the objective functions. The vertical axis marks the normalized objective function values. Two different types of information are plotted on the figure. The vertical bar for the objective function *k* represents the range covering the minimum and maximum values of the *k-th* objective function in the Pareto-optimal set, not in the obtained non-dominated set. Each cross-line, connecting all three objective bars, as shown in Fig. 3.3, corresponds to a solution from the obtained non-dominated set. When all solutions from the non-dominated set are plotted in this way, the plot provides a number of types of information:

1. For each objective function, the extreme function values provide a qualitative assessment of the spread of the obtained solutions. An algorithm which spreads

its solutions over the entire bar is considered to be good in finding diverse solutions.

2. The extent to which the cross-lines 'zigzag' shows the trade-off among the objective functions associated with the obtained non-dominated solutions. An algorithm having a large change of slope between two consecutive objective function bars is considered to be good in terms of finding good trade-off non-dominated solutions.

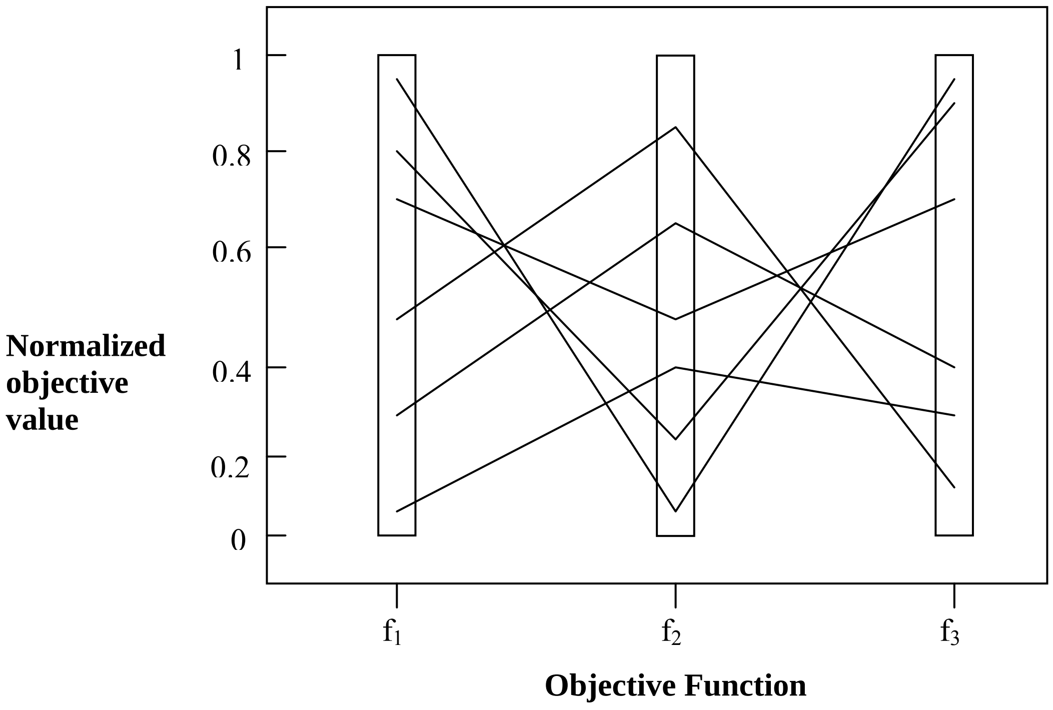


Fig. 3.3 The value path method [38]

3.2.3 Bar Chart Method

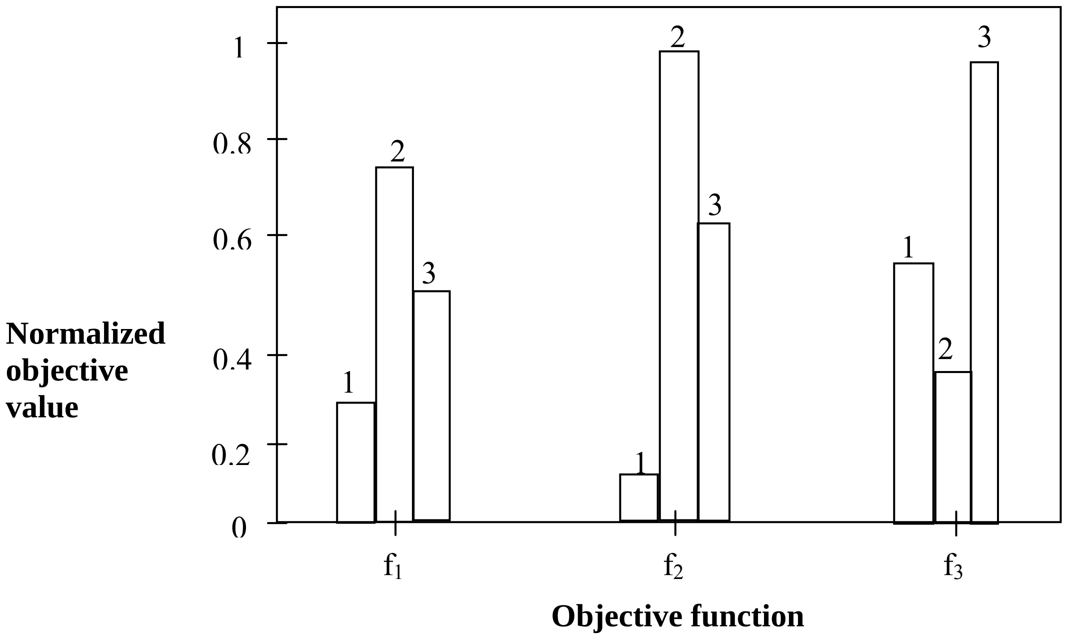


Fig. 3.4 The bar chart method

Another useful way to represent different non-dominated solutions is to plot the solutions as a bar chart [38]. First, the obtained non-dominated solutions are arranged in a particular order. Thereafter, for each objective function, the function value of each solution is plotted with a bar, in the same order. Since the objectives can take different ranges of values, it is customary to plot a bar chart diagram with the normalized objective values. In this way, if there are *N* obtained non-dominated solutions, *N* different bars are plotted for the objective functions. For example, Fig. 3.4 shows a typical bar chart plot of three objective functions and three different non-dominated solutions. Since bars are plotted, the diversity in different solutions for each objective can be directly observed from the plot. However, if *N* is large, it becomes difficult to get an idea of the trade-offs among different objective functions captured in the obtained solutions.

3.3 Metric of Performance

Comparing two multi-objective optimization algorithms experimentally always involves the notion of performance. In the case of multi-objective optimization, the definition of quality is substantially more complex than for single-objective optimization problems, because the optimization goal itself consists of the following multiple objectives [43].

- The distance of the resulting non-dominated set to the Pareto-optimal front should be minimized. In other words, solutions as close to the Pareto-optimal solutions as possible are required.
- A good (in most cases uniform) distribution of the solutions found is desirable.
- The extent of the obtained non-dominated front should be maximized, i.e., for each objective, a wide range of values should be covered by the non-dominated solutions.

A multi-objective optimization method will be termed as a good one if the above goals are adequately satisfied. Thus, a good multi-objective optimization method generates solutions close to the true Pareto-optimal front, as well as solutions that span the entire Pareto-optimal region uniformly.

Several individual metrics aiming at measuring the achievement of the previous goals by the non-dominated solution set derived from a specific multi-objective algorithm have been proposed in the literature [38, 43, 44]. Some of these are reviewed below.

Let Y' and Y'' be two sets of non-dominated objective vectors, Y_p be a Pareto-optimal set obtained from the true Pareto-optimal front Y, $\sigma > 0$ be a neighbourhood parameter (chosen appropriately for the problem at hand) and ||.|| be a distance metric.

1. Function M_1^* gives the average distance to the Pareto-optimal set Y_p [43]:

$$M_1^*(Y') = \frac{1}{|Y'|}\sum_{y' \in Y'} \min\{\|y' - y_p\| \mid y_p \in Y_p\} \qquad 3.3$$

2. Function M_2^* takes into account the distribution in combination with the number of non-dominated solutions found [43]:

$$M_2^*(Y') = \frac{1}{|Y'|-1}\sum_{y_1' \in Y'} \left|\{y_2' \in Y' \mid \|y_1' - y_2'\| > \sigma\}\right| \qquad 3.4$$

3. Function M_3^* considers the extent of the non-dominated set Y' [43]:

$$M_3^*(Y') = \sqrt{\sum_{i=1}^{M} \max\{\|y_1'[i] - y_2'[i]\| \, | y_1', y_2' \in Y'\}} \qquad 3.5$$

where M is the number of objectives and $y_1'[i]$ is the *i-th* objective function value of y_1'.

While M_1^* is intuitive, M_2^* and M_3^* need further explanation. The distribution metrics give a value within the interval [0, $|Y'|$]. The higher the value of the metric, the better the distribution for an appropriate neighbourhood parameter (e.g., $M_2^*(Y') = |Y'|$ means that, for each objective vector, there is no other objective vector within a σ-distance to it). Function M_3^* uses the maximum extent in each dimension to estimate the range to which the non-dominated front spreads out. In the case of two objectives, this equals the distance of the two outer solutions.

The previous metrics allows us to determine the absolute, individual quality of a non-dominated front. On the other hand, other metrics whose aim is to compare the performance of two different multi-objective algorithms by comparing the non-dominated sets generated by each of them have also been introduced in the literature. One of the most used among these metrics was proposed in [43], which compares a pair of non-dominated sets by computing the fraction of each set that is dominated by the other:

$$C(Y', Y'') = \frac{|\{y'' \in Y'' | \exists\, y' \in Y' \; s.t. \; y' \succ y''\}|}{|Y''|} \qquad 3.6$$

where $y' \succ y''$ indicates that the solution y′ dominates the solution y''.

Hence, the value $C(Y', Y'') = 1$ means that all the solutions in Y'' are dominated by solutions in Y'. The opposite, $C(Y', Y'') = 0$, represents the situation where none of the solutions in Y'' are dominated by the set Y'. Note that both $C(Y', Y'')$ and $C(Y'', Y')$ have to be considered, since $C(Y', Y'')$ is not necessarily equal to $1 - C(Y'', Y')$.

3.4 Searching for Preferred Solutions

Pareto multi-objective optimization algorithms are capable of finding multiple and diverse Pareto optimal (or near Pareto-optimal) solutions in a single simulation run. The next step is to select a solution among the obtained non-dominated solutions. Some possible approaches are reviewed below.

3.4.1 Compromise Programming Approach

In this approach, the algorithm picks a solution which is minimally located from a given *reference point* [45]. The user has to fix a distance metric *d()* and a *reference point* **z** for this purpose. A couple of commonly used metrics are presented below:

l_p – metric:

$$d(f,\mathbf{z}) = \left(\sum_{m=1}^{M} \left| f_m(\mathbf{x}) - z_m \right|^p \right)^{1/p} \quad 3.7$$

Tchebycheff metric:

$$d(f,z) = \max_{m=1}^{M} \frac{\left| f_m(\mathbf{x}) - z_m \right|}{\max_{\mathbf{x}\in S} \left| f_m(\mathbf{x}) - z_m \right|} \quad 3.8$$

where *S* is the entire search space. The reference point **z** is usually comprised of the individual best objective function values **z** = $(f_1^*, f_2^*, \ldots, f_M^*)^T$. Since this solution is usually a 'non-existent solution', the user is interested in choosing a feasible solution, which is closest to this reference solution.

3.4.2 Pseudo-Weight Vector Approach

In this approach, a pseudo-weight vector is calculated for each obtained non-dominated solution [38]. From the obtained set of solutions, the minimum f_i^{min} and maximum f_i^{max} values of each objective function *i* are noted. Thereafter, the following equation is used to compute the weight w_i for the *i-th* objective function:

$$w_i = \frac{\left(f_i^{\max} - f_i(\mathbf{x})\right) / \left(f_i^{\max} - f_i^{\min}\right)}{\sum_{m=1}^{M} \left(f_m^{\max} - f_m(\mathbf{x})\right) / \left(f_m^{\max} - f_m^{\min}\right)} \quad 3.9$$

This equation assumes a minimization problem and calculates the relative distance of the solution from the worst (maximum) value in each objective function. Thus, for the best solution for the *i-th* objective, the weight w_i is a maximum. The denominator in the right side of the above equation ensures that the sum of all weight components for a solution is equal to one.

Once the weight vectors for each solution in the non-dominated set are calculated, a simple strategy would be to choose the solution closest to a user-preferred weight vector.

3.5 Summary of the Chapter

This chapter firstly described the multi-objective optimization problem and how it is important in real-world applications. It explained the differences between the two approaches (classical and Pareto based algorithms) of solving the multi-objective optimization problem. This chapter further explained how to compare two solutions of the MOOP using the domination concept. In addition, it described the non-dominated set, how to illustrate the non-dominated set graphically and how to compare the performances of two multi-objective optimization algorithms. Finally, it explained the selection of a solution from the non-dominated solutions obtained by a multi-objective optimization algorithm.

The methods discussed in this chapter have been successfully applied to the design of a new general purpose Pareto-strength ant colony optimization algorithm PSACO (see section 5.6) for multi-objective optimization which may also be applied to multi-objective automatic hose/pipe routing (see section 6.4). This algorithm has also been compared with another multi-objective ant colony optimization algorithm P-ACO (see section 4.3.1) using the illustrative representation and performance metrics (see section 6.4). Further, the methods described in section 0 have been used to select a solution from the non-dominated solutions obtained in multi-objective ant colony optimization algorithms (see section 6.4.7).

4 ANT COLONY OPTIMIZATION

Initially, this chapter presents a brief introduction about ant colony optimization. Next it describes the two most commonly used varieties of ant colony optimization: the Ant System (AS) (section 4.1) and the Ant Colony System (ACS) (section 4.2). Section 4.3 briefly reviews the existing multi-objective ant colony algorithms and recent algorithms that have been developed during our research work. Among these algorithms, Pareto Ant Colony Optimization (P-ACO) could be considered as the best one (for compromise solutions) and it is explained in detail in section 4.3.1.

Some ant species are able to find the shortest path between their nest and a food source. While walking between their nest and a food source, these ants deposit a chemical called pheromone. If no pheromone trails are available, ants move randomly, but in the presence of pheromones they have a tendency to follow the trail. In practice, choices between different paths occur when several paths intersect. Then, ants choose the path to follow by a probabilistic decision biased by the amount of pheromones: the stronger the pheromone trail, the higher the desirability. Over time, the pheromone trail evaporates and it reduces intensity if no more pheromone is laid down by ants. In this way, less promising paths progressively lose pheromone because of being visited by fewer ants. This behaviour allows ants to identify the shortest paths between their nest and the food source.

Ant colony optimization (ACO) algorithms imitate the behaviour of real ants to solve difficult combinatorial optimization problems. They are based on a colony of artificial ants (computational agents) that work cooperatively and communicate through artificial pheromone trails [26]. In each cycle (or generation), some ants constructs a solution to the problem by travelling on a network. Each edge of the network represents the possible step that an ant can make and has associated two kinds of information that guide the ant’s movement:

1. *Heuristic information* – measures the heuristic preference of moving from node i to node j. This information is not modified by the artificial ants during the algorithm run.

2. *Pheromone trail information (artificial)* – mimics the real pheromone that natural ants deposit. This information is modified during the algorithm run depending on the solutions found by the ants.

In ant based systems, communication often takes place in the form of *stigmergy*. Stigmergy is a term used to indicate interactions through the environment. This form of communication does not require direct contact between the individual agents (ants). Interaction occurs when one agent alters its environment in some way, and other individuals later on respond to this change.

ACO algorithms imitate the foraging behaviour of natural ants and allow the application of this search metaphor to the finding of the solutions of hard combinatorial optimization problems like the Travelling Salesman Problem [27, 28, 29], the Quadratic Assignment Problem [30], the Job Shop Scheduling Problem [27]. Later scientists have applied them to many different discrete optimization problems [34, 35, 36, 65, 67, 69, 70, 71, 72, 73, 74, 75].

Several ACO algorithms have been proposed and are included within the ACO meta-heuristic such as the Ant System (AS) [27], the Ant Colony System (ACS) [31], the Max-Min Ant System [29], the Rank-based Ant System (rankAS) [33], and the Best-worst Ant System [32].

Among these algorithms, the former two algorithms are commonly used in most research work [65, 67, 68, 69, 70, 71, 74, 75] and are used in our approaches too. These two ACO algorithms are briefly explained in the following sections.

4.1 Ant System (AS)

The Ant System (AS) was first proposed by Dorigo and his colleagues [27, 28] as a multi-agent approach to difficult combinatorial problems such as the Travelling Salesman Problem [27, 28, 29], the Quadratic Assignment Problem [30] and the Job-Shop Scheduling problem [27].

When applying AS to finding the optimal path between two vertices S (Start node) and E (End node) of a network, a trail strength is associated with each edge to represent the pheromone strength. Initially, all ants are set on the start node (*S*) and they construct tours to the end node (*E*). At each node, the ants know the heuristic knowledge about its position (e.g. the straight line distance to the end node), the trail strength (pheromone strength) on the connecting edges, and which nodes have already been visited. Based on this knowledge the ants choose the next node (turn) probabilistically (see eq. 4.1). A global updating rule is implemented after the number of allocated turns (N_T) (at the end of the current cycle) using the quality of the solution produced by each successful ant (ant that was able to reach the end node *E*): a fraction of pheromones evaporate on all edges (edges that are not refreshed become less desirable); each ant that was able to finish a complete tour deposits an amount of pheromone on edges which belong to its tour in proportion to the quality of its tour (quality is defined according to the problem) in other words, edges which belong to high quality paths receive the greater amount of pheromone) (see eqs. 4.2 and 4.3).

The state transition rule used by the ant system, called a *random-proportional rule*, is given by eq. 4.1 and gives the probability with which ant *'i'* in node *'r'* chooses to move to neighbour node *'s'* [27],

$$p_i(r,s) = \begin{cases} \dfrac{[\tau(r,s)]^{\alpha}.[\eta(r,s)]^{\beta}}{\sum\limits_{u \in J_i(r)} [\tau(r,u)]^{\alpha}.[\eta(r,u)]^{\beta}}, & \textit{if } s \in J_i(r) \\ 0, & \textit{otherwise} \end{cases} \qquad 4.1$$

where $\tau(r, s)$ is the pheromone level on edge (r, s),

$\eta(r, s)$ is the inverse of the distance from node s to the end node (heuristic information),

$J_i(r)$ is the set of neighbour nodes of r that remain to be visited by ant i positioned on the node r,

α and β (> 0) are parameters which determine the importance of pheromone and heuristic information, respectively.

The global updating rule is implemented after the number of allocated turns (*N_T*) (at the end of the current cycle) using the quality of the solution produced by each successful ant (ant that was able to reach the end node *E*) as in eqs. 4.2 and 4.3.

$$\tau(r,s) \leftarrow (1-\rho).\tau(r,s) + \sum_{i=1}^{m} \Delta\tau_i(r,s) \qquad 4.2$$

where

$$\Delta\tau_i(r,s) = \begin{cases} Q(i), & \textit{if } (r,s) \in \textit{tour done by ant } i \\ 0, & \textit{otherwise} \end{cases} \qquad 4.3$$

and $0 < \rho < 1$ is a pheromone decay parameter. *Q(i)* measures the quality of an ant's solution (better solutions get a higher value for *Q(i)*) and *m* is the number of successful ants within the stipulated number of turns *N_T*.

Although the Ant System (AS) showed better performances than those of some general purpose heuristic algorithms for smaller size problems, it does not converge to the best known solution for the benchmark problems such as Travelling Salesman problem when the number of cities involved increased [31]. However, in these larger problems, the Ant System (AS) was able to find good solutions which are closer to the best known solution [31].

4.2 Ant Colony System (ACS)

The Ant Colony System [31] is one of the first successors of AS and was introduced to improve the performance of AS, that was able to find good solutions within a reasonable time only for small problems. ACS is based on AS and it introduces three major modifications into AS.

1. ACS uses a different transition rule, the *pseudo-random proportional rule*: an ant *i* positioned on city *r* chooses to move to neighbour node *s* by applying the rule given by eq. 4.4,

$$s = \begin{cases} \arg\max_{u \in J_i(r)} \left\{ [\tau(r,u)]^{\alpha}.[\eta(r,u)]^{\beta} \right\} & \textit{if } q \le q_0 \ (\text{exp}\textit{loitation}) \\ S & \textit{otherwise} \ (\text{exp}\textit{loration}) \end{cases} \qquad 4.4$$

where *$\tau(r, s)$* is the pheromone level on the edge (r, s),

$\eta(r, s)$ is the inverse of the distance from node s to the end node,

$J_i(r)$ is the set of neighbour nodes of r that remain to be visited by ant i positioned on node r,

α and β are parameters indicating the importance of pheromone and heuristic information, respectively,

q is a random number uniformly distributed in [0, 1],

q_0 is a parameter ($0 \leq q_0 \leq 1$) indicating the relative weighting of exploitation versus exploration,

S is a random variable selected according to the probability distribution given in eq. 4.1.

As it can be seen, the rule has a double aim: when $q \leq q_0$, it exploits the available knowledge, choosing the best option with respect to the heuristic information and the pheromone trail. However, if $q > q_0$, it applies a controlled exploration, as done in AS. In summary, the rule establishes a trade-off between the exploration of new connections and the exploitation of the information available at the moment.

2. Only the daemon (and not the individual ants) triggers the global pheromone update, i.e., an off-line pheromone trail update is done. To do so, ACS only considers a single ant, the one who generated the global best solution (*global-best-tour*). The global pheromone update rule is given by eq. 4.5,

$$\tau(r,s) = (1-\rho).\tau(r,s) + \rho.\Delta\tau(r,s) \qquad 4.5$$

where

$$\Delta\tau(r,s) = \begin{cases} Q(i_b) & if\ (r,s) \in global-best-tour \\ 0 & otherwise \end{cases} \qquad 4.6$$

and $0 < \rho < 1$ is a pheromone decay parameter. *$Q(i_b)$* measures the quality of the best ant's solution.

3. While building a solution, ants visit edges and change their pheromone level by applying the local updating rule given by eq. 4.7,

$$\tau(r,s) = (1-\rho).\tau(r,s) + \rho.\tau_0 \qquad 4.7$$

where ρ is the pheromone evaporation rate and τ_0 is the initial pheromone value.

4.3 Multi-objective Ant Colony Optimization (MOACO)

Previously, researchers have designed ACO algorithms to deal with multi-objective problems [46, 47, 48, 49, 50, 51, 52]. Most of them are designed to solve a concrete multi-objective problem such as scheduling, vehicle routing, and portfolio selection. Furthermore, most of these algorithms are designed for bi-criterion optimization problems and it is difficult to extend them to more general multi-objective ant colony optimization algorithms.

Mariano et al. [52] described an Ant-Q algorithm called MOAQ that can solve multiple objective optimization problems. MOAQ considers a family of agents for each objective function involved. Each family of agents finds solutions that depend on solutions found by the rest of the families, creating a negotiation mechanism and finding compromise solutions for all the objectives involved. The compromise solutions are evaluated in the Pareto sense, assigning rewards to the non-dominated solutions fitting all problem constraints, and punishments to the solutions violating any of them.

Iredi et al. [51] studied the ACO methods for bi-criterion optimization when the objectives cannot be ordered by importance. A multi-colony approach (BicriterionAnt) is proposed where the ant colonies are forced to search different regions of the non-dominated front. Two heterogeneous colonies are used where the ants in a colony weight the relative importance of the two optimization criteria differently so that they are able to find different solutions along the Pareto front. Cooperation among the colonies is done by exchanging solutions in the global non-dominated front that are in regions which belong to other colonies.

Pareto Ant Colony Optimization (P-ACO), proposed in [49], was originally applied to solving the multi-objective portfolio selection problem. It is based on classical ACS but the global pheromone update is performed by using two different ants, the best ant and the second-best solution generated in the current iteration for each objective *m*. In P-ACO, several pheromone matrices are considered, one for each objective *k*. At each iteration, each ant computes a set of weights $w = (w_1, w_2, \ldots, w_M)$, and uses them to combine the pheromone trails.

A multiple ant colony system (MACS-VRPTW) was developed in [50] to solve vehicle routing problems with time windows. As in the P-ACO, it is also based on classical ACS. MACS-VRPTW is based on setting up a preference to minimize one objective (the number of tours) over the other (the travel time). This solution is defined as the first of a lexicographic order on the values of the objectives. It defines two different colonies, ACS-VEI and ACS-TIME, whose activities are coordinated by the global MACS-VRPTW algorithm in order to optimize both objectives simultaneously. The former colony tries to diminish the number of vehicles used while the latter optimizes the feasible solutions obtained by the former. Each colony uses an independent pheromone trail matrix for its specific objective, and colonies collaborate by sharing the best solution found by their cooperative action. The global algorithm kills and runs again the two colonies each time a new best solution containing fewer vehicles than the previous one is obtained.

The Multiple Ant Colony System (MACS) [46] was proposed as a variation to the MACS-VRPTW algorithm. It is also based on ACS but, contrary to its predecessor, MACS uses a single pheromone matrix and several heuristic information functions.

Multi-objective Network ACO (MONACO) [47] was designed to optimize the dynamic problem of message traffic in a network. In [53] this algorithm is changed for implementation in a static environment. The algorithm takes the classical AS as the base but uses multi-pheromone trail matrices. Each ant uses the multi-pheromone trail and single heuristic information to choose the next node to visit.

In [48] the author introduced COMPETants to deal with bi-objective transportation problems. The algorithm is based on rankAS and uses two ant colonies, each with its own pheromone matrix and heuristic information. The number of ants in each colony is not fixed. When every ant has constructed its solution, the colony which has constructed better solutions gets more ants for the next iteration.

In [53] previous multi-objective ant colony algorithms were reviewed and experimentally tested in several instances of the bi-objective travelling salesman problem, comparing their performance with that of two well-known multi-objective genetic algorithms (MOGAs) NSGA-II [54] and SPEA2 [55]. According to the results published in this paper, MOACO algorithms are very competitive compared to the MOGAs implemented. MOACO algorithms offer good sets of non-dominated solutions which almost always dominate the solutions returned by NSGA-II and SPEA2. In addition, the Pareto fronts derived by the MOGAs do not dominate any of the fronts given by MOACO algorithms. Among the MOACO algorithms [53], P-ACO could be considered as the algorithm with the best global performance as fewer of its obtained Pareto fronts are dominated by the remainder of the MOACO algorithms while they dominate the remainder to some degree. Further, P-ACO generates extremely good solutions at the central part of the Pareto front (or a good set of compromised solutions).

Recent research (during the period of the research carried out for this thesis) showed that MOACO has been applied to different areas and new modifications have been proposed to MOACOs. The following paragraphs briefly review the recent publications relating to MOACO algorithms.

Alam et al. [65] applied MOACO to the problem of generating safe flight trajectories under weather hazards. The problem of weather avoidance in a Free Flight environment is formulated as: ‘given a start node and an end node in a three dimension mesh, find routes which minimize interaction with bad weather cells, minimize heading changes and minimize distance travelled’. In this paper, two different pheromone mechanisms have been applied; one uses a dynamic weighted sum of three objective functions and the other uses the concept of strength. In the second approach, the strength parameter defined in SPEA2 [55] was used, where each individual *i* in the current set of solutions is assigned a strength value $S(i) \in [0, 1]$. $S(i)$ is the number of ants that are dominated by or equal to *i*, divided by the total number of ants plus one.

In [66] the authors proposed four generic variants of ant colony optimization to solve multi-objective optimization problems and compare the four versions when applied to the multi-objective knapsack problem.

Variant 1: m-ACO$_1$ (m+1, m)

For this variant, the number of colonies is set to *m+1* and the number of pheromone matrices is set to m, where *m* is the number of objectives that need to be optimized. Each colony considers a single different objective, using its own pheromone matrix and heuristic information to build solutions; an extra ant colony is added, that aims at optimizing all the objectives. The *i*th pheromone factor considered by the *i*th objective, f_i, is defined with respect to the *i*th pheromone matrix, depending on the application. The *m+1*th pheromone factor considered by the extra ant colony is the pheromone factor of the *r*th single-objective colony, where *r* is randomly chosen. This colony considers, at each construction step, a randomly chosen objective to optimize. The *i*th heuristic factor considered by the *i*th single-objective colony that aims at optimizing the *i*th objective function f_i, is the *i*th heuristic information. The *m+1*th heuristic factor considered by the extra multi-objective colony is the sum of the heuristic information associated with all the objectives. For each single-objective colony, pheromone is laid on the components of the best solution found by the *i*th colony during the cycle, where quality of a solution is evaluated with respect to the *i*th objective, f_i only. The multi-objective colony maintains a set of solutions: a best solution for each objective. It lays pheromone on each pheromone structure relatively to the correspondent objective with the same formulae defined for the other colonies.

Variant 2: m-ACO$_2$ (m+1, m)

This second variant is very similar to the first one, and considers *m + 1* colonies and *m* pheromone matrices: a single-objective colony is associated with every different objective, and the behaviour of these single-objective colonies is defined as in variant 1; there is also an extra multi-objective colony, that aims at optimizing all objectives. The only difference between variants 1 and 2 lies in the way this multi-objective colony exploits the pheromone structures of other colonies to build solutions. For this multi-objective colony, the *m+1*th pheromone matrix is defined as the sum of every pheromone matrix of every colony.

Variant 3: m-ACO$_3$ (1, 1)

The pheromone factor considered by the ants of the single colony is defined with respect to the single pheromone structure, and the heuristic factor considered by the single colony is the sum of heuristic information associated with all the objectives. Once the colony has computed a set of solutions, every non-dominated solution (belonging to the Pareto set) is rewarded. Every component belonging to at least one solution of the Pareto set receives a same amount of pheromone. Indeed, these components belong to non comparable solutions.

Variant 4: m-ACO$_3$ (1, m)

In the last variant, there is only one colony but *m* pheromone matrices. At each step of the construction of a solution, ants randomly choose an objective $r \in \{1, ..., m\}$ to optimize. The pheromone factor is defined as the pheromone factor associated with the randomly chosen objective *r*. The heuristic factor considered by the single colony is the sum of the heuristic information associated with all the objectives. Once the colony has computed a set of solutions, the *m* best solutions with respect to the *m* different objectives are used to reward the *m* pheromone matrices.

Pinto et al. [67] proposed two modifications to the Ant Colony System (ACS) [31] and the Max-Min Ant System [29] for handling multi-objective optimization problems. In the first case, the proposed algorithm (MOACS) uses a colony of ants and a pheromone matrix for the construction of solutions at every cycle. After completing the each cycle, a known Pareto front Y_{known} is updated including the best non-dominated solutions that have been calculated so far. If the state of the Pareto front Y_{known} is changed, the pheromone matrix is reinitialized to improve exploration in the decision space X. Otherwise, the pheromone matrix is globally updated using the solutions of Y_{known} to better exploit the knowledge of the best known solutions. Note that only the links of the solutions found in Y_{known} are used to update the pheromone matrix. The modification to the Max-Min Ant System (M-MMAS) uses the same general ideas used for the MOACS.

In [68], the authors investigate the effect of elitism on multi-objective ant colony optimization algorithms (MOACOs). Elitism is implemented through the use of local, global and mixed non-dominated solutions. Further, an adaptation strategy is introduced to control the effect of elitism. With this strategy, the solutions most recently added to the global non-dominated archive are given a higher priority in defining the pheromone information. For this adaptive technique, each solution in the archive is assigned an age to indicate how long it has existed in the archive. This value is used to adjust the amount of pheromone that an aging ant will deposit. The experimental work of this research was conducted using a suite of multi-objective travelling salesman problems, each with two objectives.

Chaharsooghi et al. [69] presented a modified ant colony optimization (ACO) algorithm for solving the knapsack multi-objective problem to achieve the best layer of non-dominated solutions. A new pheromone updating rule is proposed for the multi-objective case which can increase the learning of the algorithm and consequently increases effectiveness. This approach uses multi-pheromone matrices, where each matrix represents the desirability of the solution components with respect to one objective. The pheromone updating process is achieved in two phases. In the first phase, each colony updates its own constructed solution according to the best-so-far strategy, regardless of the other objective functions. In the second phase, the global updating is accomplished for all the constructed solutions as follows: each constructed solution in the current cycle is compared with all former non-dominated solutions; if it is a non-dominated solution, the quantity of pheromone in all edges, which constructed it, will be increased by $t \times \delta$ where δ is a small positive number and t is the number of the current cycle, otherwise it is decreased by $t \times \delta$. In [72], the same authors (Chaharsooghi et al.) applied the same approach to the problem of multi-objective resource allocation.

Yagmahana et al. [70] studied the flow shop scheduling problem with multi-objectives of makespan, total flow time and total machine idle time. The ant colony optimization

(ACO) algorithm was proposed for the solution of this problem which is known to be of the NP-hard type.

The task of Supply Chain Management (SCM) is to deploy resources across a supply chain to produce high-quality goods as inexpensively as possible when the customer wants them. Taking the dependencies of the underlying production techniques into account, the SCM presents itself as an NP-hard problem. Sun et al. [71] described a multi-objective supply chain model including measurements of costs, customer service fill rates and delivery flexibility and used Ant Colony Optimization (ACO) to solve this multi-objective optimization problem. The performance of each echelon is optimized considering customer demand, production lead-time, and supply lead times throughout the supply chain.

Panahi et al. [73] considered an open shop scheduling problem that minimizes bi-objectives, namely makespan and total tardiness. The authors proposed a method based on multi-objective simulated annealing and ant colony optimization, in order to solve the given problem. This proposed algorithm is based on the concept of the Pareto dominancy. It uses an archive with a predefined size to store dominant solutions. A multi-objective simulated annealing algorithm was used to initialize the first population of the multi-objective ant colony optimization (MOACO) algorithm. At the end of each generation of the MOACO, the pheromone trail matrix is updated by elitist agents.

Colson et al. [74] presented a framework for an intelligent supervisory controller that utilizes ant colony optimization (ACO) methods for alternative energy distributed generation (AEDG) micro-grid dispatch control. The novelty of this work is the application of ACO to the rapid micro-grid power management problem given complex constraints and objectives including environmental, fuel/resource availability, and economic considerations. Given the compound nature of the multi-objective, multi-constraint energy management problem for integrated AEDG systems, this paper develops a constraint satisfaction problem (CSP) algorithm capable of finding Pareto optimal dispatch solutions.

Airport Ground Service Scheduling (AGSS) problems can be formulated as Vehicle Routing Problems with Tight time windows, Short travel time and Re-used Vehicles

(VRPTSR). Du et al. [75] presented a model with multiple objectives to minimize the number of vehicles used, the total start time of serving flights and the total flow time of vehicles for VRPTSR. An ACO algorithm with MAX-MIN and a Rank-based Ant System is proposed; an efficient heuristic called Earliest Due Date First (EDD) is incorporated into ACO as a comparative ant in order to improve the performance of ACO.

Table 4.1 shows the taxonomy for the previous MOACO algorithms and the proposed PSACO algorithm.

TABLE 4.1 A taxonomy for MOACO algorithms

MOACO Algorithm	**Year**	**Use of domination concept for pheromone updating**	**Use of density information for pheromone updating**	**Single pheromone matrix for all objectives**	**Can be extended to multiple objectives (more than 2 objectives)?**	**Several heuristic matrices**
MOAQ [52]	1999	No	No	No	Yes	Yes
BicriterionAnt [51]	1993	No	No	No	No	Yes
MACS [46]	2003	No	No	Yes	No	Yes
MONACO [47]	2003	No	No	No	Yes	No
COMPETants [48]	2003	No	No	Yes	No	Yes
P-ACO [49]	2004	No	No	No	Yes	No
Pinto et al. [67]	2005					
Alam et al. [65]	2006	Yes	No	Yes	Yes	No
Alaya et al. [66]	2007	No	No	No	Yes	No
Bui et al. [68]	2008	Yes	No	No	Yes	No
Chaharsooghi et al. [69]	2008	No	No	No	Yes	Yes
Panahi et al. [73]	2008	Yes	No	Yes	Yes	No
Colson et al. [74]	2009	No	No	Yes	Yes	No
Proposed PSACO	**2009**	**Yes**	**Yes**	**Yes**	**Yes**	**No**

Since recent MOACO algorithms were not available during the period of this research and the proposed MOACO algorithm (PSACO) was designed independently of the recent publications, the proposed MOACO algorithm is compared according to the conclusions discussed in [53]. Thus, P-ACO is selected for comparison with the proposed multi-objective ant colony optimization algorithm – the Pareto Strength Ant Colony Optimization (PSACO) algorithm (section 5.6). The following section describes P-ACO in more detail.

4.3.1 Pareto Ant Colony Optimization (P-ACO)

P-ACO uses several pheromone matrices τ_m ($m = 1, 2, ..., M$), one for each objective m. At every iteration, each ant generates a set of weights $w = (w_1, w_2, ..., w_M)$ and uses them to calculate the combined pheromone value from all the objectives.

The state transition rule used by P-ACO is given by eq. 4.8 and gives the probability with which ant i in node r chooses to move to node s [49, 53],

$$s = \begin{cases} \arg\max\limits_{u \in J_i(r)} \left\{ [\sum_{m=1}^{M} w_m.\tau_m(r,u)]^{\alpha}.[\eta(r,u)]^{\beta} \right\} & if\ q \le q_0 \\ S & otherwise \end{cases} \qquad 4.8$$

where $\tau_m(r, s)$ is the pheromone level on the edge (r, s) with respect to objective m,

$\eta(r, s)$ is the inverse of the distance from node s to the end node,

$J_i(r)$ is the set of neighbour nodes of r that remain to be visited by ant i positioned on node r,

α and β are parameters indicating the importance of pheromones and heuristic information, respectively,

w_m's are random weights selected from [0, 1],

M is the number of objectives,

q is a random number uniformly distributed in [0, 1],

q_0 is a parameter ($0 \le q_0 \le 1$) indicating the relative weighting of exploitation versus exploration,

S is a random variable selected according to the probability distribution given in eq. 4.9.

$$p_i(r,s) = \begin{cases} \dfrac{[\sum_{m=1}^{M} w_m.\tau_m(r,s)]^{\alpha}.[\eta(r,s)]^{\beta}}{\sum_{u \in J_i(r)} [\sum_{m=1}^{M} w_m.\tau_m(r,u)]^{\alpha}.[\eta(r,u)]^{\beta}}, & if\ s \in J_i(r) \\ 0, & otherwise \end{cases} \qquad 4.9$$

Every time an ant travels an edge *(r, s)* it performs the local pheromone update in each pheromone matrix, i.e., for each objective *m*, as follows:

$$\tau_m(r,s) = (1-\rho).\tau_m(r,s) + \rho.\tau_0 \qquad 4.10$$

where ρ is the pheromone evaporation rate,

τ_0 is the initial pheromone value.

The global updating rule is implemented after the number of allocated turns (N_T) using the best ant and the second-best ant with respect to each of the objectives as in eq. 4.11.

$$\tau_m(r,s) = (1-\rho).\tau_m(r,s) + \rho.\Delta\tau_m(r,s) \qquad 4.11$$

where

$$\Delta\tau_m(r,s) = \begin{cases} 15 & if\ (r,s) \in best\ and\ \mathrm{sec}ond-best\ solutions \\ 10 & if\ (r,s) \in best\ solution \\ 5 & if\ (r,s) \in \mathrm{sec}ond-best\ solution \\ 0 & otherwise \end{cases} \qquad 4.12$$

During the process, the non-dominated solutions found are stored in an external set as usually done in elitist (second generation) MOGAs.

Algorithm P-ACO [49, 53]

```
Inputs
    $\alpha$, $\beta$, $\rho$, $q_0$, Initial pheromone level ($\tau_0$), Archive
    size ($N_A$), Number of ants (N), Maximum cycles ($N_c$),
    Maximum turns or jumps ant can perform within a cycle
    ($N_T$), Start node (S), End node (E)
Output
    Non-dominated set ($A^*$)
Variables
    Cycle no (t), Turns remained (turns_remain), Archive
    ($A_t$), Population ($P_t$), Set of ants ($X_t$), Number of
    successful ants in the current cycle ($n_s$)
Initialize
    t = 1, turns_remain = $N_T$, $A_t = \Phi$, $P_t = \Phi$, $X_t = \Phi$, $n_s$ =
    0; Set pheromone level of each edge to $\tau_0$
Repeat
    Release a new set of ants $X_t$(of size N) to the colony
    from the start node (S)
    Repeat
        For each ant 'a' in $X_t$
            If ant 'a' does not reach to the end node
            (E)
                If ant's ('a') next set of feasible
                nodes is not empty
                  Move to the next node using eqs. 4.8
                  and 4.9
                  Apply the local updating to the
                  visited edge using eq. 4.10
                  Store this node in ant's ('a')
                  visited cities
                Else // Ant 'a' lost
                  Start ant 'a' again from start node
                  (S)
```

```
          Else If ant ‘a’ reaches to the end node (E)
               Mark ant ‘a’ as sussessful
               Copy the solution produced by ant ‘a’
               to $P_t$
               Ant ‘a’ stops exploring
               $n_s = n_s + 1$
     If ($n_s$ = N) // All ants in current cycle reached
     to (E)
          Break // Exit from Repeat … Until
          (turnsRemain == 0)
     turns_remain = turns_remain - 1
  Until (turns_remain == 0)
  Apply the global pheromone updating (eqs. 4.11 and
  4.12) using the best and second-best solution in $P_t$
  Copy all non-dominated solutions in $P_t$ and $A_t$ to $A_{t+1}$
  If $|A_{t+1}| > N_A$
     Reduce the size of $A_{t+1}$ to $N_A$ by removing crowded
     solutions
  Remove the current set of ants from the colony
  ($C_t = \Phi$)
  Set $A_t = \Phi$ and $P_t = \Phi$
  t = t + 1
  turns_remain = $N_T$
  $n_s = 0$
Until (t <= $N_c$)
Return $A_{Nc}$ as $A^*$
```

4.4 Summary of the Chapter

In this chapter, ant colony optimization has been described and the most commonly used ACO algorithms the Ant System (AS) and the Ant Colony System (ACS), were introduced. The chapter then reviewed the existing multi-objective ant colony optimization algorithms. Finally, P-ACO, the best MOACO algorithm for generating a good set of solutions in the central part of the Pareto front (compromised solutions) was described.

5 AUTOMATIC HOSE/PIPE ROUTING IN 3D SPACE USING THE ACO AND THE PROPOSED ACO ALGORITHMS

The research work reported in this thesis uses ant colony optimization algorithms for simulation work carried out in finding the optimal layout of automatic hose/pipe routing. Initially, this chapter explains the simulation of automatic hose/pipe routing in a 3D CAD model using ant colony optimization. Artificial ants travel from a start point to an end point on a randomly generated network of points or network of grid points. For generating these networks, collision free edges must be obtained. In this research, the C++ library RAPID was used for collision detection and this library needs the tessellated model of the original CAD model. Section 5.1 introduces the tessellated format of a CAD model. Next, the collision detection library and its use are discussed in section 5.2. Next, the steps involved in hose routing with ant colony optimization are discussed in section 5.3. Then two (proposed) modifications are explained for reducing the size of the search space (section 5.4) and avoiding the stagnation problem of the ant colony algorithm (section 5.5). Section 5.6 introduces a new multi-objective optimization algorithm – the Pareto Strength Ant Colony Optimization algorithm (PSACO). Section 5.7 explains the multi-objective hose routing problem and how P-ACO and PSACO are applied to this problem. Finally, multi-colony ant systems are proposed for (simultaneous) multi-hose routing in section 5.8.

Initially, an ant colony optimization algorithm (described in Chapter 4) is developed for minimizing the total length of pipes and avoiding the obstacles. The proposed ant colony optimization algorithm is tested on randomly generated networks of points and networks of grid points located in the free space of 3D CAD models. Other ant colony algorithms proposed in this chapter are tested on randomly generated networks. For generating these networks, it is required to test whether a path between two points is collision free. For this, the tessellated model of the original CAD model was obtained and passed to the C++ collision detection library – RAPID.

5.1 The Tessellated Format

The STL (STereoLithography) format or tessellated format [11] is an ASCII or binary file used in manufacturing to represent 3D models. It is a list of triangular planes that describes a computer generated solid model. This is the standard input for most prototyping machines. The STL file defines an object's surfaces as a set of adjacent triangles as shown in Fig. 5.1. This file basically contains the X, Y and Z Cartesian coordinates of each vertex of the triangles, as well as the coordinates of the vectors normal to the triangles. With the tessellated format, each edge is shared only by two triangles. The tessellated model is an approximation to the real model and the accuracy of the tessellated model depends on the number of triangles used. In most CAD packages the number of triangles generated for the tessellated model can be controlled. Models used in this research were generated using the CAD package Pro/Engineer and its programming toolkit Pro/Toolkit.

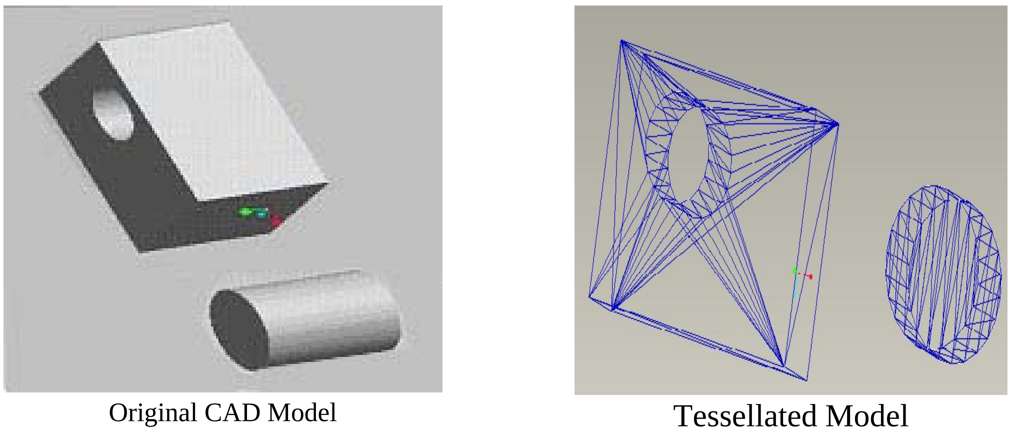


Fig. 5.1 3D Model generated in tessellated format

Standard collision detection software (such as RAPID) requires polygonal models composed entirely of triangles that are an approximated model of the original model. Thus, the tessellated representation of a 3D model can be passed to the collision detection program. Also, most CAD software support the tessellated format.

5.2 Collision Detection Library – RAPID

RAPID (Robust and Accurate Polygon Interface Detection) [59] is a C++ library developed at the Department of Computer Science, University of North Carolina, for interference detection (or collision detection) of large environments composed of unstructured models.

- It is applicable to polygon soups [59] - models that contain no adjacency information and obey no topological constraints. The models may contain cracks, holes, self-intersections, and non-generic (e.g., coplanar and collinear) configurations.
- It is numerically robust - the algorithm is not subject to conditioning problems and requires no special handling of non-generic cases (such as parallel faces).
- The RAPID library is free for non-commercial use. It has a very simple user interface: the user needs to be familiar with only about five function calls.

RAPID accepts only polygonal models composed entirely of triangles, but does not require the model to have any particular structure. For example, some collision detection systems require the shapes to be well-formed solids – the surfaces must be “closed” so that there are a well-defined inside and outside.

5.2.1 Basic Usage of RAPID

To use the RAPID library [60] in a C++ project, one must include the header file “RAPID.H” that includes all necessary structures and functions. In addition, the following header files need to be included in the C++ project: MATVEC.H, MOMENTS.H, OBB.H, OVERLAP.H, RAPID_PRIVATE.H and RAPID_VERSION.H.

In RAPID, a model is a collection of triangles; each triangle has three vertices; each vertex has three coordinates. These coordinates are given with respect to the “model coordinate system” or within the “model space”.

A model’s placement in world space is defined as the placement of the model’s coordinate axes within the world space, which are specified as a rotation, *R*, followed by a translation, *T*. Given the placement of a model with R and T, the location in world space of a vertex of the model can be determined as follows:

$$X_w = RX_m + T \tag{5.1}$$

where X_m is a point in the model coordinate system, and X_w is the world coordinate of the same point.

The basic function of RAPID is to indicate whether two objects m1 and m2 are in physical contact in world space. The corresponding code is:

```
int RAPID_Collide (
     double R1[3][3], double T1[3], RAPID_model* m1,
     double R2[3][3], double T2[3], RAPID_model* m2,
     int flag)
```

Here `R1` and `T1` represent the orientation (rotation and translation) of model `m1` in the world space and `R2` and `T2` the orientation of model `m2` in the world space. This function returns "RAPID_OK", which is 0, on success. A non-zero value indicates that the call failed, and the returned value itself is the error code. After calling this function, the number of pair-wise intersecting triangles can be found in the global variable "RAPID_num_contacts". If this variable is 0, the models `m1` and `m2` are not touching. If it is non-zero, they are touching. The variable 'flag' can be set as "RAPID_FIRST_CONTACT" or "RAPID_ALL_CONTACTS". If it is set as "RAPID_FIRST_CONTACT", the collide routine searches for contacts until it locates the first one. In the case of "RAPID_ALL_CONTACTS", the function checks for all contacts which is useful for complete knowledge about which triangles collide with others.

RAPID acquires a model by adding its triangles to a RAPID_model object. For example, the following code adds a pyramid to the RAPID_model `'m'`. Notice that the square base of the pyramid must be built as two triangles.

```
double p0[3] = {0.0, 0.0, 1.0}; // top of pyramid
double p1[3] = {-.5, -.5, 0.0); // SW corner
double p2[3] = {+.5, -.5, 0.0); // SE corner
double p3[3] = {+.5, +.5, 0.0); // NE corner
double p4[3] = {-.5, +.5, 0.0); // NW corner

RAPID_model* m = new RAPID_model;
m->BeginModel();
m->AddTri(p1, p2, p0, 0); // south face
m->AddTri(p2, p3, p0, 1); // east face
m->AddTri(p3, p4, p0, 2); // north face
m->AddTri(p4, p1, p0, 3); // west face
m->AddTri(p1, p4, p2, 4); // bottom face
m->AddTri(p2, p4, p3, 5); // bottom face
m->EndModel();
```

Notice that each triangle is given an id number, as it is added to RAPID's object. When RAPID reports contacts, it is these id numbers that are inserted into the contact_pair structures.

The `m->BeginModel()` tells RAPID to prepare the object `'m'` for the addition of triangles. Each subsequent `m->AddTri(...)` adds a triangle to the object `'m'`. RAPID stores a copy of the triangles in `'m'`. When `m->EndModel()` is called, RAPID knows that no further triangles will be added, and it then performs any necessary pre-processing.

The RAPID_model object can be destroyed with the usual C++ syntax,

```
delete m1;
delete m2;
```

5.3 Hose Routing with Ant Colony Optimization

Hose routing with the ant colony optimization algorithm is implemented by the following steps.

1. Generate the tessellated representation of the original 3D model.

2. Generate a network of valid paths (edges) using randomly generated points or grid points in the free space, from the start point to the end point.
3. Obtain the best layout of hoses ($\boldsymbol{X_G}$) between the commodities (start and end point pairs) using the ant colony optimization algorithm.
4. Refine the solution ($\boldsymbol{X_G}$) obtained using the entire search space.

In the first step, the tessellated representation of the obstacles is obtained as a text file from the CAD package. In the second step, this text file is passed to a program which incorporates the collision detection library RAPID. The following inputs are also supplied to this program:

- world size of the paths to be explored, given by the maximum and minimum of each axis coordinate - X^{min}, X^{max}, Y^{min}, Y^{max}, Z^{min}, and Z^{max},
- coordinates of the start (S) and the end (E) points,
- number of random points or grid points
- radius (r) of the hose pipes,
- text file containing the tessellated model of the original 3D model.

This program then generates a network of valid paths from randomly generated or grid points from the world, and the start point and the end point (See Fig. 5.2). When connecting two points, the program checks, with the aid of the C++ library, RAPID, that the path between two points is collision free (the axis of the hose cylinder lies on the line connecting the two points). For simplicity, a rectangular hexahedron is used that is centred on the line segment between the two points such that the cylindrical hose could be laid within it. This network data are stored in a text file for use in the next step.

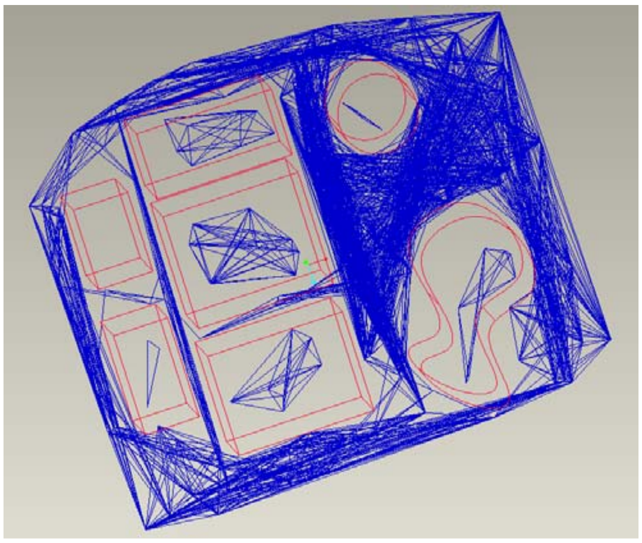


Fig. 5.2 A random point network generated in a CAD model

During the third step, the best layout of hoses (***X_G***) between the start and the end points is obtained using one of the proposed algorithms discussed in Chapter 4.

In the fourth step, the program refines the solution ***X_G*** (obtained using the entire search space). For this, the algorithm generates a network of random points near the neighbourhood of solution ***X_G*** and searches a better solution using again ant colony optimization.

5.4 The Proposed Reduced Sized Search Space for Ant Colony Optimization

A new modification of the Ant System (MAS) is introduced to reduce the size of the search space of the ant colony algorithms. This modification works very well when the problem becomes more complicated, i.e., the number of edges is increased.

The motivation of this modification is to reduce the size of the search space while generating the paths to the problem. For this, the algorithm tries to reduce the number of possible edges that need to be explored during each turn.

In ant colony optimization $J_i(r)$ (see eq. 4.1) is the set of neighbour nodes of r that remain to be visited by ant i positioned on node r. This list may include some nodes for which, if ant i were to visit them, the travel length would be greater than the current best length (or the quality would be less than the current best quality) of the best solution found so far. The proposed modification throws out these unwanted nodes from $J_i(r)$.

Assume that L_c is the current best length found by MAS and L_i is the distance travelled up to node *r* by ant *i*. Assume also that *x* is a node connected to node *r* that has not been visited by ant *i* and the distance between nodes *x* and *r* is L_x. In MAS, the contender node list $J_i^*(r)$ is obtained as follows:

$$J_i^*(r) = \{x \mid L_i + L_x < L_c\} \tag{5.2}$$

In standard ant colony optimization, there is a possibility that ants can choose a node *x* with $L_k + L_x \geq L_c$ which increases the search space and reinforces pheromones on unwanted edges.

5.5 The Proposed Explorer Ants for Avoiding Stagnation (N_MAS)

Stagnation [58] occurs when a network reaches its convergence (or equilibrium) state; an optimal (local) path p_0 is chosen by all the ants and this recursively increases an ant's preference for p_0. In order to avoid this situation, the following two types of ants are introduced into the algorithm:

a) Explorers – these ants negatively smell the high pheromone edges, i.e., these ants are attracted towards low pheromone edges and hence search for new paths.
b) Followers – these ants tends to choose high pheromone edges, i.e., these ants follow the paths found by previous ants.

In order for the proposed MAS algorithm to avoid the stagnation problem, it is necessary to introduce a new state transition rule for explorers. Thus, the state transition rule is changed for MAS as follows:

$$p_i(r,s) = \begin{cases} \dfrac{[1/\tau(r,s)].[\eta(r,s)]^{\beta}}{\sum_{u \in J_i(r)} [1/\tau(r,u)].[\eta(r,u)]^{\beta}}, & if\ s \in J_i(r) \\ 0, & otherwise \end{cases} \tag{5.3}$$

where $\tau(r, s)$, $\eta(r, s)$, $J_i(r)$ are defined as in eq. 4.1 and β is a parameter which determines the relative importance of pheromone versus distance ($\beta > 0$).

Notice that, the reciprocals of pheromone values are used in eq. 5.3 for explorer ants. As a result, these ants are attracted towards low pheromone edges. The objective of these ants (explorers) is to try to explore new paths when the algorithm exhibits a stagnated behaviour.

For the second type of ants (followers), the state transition rule remains unchanged as given in the eq. 4.1.

5.6 The Proposed MOACO Algorithm - Pareto Strength Ant Colony Optimization (PSACO)

In this thesis, a new Pareto strength ant colony optimization algorithm (PSACO) is proposed independently from [65, 66, 67]. The algorithm updates the pheromones based on the domination concept introduced in SPEA2 [55]. A single pheromone matrix is used for all of the objectives in the problem which is in contrast to P-ACO [49, 53]. Further, this algorithm is based on the classical ant colony algorithm AS [27, 28].

Most of the previous MOACO algorithms (except recently published) have not used the domination concept in Pareto optimization to update the pheromones. For example, the best and the second-best ants of the each objective are used for updating the pheromone matrix corresponding to each objective in P-ACO. Further, this algorithm needs more memory as the number of objectives is increased as it needs a separate pheromone matrix for each of the objectives. Another problem with most of the earlier approaches is that they were developed for bi-criteria optimization problems and cannot be extended to any number of objectives. For example, MACS uses a single pheromone matrix and two heuristic functions for bi-criteria optimization and the pseudo-random proportional rule of MACS cannot be extended to multiple objectives.

In PSACO, when an ant selects the next move, it uses the random propositional rule as in AS (Ant System) as defined in eq. 4.1. The major change of this algorithm is in the pheromone updating procedure. In a multi-objective problem, one cannot evaluate the quality of a solution according to just one objective as all the objectives are equally important. The current state of the art in multi-objective optimization methods for handling this type of problems is the domination concept: when assigning a quality measurement to a solution the number of individual solutions that it dominates should be considered together with the number of individual solutions by which it is dominated. In addition, a multi-objective optimization algorithm must take into account the diversity of the solutions, i.e., the final solutions produced by the algorithm must represent the whole Pareto front.

As in SPEA2 [55], PSACO maintains two solution sets: *population* P_t (of size N_P) and *archive* A_t (of size N_A) for each cycle t. Population P_t contains the set of solutions produced by the current set of ants (current cycle). Archive A_t is an external set that includes a fixed number of solutions (N_A) containing the best non-dominated solutions that have been found from the beginning of a simulation run. Whenever the number of non-dominated individuals is less than N_A, the archive A_t is filled up by current best dominated solutions.

When evaluating the quality of a solution, PSACO takes into account both dominated and dominating solutions. More specifically, each individual solution *i* in the archive and the population is assigned a strength value *S(i)* [55], representing the number of solutions it dominates (see eq. 5.7):

$$S(i) = \left|\{j \mid j \in P_t \cup A_t \wedge i \succ j\}\right| \qquad 5.7$$

where |.| denotes the cardinality of a set and the symbol $i \succ j$ indicates that solution *i* dominates solution *j*. On the basis of the *S* values, the raw fitness *R(i)* of an individual solution *i* is calculated as in eq. 5.8:

$$R(i) = \sum_{\substack{j \in P_t \cup A_t \\ j \succ i}} S(j) \qquad 5.8$$

It is important to note that *R(i) = 0* corresponds to a non-dominated solution while a high value of *R(i)* means that *i* is dominated by many solutions (which in turn dominate many solutions). Furthermore, the raw fitness *R(i)* needs to be minimized.

In addition to the raw fitness value, additional density information is incorporated to discriminate between solutions having identical raw fitness values. The density information is calculated using the *k-th* nearest neighbour method [56] as in eq. 5.9.

$$D(i) = \frac{1}{\sigma_i^k + 2} \qquad 5.9$$

where σ_i^k is the distance between the solution *i* and its *k-th* nearest neighbour in the population P_t or the archive A_t. Usually *k* is equal to the square root of the sample size (i.e. $\sqrt{N_P + N_A}$). In the denominator, two is added to ensure that $0 < D(i) < 1$.

The quality of a solution *i* in PSACO is evaluated as in eq. 5.10

$$Q(i) = \frac{1}{R(i) + D(i)} \qquad 5.10$$

This *Q(i)* value of a solution is used for the pheromone updating of eqs. 4.2 and 4.3 in AS.

The next important aspect in PSACO is the selection of solutions for the next cycle's archive A_{t+1}. The algorithm uses the same method adopted in SPEA2 [56]. The first step is to copy all non-dominated solutions (i.e., those which have quality *Q(i)* more than one) from the current archive A_t and the current population P_t to the next cycle's archive A_{t+1}:

$$A_{t+1} = \{i \in P_t \cup A_t | Q(i) > 1\} \qquad 5.13$$

If the non-dominated front fits exactly into the archive ($|A_{t+1}| = N_A$) the filling up of the archive is completed. Otherwise, there can be two situations: $|A_{t+1}| < N_A$ or $|A_{t+1}| > N_A$. In the first case the best N_A - $|A_{t+1}|$ dominated solutions (with highest *Q(i)*) from A_t and P_t) are copied into the new archive. In the second case, solutions are iteratively removed from A_{t+1} until $|A_{t+1}| = N_A$. A solution with minimum distance to another solution is chosen for removal at each stage. If there are several solutions with minimum distance the tie is broken by considering the second smallest distances and so forth.

Proposed Algorithm - PSACO

```
Inputs
    $\alpha$, $\beta$, $\rho$, Initial pheromone level ($\tau_0$), Archive size
    ($N_A$), Number of ants (N), Maximum cycles ($N_c$), Maximum
    turns or jumps ant can perform within a cycle ($N_T$),
    Start node (S), End node (E)
Output
    Non-dominated set ($A^*$)
Variables
    Cycle no (t), Turns remained (turns_remain), Archive
    ($A_t$), Population ($P_t$), Set of ants ($X_t$), Number of
    successful ants in the current cycle ($n_s$)
Initialize
    t = 1, turns_remain = $N_T$, $A_t = \Phi$, $P_t = \Phi$, $X_t = \Phi$, $n_s$ =
    0; Set pheromone level of each edge to $\tau_0$
Repeat
    Release a new set of ants $X_t$(of size N) to the colony
    from the start node (S)
    Repeat
        For each ant 'a' in $X_t$
            If ant 'a' does not reach to the end node
            (E)
                If ant's ('a') next set of feasible
                nodes is not empty
                    Move to the next node using eq.
                    4.1
                    Store this node in ant's ('a')
                    visited cities
                Else // Ant 'a' lost
                    Start ant 'a' again from start
                    node (S)
            Else If ant 'a' reaches to the end node (E)
                Mark ant 'a' as successful
                Copy the solution produced by ant 'a'
                to $P_t$
                Ant 'a' stops exploring
                $n_s = n_s + 1$
        If ($n_s$ = N) // All ants in current cycle reached
        to (E)
            Break // Exit from Repeat … Until
            (turnsRemain == 0)
        turns_remain = turns_remain - 1
    Until (turns_remain == 0)
    For each solution 'i' in $P_t$ or $A_t$
        Compute R(i), D(i) and Q(i) using eqs. 5.8, 5.9
        and 5.10 respectively
    Apply the global pheromone updating rule (eqs. 4.2 and
    4.3) using the quality Q(i) of each solution 'i' in $P_t$
```

```
        Copy all non-dominated solutions in $P_t$ and $A_t$ to $A_{t+1}$
        (eq. 5.13)
        If $|A_{t+1}| < N_A$
            Copy the best $N_A - |A_{t+1}|$ solutions in $P_t$ and $A_t$ to
            $A_{t+1}$
        Else
            Reduce the size of $A_{t+1}$ to $N_A$ by removing crowded
            solutions
        Remove the current set of ants from the colony ($C_t = \Phi$)
        Set $A_t = \Phi$ and $P_t = \Phi$
        $t = t + 1$
        turns_remain = $N_T$
        $n_s = 0$
Until ($t <= N_c$)
Return $A_{N_c}$ as $A^*$
```

The following section describes a multi-objective optimization problem that arises in hose/pipe routing with three objective functions with the addition of a new term to the random proportional rule and the pseudo random proportional rule of P-ACO and PSACO to fit the multi-objective problem.

5.7 Multi-Objective Hose Routing with MOACO

Initially an ant colony algorithm was proposed for single hose routing in 3D space and was published in [12, 13]. The search space is defined by the set of grid points as well as a set of random points in the free space defined in the CAD model. This problem is converted into finding the shortest path between two points (start and end) in a network generated by grid points or random points in the free space. There are many algorithms (e.g. A* algorithm [61], Dijkstra's algorithm [5]) that can be used to solve this problem. The major problem with the existing conventional path finding algorithms is that they cannot be extended to multiple objectives problems. For such type of problems, unconventional population-based algorithms such as ant colony algorithms or genetic algorithms produce favourable results. The aim was to extend the ant colony algorithm introduced in [12, 13] and summarized in this thesis to deal with multi-objective problems. This type of problem has a wide range of applications such as hose/pipe harness, electrical and hydraulic wiring.

The proposed MOACO algorithms attempted to optimize the following objectives in automatic hose/pipe routing:

1. Total length of the hoses between the start and the end nodes
2. The number of bends
3. The angles of the bends (the algorithm tries to keep the angle of each bend close to those in a pre-specified catalogue of angles of bends).

This scenario can be modelled as follows. Let *H = (V, E)* be an edge-weighted undirected graph representing a network in which the nodes represent terminating nodes (start or end nodes) or intermediate nodes. Let *S* and *E* be the start and end nodes, respectively. Then the algorithm needs to find a path (*P*) such that

$$\min(L), \min(N_B) \textit{ and } \min(\sum_{i=1}^{N_B} \phi(\theta_i)) \qquad 5.17$$

where *L* is the length of path *P*

N_B is the number of bends in path *P*

θ_i's are the angles between adjacent edges (see Fig. 5.3)

$\phi(\theta)$ is defined as follows:

$$\phi(\theta) = \min\{|\theta - \omega_1|/180, |\theta - \omega_2|/180, \ldots, |\theta - \omega_m|/180\} \qquad 5.18$$

where ω_1, ω_2, …, ω_m are the pre-specified catalogue angles. Each component is divided by 180 for normalizing the function and *m* is the number of catalogue angles.

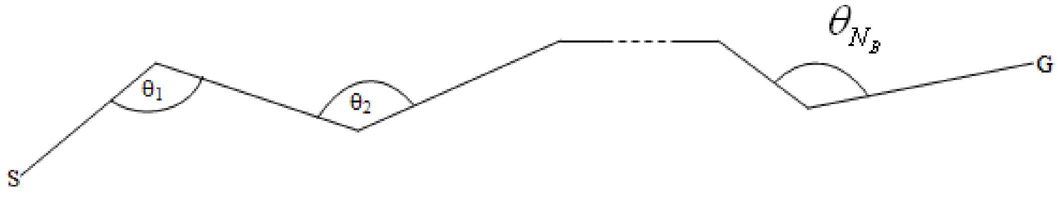


Fig. 5.3 A path between start (S) and end (E) nodes with bend angles

Then the multi-objective minimization problem can be stated as follows:

Find a path *P* in *H* between *S* and *E* such that

$$Min \quad \begin{aligned} f_1 &= 1 - \frac{SE}{L} \\ f_2 &= 1 - \frac{1}{N_B} \\ f_3 &= \frac{\sum_{i=1}^{N_B} \phi(\theta_i)}{N_B} \end{aligned} \qquad 5.19$$

Function f_1 deals with the minimization of the total length of the hoses. *SE* is the length of the straight line joining the start and end nodes. This is the absolute minimum length without considering the obstacles and is used to normalize the total length of the hoses (L). Function f_2 minimizes the number of bends (N_B). The term ($1 - 1/N_B$) is used instead of N_B for normalization such that the term reaches 0 when there is only one bend and tends to unity as the number of bends increases. Here it is assumed that there is no straight path between the start and end nodes. Function f_3 minimizes the sum of the absolute differences between angles of bends (θ_i's) and the closest angle from the pre-specified catalogue of angles of bends (see Fig. 5.3). $\phi(\theta_i)$ is defined as in eq. 5.18. When defining $\phi(\theta_i)$, each component is divided by 180 to normalize the function as the maximum absolute difference of θ and ω is 180^0.

5.7.1 Modified Random-Proportional Rule

Since the experiments carried-out in this thesis for multi-objective optimization are based on networks created from randomly generated points, the angles between two connected edges are also not known and random. Therefore, angles between two connected edges are not from the given catalogue of angles. As a result of this, the algorithm must select an angle which is closer to one of the catalogue angles. For this, a new term is added as the second heuristic function to the random proportional rule (eq. 4.1) and the pseudo-random proportional rule (eq. 4.4) for selecting edges with angles which are closer to one of the pre-specified catalogue angles as follows:

Random proportional rule:

$$p_i(r,s) = \begin{cases} \dfrac{[\tau(r,s)]^{\alpha}.[\eta(r,s)]^{\beta}[1/\phi(r,s,\theta)]^{\gamma}}{\sum_{u\in J_i(r)}[\tau(r,u)]^{\alpha}.[\eta(r,u)]^{\beta}[1/\phi(r,s,\theta)]^{\gamma}}, & \text{if } s \in J_i(r) \\ 0, & \text{otherwise} \end{cases} \qquad 5.22$$

Pseudo-random proportional rule:

$$s = \begin{cases} \arg\max_{u\in J_i(r)} \left\{[\tau(r,u)]^{\alpha}.[\eta(r,u)]^{\beta}[1/\phi(r,s,\theta)]^{\gamma}\right\} & \text{if } q \le q_0 \text{ (exploitation)} \\ S & \text{otherwise (exploration)} \end{cases} \qquad 5.23$$

In these formulae, the new term $\phi(r, s, \theta)$ is defined as in eq. 5.18 and θ is the angle between the edge *(r, s)* and the edge visited immediately before by ant *i*. γ is a parameter that indicates the relative importance of how close the angle θ is to one of the angles in the pre-specified catalogue of angles of bends. If γ is set to zero the ant calculates the probability based on the problem heuristic (distance to the end node *E*) and the pheromones laid by previous ants, just like in the original ant system. As γ is increased, the probability of choosing an edge that makes an angle (with the edge visited immediately before) that is closer to an angle in the pre-specified catalogue of angles of bends is increased.

The set $J_i(r)$, the set of neighbour nodes of *r* that remain to be visited by ant *i*, was also modified to suit our problem. First the algorithm filters neighbour nodes (say *x*) of *r* that have not been visited by ant *i* such that the edge '*rx*' makes an angle (with the edge visited immediately before by ant *i*) that differs from one of the pre-specified catalogue angles of bends by less than a certain small angle (e.g. 5^0). This set is defined as follows:

$$J_i^{(1)}(r) = \{x | \phi(r,x,\theta) \le \varepsilon\} \qquad 5.24$$

where $\phi(r, x, \theta)$ is defined as in eq. 5.18 and θ is the angle between the edge *(r, x)* and edge visited immediately before by ant *i*.

If the set $J_i^{(1)}(r)$ is empty, the algorithms explore the other neighbouring nodes of r. i.e.,

$$J_i^{(2)}(r) = \{x | \phi(r, x, \theta) > \varepsilon\} \qquad 5.25$$

Here ε is a small value between 0 and 1. For example, if the designer selects the maximum tolerance as 5 degrees, ε is set as $5/180$.

5.8 The Proposed Multi-Colony Ant Systems for Multi-Hose Routing

Two types of multi-colony ant systems (MCAS-MHR-1 and MCAS-MHR-2) are proposed for multi-hose routing; both are extensions to the ant system (AS). In MCAS-MHR-1, the task of each colony is to search for an optimal path between two points such that it shares other colonies' optimal paths (bundling) as much as possible. MCAS-MHR-1 uses only a single pheromone matrix for all the colonies. Pheromone updating is based on a weighted sum of total path lengths and shared path lengths between the paths. MCAS-MHR-2 is very similar to MCAS-MHR-1, but it uses a separate pheromone matrix for each colony and adds an additional term to the random proportional rule defined in the original ant system (AS) so that ants prefer to select paths that have been used by not only the same colony but also by the ants of other colonies.

The combinatorial optimization version of this problem consists of finding the optimum set of paths between the commodities and maximising the shared (or common) lengths of these paths. i.e., solution needs to be found such that

$$\min \sum_{c=1}^{n} l(P_c) \quad \textit{and} \quad \max \sum_{c=1}^{n} \sum_{c' \neq i, c'=1}^{n} l(P_c \cap P_{c'}) \qquad 5.26$$

where P_c and $P_{c'}$ ($c \neq c'$) are two paths between (S_c, E_c) and ($S_{c'}$, $E_{c'}$) respectively,
$l()$ represents the length of a path,
$P_c \cap P_{c'}$ represents common edges (or shared edges) between P_c and $P_{c'}$.

The multi-colony ant systems proposed in this thesis use *n* colonies to explore paths between *n* commodities. Each ant in a colony *c* explores paths between the start node S_c and the end node E_c by cooperating with other ants in colony *c*. While the ants of colony *c* are walking along edges, they try to maximize the use of common edges that are being used by ants of other colonies. As there is no direct communication between ants, this is obtained by the pheromone communication system of the proposed ant systems.

The following sections (sections 5.8.1 and 5.8.2) introduce the two versions of multi-colony ant systems MCAS-MHR-1 and MCAS-MHR-2 proposed for multi-hose Routing.

5.8.1 MCAS-MHR-1 with Single Pheromone Matrix

The problem in which we are interested is to identify the paths between the commodities with maximum possible length of common edges. Therefore, in the proposed algorithm, the common edges of the paths receive a higher amount of pheromone when pheromone updating occurs. As a result, when an ant of a later cycle encounters a shared edge, it has a higher probability of choosing it than choosing a non-shared edge.

In this approach a single pheromone matrix is used for all the colonies. When an ant of a colony is selecting the next move, it uses the random propositional rule as in the ant system defined in eq. 4.1. However, the most noticeable change to this algorithm is the pheromone updating procedure. The implementation of the pheromone updating of a path P_{ck} produced by an ant *k* of colony *c* is based on

- the length of the path P_{ck}.
- the total shared length of the path P_{ck} with each path $P_{c'k'}$ produced by each successful ant k' of each colony c' ($c' \neq c$ and $c' = 1, 2, \ldots, n$).

The global updating rule is implemented as follows. Ants which were able to complete their tour within the number of allocated turns (N_T) allow the updating of pheromone levels of their visited edges according to the following equation:

$$\tau(r,s) \leftarrow (1-\rho)\tau(r,s) + \sum_{c=1}^{n}\sum_{k=1}^{m_c}\Delta\tau_{ck}(r,s) \qquad 5.28$$

where $$\Delta\tau_{ck}(r,s) = \begin{cases} Q_{ck}, & if\,(r,s) \in tour\ done\ by\ ant\ k\ of\ colony\ c \\ 0, & otherwise \end{cases} \qquad 5.29$$

where ($0 < \rho < 1$) is a pheromone decay parameter, *n* is number of colonies (or number of commodities), m_c is the number of ants in colony *c* that were able to complete their tours within the stipulated number of turns N_T and Q_{ck} is the pheromone contribution of edges on path (P_{ck}) produced by ant *k* of colony *c* and is defined as follows:

$$Q_{ck} = w_1 * \frac{1}{L_{ck}} + w_2 * \frac{s_{ck}}{L_{ck}} \qquad 5.30$$

where L_{ck} is the length of the path P_{ck} and s_{ck} is the total shared length of path P_{ck} with paths produced by ants of other colonies (i.e., other than colony *c*) which is defined as follows:

$$s_{ck} = \sum_{\substack{c'=1 \\ c'\neq c}}^{n}\sum_{k'=1}^{m_{c'}} l(P_{ck} \cap P_{c'k'}) \qquad 5.31$$

where *l()* is the length of the path, $P_{ck} \cap P_{c'k'}$ represents common edges (or shared edges) between P_{ck} and $P_{c'k'}$, $m_{c'}$ is the number of ants in colony *c'* that were able to complete their tours within the stipulated number of turns N_T, w_1 and w_2 ($w_1 + w_2 = 1$) are two weights that measure the importance of the length of path P_{ck} and the total shared length s_{ck}, respectively.

In eq. 5.30, the weighted sum approach is used for pheromone updating and hence this algorithm uses weighted sum multi-objective optimization with the following optimizing criteria:

1. Minimizing the length of each path
2. For each path, maximizing the path length shared with other paths produced by ants in other colonies.

It is noticeable in eq. 5.28 that shared edges obtain more pheromones than non-shared edges. For example, consider the following two paths P_1 ($S_1 \rightarrow A \rightarrow B \rightarrow E_1$) and P_2 ($S_2 \rightarrow A \rightarrow B \rightarrow E_2$) (See Fig. 5.4).

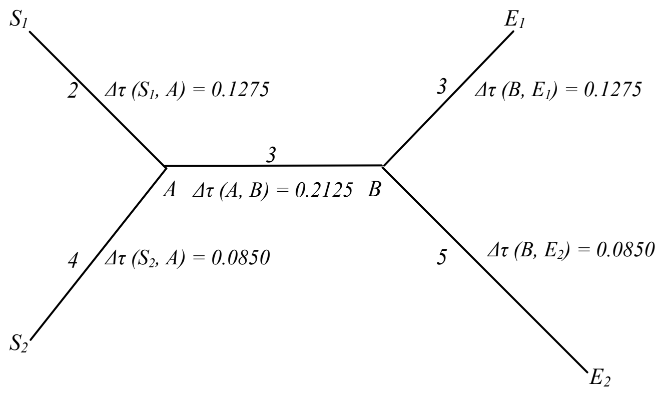

S_1
E_1
2
$\Delta\tau\ (S_1, A) = 0.1275$
3
$\Delta\tau\ (B, E_1) = 0.1275$
3
A
$\Delta\tau\ (A, B) = 0.2125$
B
4
$\Delta\tau\ (S_2, A) = 0.0850$
5
$\Delta\tau\ (B, E_2) = 0.0850$
S_2
E_2

Fig. 5.4 Demonstration of pheromone strengths of shared and non-shared edges

For the simplicity of calculation, let us assume that there is only one ant in each colony c_1 and c_2 (the ant from colony c_1 travels from S_1 to E_1 and the ant from colony c_2 travels from S_2 to E_2). Then the pheromone contribution (Q_{11}) on path P_1 ($S_1 \rightarrow A \rightarrow B \rightarrow E_1$) and the pheromone contribution (Q_{21}) on path P_2 ($S_2 \rightarrow A \rightarrow B \rightarrow E_2$) are calculated according to eq. 5.30 with w_1 = 0.99 and w_2 = 0.01 as follows:

$$F_{11} = 0.99 * \frac{1}{8} + 0.01 * \frac{3}{8} = 0.1275 \text{ (with } L_{11} = 2+3+3 = 8 \text{ and } s_{11} = 3) \qquad 5.32$$

$$F_{21} = 0.99 * \frac{1}{12} + 0.01 * \frac{3}{12} = 0.0850 \text{ (with } L_{21} = 4+3+5 = 12 \text{ and } s_{21} = 3) \qquad 5.33$$

From eq. 5.28, the shared edge (*AB*) of paths P_1 and P_2 obtains combined pheromones from both paths P_1 and P_2 that is calculated as 0.1275 + 0.085 = 0.2125, while the non-shared edges of paths P_1 and P_2 receive 0.1275 and 0.0850 as pheromone values, respectively.

5.8.2 MCAS-MHR-2 with Multiple Pheromone Matrices

Unlike MACS-MHR-1, this algorithm uses a separate pheromone matrix for each colony (or commodity). This algorithm is similar to the multi-colony ant algorithm for the edge disjoint path problem described in [57] where the algorithm attempted to find disjoint paths between the commodities (the opposite of our approach). In this approach, an ant that encounters a pheromone trail left by an ant of the same type still has a high probability of following it. However, when it encounters an edge that was shared by other paths, it is more attracted to that edge than to an edge of non-shared paths.

To implement the ants' attraction towards foreign pheromones the random propositional rule defined in eq. 4.1 has been modified appropriately as follows:

$$p_{ck}(r,s) = \begin{cases} \dfrac{[\tau_c(r,s)]^{\alpha}.[\eta(r,s)]^{\beta}[\phi_c(r,s)]^{\gamma}}{\sum_{u \in J_k(r)}[\tau_c(r,u)]^{\alpha}.[\eta(r,u)]^{\beta}[\phi_c(r,s)]^{\gamma}}, & if\ s \in J_k(r) \\ 0, & otherwise \end{cases} \qquad 5.35$$

where $\tau_c(r, s)$ is the pheromone trail left by colony c on the edge (r, s),

$\eta(r, s)$ is the inverse of the distance from node s to the end node (E_c) of colony c,

$J_k(r)$ is the set of neighbour nodes of r that remain to be visited by ant k positioned on node r,

α and β are parameters indicating the importance of pheromones and heuristic information, respectively,

$\phi_c(r, s)$ represents the amount of pheromone trail not belonging to colony c on the edge (r, s) and is known as foreign pheromone, and is defined as the sum of the pheromone trails left by all other colonies on the edge (r, s), i.e.,

$$\phi_c(r,s) = \sum_{\substack{c'=1 \\ c' \neq c}}^{n} \tau_{c'}(r,s) \qquad 5.36$$

γ is a parameter that indicates the relative importance of foreign pheromone trails left by other colonies. If γ is set to zero, the ant calculates the probability based on the problem heuristic and the pheromones of its own colony in the manner identical to the original ant system (see eq. 4.1). If γ is increased, the probability of choosing an edge with a large amount of foreign pheromone trail is also increased and thus the ant tends to select the edges shared with the previous paths.

The global updating rule is implemented as follows. The ants which were able to complete their tour within the number of allocated turns (N_T) allow the updating of pheromone levels of their visited edges according to:

$$\tau_c(r,s) \leftarrow (1-\rho)\tau_c(r,s) + \sum_{k=1}^{m_c} \Delta\tau_{ck}(r,s) \qquad 5.38$$

where m_c is the number of ants in colony c that were able to complete their tours within the stipulated number of turns N_T and $\Delta\tau_{ck}(r, s)$ is defined as in eqs. 5.29, 5.30 and 5.31.

In the ant colony algorithms presented above, it is necessary to compare two paths produced by the same colony and compare the entire solution produced by all colonies in the current cycle with the previously generated solution. The following methods are used for comparing them.

A. Finding the best path of a colony in a cycle

The pheromone contribution F_{ck} defined in eq. 5.30 is used to compare two paths in the same colony, i.e., $Q_{ck} > Q_{ck'}$ means that ant k produces a better path than ant k'.

B. Updating the total solution

Once the best path for each colony *c* is identified, it is necessary to find whether or not this solution is an improvement over the best solution produced in the previous cycles. To do this, two metrics, *strength_1* and *strength_2*, are defined for a solution $\{P_j \mid j = 1, 2, \ldots, n\}$ of the multi-path problem.

strength_1 is the reciprocal of the total length of the solution:

$$strength\_1 = \frac{1}{L} \tag{5.39}$$

strength_2 is the weighted measure of the total length and total shared distance:

$$strength\_2 = w_1 * \frac{1}{L} + w_2 * \frac{s}{L} \tag{5.40}$$

where $$L = \sum_{j=1}^{n} l(P_j), \quad l - length\ of\ the\ P_j$$

$$s = \sum_{i=1}^{n} \sum_{\substack{j=2 \\ j \neq i}}^{n} l(P_i \cap P_j)$$

w_1 and w_2 ($w_1 + w_2 = 1$) are two weights that measure the importance of the total length L and shared length s, respectively.

The solution produced by a cycle is considered to be better than the previous solution if both *strength_1* and *strength_2* have improved or if one of the criteria has improved whilst the other criterion has remained unchanged.

5.9 Summary of the Chapter

In this chapter, the tessellated model of a 3D CAD model and the C++ collision detection library RAPID have been described. It was then explained how they are used to check for edge collision with the given CAD model. A description was given of how the tessellated model, the RAPID library and ant colony optimization are used in hose routing.

Fig. 5.5 shows the various steps involved in automatic hose/pipe routing suggested in this thesis.

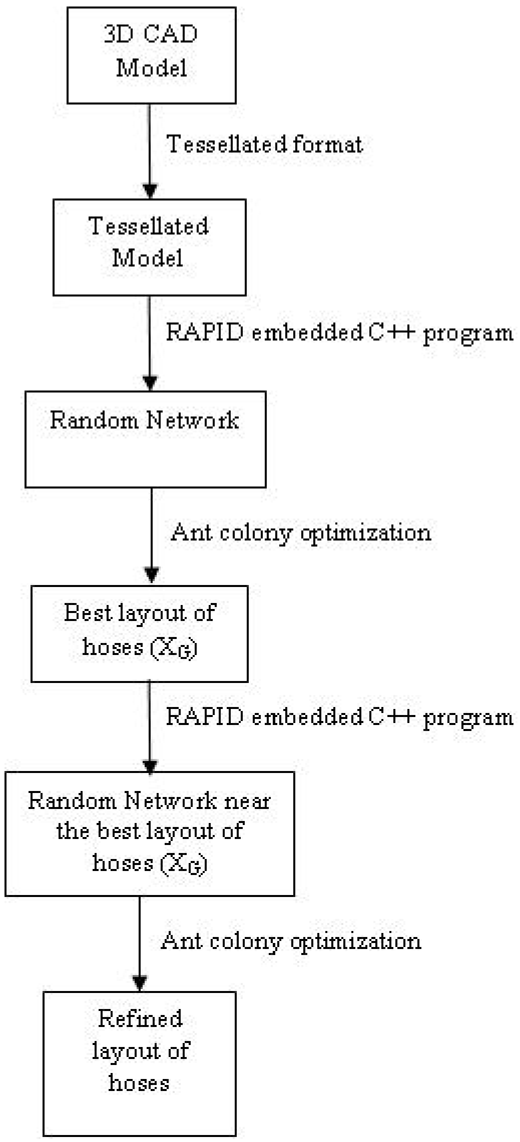

3D CAD Model
Tessellated format
Tessellated Model
RAPID embedded C++ program
Random Network
Ant colony optimization
Best layout of hoses (X_G)
RAPID embedded C++ program
Random Network near the best layout of hoses (X_G)
Ant colony optimization
Refined layout of hoses

Fig. 5.5 Steps involved in automatic hose/pipe routing in a 3D CAD model using ACO

Next two modifications were introduced into ant colony optimization for reducing the size of the search space while generating solutions to a given problem and for avoiding the stagnation problem. Then this chapter briefly reviewed the existing multi-objective ant colony optimization algorithms. In addition, a taxonomy of existing algorithms and the proposed MOACO algorithm (PSACO) was presented. P-ACO, the best MOACO algorithm for generating a good set of solutions in the central part of the Pareto front (compromised solutions) and the proposed algorithm (PSACO) for MOACO were described in detail. How these two algorithms (P-ACO and PSACO) can be applied to a multi-objective hose routing optimization problem was explained. Finally, two multi-colony ant systems were proposed for simultaneous multi-hose routing.

The algorithms presented in this chapter have been successfully applied to obtain the results of this thesis. Initially ant colony optimization algorithms have been applied to single-objective optimization (minimizing the length of the path) in section 6.1. Results of the proposed methods for reducing the size of the search space and avoiding the stagnation problem were presented in sections 6.2 and 6.3, respectively. Results of the proposed Pareto strength ant colony optimization (PSACO) and P-ACO were presented in section 6.4. Finally in section 6.5, results of the proposed multi-hose routing algorithms were presented.

6 RESULTS AND DISCUSSION

This chapter shows the experimental results of the simulations carried out for the automatic hose/pipe routing of 3D models using ant colony optimization. In section 6.1, the results of the preliminary experiments are presented and the strengths and weaknesses of the Ant System (AS) on grid-based and random-based networks are discussed. In section 6.2, the results of the reduced size search space for ant colony optimization (MAS) are presented and discussed. The results of the introduction of explorer ants to solve the stagnation problem of the ant colony optimization algorithm are presented and discussed in section 6.3. In section 6.4, the results of the multi-objective ant colony optimization algorithm (PSACO) proposed in section 5.6 are presented and discussed. In this section, the performance of PSACO is compared with the current best Pareto ant colony optimization algorithm (P-ACO), using the illustrative representation methods for the non-dominated sets (described in section 3.2), the performance comparison methods for two multi-objective optimization algorithms (described in section 3.3), and the methods for selecting a solution from the non-dominated solutions (described in section 3.4). Further, computation times for the two algorithms (PSACO and P-ACO) are compared and refinement of the obtained solution is discussed in section 6.4. Finally, results obtained by the proposed multi-colony ant systems (see section 5.8) for multi-hose routing are discussed in section 6.5.

Initially, the Ant System (AS) was implemented for grid-based networks and random-based networks and their strengths and weaknesses were investigated empirically. Preliminary experiments were restricted to a single objective function (finding the shortest path from start to goal) and avoiding the obstacles. The CAD software package Pro/Engineer was used for generating the 3D models and its programming toolkit Pro/Toolkit was used for obtaining the tessellated format (STL format) of the generated model. The C++ library, RAPID, was used to generate a network of paths which are collision-free with objects of the CAD model. The same procedure was followed for the generation of the random networks of collision-free paths for the other experiments carried-out in this thesis.

6.1 Results Obtained for Grid-Based and Random-Based Ant Systems

Initially, a feasibility study was carried out for the application of the Ant System to hose/pipe routing. The purpose of this experiment was to optimize a single objective function (total length of the pipes) avoiding collision of the pipes with objects in the CAD model.

The parameter settings for the ant systems were:

Number of ants (N) = 10
Initial pheromone level for each edge (τ_0) = 100
Maximum number of turns an ant can perform within a cycle (N_T) = 100
Maximum number of cycles (N_c) = 100
$\alpha = 1$, $\beta = 5$ and $\rho = 0.01$

These parameter settings were selected based on earlier research carried-out for ant colony optimization [30, 31], where it was found that these parameter settings achieve good results in general.

All the simulations were conducted on a Pentium IV PC (Processor speed = 3.0 GHz, Memory = 512 MB) in the Microsoft Windows XP environment using Microsoft Visual C++ (Version 6.0).

The performance of the algorithms was defined by the time (seconds) taken by a run and the length of the optimal path obtained.

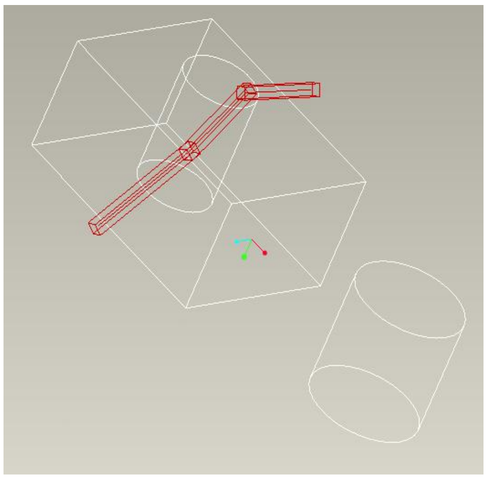
The algorithms were tested on 5 models and for each model, the grid-based version was tested on 3 different step sizes (increment values of x, y and z coordinates) 10, 25 and 50. The random-based version was tested for 100, 150, 200 and 500 random points. All the simulations were carried out for 100 cycles and averaged over 10 trials varying the random seed across the trials. In the figures below, the best paths obtained over 10 trials are shown for each of the two versions.

6.1.1 Model 1 - Hose routing in an environment with a hole in a cube

The proposed ant colony algorithm was tested in an environment consisting of a cube containing a hole (see Fig. 6.1). Hose segments needed to be laid inside this hole in order to obtain the optimal path.

Table 6.1 Comparison of grid-based and random-based - hole in a cube model

	Grid-based			Random-based			
Step Size	10	25	50	N/A	N/A	N/A	N/A
No of points	18081	1377	225	100	150	200	500
Avg. Length	**244.32**	251.66	378.80	292.70	299.00	262.49	**245.22**
St. Dev. (Len.)	4.21	4.35	11.24	51.20	54.00	17.51	3.40
Best (Length)	**237.79**	247.23	367.96	256.80	257.70	241.32	**238.85**
Avg. Time (s)	993.70	49.40	3.50	8.90	21.40	32.00	220.10


(a) Grid-based

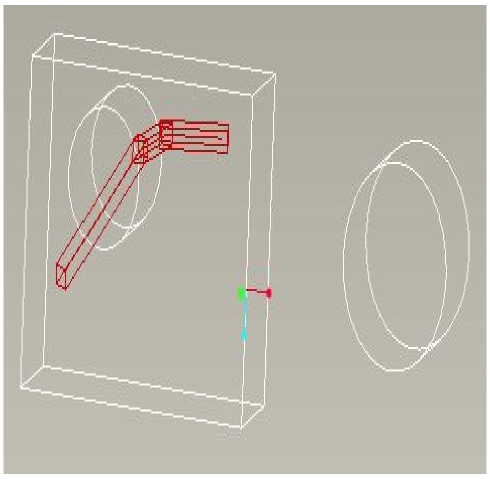

(b) Random-based

Fig. 6.1 Model 1: Hole in a cube
{X_{min} = -250, X_{max} = 150, Y_{min} = -50,Y_{max} = 150, Z_{min} = -200, Z_{max} = 0; S = (-200, 150, -100); E = (-100, -50, -150); Radius = 5}

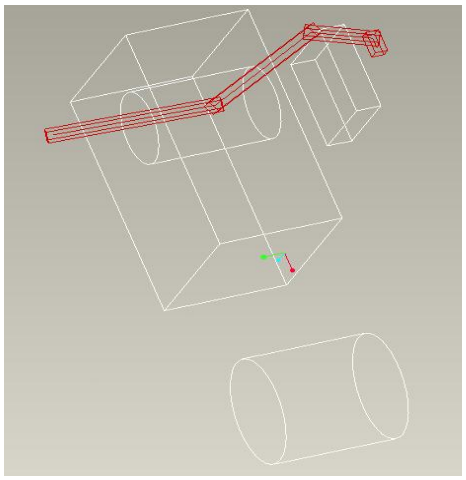
Table 6.1 shows the comparison of the two versions over 10 trials for each value of the step size (grid-based) and each number of random points (random-based). According to Table 6.1, the best solution generated by the random-based version (with 500 random points) is very close to the best solution generated by the grid-based version (with step size = 10 and 18081 points). However, the average computational time taken by the random-based version is comparatively less than for the grid-based version (220.1 sec. against 993.7 sec.). The random-based version is more than 4 times faster.

6.1.2 Model 2 - Hose routing in an environment with a hole in a cube where the optimal path is blocked by an obstacle

In this simulation, the optimal path found in the earlier case was blocked by a cubic obstacle and the target point was placed behind the obstacle (see Fig. 6.2 and Table 6.2).

Table 6.2 Comparison of grid-based and random-based - hole in a cube models where the optimal path is blocked by a cubic obstacle

	Grid-based			Random-based			
Step Size	10	25	50	N/A	N/A	N/A	N/A
No of points	18081	1377	225	100	150	200	500
Avg. Length	322.70	**305.08**	400.00	397.70	368.10	376.50	**312.60**
St. Dev. (Len.)	20.96	7.24	0.00	43.10	29.88	41.70	15.14
Best (Length)	**290.55**	299.34	400.00	339.40	329.54	324.40	**291.51**
Avg. Time (s)	952.20	64.40	5.00	12.30	27.30	49.90	318.30


(a) Grid-based

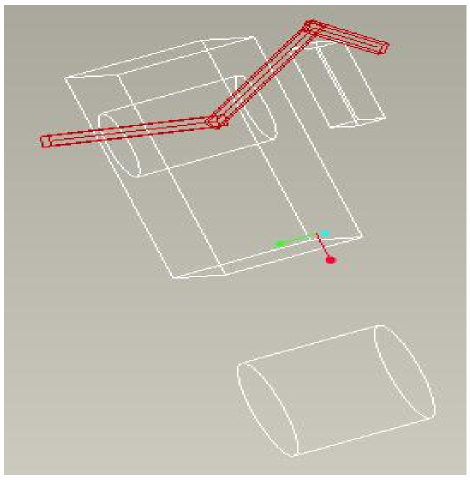

(b) Random-based

Fig. 6.2 Model 2: Optimal path is blocked by a cubic obstacle {X_{min} = -250, X_{max} = 150, Y_{min} = -50, Y_{max} = 150, Z_{min} = -200, Z_{max} = 0; S = (-200, 150, -100); E = (-200, -100, -150); Radius = 5}

The best average length for the grid-based version is obtained with step size 25 (1377 points) (see Table 6.2). However, the best path was produced with the step size 10. The average path length of the random-based version with 500 points (312.6) is relatively close to the average path length of the grid-based version with step size 25 (305.08) and the lengths of the best paths in both versions are very close.

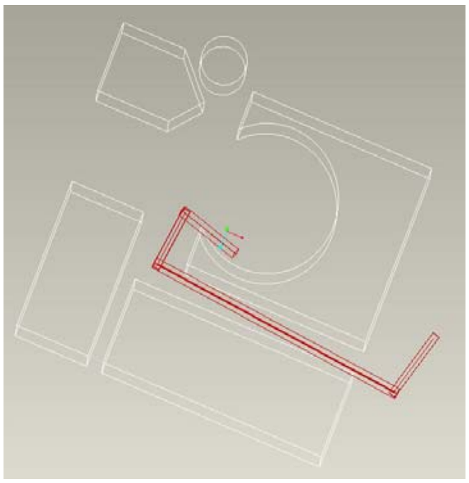
This experiment shows that when selecting the right grid (or step) size, the grid-based version performs very well even with relatively large step sizes.

6.1.3 Model 3 - Hose routing in an environment with a U-shape obstacles

In this experiment, a U-shape obstacle was placed in the environment and the environment was made more complex by introducing other objects. Furthermore, the start and the target points were placed such that only one path existed between them. Note that the z coordinates of the search space were restricted to the top and the bottom of the obstacles (See Fig. 6.3 and Table 6.3).

Table 6.3 Comparison of grid-based and random-based models with a u-shaped obstacle

	Grid-based			Random-based			
Step Size	10	25	50	N/A	N/A	N/A	N/A
No of points	55451	4205	675	100	150	200	500
Avg. Length	Failed	**596.28**	Failed	1260.00	925.00	761.60	**710.30**
St. Dev. (Len.)		22.42		405.00	328.00	174.80	40.00
Best (Length)		**551.31**		698.00	654	619.10	**608.9**
Avg. Time (s)		173.25		13.20	38.10	89.70	804.60


(a) Grid-based

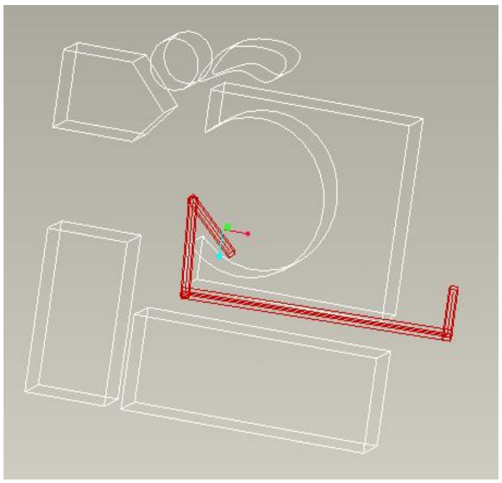

(b) Random-based

Fig. 6.3 Model 3: U-shaped obstacle
{X_{min} = -300, X_{max} = 400, Y_{min} = 0, Y_{max} = 100, Z_{min} = -300, Z_{max} = 400; S = (50, 25, -50); E = (350, 25, -50); Radius = 5}

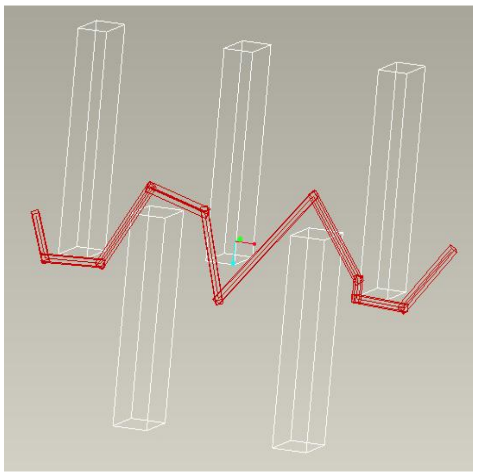
In this experiment, the grid-based version failed in all the trials with the step sizes 10 and 50. The grid-based version failed for these two step sizes because none of the grid lines were laid in any part of the optimal path when generating the network of grid lines. However, it was successful with step size 25 and generated the best average length (596.28) and best optimal path length (551.31). This experiment demonstrates that if the right resolution is selected, the grid-based version performs well in terms of both optimal length and the computational time.

6.1.4 Model 4 - Hose routing in an environment with parallel walls

In this experiment, two 3D points were selected and the shortest path between them was blocked by 5 parallel walls (see Fig. 6.4 and Table 6.4).

Table 6.4 Comparison of grid-based and random-based models containing parallel walls

	Grid-based			Random-based			
Step Size	10	25	50	N/A	N/A	N/A	N/A
No of points	40931	3125	507	100	150	200	500
Avg. Length	Failed	**1096.90**	Failed	1021.90	1025.40	986.20	**963.61**
St. Dev. (Len.)		91.30		41.60	57.20	45.90	9.07
Best (Length)		**1007.70**		968.90	963.00	**938.00**	948.09
Avg. Time (s)		133.67		11.10	28.30	59.40	316.70


(a) Grid-based

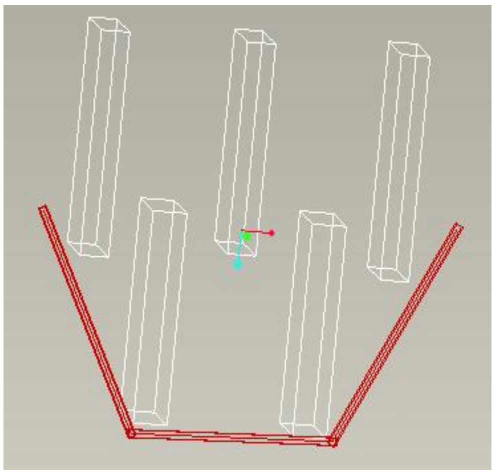
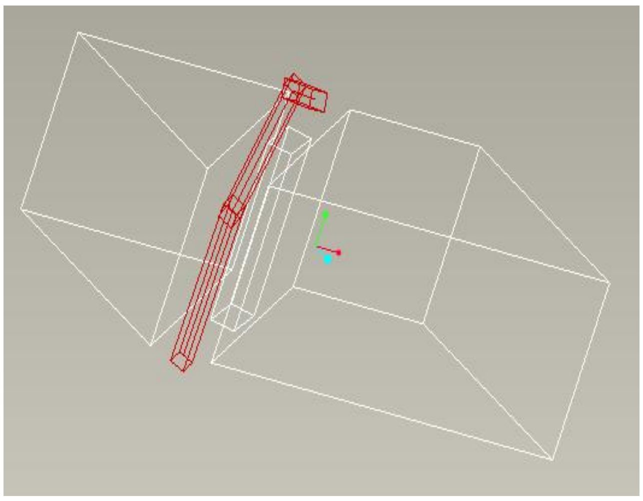

(b) Random-based

Fig. 6.4 Model 4: Parallel walls
{$X_{min} = -300$, $X_{max} = 300$, $Y_{min} = 0$, $Y_{max} = 100$, $Z_{min} = -300$, $Z_{max} = 300$; S = (-300, 25, 0); E = (300, 50, -25); Radius = 5}

Here also, the grid-based version failed for step sizes 10 and 50 as in the previous experiment. Here also grid-based version failed because it is not possible to find a connected path between start point (S) and end point (E) in the network of grid lines generated using the step sizes 10 and 50. Even though the grid-based version was successful with step size 25, the average length and the best length are higher than the respective values for the random-based version. The average computational time for the random-based version is low for all cases except for 500 random points.

6.1.5 Model 5 - Hose routing in an environment with a diagonal empty space



Random-based

Fig. 6.5 Model 5: Diagonal empty space
{X_{min} = -50, X_{max} = 350, Y_{min} = -50, Y_{max} = 150, Z_{min} = -200, Z_{max} = 50; S = (25, 0, 0); E = (180, 100, -185); Radius = 5}

Table 6.5 Comparison of grid-based and random-based models with a diagonal empty space

	Grid-based			Random-based			
Step Size	10	25	50	N/A	N/A	N/A	N/A
No of points	22386	1683	270	100	150	200	500
Avg. Length	Failed	Failed	Failed	320.24	313.32	327.17	**295.23**
St. Dev. (Len.)				13.70	9.21	20.44	9.86
Best (Length)				302.38	299.14	303.08	**275.39**
Avg. Time (s)				7.60	16.60	27.50	155.70

In this simulation, a diagonal empty space was placed between two objects and the straight path between the start point and the end point was blocked by a cubic shaped object (see Fig. 6.5 and Table 6.5). The grid-based version failed for all 3 step sizes. Here also the grid-based version failed because it is not possible to find a connected path between start point (S) and end point (E) in the network of grid lines generated using the step sizes 10, 25 and 50. The random-based version was successful in each case and produced reasonable results.

6.1.6 Discussion (Grid-Based and Random-Based Ant Systems)

The first phase of our research has been applied to automatic 3D hose/pipe routing where the world is represented as two versions, grid-based and random-based for single hose routing in 3D space. The search space is defined by a set of grid points as well as a set of random points in free space. The problem is converted into finding the shortest path between two points (start and end) in a network generated by the grid points or random points in the free space. There are many algorithms (e.g., A* algorithm [61], Dijkstra's algorithm [5]) that can be used to solve this problem. The major problem with existing conventional path finding algorithms is that they cannot be extended to multiple-objective problems. For such types of problem, unconventional algorithms such as ant colony algorithms, genetic algorithms are preferable. The aim of the ant colony algorithm introduced in this thesis was for it to be extendable to multiple-objective problems. This type of problem has a wide area of applications such as hose/pipe harness, electrical and hydraulic wiring.

The problem presented here and the travelling salesman problem (TSP) are quite similar; however there are some differences. In the TSP, paths must be found such that each ant must travel to each city once and must finally come back to the start city. In the case described here, ants must start from the start point and need to finally reach the end point. The constraints that each ant must travel to each point and that ants must finally come back to the start point are not imposed. However, it must be guaranteed that when an ant has visited a point, it must not visit that point again. To this end, cycles were removed from the ants' paths before applying the global updating rule. For the TSP, the global updating rule is applied after all ants have completed a tour (i.e., each and every ant must come back to the start city). Hence, for the TSP, the algorithm knows when to apply the global updating rule. In the experiments described above, this is not always possible, as some ants may get lost. Thus, a new parameter, N_T, was introduced into the algorithm. This parameter was set such that most of the ants of the current population were able to reach the end point.

The above simulation results show the strengths and weaknesses of the grid-based and the random-based versions of the ant colony algorithm for automatic 3D hose routing. The use of the RAPID library greatly helps the algorithm to detect collisions when laying the hoses. The simulation study also indicates that the proposed grid-based and random-based versions of the ant colony algorithms are of practical use because the required computational times are reasonably low.

However, in the grid-based version, the resolution or the size of the grid plays an important role in the determination of the optimal path and affects the computational time. If none of the grid line falls on the optimal path when constructing the road map, the algorithm fails to obtain the optimal path (See Tables 6.3, 6.4 and 6.5). Thus, selecting the right size of grid (or step size) is important for the grid-based version.

The advantage of using the random-based version is that it did not fail for any of the tested models and produced a reasonably good solution to the problem in comparably lesser time. Furthermore, in the case of the grid-based version, as the step size is decreased, the amount of memory needed to store the road map increases drastically as does the computation time.

6.2 Results Obtained for Ant Colony Optimization with a Reduced Size Search Space

In this section, results of the new modification (MAS) introduced in section 5.4 are explored. AS (Ant system) and MAS were implemented and their strengths and weaknesses were investigated experimentally. The performances of the two algorithms were compared in terms of the quality of the best path obtained (see section 6.2.1), the number of turns performed during the allocated time (see section 6.2.2), the total number of nodes in all contender node lists $J_i^*(r)$ during each cycle and the average number of alternatives (see section 6.2.3) and the time spent on each cycle (see section 6.2.4).

The parameter settings for both algorithms were:

Number of ants (N) = 10

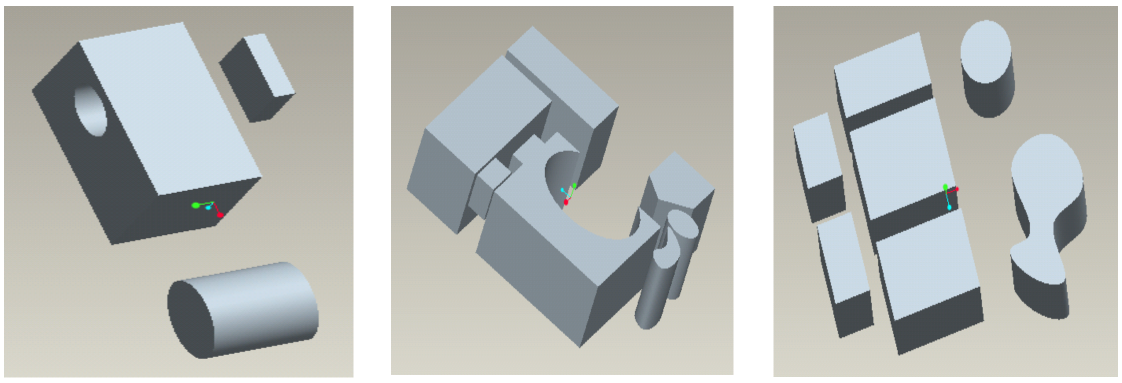
Initial pheromone level for each edge (τ_0) = 100

$\alpha = 1$, $\beta = 5$ and $\rho = 0.01$

The stopping criterion for each of the experiments was set according to the problem and the complexity of the network.

All the simulations were carried out on a Pentium IV PC (Processor speed = 3.0 GHz, RAM = 512 MB) in the Microsoft Windows XP environment using Microsoft Visual C++ (.Net version).

Both algorithms (AS and MAS) were tested on the following three Pro/Engineer models (see Fig. 6.6).



(a) Model 1 (b) Model 2 (c) Model 3

Fig. 6.6 Tested Pro/Engineer Models

6.2.1 Quality of the best path obtained

For each model, both algorithms (AS and MAS) were tested with networks of 200, 400 and 800 random points. All the simulations were carried out for a specific time (T) over 100 trials with different random seeds in each trial. T was determined by running MAS for 4000 turns (40 cycles) and the time elapsed to obtain the best length was recorded for each trial. *T* was then made roughly equal to the average time taken to obtain the best solution with MAS over 50 trials.

For each trial, the best length given by AS or MAS, the list of nodes in the corresponding best path and the number of turns (iterations) were recorded in text files.

In this section, the best lengths obtained for AS and MAS are compared statistically over the results obtained from 100 trials. The following statistics are considered:

Avg. Len.	– Average of the best lengths over 100 trials
Std. Dev.	– Standard deviation of the best lengths
Minimum Len.	– Minimum best length obtained
Maximum Len.	– Maximum best length obtained
First Quartile	– First quartile of best lengths
Median	- Median of best lengths
Third Quartile	– Third quartile of best lengths
Range	– Difference between the maximum and minimum best lengths

A highlighted (in bold) statistical value in the tables (see Tables 6.6 – 6.8) indicates that it exceeds the corresponding value of the other algorithm. Values are italicized when the two algorithms obtained the same results.

Histograms are also used to compare the best lengths of AS and MAS (see Figs. 6.7 – 6.15).

Experimental results (Quality of the best path obtained)

Model 1

Both algorithms were tested on an environment consisting of a cube containing a hole (See Fig. 6.6). The optimal path was first determined by laying pipe segments through the hole. The optimal path was then blocked by a cubic obstacle and the end node was placed behind this cubic obstacle.

The following input parameters were supplied to the program:

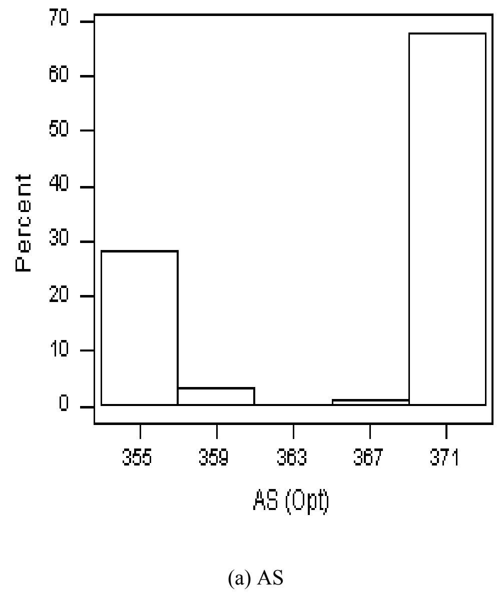
X_{min} = -250, X_{max} = 150, Y_{min} = -100, Y_{max} = 150, Z_{min} = -200, Z_{max} = 0,

S = (-125, 150, -100), *E* = (-125, -100, -150), Radius (*r*) = 5,

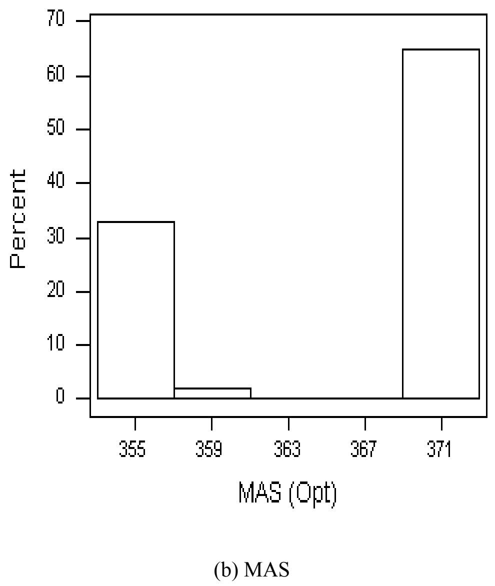
Maximum number of turns an ant can perform within a cycle (N_T) = 100

Table 6.6 Comparison of best lengths obtained in AS and MAS – Model 1

	AS			MAS		
No of points	200	400	800	200	400	800
Time *T* (in Sec.)	5	6	50	5	6	50
Avg. Len.	365.61	317.79	349.43	**365.06**	**315.14**	**342.03**
Std. Dev.	**6.34**	6.47	11.84	6.60	**1.84**	**10.04**
Minimum Len.	*355.97*	*310.62*	322.71	*355.97*	*310.62*	**316.27**
Maximum Len.	*369.87*	339.70	380.92	*369.87*	**321.27**	**358.55**
First Quartile	*355.97*	*314.68*	342.12	*355.97*	*314.68*	**333.51**
Median	*369.87*	*314.68*	353.45	*369.87*	*314.68*	**343.97**
Third Quartile	*369.87*	319.82	354.79	*369.87*	**314.68**	**349.69**
Range	*13.90*	29.08	58.21	*13.90*	**10.65**	**42.28**

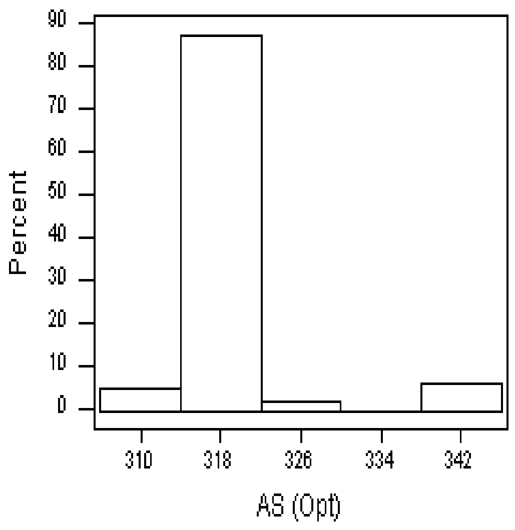


(a) AS

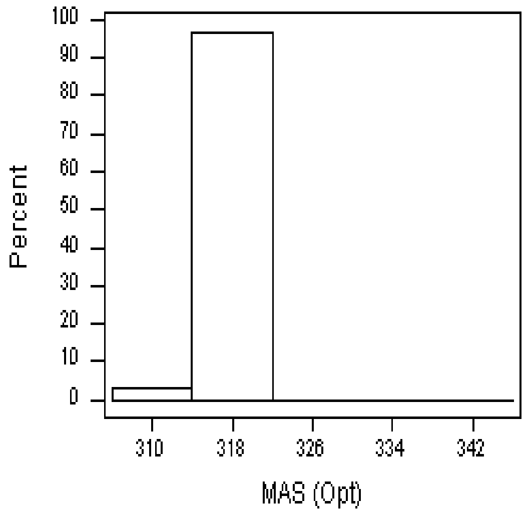


(b) MAS

Fig. 6.7 AS and MAS histograms for a 200-point network over 100 trials - Model 1

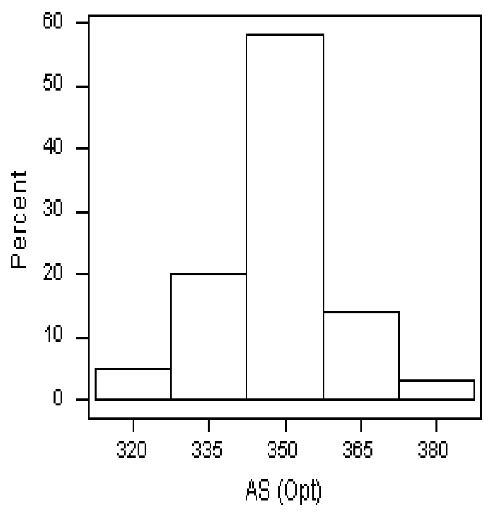


(a) AS

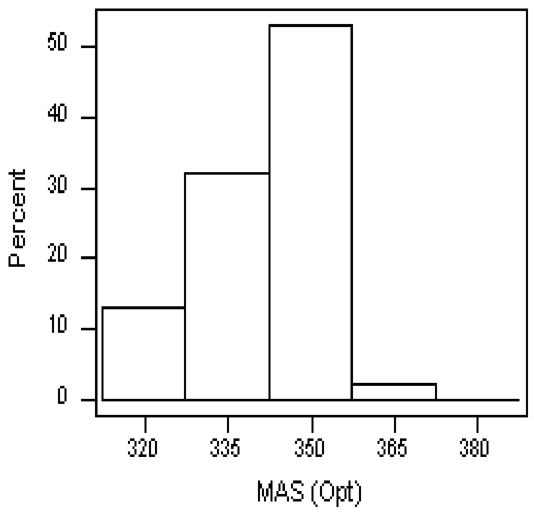


(b) MAS

Fig. 6.8 AS and MAS histograms for a 400-point network over 100 trials - Model 1



(a) AS



(b) MAS

Fig. 6.9 AS and MAS histograms for an 800-point network over 100 trials - Model 1

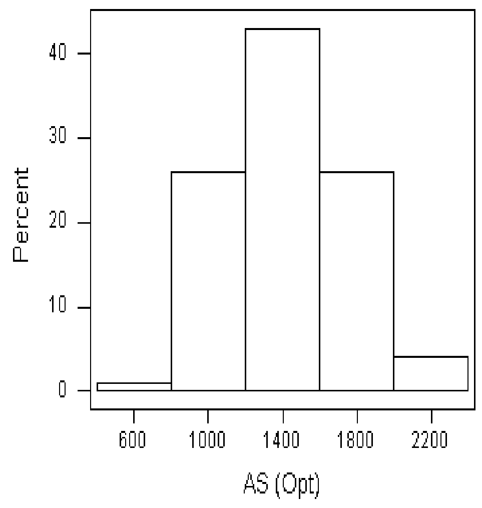
Model 2

In this model, a U-shape obstacle was placed in the environment and the environment was made more complex by introducing other objects. Furthermore, the start and the end nodes were placed such that only one path existed between them. Note that the *z* coordinates of the search space were restricted to the top and the bottom of the obstacles (see Fig. 6.6).

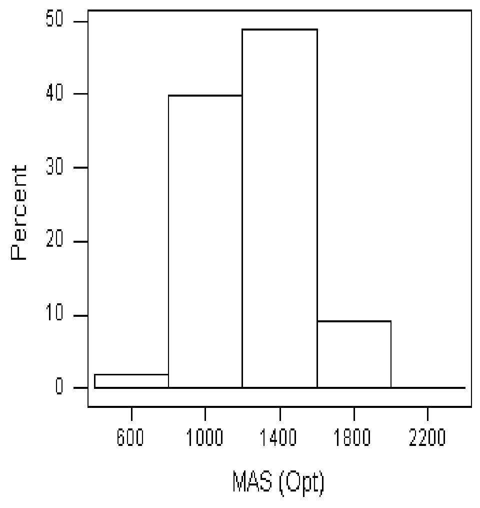
$X_{min} = -300, X_{max} = 400, Y_{min} = 0, Y_{max} = 100, Z_{min} = -300, Z_{max} = 400,$ $S = (50, 25, -50), E = (350, 25, -50)$, Radius ($r$) = 5, Maximum number of turns an ant can perform within a cycle (N_T) = 100

Table 6.7 Comparison of best lengths obtained in AS and MAS – Model 2

	AS			MAS		
No of points	200	400	800	200	400	800
Time *T* (in Sec.)	10	25	150	10	25	150
Avg. Len.	1439.9	1359.6	1909.8	**1262.0**	**1142.5**	**1193.1**
Std. Dev.	325.4	331.3	644.4	**253.1**	**227.3**	**353.3**
Minimum Len.	**752.8**	**782.5**	714.8	796.7	799.1	**682.9**
Maximum Len.	2225.2	2295.4	3433.7	**1896.4**	**1816.9**	**2402.8**
First Quartile	1170.3	1115.1	1424.1	**1040.2**	**969.0**	**935.5**
Median	1452.1	1312.0	1844.3	**1246.8**	**1076.1**	**1117.5**
Third Quartile	1683.8	1586.7	2364.7	**1453.5**	**1310.2**	**1339.0**
Range	1472.4	1512.9	2718.9	**1099.7**	**1017.8**	**1719.9**

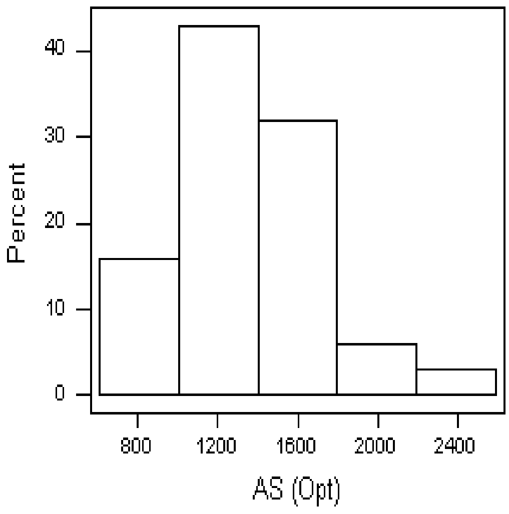


(a) AS

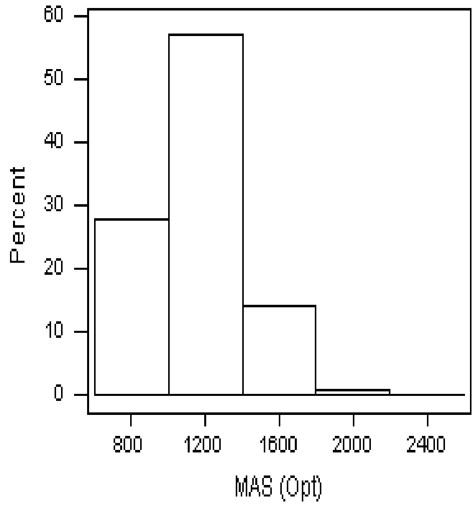


(b) MAS

Fig. 6.10 AS and MAS histograms for a 200-point network over 100 trials - Model 2

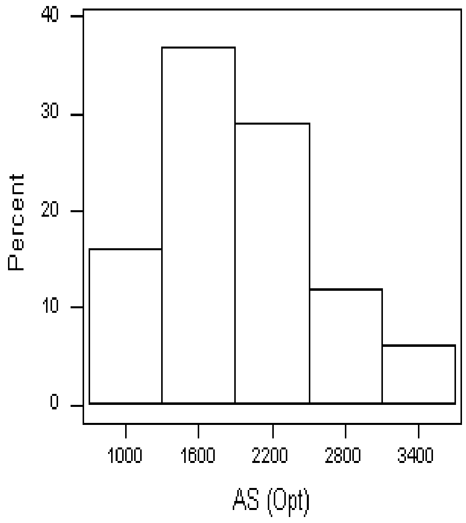


(a) AS

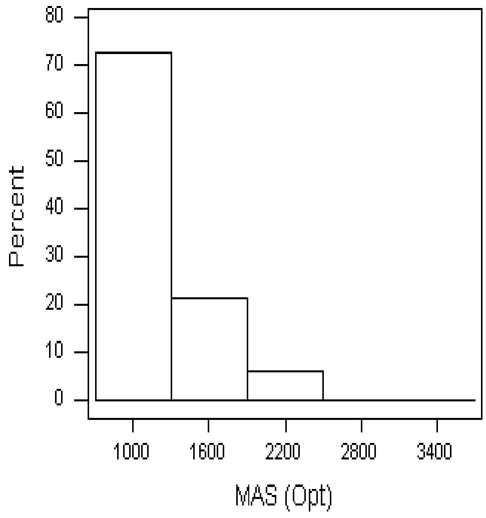


(b) MAS

Fig. 6.11 AS and MAS histograms for a 400-point network over 100 trials - Model 2



(a) AS



(b) MAS

Fig. 6.12 AS and MAS histograms for an 800-point network over 100 trials - Model 2

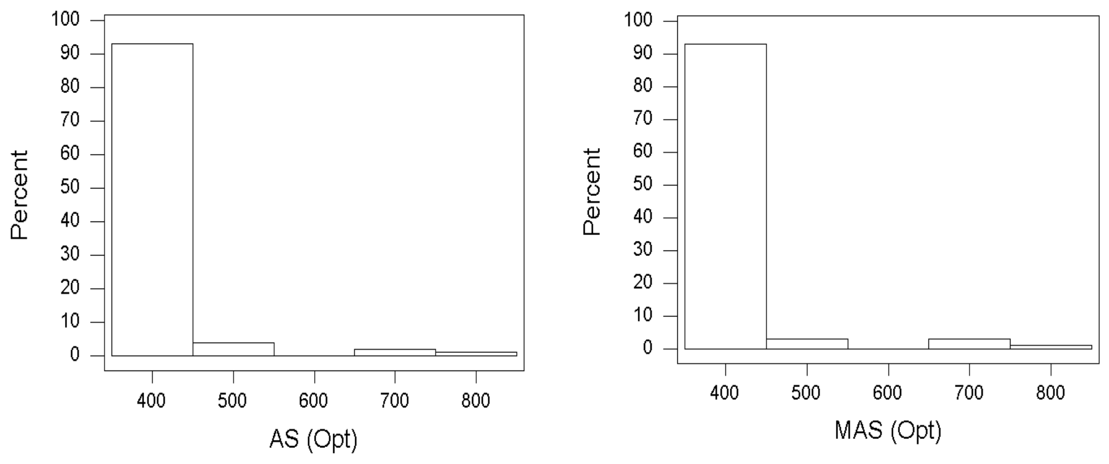
Model 3

In this model, there were two narrow passages for the routing of the optimum path between the start and the end nodes (see Fig. 6.6) and the environment was made more complex by introducing other objects. Note that the z coordinates of the search space were restricted to the top and the bottom of the obstacles.

$X_{min} = -125$, $X_{max} = 400$, $Y_{min} = 0$, $Y_{max} = 100$, $Z_{min} = -250$, $Z_{max} = 200$, $S = (-75, 50, -60)$, $E = (250, 50, -60)$, Radius (r) = 5, Maximum number of turns an ant can perform within a cycle (N_T) = 100

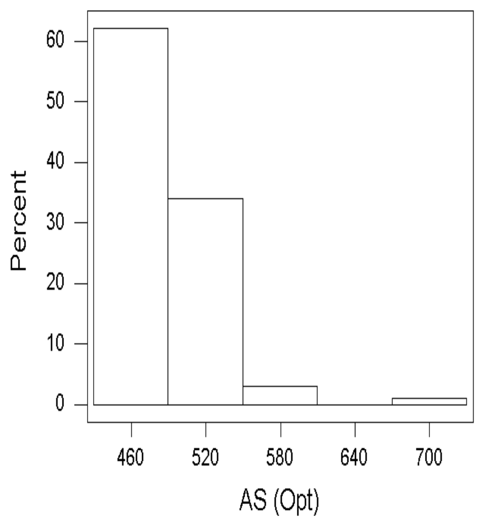
Table 6.8 Comparison of best lengths obtained in AS and MAS – Model 3

	AS			MAS		
No of points	200	400	800	200	400	800
Time *T* (in Sec.)	1	5	20	1	5	20
Avg. Len.	**437.66**	489.07	447.55	440.23	**475.96**	**444.13**
Std. Dev.	**57.73**	33.24	12.10	62.58	**14.58**	**10.56**
Minimum Len.	*423.27*	*463.37*	*432.96*	*423.27*	*463.37*	*432.96*
Maximum Len.	816.62	701.30	482.54	**760.89**	**517.25**	**465.08**
First Quartile	*423.27*	*463.37*	*432.96*	*423.27*	*463.37*	*432.96*
Median	*423.27*	485.49	*450.43*	*423.27*	**468.15**	*450.43*
Third Quartile	**423.27**	494.29	456.12	430.34	**485.49**	**450.48**
Range	393.35	237.93	49.58	**337.62**	**53.88**	**32.12**

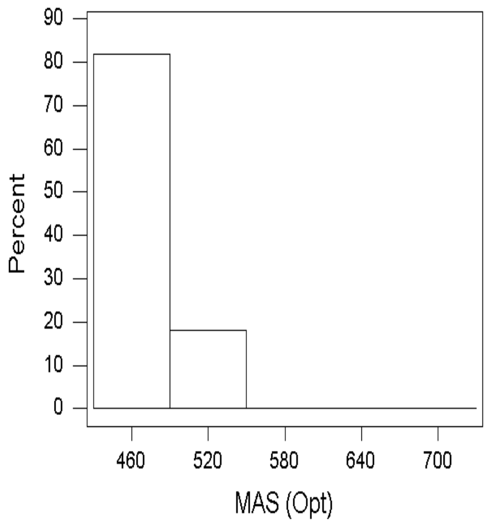


(a) AS (b) MAS

Fig. 6.13 AS and MAS histograms for a 200-point network over 100 trials - Model 3

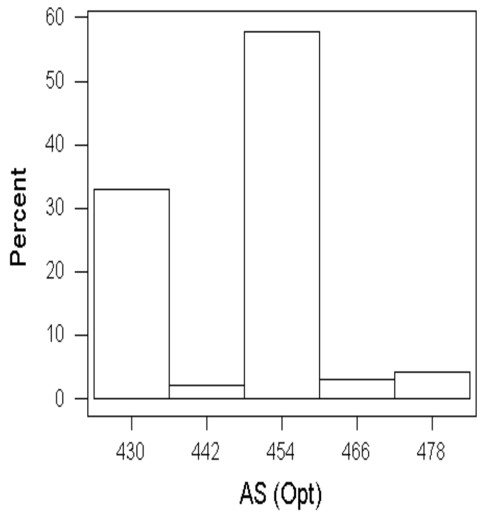


(a) AS

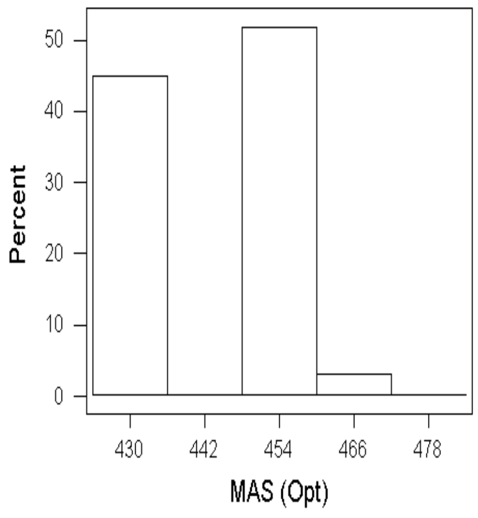


(b) MAS

Fig. 6.14 AS and MAS histograms for a 400-point network over 100 trials - Model 3



(a) AS



(b) MAS

Fig. 6.15 AS and MAS histograms for an 800-point network over 100 trials - Model 3

Discussion (Quality of the best path obtained)

When comparing the results obtained in Tables 6.6 – 6.8, MAS outperforms AS according to the statistics except in a small number of cases. Furthermore, it is noticeable that MAS produces good results when the routing problem becomes more complex. For example, MAS produces better statistics or equal results when networks of 800 random points were used. When comparing the average lengths of the two algorithms, MAS produces better average results except in one case. In most cases, the standard deviation obtained by MAS is low which implies that variation between the solutions produced by MAS is low. This is also confirmed by the range statistic values. When considering the minimum lengths obtained, AS produced the better value in one case, MAS produced the better value in another case and in 5 cases the results were the same for both algorithms. Normally it is difficult to compare single results as both methods are stochastic and hence, for a particular trial, one algorithm may behave very well by chance. Quartile values also indicate how the optimum lengths obtained from the two algorithms are distributed. For example, the third quartiles of AS and MAS for the 800-point network of Model 2, are 2364.7 and 1339.0 (see Table 6.7) respectively. This implies that 75% of the lengths obtained for MAS are less than 1339.0, whereas 75% of the lengths for AS are below a much higher value of 2364.7.

When comparing the histograms of the obtained best lengths (see Figs. 6.7 – 6.15) for AS and MAS, it is noticeable that in most cases, the percentages of the leftmost bars of the histograms for MAS are higher than the corresponding percentages for AS. For example, in Fig. 6.12, this value is nearly 71% for MAS and the corresponding value for AS is about 15%. This indicates that MAS produces better solutions than AS. The minimum value (782.5) for AS for the network of 400 points for Model 2 (see Table 6.7) is less than the corresponding value for MAS (799.1). However, when comparing the corresponding histograms (see Fig. 6.11), the percentage of the leftmost bars for MAS (about 28%) is higher than the respective value for AS (about 16%). It is evident that MAS produces good results for these statistics when the number of points used is higher or the problem becomes complex. Furthermore, it can be noted that the percentages of the rightmost bars of the histograms of MAS are zero for most cases.

In addition to these results, a two-tailed t-test with a 95% confidence level was applied to both methods for the following hypotheses:

$$H_0: \mu_{MAS} \neq \mu_{AS} \text{ Vs. } H_1: \mu_{MAS} < \mu_{AS}$$

where H_0 is the null hypothesis, H_1 is the alternative hypothesis, μ_{MAS} and μ_{AS} are the mean best lengths of MAS and AS, respectively.

Tables 6.9 – 6.11 show the test statistics for the two-tailed t-test for the 3 models. In these tables, 95% confidence intervals for (μ_{MAS} - μ_{AS}) are also included. Negative numbers in these intervals (highlight in bold) imply that the first mean (μ_{MAS}) is smaller than the second mean (μ_{AS}) or the optimum lengths produced by MAS are smaller than those for AS with a 95% confidence level.

Table 6.9 Two-tailed t-test and confidence interval – Model 1

No of points		200	400	800
95% Confidence Interval (μ_{MAS} - μ_{AS})		(-2.36, 1.25)	**(-3.98, -1.32)**	**(-10.5, -4.3)**
T value		-0.61	-3.94	-4.77
Degree of freedom		197	114	192
Alternative hypothesis	Accept. Interval	(0, 0.05)	(0, 0.05)	(0, 0.05)
	P value	0.27	0.0001	0.0000

Table 6.10 Two-tailed t-test and confidence interval – Model 2

No of points		200	400	800
95% Confidence Interval (μ_{MAS} - μ_{AS})		**(-259, -97)**	**(-296, -138)**	**(-862, -572)**
T value		T = -4.31	-5.41	-9.75
Degree of freedom		186	175	153
Alternative hypothesis	Accept. Interval	(0, 0.05)	(0, 0.05)	(0, 0.05)
	P value	0.0000	0.0000	0.0000

Table 6.11 Two-tailed t-test and confidence interval – Model 3

No of points		200	400	800
95% Confidence Interval (μ_{MAS} - μ_{AS})		(-14.2, 19.4)	**(-20.3, -5.9)**	**(-6.6, -0.2)**
T value		0.30	-3.61	-2.13
Degree of freedom		196	135	194
Alternative hypothesis	Accept. Interval	(0, 0.05)	(0, 0.05)	(0, 0.05)
	P value	0.62	0.0002	0.017

According to these results, there is sufficient evidence, at the 0.05 level of significance, to conclude that the average best lengths for MAS is smaller than the corresponding value for AS, in most cases. It is noticeable that these results are significantly better when the complexity of the problem is increased.

6.2.2 Number of turns

In this section, the numbers of turns (iterations) performed within the specified time (*T*) for AS and MAS are statistically analyzed. The following statistics are considered:

Avg. No of Turns – Average number of turns over 100 trials
Std. Dev. – Standard deviation of the turns
Min. No of Turns – Minimum number of turns
Max. No of Turns – Maximum number of turns

A highlighted (in bold) statistical value in the tables (see Tables 6.12 – 6.14) indicates that it exceeds the corresponding value of the other algorithm.

Experimental results (Number of turns)

Table 6.12 Comparison of number of turns for AS and MAS – Model 1 (see Fig. 6.6)

	AS			MAS		
No of points	200	400	800	200	400	800
Time *T* (in Sec.)	5	6	50	5	6	50
Avg. No of Turns	886.33	428.41	1006.6	**1135.5**	**690.12**	**1998.2**
Std. Dev.	24.10	42.06	**42.3**	**21.8**	**23.94**	61.2
Min. No of Turns	807.00	202.0	902.0	**1084.0**	**604.0**	**1800.0**
Max. No of Turns	923.00	500.0	1100.0	**1181.0**	**704.0**	**2104.0**

Table 6.13 Comparison of number of turns for AS and MAS – Model 2 (see Fig. 6.6)

	AS			MAS		
No of points	200	400	800	200	400	800
Time *T* (in Sec.)	10	25	150	10	25	150
Avg. No of Turns	963.47	605.09	489.42	**1823.4**	**1916.0**	**2247.6**
Std. Dev.	**35.46**	**11.98**	**45.29**	102.4	249.7	607.9
Min. No of Turns	900.0	591.00	400.0	**1605.0**	**1500.0**	**1109.0**
Max. No of Turns	1020.0	700.00	600.0	**2062.0**	**2502.0**	**3900.0**

Table 6.14 Comparison of number of turns for AS and MAS – Model 3 (see Fig. 6.6)

	AS			MAS		
No of points	200	400	800	200	400	800
Time *T* (in Sec.)	1	5	20	1	5	20
Avg. No of Turns	213.72	577.99	818.64	**241.6**	**883.00**	**1565.6**
Std. Dev.	**6.44**	28.44	**36.55**	12.92	**27.57**	50.2
Min. No of Turns	200.0	504.0	703.00	**219.0**	**804.0**	**1404.0**
Max. No of Turns	227.0	602.0	900.00	**272.0**	**945.0**	**1673.0**

Discussion (Number of turns)

According to these tables, it is evident that MAS performs a greater number of turns (or iterations), and hence tends to give better solutions than AS, within the given fixed time. In all cases, the average number of turns for MAS is higher than the corresponding value for AS. For example, this value for MAS is 2247.6, for the network of 800 points of Model 2 (see Table 6.13), while the corresponding value for AS is 489.42. This implies that MAS is able to explore more good solutions within a given time. It is also noticeable that the maximum number of turns taken by AS is less than the minimum number of turns taken by MAS except in one case. This means that MAS almost always performs a higher number of turns than AS within the given time. This also shows that MAS has the ability to explore a greater scope of the search space.

6.2.3 Total No of nodes in all contender node lists and average No of alternatives

The total number of nodes in all the contender node lists [$J_i^*(r)$] (see section 5.4) during each cycle is given by:

$$n\_AS = \sum_{i=1}^{no\_of\_ants} \sum_{r \in i\,th\ ant\ path} n(J_i(r)) \quad for \quad AS \qquad 6.1$$

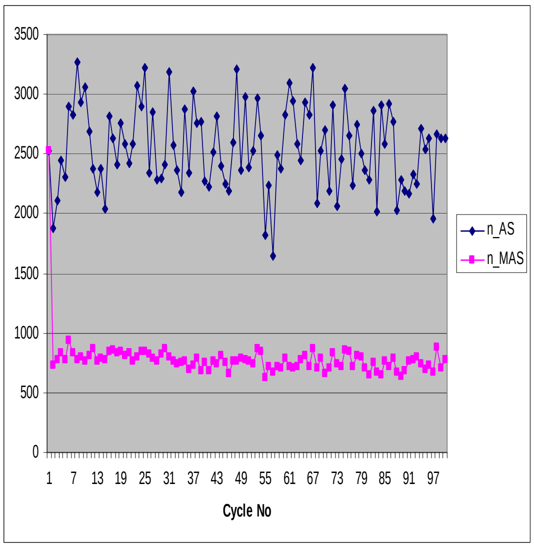
and

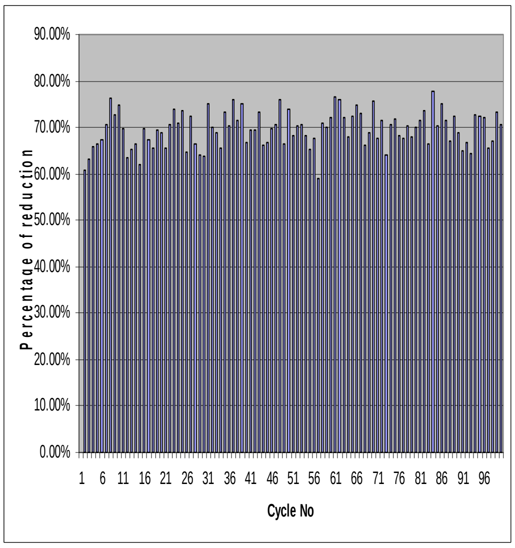
$$n\_MAS = \sum_{i=1}^{no\_of\_ants} \sum_{r \in i\,th\ ant\ path} n(J_i^*(r)) \quad for \quad MAS \qquad 6.2$$

where *n(A)* is the number of elements in set A.

For this experiment, Model 1 with the same parameter values as in the first 2 experiments (sections 6.2.1 and 6.2.2) was used. However, the stopping criterion was changed: in this case, the program was run for 100 cycles (or 10,000 turns). The program was set to find the best path by using AS and recording both *n_AS* and *n_MAS*. When computing $J_i^*(r)$, the current best value produced by AS rather than MAS is used as the program is unable to run the two algorithms simultaneously.

<u>Experimental results</u> <u>(Total No of nodes in all contender node lists)</u>

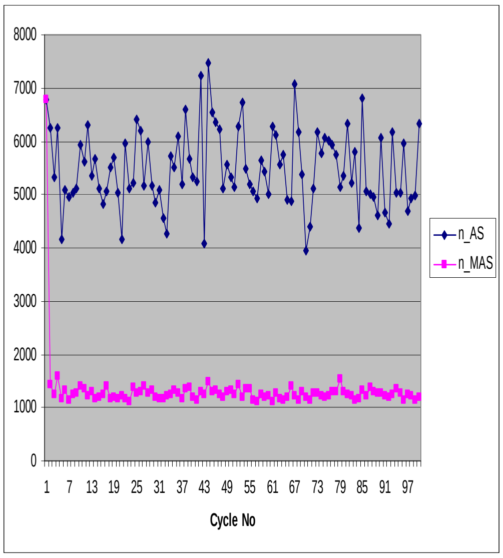


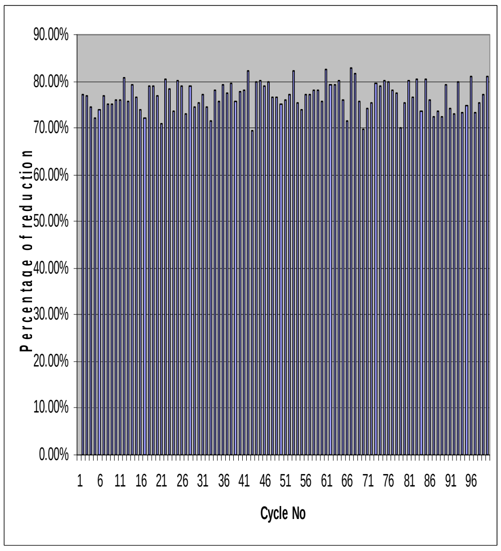


(a) *n_AS* vs. *n_MAS*

(b) Percentage of Node Reduction

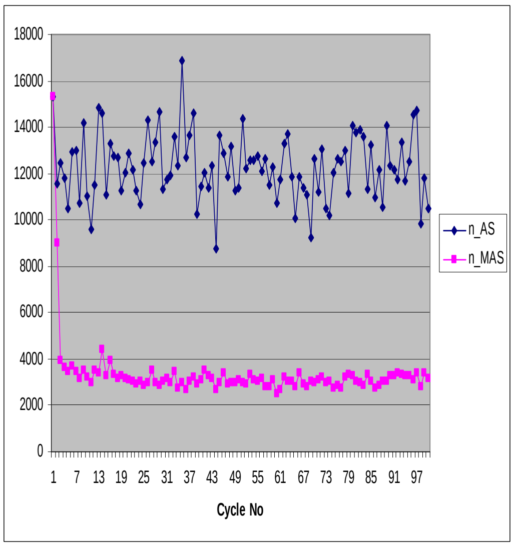
Fig. 6.16 Comparison of number of nodes in contender node lists [$J_i^*(r)$] per cycle for AS and MAS for a 200-point network - Model 1 (Typical run)

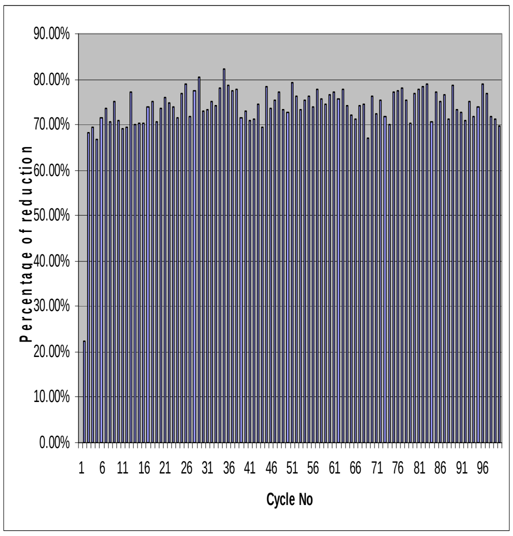




(a) *n_AS* vs. *n_MAS* (b) Percentage of Node Reduction

Fig. 6.17 Comparison of number of nodes in contender node lists [$J_i^*(r)$] per cycle for AS and MAS for a 400-point network - Model 1 (Typical run)





(a) *n_AS* vs. *n_MAS* (b) Percentage of Node Reduction

Fig. 6.18 Comparison of number of nodes in contender node lists [$J_i^*(r)$] per cycle for AS and MAS for a 800-point network - Model 1 (Typical run)

Discussion (Total No of nodes in all contender node lists)

According to Figs. 6.16 – 6.18, it is clear that *n_MAS* values are very much lower than *n_AS* values. The right-hand side of each figure shows the percentage of node reduction if MAS is used instead of AS. This shows that the total number of nodes in all contender node lists per cycle is reduced by more than 60% after 3 or 4 cycles. This is also a good indication that MAS is able to search for good solutions and removes the unwanted edges from the contender node lists and hence avoids reinforcing pheromone values of unwanted edges. Note that during the first cycle, *n_AS* and *n_MAS* are the same as no information of the best length is available.

<u>**Experimental results** (Average No of alternatives)</u>

Further, the average number of alternatives [62] that an ant has for choosing the next node was computed for an 800-point network of Model 1. Although some edges may be connected to a node, if the pheromone levels of these edges are smaller than a threshold λ after a number of cycles, the probability of the ants selecting such edges is negligible. These edges were removed from the count.

For ant *i* that is placed on node *r* let

$$D_{AS}^{(i)}(r) = \{ s \mid p_i(r,s) > \lambda,\ s \in J_i(r) \} \qquad 6.3$$

and

$$D_{MAS}^{(i)}(r) = \{ s \mid p_i(r,s) > \lambda,\ s \in J_i^{*}(r) \} \qquad 6.4$$

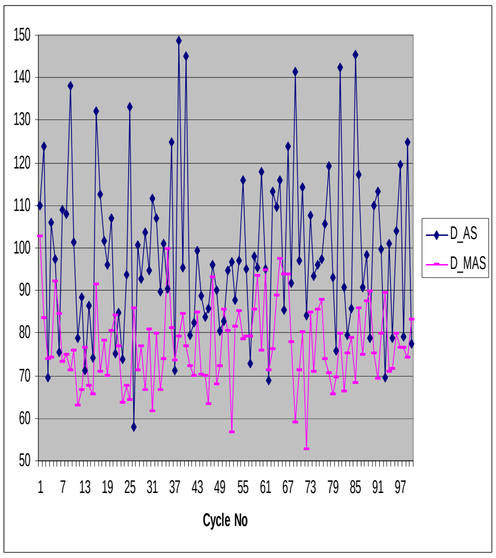
be the numbers of possible next nodes (that have not yet been visited) that have a probability > λ of being chosen, for AS and MAS, respectively. Then, the average numbers of alternatives with probability > λ during a cycle are

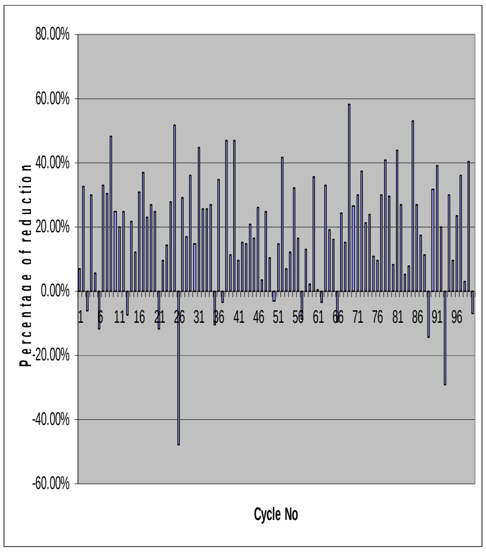
$$D\_AS = \frac{\sum_{i=1}^{no\_of\_ants} \sum_{r \in i\,th\ ant\ path} n(D_{AS}^{(i)}(r))}{no\_of\_ants} \quad for\ AS \qquad 6.5$$

and

$$D\_MAS = \frac{\sum_{i=1}^{no\_of\_ants} \sum_{r \in i\,th\ \ ant\ \ path} n(D_{MAS}^{(i)}(r))}{no\_of\_ants} \quad for\ MAS \tag{6.6}$$

where *n(A)* is the number of elements in set A.





(a) *D_AS* vs. *D_MAS* (b) Percentage reduction of alternatives

Fig. 6.19 Comparison of average numbers of alternatives, *D*, for an 800-point network of Model 1 (Typical run) (λ = 0.01)

Discussion (Average No of alternatives)

According to Fig. 6.19, the average number of alternatives for MAS is well below that for AS in most of the cycles. The right-hand side of the figure shows the percentage reduction in the average number of alternatives *D* for MAS with respect to AS. Negative percentages indicate that the *D* values for AS are less than for MAS. Furthermore, the average values of the *D* values for all cycles for AS and MAS are 98.57 (Std. Dev. 19.15) and 77.11 (Std. Dev. 9.61) respectively. According to these average values, each ant has roughly 99 and 77 alternatives per cycle for AS and MAS, respectively.

6.2.4 Time spent on a cycle

In this experiment, time spent on each cycle was measured. For this, 200-, 400- and 800-point networks of Model 1 were used with the same parameters as for experiment 1 (See Section 6.2). The stopping criterion was changed such that the program was run for 100 cycles.

<u>**Experimental results** (Time spent on a cycle)</u>

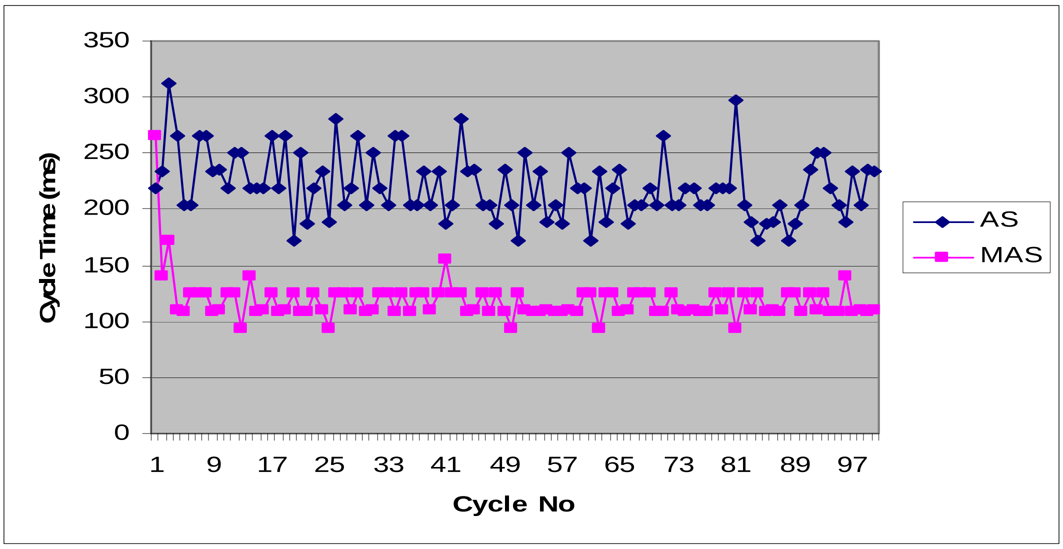


Fig. 6.20 Comparison of cycle times, for a 200-point network of Model 1 (Sample run)

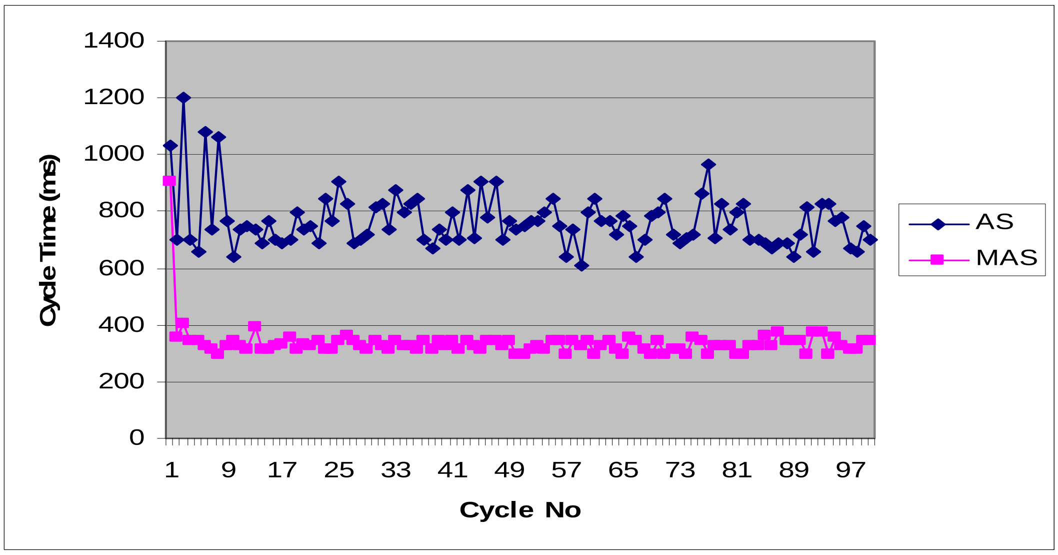


Fig. 6.21 Comparison of cycle times, for a 400-point network of Model 1 (Sample run)

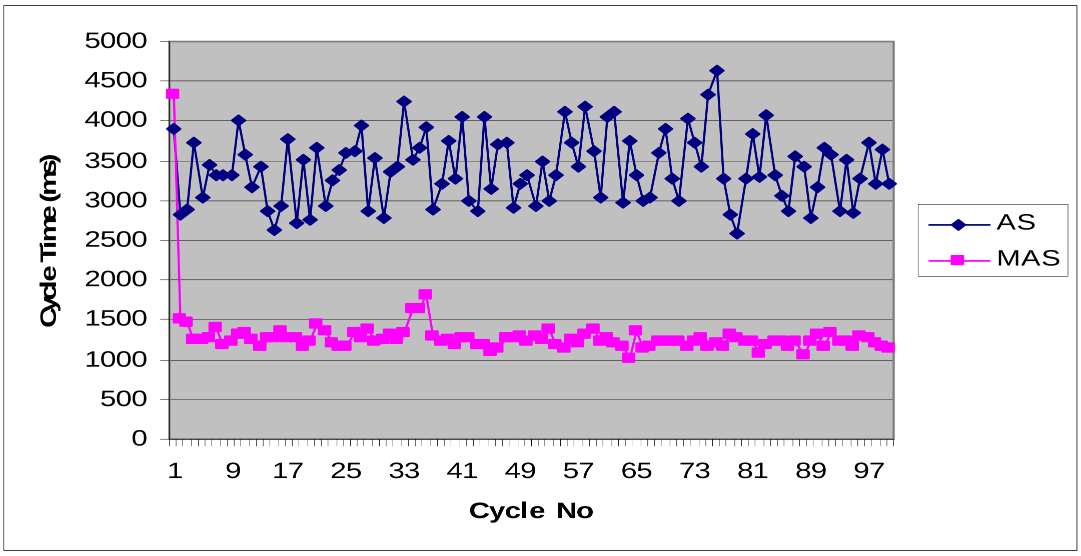


Fig. 6.22 Comparison of cycle times, for a 800-point network of Model 1 (Sample run)

Table 6.15 Comparison of average cycle time for AS and MAS – Model 1

No of Points	200		400		800	
Algorithm	AS	MAS	AS	MAS	AS	MAS
Average Cycle Time (ms)	220.47	**118.13**	765.15	**335.63**	3398.6	**1288.75**
Std. Dev. (Cycle Time)	29.13	**19.19**	97.07	**61.82**	433.41	**327.35**

Discussion (Time spent on a cycle)

Figs. 6.20 – 6.22 show that cycle times for MAS are very low compared to those for AS in all the three cases. Table 6.15 shows the average cycle time and its standard deviation. These figures demonstrate that the average cycle time can be reduced approximately by half when using MAS. Hence, for a given run time, it is possible to run a higher number of cycles for MAS than for AS and thus achieve better results for MAS than for AS.

6.3 Results Obtained with the Introduction of Explorer Ants for Avoiding Stagnation

In this experiment, the introduction of two types of ants (explorers and followers) into MAS (N_MAS) in order to avoid the stagnation behaviour is examined (see section 5.5

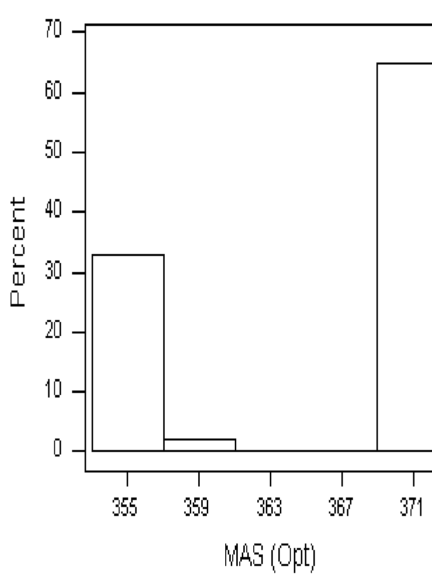
also).

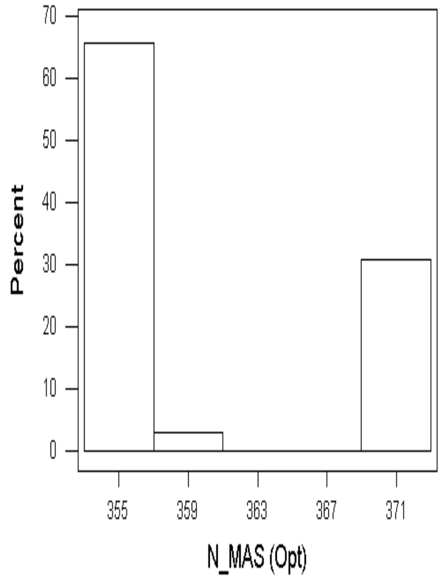
Each of the three models (see Fig. 6.6) was tested with a 200-, 400- and 800- random point network for 100 trials. For each trial, 5 explorers and 5 followers were used. Values for the other parameters and the stopping criterion were the same as for experiment 1 in section 6.2.

Experimental results (Introduction of Explorer Ants for Avoiding Stagnation)

Table 6.16 Comparison of best lengths obtained by MAS and N_MAS – Model 1

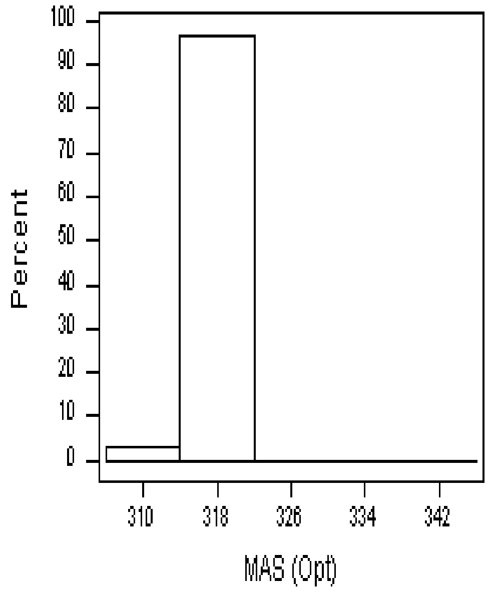
	MAS			N_MAS		
No of points	200	400	800	200	400	800
Time *T* (in Sec.)	5	6	50	5	6	50
Avg. Len.	365.06	315.14	342.03	**360.36**	**314.72**	**340.94**
Std. Dev.	6.60	1.84	**10.04**	**6.42**	**1.64**	10.14
Minimum Len.	*355.97*	*310.62*	**316.27**	*355.97*	*310.62*	318.92
Maximum Len.	*369.87*	*321.27*	*358.55*	*369.87*	*321.27*	*358.55*
First Quartile	*355.97*	*314.68*	333.51	*355.97*	*314.68*	**332.49**
Median	369.87	*314.68*	343.97	**355.97**	*314.68*	**343.78**
Third Quartile	*369.87*	*314.68*	**349.69**	*369.87*	*314.68*	350.64
Range	*13.90*	*10.65*	42.28	*13.90*	*10.65*	**39.63**

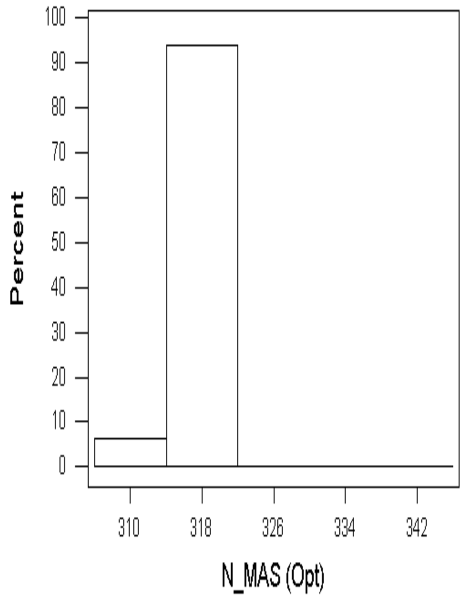




(a) MAS (b) N_MAS

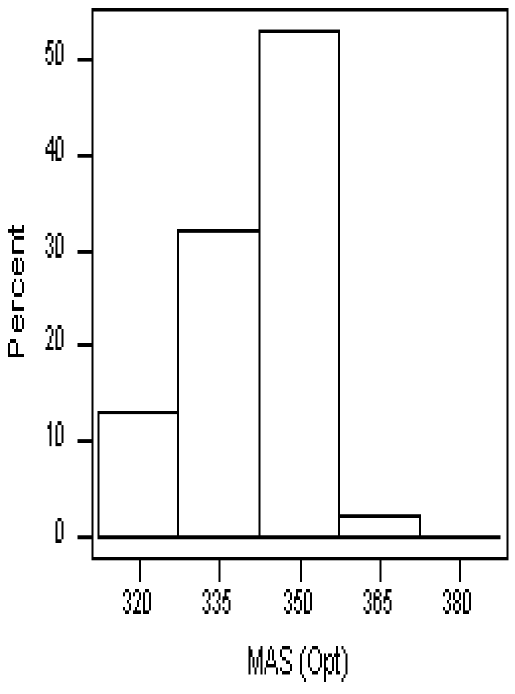
Fig. 6.23 MAS and N_MAS histograms for a 200-point network over 100 trials - Model 1

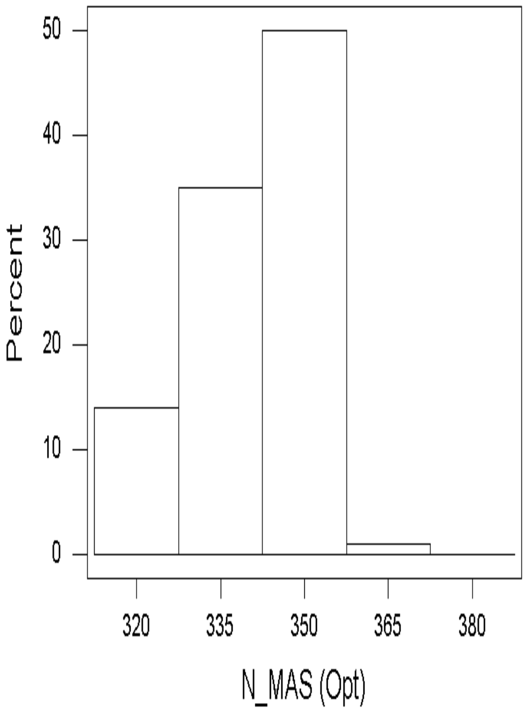




(a) MAS (b) N_MAS

Fig. 6.24 MAS and N_MAS histograms for a 400-point network over 100 trials - Model 1



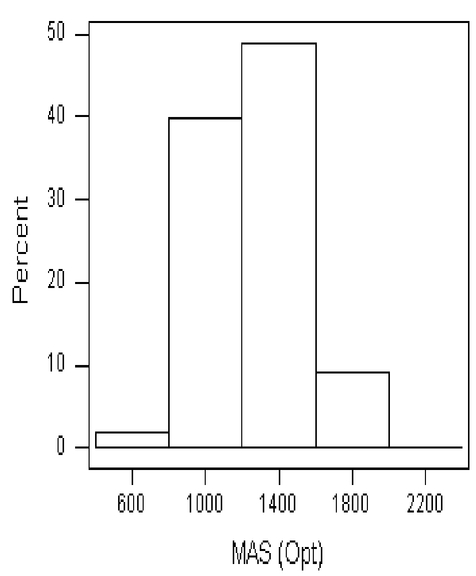


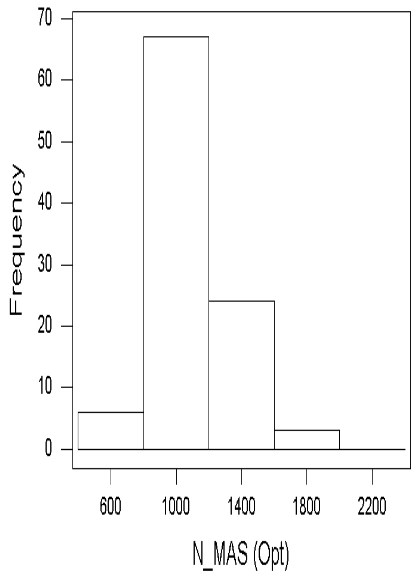
(a) MAS (b) N_MAS

Fig. 6.25 MAS and N_MAS histograms for a 800-point network over 100 trials - Model 1

Table 6.17 Comparison of best lengths obtained in MAS and N_MAS – Model 2

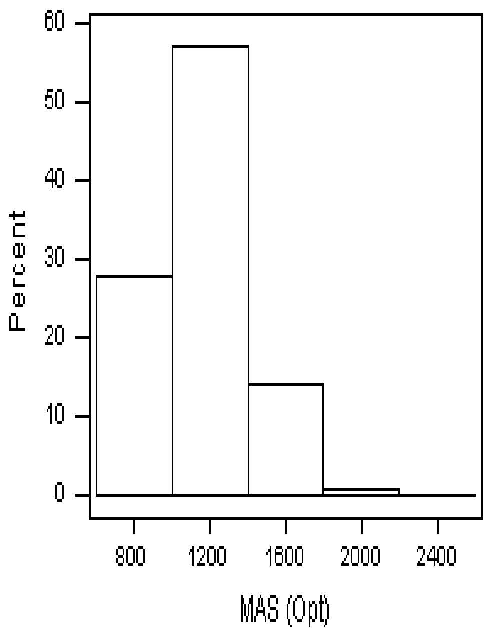
	MAS			N_MAS		
No of points	200	400	800	200	400	800
Time *T* (in Sec.)	10	25	150	10	25	150
Avg. Len.	1262.0	1142.5	1193.1	**1098.0**	**1066.3**	**1104.5**
Std. Dev.	253.1	227.3	353.3	**208.9**	**180.7**	**322.7**
Minimum Len.	796.7	799.1	682.9	**752.3**	**784.7**	**671.2**
Maximum Len.	1896.4	1816.9	**2402.8**	**1691.9**	**1646.2**	2533.5
First Quartile	1040.2	969.0	935.5	**958.4**	**952.8**	**863.4**
Median	1246.8	1076.1	1117.5	**1085.9**	**1032.0**	**993.9**
Third Quartile	1453.5	1310.2	1339.0	**1219.2**	**1153.0**	1360.4
Range	1099.7	1017.8	**1719.9**	**939.6**	**861.5**	1862.3

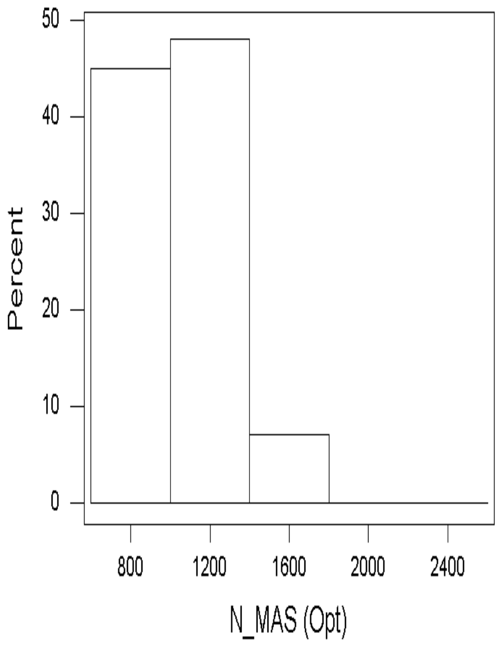




(a) MAS (b) N_MAS

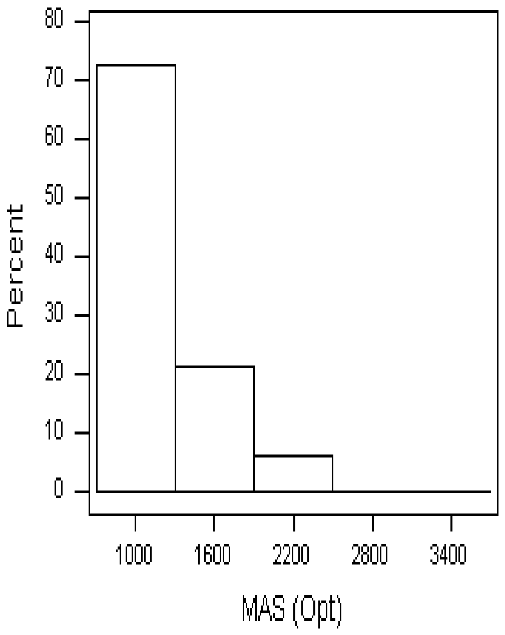
Fig. 6.26 MAS and N_MAS histograms for a 200-point network over 100 trials - Model 2



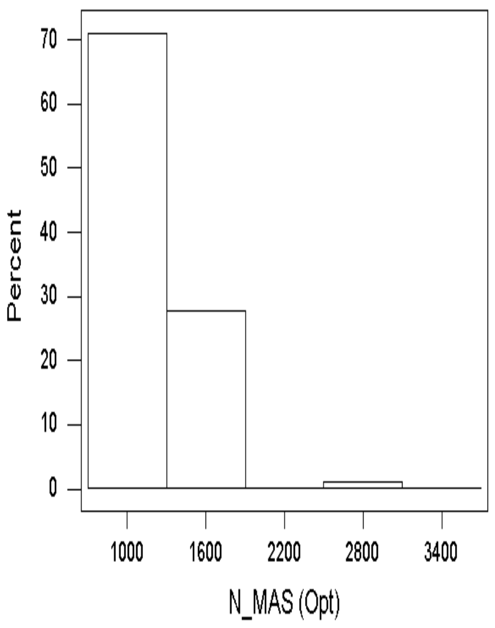


(a) MAS (b) N_MAS

Fig. 6.27 MAS and N_MAS histograms for a 400-point network over 100 trials - Model 2



(a) MAS

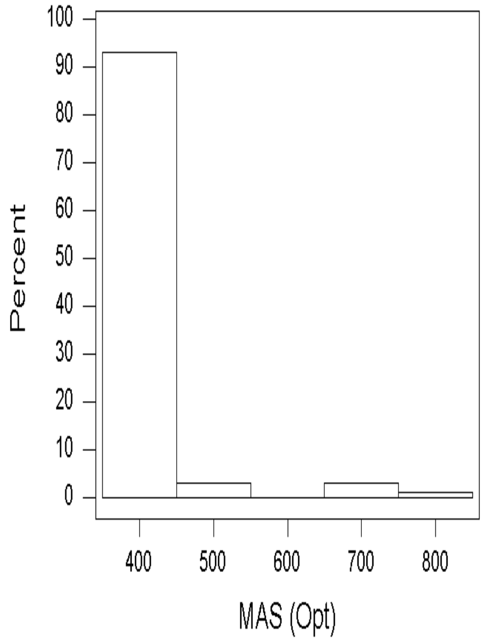


(b) N_MAS

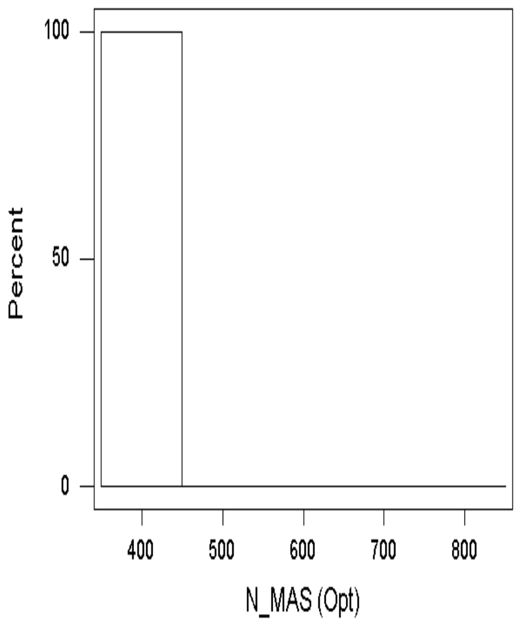
Fig. 6.28 MAS and N_MAS histograms for a 800-point network over 100 trials - Model 2

Table 6.18 Comparison of best lengths obtained in of MAS and N_MAS – Model 3

	MAS			N_MAS		
No of points	200	400	800	200	400	800
Time *T* (in Sec.)	1	5	20	1	5	20
Avg. Len.	440.23	475.96	444.13	**423.27**	**470.34**	**440.89**
Std. Dev.	62.58	14.58	10.56	**0.00**	**11.34**	**9.59**
Minimum Len.	423.27	*463.37*	*432.96*	**423.27**	*463.37*	*432.96*
Maximum Len.	760.89	*517.25*	*465.08*	**423.27**	*517.25*	*465.08*
First Quartile	423.27	*463.37*	*432.96*	**423.27**	*463.37*	*432.96*
Median	423.27	468.15	450.43	**423.27**	**463.37**	**432.96**
Third Quartile	430.34	485.49	450.48	**423.27**	**484.91**	**450.43**
Range	337.62	*53.88*	*32.12*	**0.00**	*53.88*	*32.12*

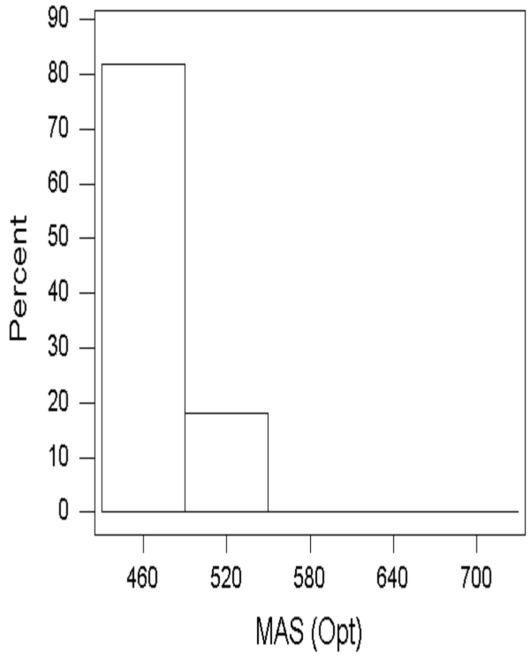


(a) MAS

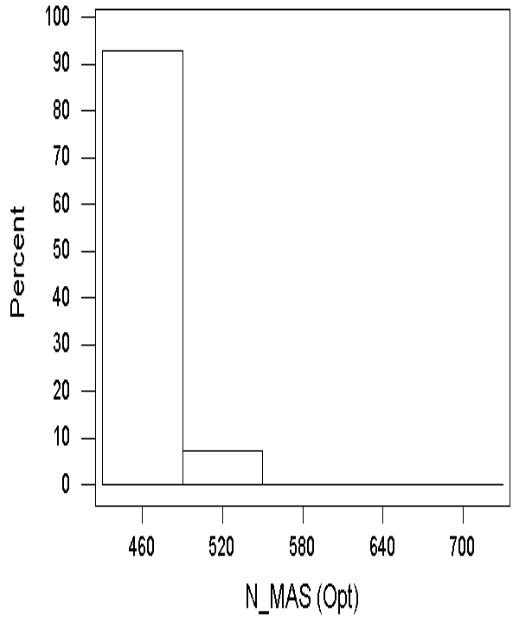


(b) N_MAS

Fig. 6.29 MAS and N_MAS histograms for a 200-point network over 100 trials - Model 3

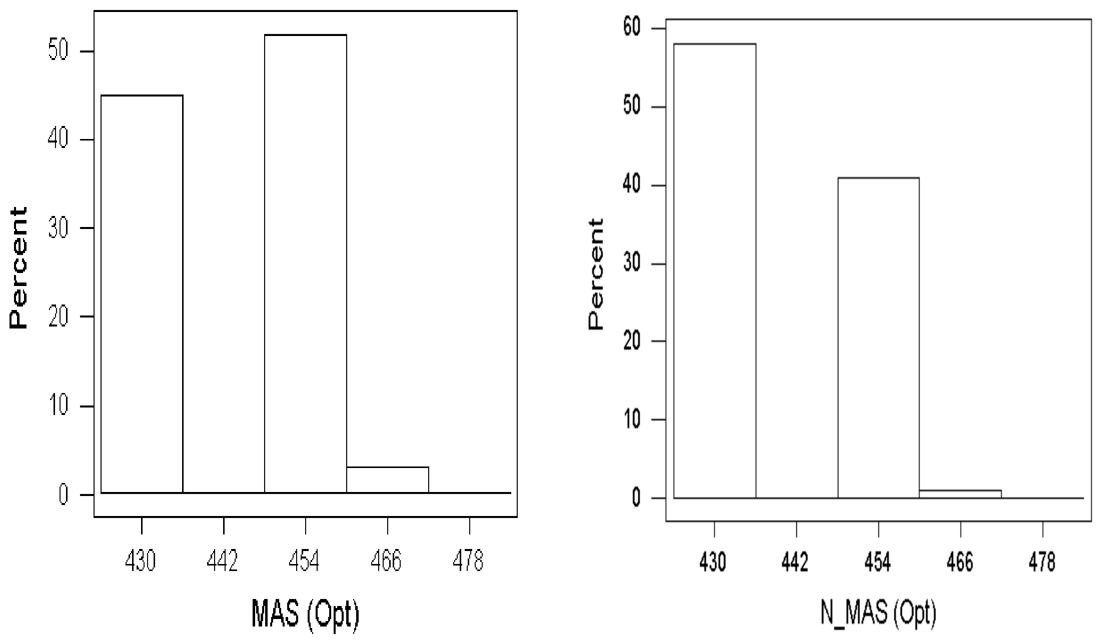


(a) MAS



(b) N_MAS

Fig. 6.30 MAS and N_MAS histograms for a 400-point network over 100 trials - Model 3



(a) MAS (b) N_MAS

Fig. 6.31 MAS and N_MAS histograms for a 800-point network over 100 trials - Model 3

Discussion (Introduction of Explorer Ants for Avoiding Stagnation)

According to the Tables 6.16 – 6.18, the average values of the best lengths are improved when the two types of ants (explorers and followers) are introduced into MAS. Standard deviations of the best lengths are also reduced except in one case. Note that, the standard deviation for the network of 200 points of Model 3 (see Table 6.18) is 0. In this experiment, N_MAS converged to 423.27 for all 100 trials. In addition to these results, the other statistics are also improved or remain the same.

When comparing the histograms of N_MAS and MAS (see Figs. 6.29 – 6.31), it can be seen that the percentage values of the two lowest bars are always higher for N_MAS than for MAS.

Further, the two-tailed t-test with a 95% confidence level was applied to N_MAS with the following hypothesises:

$$H_0: \mu_{N\_MAS} \neq \mu_{MAS} \text{ Vs. } H_1: \mu_{N\_MAS} < \mu_{MAS}$$

where μ_{N_MAS} and μ_{MAS} are the mean best lengths of N_MAS and MAS, respectively. The following tables (see Tables 6.19 – 6.21) show the test statistics of the two-sample t-test for the 3 models. In these tables, 95% confidence intervals for (μ_{N_MAS} - μ_{MAS}) are also included.

Table 6.19 Two-Tailed T-Test and Confidence Interval – Model 1

No of points		200	400	800
95% Confidence Interval (μ_{N_MAS} - μ_{MAS})		**(-6.51, -2.88)**	(-0.90, 0.07)	(-3.9, 1.7)
T value		-5.10	-1.69	-0.77
Degree of freedom		197	195	197
Alternative hypothesis	Accept. Interval	(0, 0.05)	(0, 0.05)	(0, 0.05)
	P value	0.0000	0.047	0.22

Table 6.20 Two-Tailed T-Test and Confidence Interval – Model 2

No of points		200	400	800
95% Confidence Interval (μ_{N_MAS} - μ_{MAS})		**(-229, -99)**	**(-133, -19)**	(-183, 6)
T value		-5.00	-2.62	-1.85
Degree of freedom		191	188	196
Alternative hypothesis	Accept. Interval	(0, 0.05)	(0, 0.05)	(0, 0.05)
	P value	0.0000	0.0047	0.033

Table 6.21 Two-Tailed T-Test and Confidence Interval – Model 3

No of points		200	400	800
95% Confidence Interval (μ_{N_MAS} - μ_{MAS})		**(-29.38, -4.5)**	**(-9.3, -2.0)**	**(-6.05, -0.4)**
T value		-2.71	-3.04	-2.27
Degree of freedom		99	186	196
Alternative hypothesis	Accept. Interval	(0, 0.05)	(0, 0.05)	(0, 0.05)
	P value	0.0040	0.0013	0.012

According to these results, there is sufficient evidence, at the 0.05 level of significance, to conclude that the average best length for N_MAS is smaller than the corresponding value for MAS in most of the cases.

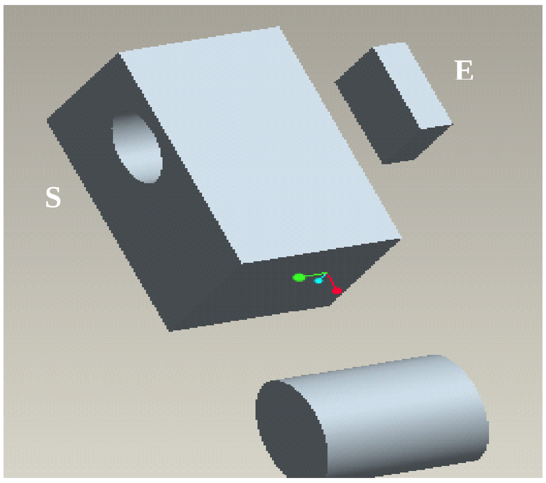
6.4 Results Obtained for Multi-Objective Ant Colony Optimization

In this section, results of the application of multi-objective ant colony optimization algorithms (P-ACO and the proposed PSACO) discussed in sections 4.3.1 and 5.6 for the multi-objective hose routing problem described in section 5.7 are investigated. Both algorithms (P-ACO and PSACO) have been tested on two Pro/Engineer models (see Fig. 6.32). Model 1 consists of a cube containing a hole. Hose segments needed to be laid inside this hole in order to obtain the optimal path, which is blocked by a cubic obstacle, and the end point is placed behind this obstacle. In Model 2, a U-shape obstacle is placed in the environment and the environment is made more complex by introducing other objects. Furthermore, the start and the end points are placed on the opposite sides of the U-shaped obstacle. Note that the z-coordinates of the search space are restricted to the top and the bottom of the obstacles.

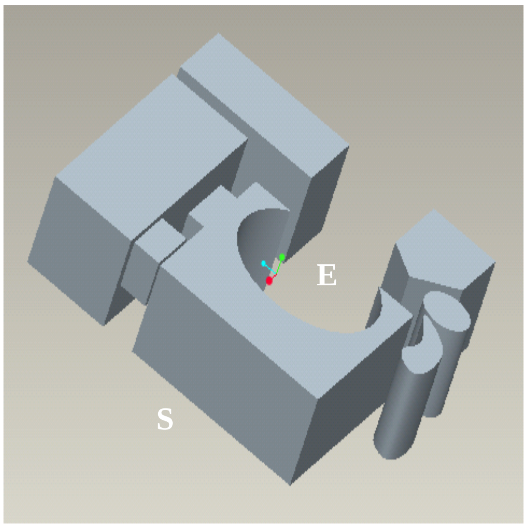
For each model, two methods (P-ACO and PSACO) are compared in terms of illustrative representation methods of non-dominated solutions, performance comparison methods described in sections 3.2 and 3.3, respectively. Further, performances are compared using the computation times of the two algorithms. In section 6.4.7 the application of the methods described in section 3.4 for selecting a final solution from the non-dominated solutions obtained by MOACO algorithms is discussed. Finally, a refining process is described and results of this process are presented in section 6.4.7.

A network of 800 random points has been generated for each model as described in step 2 of section 5.3 and has been stored in a text file. 93198 and 67975 edges have been generated for Model 1 and Model 2, respectively.

Both P-ACO and PSACO have been run twenty times on these networks for 40 cycles (N_c = 40). Each cycle consists of 100 turns (N_T = 100). All the simulations have been carried out on a Pentium IV PC (Processor speed = 3.0 GHz, RAM = 512 MB) in the Microsoft Windows XP environment using Microsoft Visual C++ (.Net version).



(a) Model 1



(b) Model 2

Fig. 6.32 Tested Pro/Engineer Models

Model 1

The parameters for the two algorithms for Model 1 have been set as follows: Number of ants $N = 20$, Archive size $N_A = 10$, Initial pheromone level $\tau_0 = 1$, $\alpha = 1$, $\beta = 5$, $\rho = 0.01$, $q_0 = 0.9$ (only for P-ACO), $\gamma = 2$, $\varepsilon = 10/180 = 0.0556$, Catalogue angles: $\omega_1 = 45^\circ$, $\omega_2 = 90^\circ$, $\omega_3 = 120^\circ$, $\omega_4 = 150^\circ$ and $\omega_5 = 180^\circ$.

Overall, 13 and 20 non-dominated solutions have been generated by P-ACO and PSACO, respectively out of 200 solutions generated in 20 runs.

6.4.1 Illustrative Representation of the Non-Dominated Solutions for Model 1 (see Fig. 6.32)

Figs. 6.33 – 6.35 illustrate the non-dominated solutions using the scatter-plot matrix, the value path and the bar chart methods, respectively.

According to the scatter-plot matrix method both algorithms have a good spread of non-dominated solutions within the objective function's extreme values. Further, PSACO has obtained smaller (better) objective values when compared with the objective values obtained by P-ACO.

The value path method shows the spread of each of the non-dominated solutions in each objective. The objective values of both algorithms do not span the entire range for each objective. This is expected as the objective space of our problem is discontinuous and discrete. For example, for the second objective, f_2 cannot achieve the value 0 as there is no solution with the number of bends $N_B = 1$ (see Fig. 6.32). However, it is noticeable that both algorithms produce solutions with a good spread between the extreme values of each objective. Further, both algorithms produce good trade-off solutions as most of the solutions have a large change of slope between two consecutive objective function bars.

The bar chart method (see Fig. 6.35) also shows the same results as the value path method. The values of each objective function do not spread over their entire region [0, 1] as the objective spaces are not continuous.

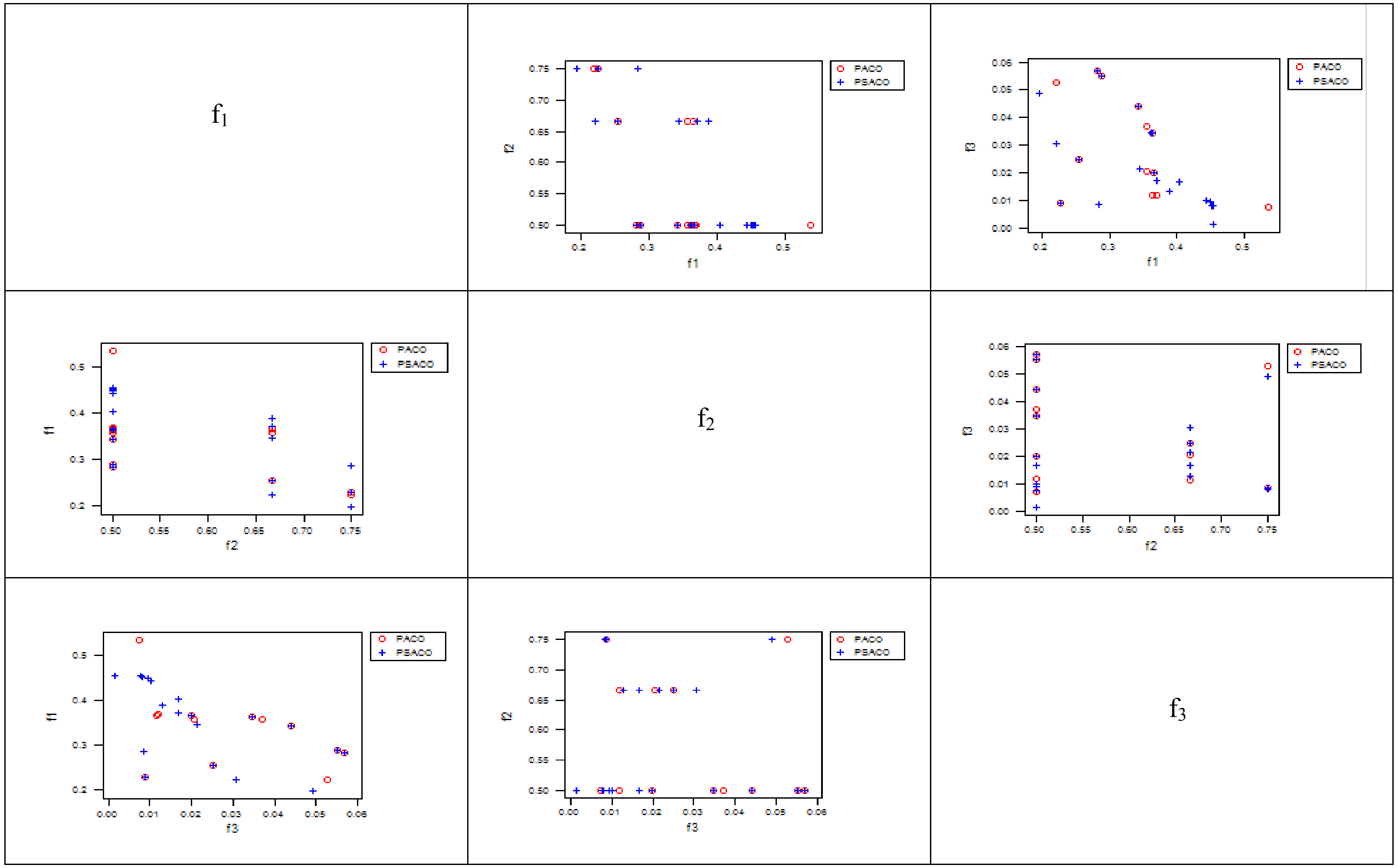
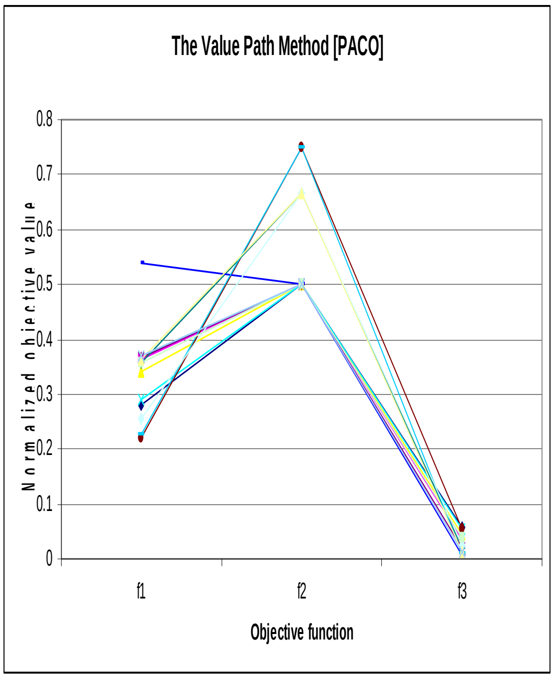


Fig. 6.33 The scatter-plot matrix method for Model 1



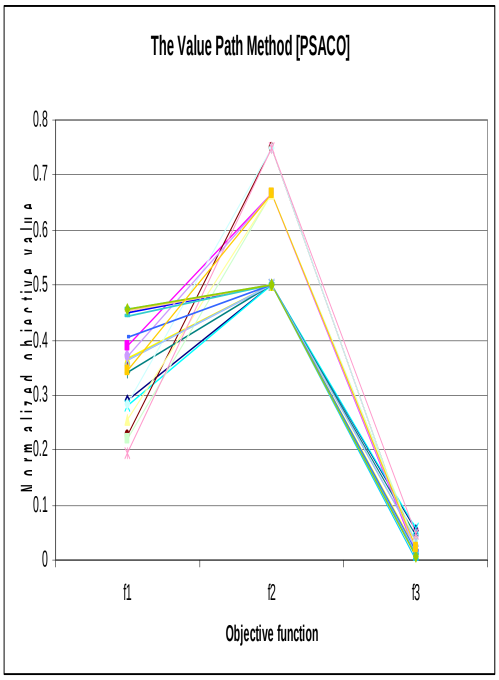
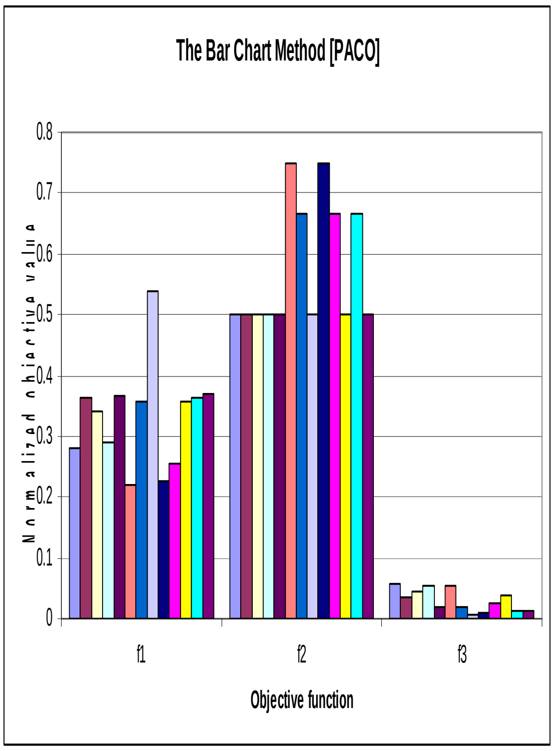


Fig. 6.34 The value path method for Model 1



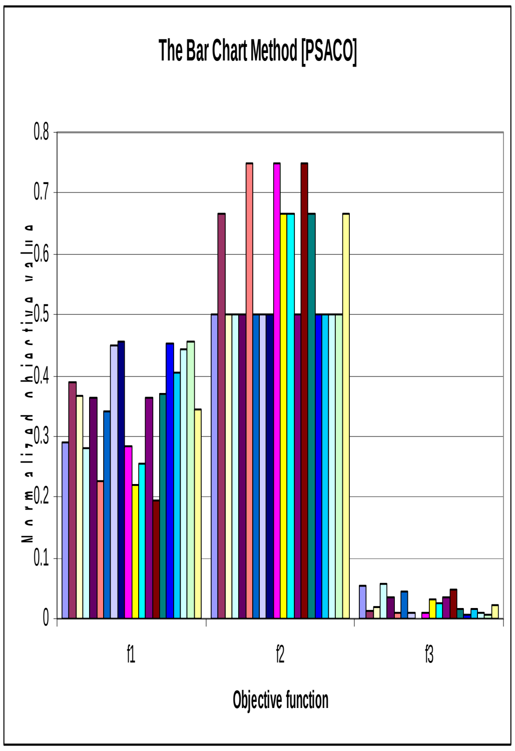


Fig. 6.35 The bar chart method for Model 1

6.4.2 Metrics of Performance of the Non-Dominated Solutions for Model 1

As the true Pareto front of the Model 1 is not known, it is not possible to compute the M_1^* metric (see section 3.3). Thus it is not possible to discover how close each non-dominated solution obtained by P-ACO and PSACO is to the true Pareto front.

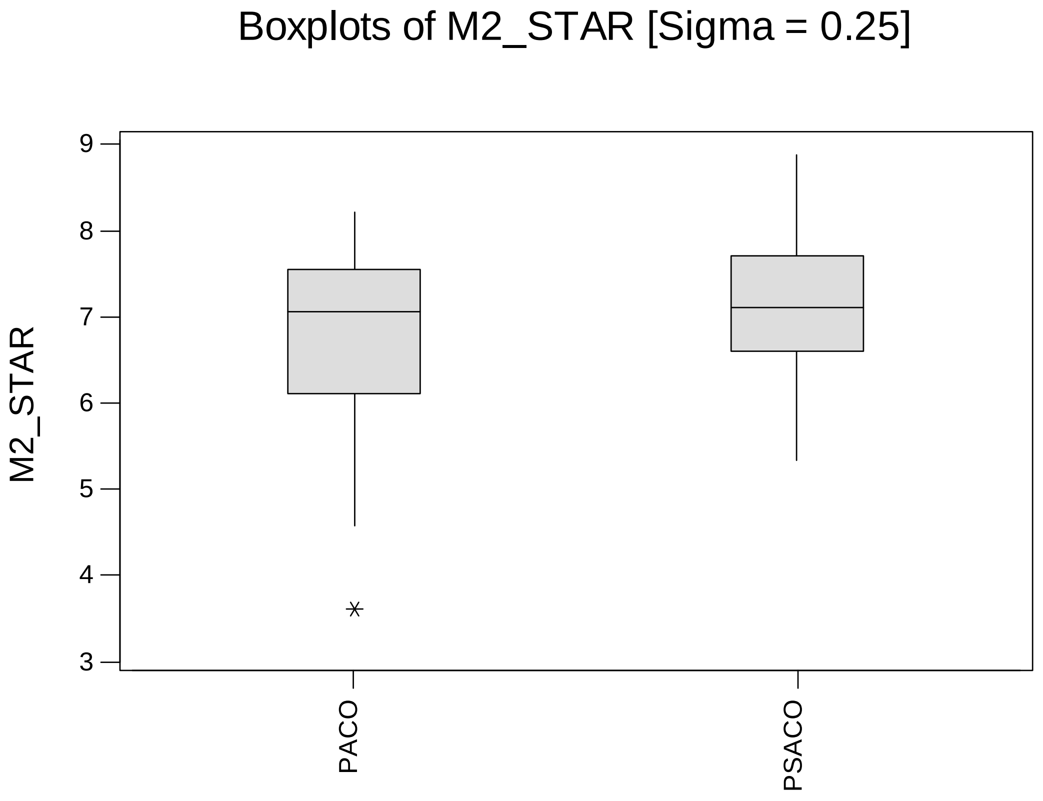


Fig. 6.36 Boxplots of the results obtained for the M_2^* metric with $\sigma = 0.25$ for Model 1

The second metric M_2^* gives an idea about the distribution of the non-dominated solutions obtained in each of the runs of the experiment. Higher M_2^* values imply a better distribution of the non-dominated solutions for the algorithm. Fig. 6.36 shows boxplots of values of M_2^* for P-ACO and PSACO with $\sigma = 0.25$. A boxplot graphically represents the minimum, maximum and median values as well as the upper and lower quartiles (upper and lower ends of the box). The boxplots show that PSACO has higher values for each of these statistics. Table 6.22 also compares the descriptive statistics of M_2^* for P-ACO and PSACO over the twenty runs of the experiment. Bold values indicate that they exceed the corresponding values of the other algorithm. It is noticeable that PSACO outperforms P-ACO for each statistics and hence PSACO generates a better distribution of non-dominated solutions.

Table 6.22 Descriptive statistics of the M_2^* metric with $\sigma = 0.25$ for Model 1

	P-ACO	**PSACO**
Mean	6.725	**7.142**
Std. Dev.	1.168	**0.916**
Q1	6.111	**6.595**
Median	7.056	**7.111**
Q3	7.556	**7.722**
Minimum	3.600	**5.333**
Maximum	8.222	**8.889**

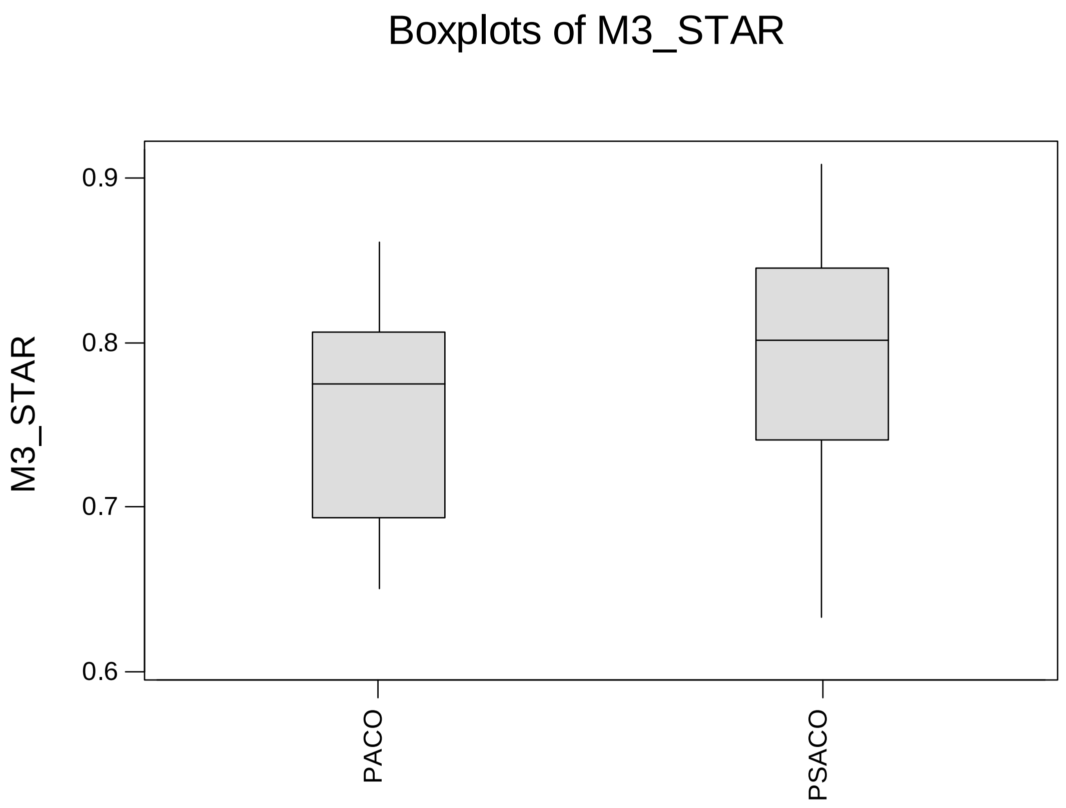


Fig. 6.37 Boxplots of the results obtained for the M_3^* metric for Model 1

M_3^* estimates the maximum extent in each dimension of the spread of the non-dominated solutions. Fig. 6.37 shows the boxplots of the M_3^* metric for P-ACO and PSACO of Model 1 for the non-dominated solutions obtained over the 20 runs of the experiment. Table (6.23) compares the descriptive statistics of the M_3^* metric for P-

ACO and PSACO. According to these, it is noticeable that the maximum extent in each dimension of the non-dominated solutions obtained by PSACO is better in most cases.

Table 6.23 Descriptive statistics of the M_3^* metric for Model 1

	PACO	PSACO
Mean	0.7556	**0.7875**
Std. Dev.	**0.0663**	0.0732
Q1	0.6935	**0.7408**
Median	0.7750	**0.8019**
Q3	0.8067	**0.8459**
Minimum	**0.6502**	0.6324
Maximum	0.8614	**0.9090**

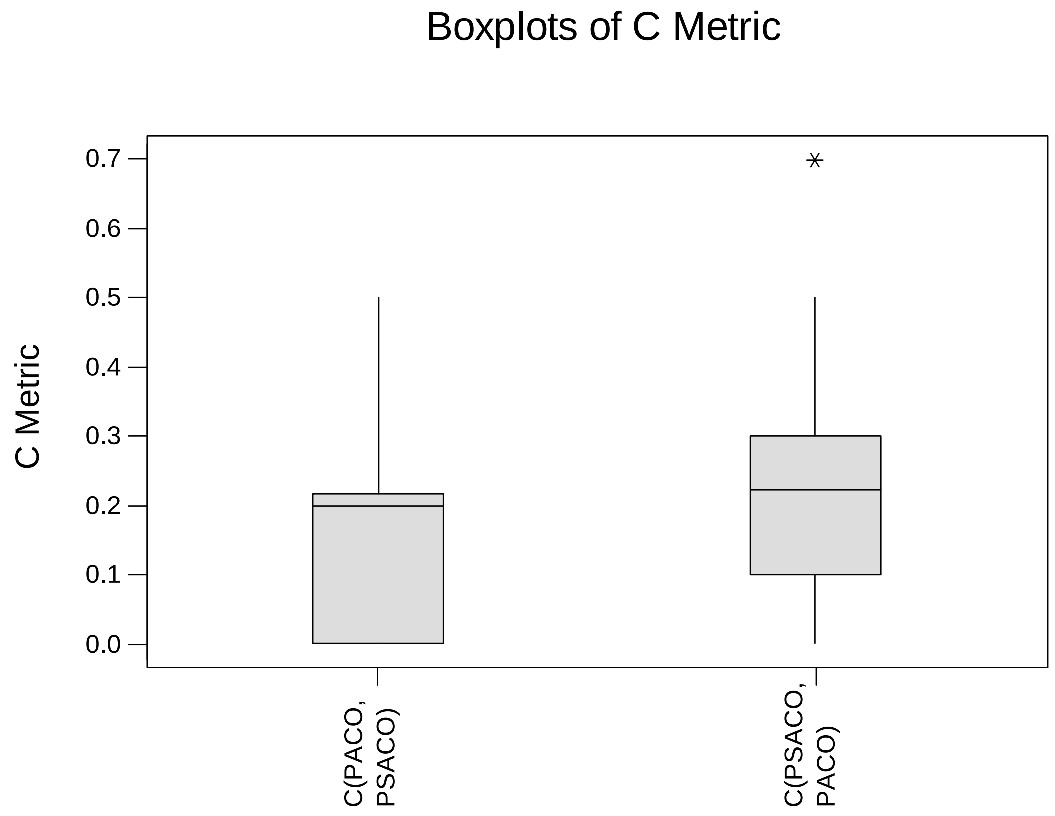


Fig. 6.38 Boxplots of the results obtained for the C metric for Model 1

The *C*-metric (defined in section 3.3) calculates the proportion of non-dominated solutions obtained in one algorithm, which are dominated by non-dominated solutions obtained in another algorithm. For example, *C(A, B) = 0.7* means that 70% of solutions in algorithm B are dominated by at least one of the solutions of algorithm A. Fig. 6.38 illustrates the boxplots of the *C*-metrics of one algorithm against the other. The left boxplot demonstrates the proportion of non-dominated solutions obtained in PSACO that are dominated by non-dominated solutions obtained in P-ACO. Similarly, the right boxplot displays the proportion of non-dominated solutions obtained in P-ACO that are dominated by non-dominated solutions obtained in PSACO. Table 6.24 compares the descriptive statistics of the *C*-metric values. From these, it is clear that, PSACO generates better non-dominated solutions. For example, the mean value for *C(PSACO, P-ACO)* is 0.2397 and hence 24% of non-dominated solutions of P-ACO are covered by PSACO solutions on average whilst it is 0.1636 (or 16%) for *C(P-ACO, PSACO)*.

Table 6.24 Descriptive statistics of the C Metric for Model 1

	C(PACO, PSACO)	**C(PSACO, PACO)**
Mean	0.1636	**0.2397**
Std. Dev.	**0.1516**	0.1750
Q1	0.0000	**0.1000**
Median	0.2000	**0.2222**
Q3	0.2167	**0.3000**
Minimum	0.0000	**0.0000**
Maximum	0.5000	**0.7000**

6.4.3 Computation Times for Model 1

In each run of the experiment, each algorithm was run for 40 cycles (each cycle consisting of 100 turns). The computation time of each run was recorded and stored in a text file. Fig. 6.39 depicts the boxplots of the computation times for P-ACO and PSACO and Table 6.25 compares the descriptive statistics of the computation times. The average times are 26.919 (min.) and 18.997 (min.) for P-ACO and PSACO, respectively. These results show that PSACO takes less computation time to perform a single run.

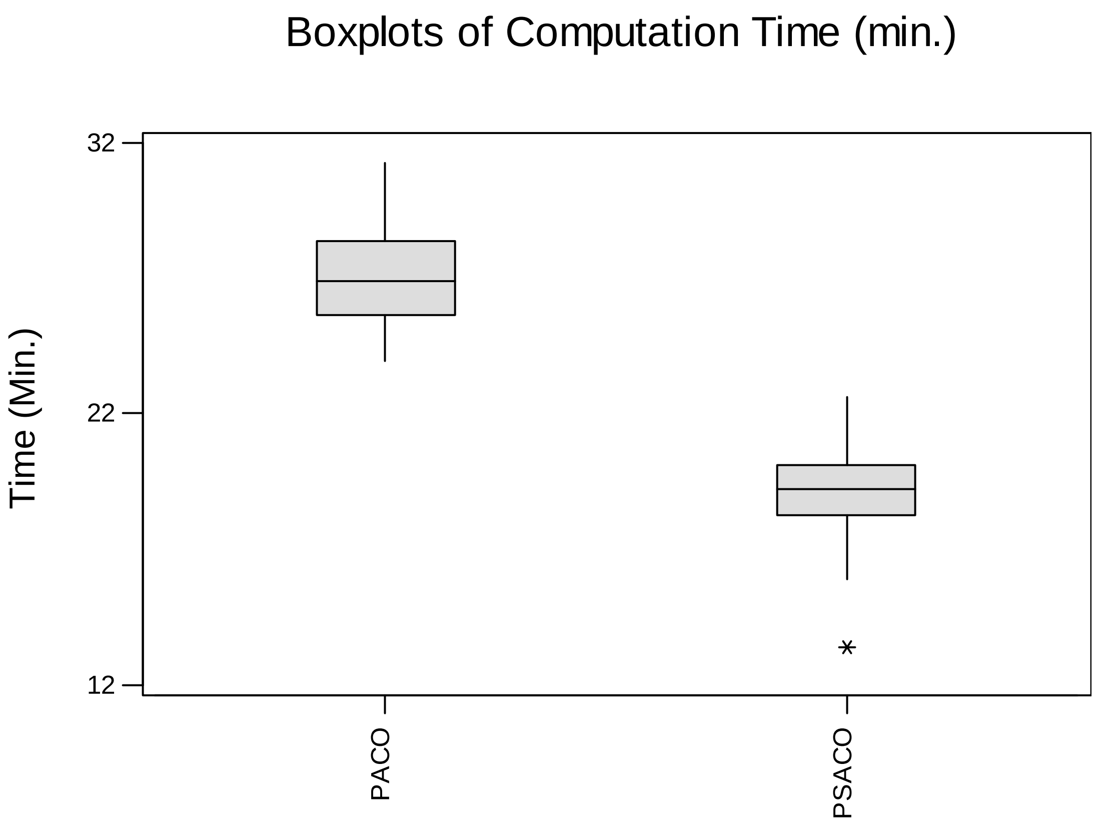


Fig. 6.39 Boxplots of computation time (min.) for Model 1

Table 6.25 Descriptive statistics of computation time (min.) for Model 1

	PACO	**PSACO**
Mean	26.919	**18.997**
Std. Dev.	1.957	**1.917**
Q1	25.624	**18.305**
Median	26.932	**19.237**
Q3	28.365	**20.107**
Minimum	23.948	**13.416**
Maximum	31.274	**22.646**

Model 2

The parameters for the two algorithms for Model 2 have been set as follows: Number of ants $N = 10$, Archive size $N_A = 5$, Initial pheromone level $\tau_0 = 1$, $\alpha = 1$, $\beta = 5$, $\rho = 0.01$, $q_0 = 0.9$ (only for P-ACO), $\gamma = 2$, $\varepsilon = 10/180 = 0.0556$, Catalogue angles: $\omega_1 = 45^\circ$, $\omega_2 = 90^\circ$, $\omega_3 = 120^\circ$, $\omega_4 = 150^\circ$ and $\omega_5 = 180^\circ$.

Overall, P-ACO and PSACO have generated 7 and 4 non-dominated solutions, respectively, out of 100 solutions generated in 20 runs.

6.4.4 Illustrative Representation of the Non-Dominated Solutions for Model 2

Figs. 6.40 – 6.42 show the overall non-dominated solutions using the scatter-plot matrix, the value path and the bar chart methods, respectively. According to the scatter-plot matrix, PSACO has obtained smaller objective values compared to P-ACO. As for Model 1, objective values are not spread over the entire range of each of the objectives [0, 1] (see also the value path and bar chart methods) as the problem considered here is also discontinuous and discrete. Both algorithms also produce good trade-off solutions as witnessed by the large slope between two consecutive objective bars in the value path method.

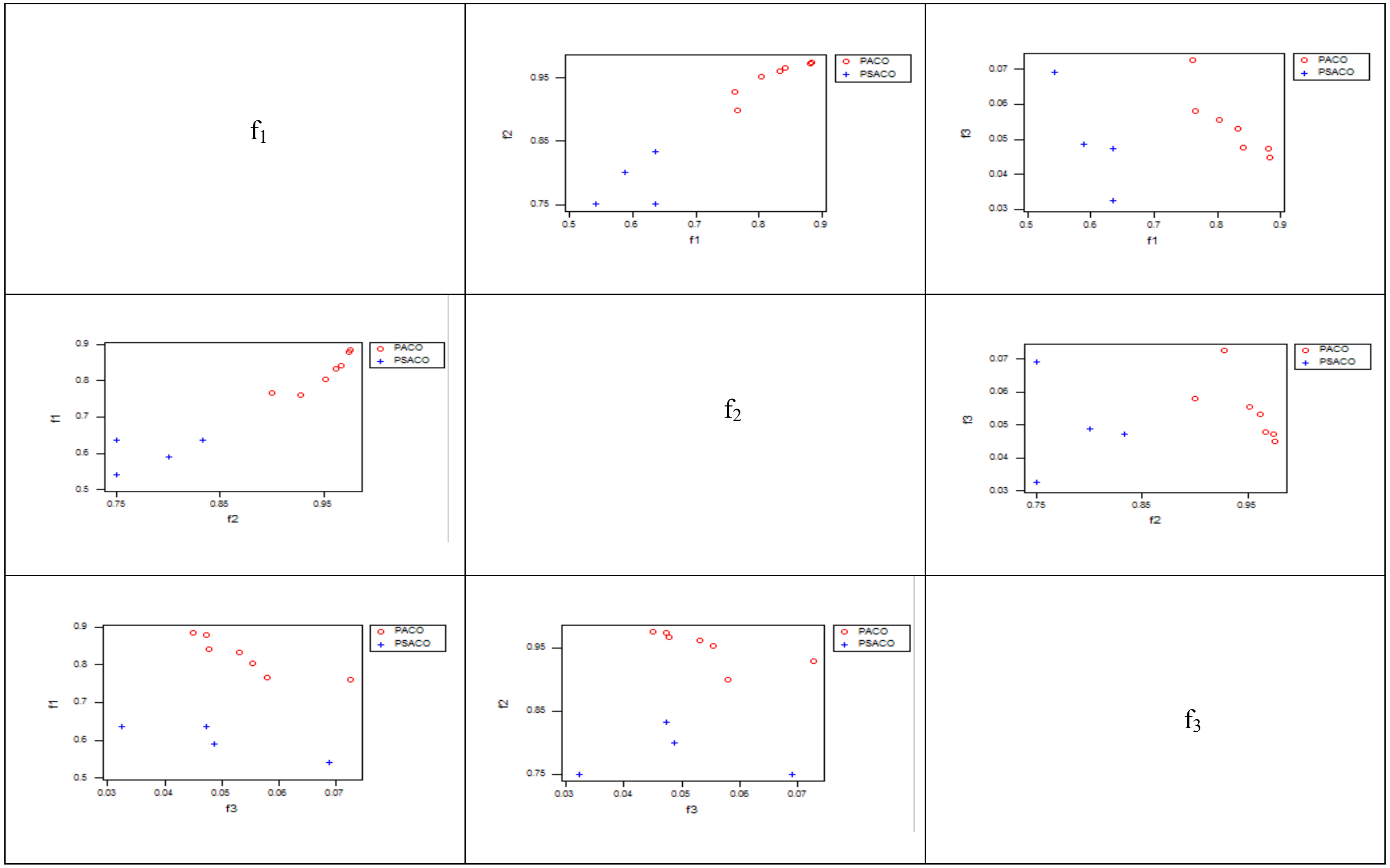


Fig. 6.40 The scatter-plot matrix method for Model 2

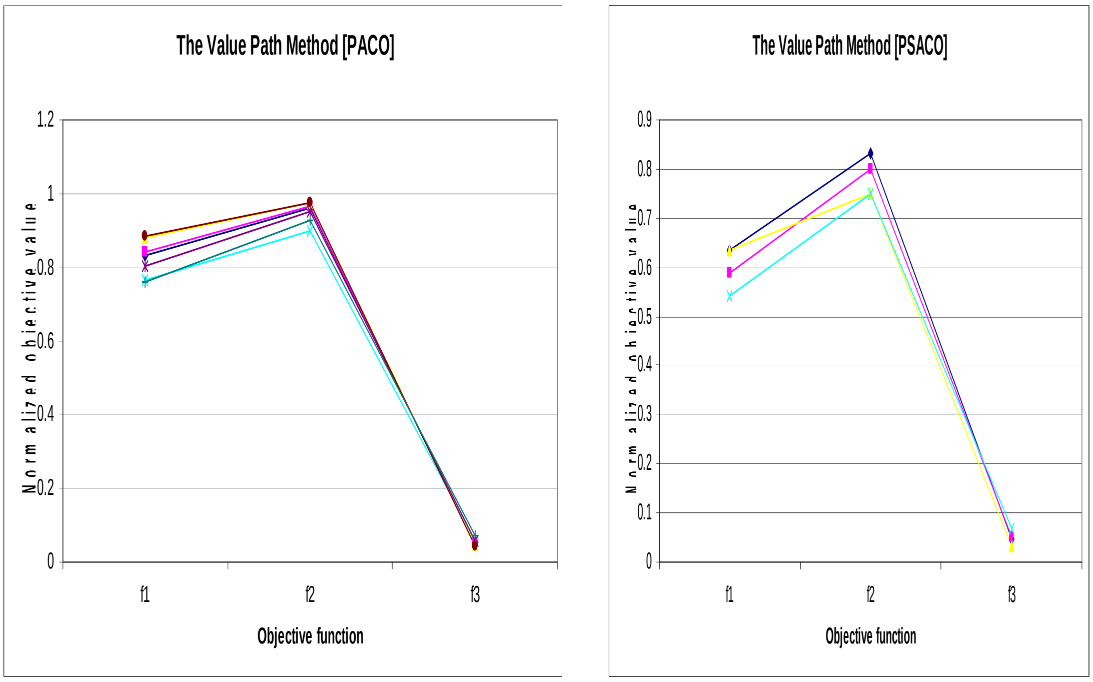


Fig. 6.41 The value path method for Model 2

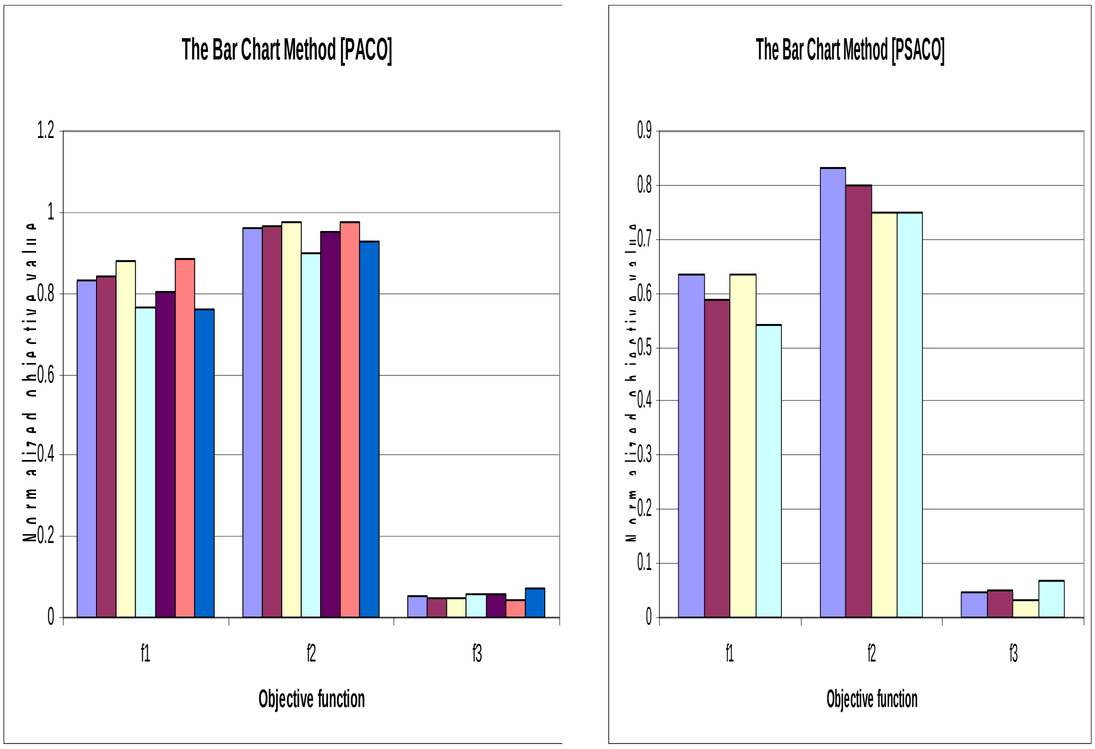


Fig. 6.42 The bar chart method for Model 2

6.4.5 Metrics of Performance of the Non-Dominated Solutions for Model 2

As for Model 1, it is not possible to compute the M_1^* metric since the true Pareto front of the Model 2 is not known.

Fig. 6.43 illustrates boxplots of values of M_2^* for P-ACO and PSACO with $\sigma = 0.10$. Further, Table 6.26 also compares the descriptive statistics of M_2^*. These results show that PSACO has obtained a better distribution of non-dominated solutions for Model 2. For P-ACO, the first quartile and the median are zero (Table 6.26) as M_2^* was zero for 15 trials out of 20.

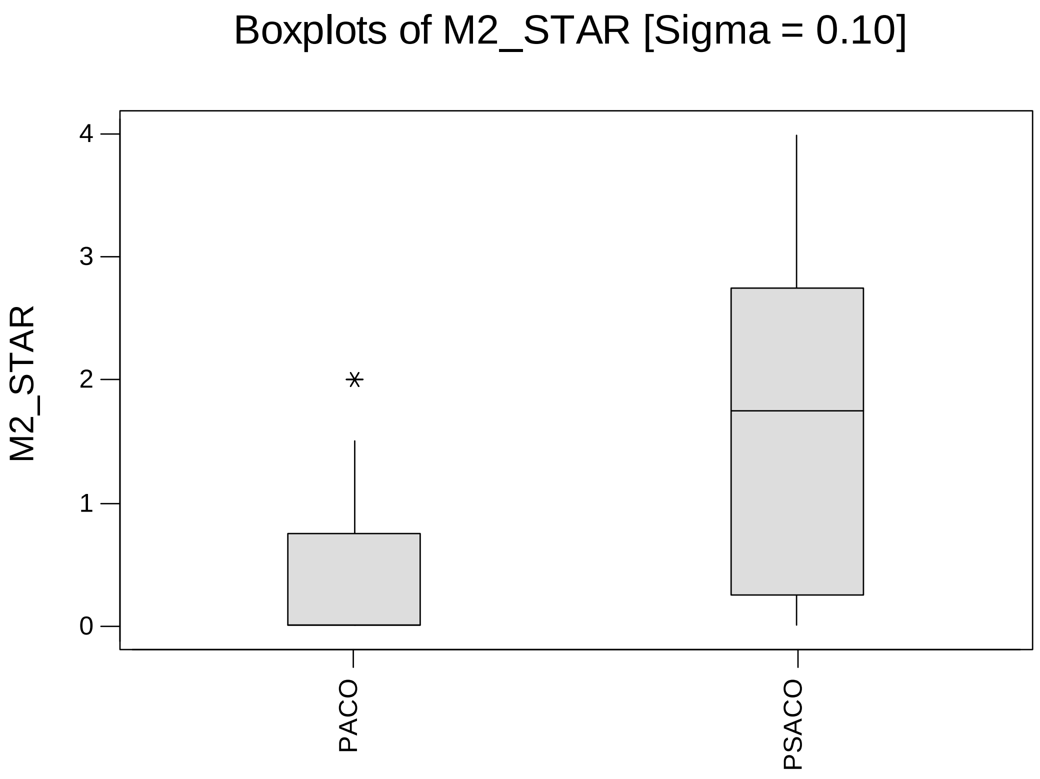


Fig. 6.43 Boxplots of the results obtained for the M_2^* metric with $\sigma = 0.10$ for Model 2

Table 6.26 Descriptive statistics of the M_2^* metric with $\sigma = 0.10$

	PACO	PSACO
Mean	0.425	**1.675**
Std. Dev.	**0.783**	1.304
Q1	0.000	**0.250**
Median	0.000	**1.750**
Q3	0.750	**2.750**
Minimum	0.000	0.000
Maximum	2.000	**4.000**

Fig. 6.44 shows the boxplots of the M_3^* metric for P-ACO and PSACO for Model 2 for the non-dominated solutions obtained over 20 runs of the experiment. Table (6.27) compares the descriptive statistics of the M_3^* metric for P-ACO and PSACO. According to the boxplots and the descriptive statistics, it is clear that the maximum extent in each dimension of the non-dominated solutions obtained is better in the case of PSACO.

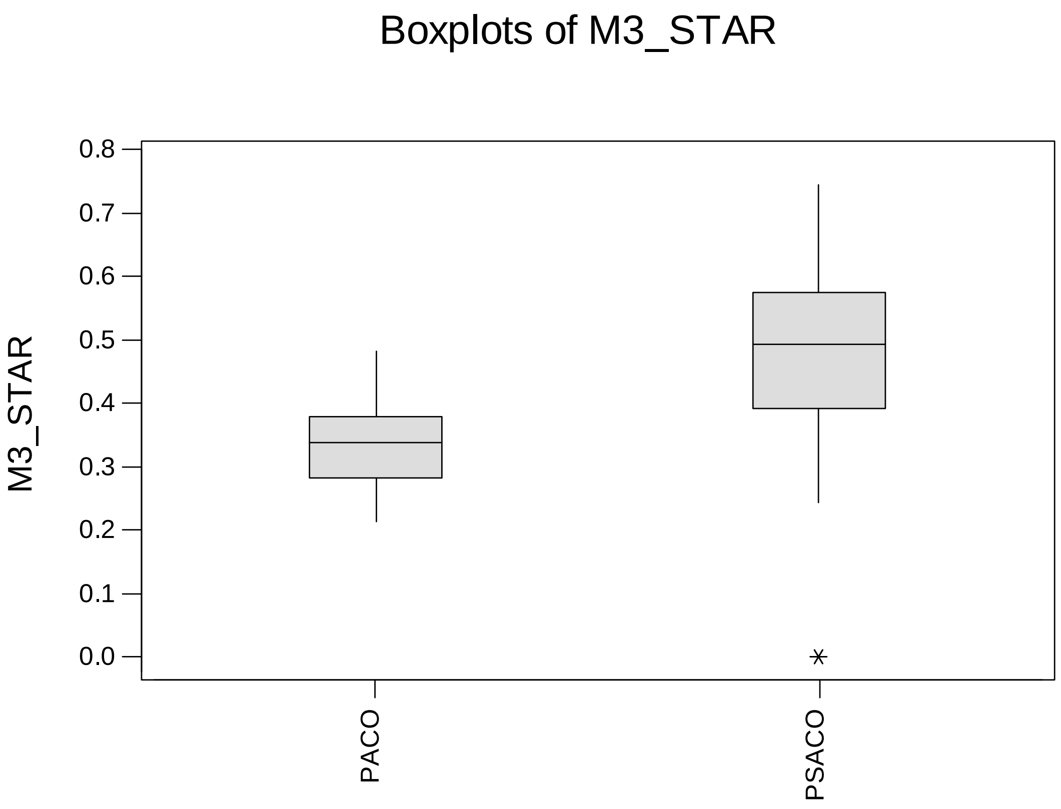


Fig. 6.44 Boxplots of the results obtained for the M_3^* metric for Model 2

Table 6.27 Descriptive statistics of the M_3^* metric for Model 2

	PACO	**PSACO**
Mean	0.3396	**0.4383**
Std. Dev.	**0.0745**	0.2212
Q1	0.2818	**0.3920**
Median	0.3374	**0.4941**
Q3	0.3784	**0.5766**
Minimum	**0.2121**	0.0000
Maximum	0.4823	**0.7468**

Fig. 6.45 illustrates the boxplots of the *C*-metrics of one algorithm against the other for Model 2. Table 6.28 shows the descriptive statistics of the *C*-metric values. Since the mean and other statistics of *C(P-ACO, PSACO)* are zero, none of the non-dominated solutions obtained in P-ACO dominate the non-dominated solutions of PSACO. Since the mean of *C(PSACO, P-ACO)* is 0.8225, 82% of non-dominated solutions obtained in P-ACO are dominated by one of the non-dominated solutions of PSACO.

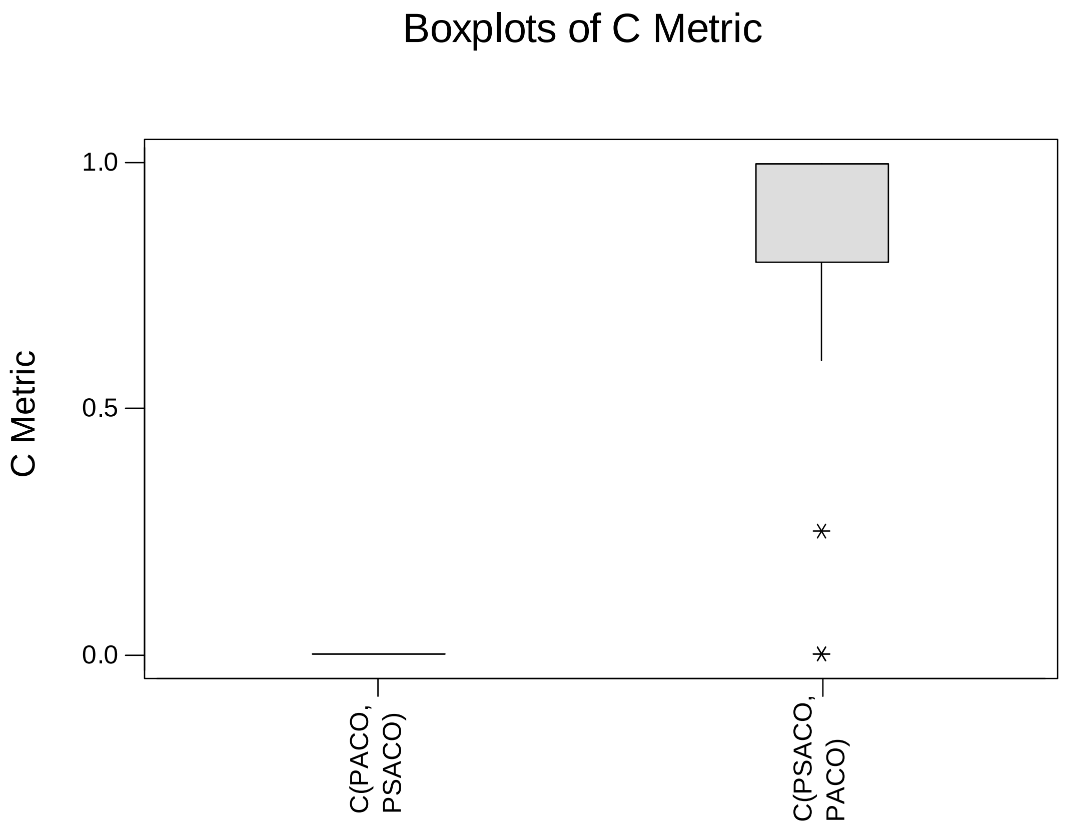


Fig. 6.45 Boxplots of the results obtained in the C metric for Model 2

Table 6.28 Descriptive statistics of the C Metric for Model 2

	C(PACO, PSACO)	**C(PSACO, PACO)**
Mean	0.0000	**0.8225**
Std. Dev.	0.0000	0.3381
Q1	0.0000	**0.8000**
Median	0.0000	**1.0000**
Q3	0.0000	**1.0000**
Minimum	0.0000	0.0000
Maximum	0.0000	**1.0000**

6.4.6 Computation Times for Model 2

As for Model 1, each algorithm was run for 40 cycles (each cycle consisting of 100 turns). The computation time of each run was recorded and stored in a text file. Fig. 6.46 illustrates the boxplots of the computation times for P-ACO and PSACO and Table 6.29 compares the descriptive statistics of the computation times. The average times for P-ACO and PSACO are 83.518 and 81.016, respectively. These results show that PSACO requires a lower computation time for a single run.

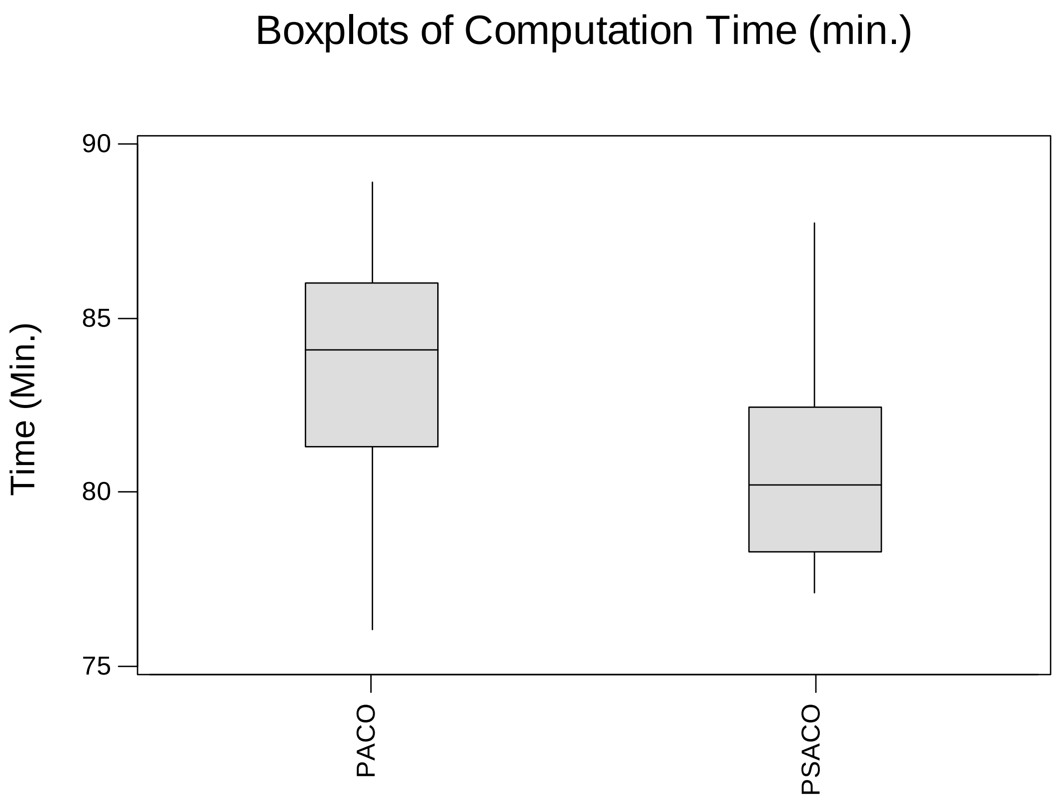


Fig. 6.46 Boxplots of computation time (min.) for Model 2

Table 6.29 Descriptive statistics of the computation time (min.) for Model 2

	PACO	**PSACO**
Mean	83.518	**81.016**
Std. Dev.	3.965	**3.256**
Q1	81.286	**78.251**
Median	84.103	**80.191**
Q3	86.030	**82.437**
Minimum	**76.005**	77.067
Maximum	88.947	**87.771**

6.4.7 Final Solution

In the case of a multi-objective optimization procedure, the higher level of information that cannot be incorporated into the optimization algorithm must be used for selecting a suitable solution (see Fig. 3.1). The approaches discussed in section 3.4 have been used for selecting a solution among the non-dominated solutions (overall) in both the algorithms, P-ACO and PSACO. For the first two metrics (*l_p-metric* and *Tchebycheff metric*), a reference point **z** is required which is comprised of individual best objective function values and thus $(0, 0, 0)^T$ is selected as the reference point **z**. Further, $p = 2$ (Euclidian distance) has been used for the *lp-metric*. Both metrics assume that there are no priorities among the objectives. In the Pseudo-Weight Vector Approach, the selection of the solution is subjective. Thus, the solution for this approach is selected such that the weight for the third objective (f_3) is the closer to zero. The purpose of this is to minimize the deviation between the bend angles and the pre-specified catalogue angles. Table 6.30 and Table 6.31 show the final solutions obtained using these approaches for Model 1 and Model 2, respectively. The last column of these tables is the sum of the deviations (absolute difference between the bend angle (θ_i) and the closest catalogue angle to θ_i (ω)). These results also show that in most cases, the final solution obtained for PSACO for these approaches is not dominated by the corresponding solution for P-ACO.

Table 6.30 Final solution obtained using various methods for Model 1

	Method	**f1**	**f2**	**f3**	**Length**	**# Bends**	$\sum_i \lvert\theta_i - \omega\rvert$
lp-Metric	PACO	0.2814	0.5	0.0570	354.79	2	20.50
	PSACO	0.2814	0.5	0.0570	354.79	2	20.50
Tchebycheff-Metric	PACO	0.2814	0.5	0.0570	354.79	2	20.50
	PSACO	0.2889	0.5	0.0551	358.55	2	19.83
Pseudo-weight	PACO	0.3699	0.5	0.0119	404.61	2	4.29
	PSACO	0.4520	0.5	0.0080	465.25	2	2.86

Table 6.31 Final solution obtained using various methods for Model 2

	Method	**f1**	**f2**	**f3**	**Length**	**# Bends**	$\sum_i \lvert\theta_i - \omega\rvert$
lp-Metric	PACO	0.7652	0.90	0.0580	1277.92	10	104.42
	PSACO	0.5415	0.75	0.0690	654.36	4	49.73
Tchebycheff-Metric	PACO	0.7652	0.90	0.0580	1277.92	10	104.42
	PSACO	0.5415	0.75	0.0690	654.36	4	49.73
Pseudo-weight	PACO	0.8039	0.95	0.0555	1529.91	21	209.74
	PSACO	0.5887	0.80	0.0488	729.47	5	43.89

6.4.8 Refining the solution

According to the solutions found in the previous section, it is noticeable that bend angles are not much closer to the pre-specified catalogue of angles of bends. For example, for the solution obtained by the *lp-metric* and the *Tchebycheff-metric* for Model 2 for PSACO each bend deviates on average by approximately 12^o ($\approx$ 49.73/4) from the closest catalogue angle (see second and fourth rows of Table 6.31). Thus, one could refine the solution further near the solution $\boldsymbol{X_G}$ (obtained using the entire search space) as follows:

Assume that $\boldsymbol{X_G} = (S, P_1, P_2, ..., P_n, E)$ is the final solution obtained using the entire search space where $P_1, P_2, ..., P_n$ are the intermediate points between S and E. Then the refining algorithm creates a network using some random points in the neighbourhood of each point P_i $(i = 1, 2, ..., n)$ (see Fig. 6.47). Next, the refining algorithm searches for non-dominated solutions in this network using PSACO.

Table 6.32 shows the overall best non-dominated solutions found for Model 2 after the refining of the *lp-metric* solution of PSACO (see second row of Table 6.31). Here the refined network is generated using 200 random points (*density = 200*) in each point P_i's neighbourhood. Each neighbourhood i is defined as a cube (of size $2r$) centred on the corresponding point P_i. For this experiment r is set to 20. As in the previous experiments, PSACO is run for 20 trials with 10 ants and an archive size (N_A) of 5.

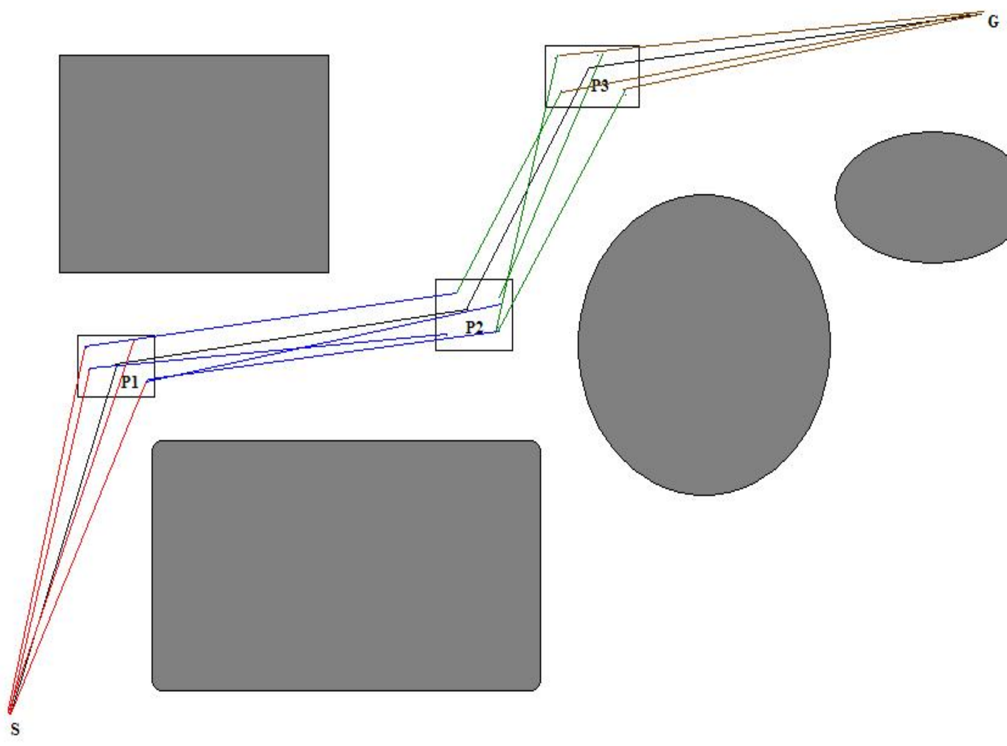


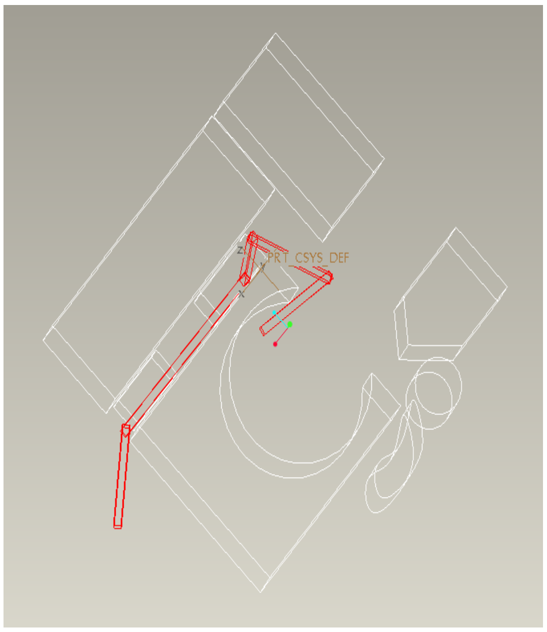
Fig. 6.47 Creating the refined network

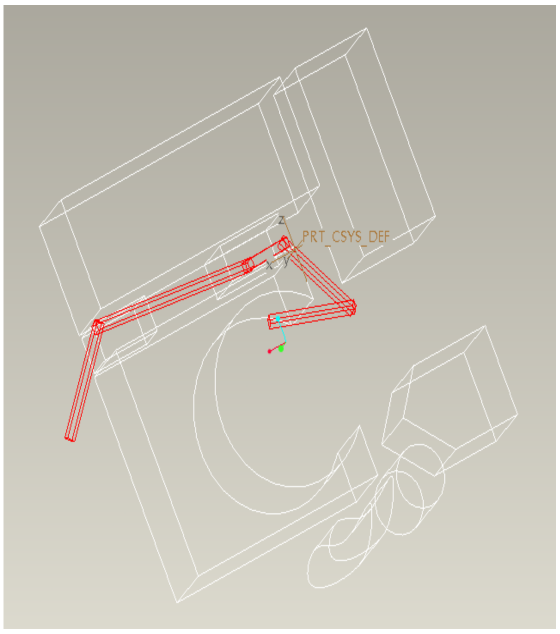
Table 6.32 Refined solutions obtained using PSACO for Model 2
(*r = 20, Density = 200*)

	f1	**f2**	**f3**	**Length**	**# Bends**	$\sum_i \lvert\theta_i - \omega\rvert$
Solution 1	0.5160	0.75	0.0104	619.79	4	7.52
Solution 2	0.4892	0.75	0.0277	587.27	4	19.93
Solution 3	0.5667	0.75	0.0093	692.41	4	6.67
Solution 4	0.5085	0.75	0.0260	610.37	4	18.73

According to Table 6.32, it is noticeable that all the refined solutions (except solution 3) dominate the previous solution found searching the entire search space (compare the solutions found in Table 6.32 with the second row of Table 6.31). In solutions 1, 2 and 4 both length and $\sum_i \lvert\theta_i - \omega\rvert$ are improved even though the number of bends is still the same. The average deviation from the catalogue angles is also improved. For example, if solution 1 is selected, this value is approximately 2^0 ($\approx 7.52/4$) per bend and was 12^0 with the solution found using the entire search space.

Fig. 6.48 shows the solutions obtained in the entire search space and in the refined search space. Table 6.33 lists the actual bend angles, their closest catalogue angles and the deviation from the closest catalogue angle. The bend angles of the refined solution are much closer to the pre-specified catalogue angles.





(i) Solution obtained searching the entire search space

(ii) Refined solution (Solution 1)

Fig. 6.48 Solutions obtained using the entire search space and the refined search space for Model 2

Table 6.33 Bend angles of the solutions obtained in the entire search space and the refined search space for Model 2

	Solution obtained using the entire search space			Refined solution (Solution 1)		
	Angle	Closest Catalogue Angle	Deviation	Angle	Closest Catalogue Angle	Deviation
Bend 1	58.82	45	13.82	45.33	45	0.33
Bend 2	80.63	90	9.37	117.29	120	2.71
Bend 3	108.26	120	11.74	118.31	120	1.69
Bend 4	134.80	120	14.80	122.79	120	2.79

Fig 6.49 shows the boxplots of the computation times for searching non-dominated solutions in the entire search space and the refined search space. Table 6.34 compares the different descriptive statistics of the computation times for finding the solutions in

the entire and refined search spaces. According to the boxplot and the table the computation times for searching in the refined search space is well below the computation time for searching in the entire search space.

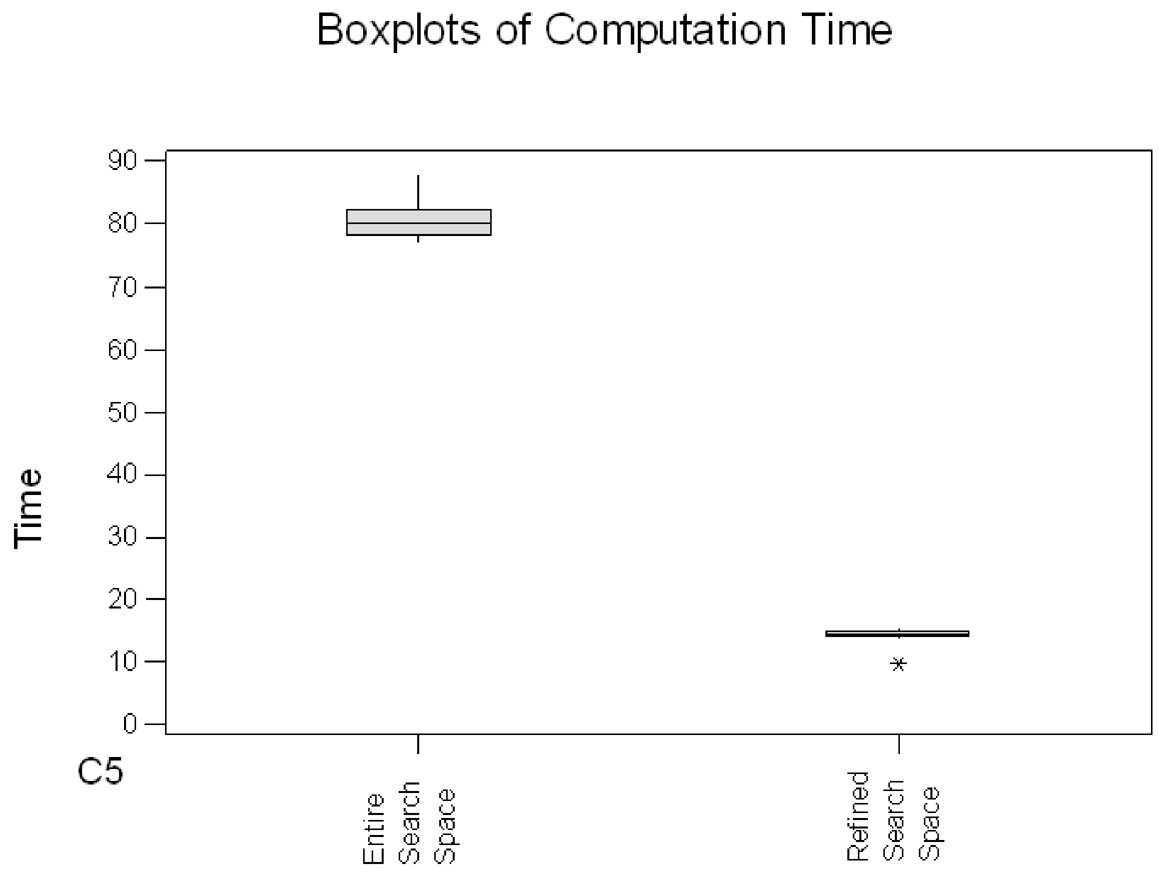


Fig. 6.49 Boxplots of computation times using the entire search space and the refined search space for Model 2

Table 6.34 Descriptive statistics of computation times using the entire search space and the refined search space for Model 2

	Entire Search Space	**Refined Search Space**
Mean	81.016	**14.242**
Std. Dev.	3.256	**1.235**
Q1	78.251	**78.251**
Median	80.191	**14.371**
Q3	82.437	**14.936**
Minimum	77.067	**9.477**
Maximum	87.771	**15.433**

Thus, if the solution obtained using the entire search space is not satisfactory, it is advisable to run the refinement algorithm to search for an acceptable solution near the neighbourhood of the solution obtained using the entire search space.

6.5 Results Obtained for the Proposed Multi-Colony Ant Systems for Multi-Hose Routing

In this section, the results of applying the proposed multi-colony ant systems MCAS-MHR-1 and MCAS-MHR-2 described in section 5.8 are presented and discussed and their strengths and weaknesses are investigated empirically.

The parameter settings for both algorithms are: number of ants for each colony = 5, initial pheromone level on each edge = 1, pheromone decay parameter $\rho = 0.01$, $\alpha = 1$ and $\beta = 5$. Other parameter settings are included in the relevant experiment results.

The termination criterion for each of the experiments was set to 100 cycles.

All the simulations were carried out on a Pentium IV PC (Processor speed = 3.0 GHz, RAM = 512 MB) in the Microsoft Windows XP environment using Microsoft Visual C++ (.Net version).

6.5.1 Experiment 1: Demonstrating the potential of MCAS-MHR-1 and MCAS-MHR-2 on a test graph

The purpose of this experiment is to demonstrate on a graph the potential of the proposed two algorithms MCAS-MHR-1 and MCAS-MHR-2 whose optimal solutions between the given commodities (pairs of start and end points) are already known.

Test graph 1 (see Fig. 6.50) is used to find the best shared paths between the commodities (S_1, E_1) and (S_2, E_2). There are 9 possible solutions for this simple graph (see Table ble 6.35). When considering the shortest paths between the two commodities (S_1, E_1) and (S_2, E_2), the choice would be solution 1. However, if paths need to be shared as much as possible while the total length of the paths is minimized, the algorithms should select solution 5.

Table 6.36 shows the percentage of runs that obtain each solution over 100 runs after 100 cycles for different values of w_1, w_2 and γ (only for MCAS-MHR-2).

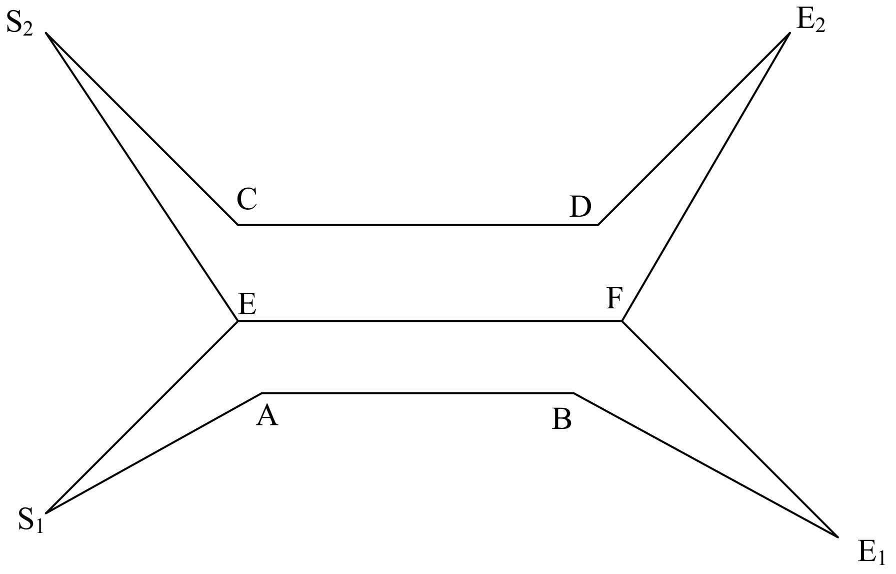


Fig. 6.50 Test graph 1 – Finding the optimum paths between commodities (S_1, E_1) and (S_2, E_2). Best solution:(S_1EFE_1, S_2EFE_2)

These results show that both algorithms obtain the disjoint solution (solution 1) when the shared length between two paths is not considered for pheromone updating (see Table 6.36, Fig. 6.51 and Fig. 6.52). However, both algorithms obtain solution 5 from more than 70% of the runs (except in one case) when there is a contribution from the shared length between two paths for the pheromone updating (see Fig. 6.51 and Fig. 6.52). The highest percentage obtained for MCAS-MHR-1 is 84% and it happened when w_1 = 0.9 and w_2 = 0.1. Compared with MCAS-MHR-1, MCAS-MHR-2 gives slightly better results for most of the γ values. For example, when w_1 = 0.9 and w_2 = 0.1, MCAS-MHR-2 obtains solution 5, from 91%, 88%, 89% and 93% of the runs for the γ values 1, 2, 4 and 5, respectively. Therefore, it possible to conclude, empirically, that the performance of MCAS-MHR-2 is better as a result of using additional information (foreign pheromones).

Table 6.35 Possible solutions for test graph 1

Solution No	**Path from S_1 to E_1**	**Path from S_2 to E_2**
1	S_1ABE_1	S_2CDE_2
2	S_1ABE_1	S_2EFE_2
3	S_1ABE_1	$S_2ES_1ABE_1FE_2$
4	S_1EFE_1	S_2CDE_2
5	S_1EFE_1	S_2EFE_2
6	S_1EFE_1	$S_2ES_1ABE_1FE_2$
7	$S_1ES_2CDE_2FE_1$	S_2CDE_2
8	$S_1ES_2CDE_2FE_1$	S_2EFE_2
9	$S_1ES_2CDE_2FE_1$	$S_2ES_1ABE_1FE_2$

Table 6.36 Percentage of each solution found on test graph 1 for different values of w_1, w_2 and γ (for MCAS-MHR-2 only)

MCAS-MHR-1																																				
(w_1, w_2)	**(1, 0)**									**(0.9, 0.1)**									**(0.8, 0.2)**									**(0.7, 0.3)**								
Solution	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
(%)	**100**	0	0	0	0	0	0	0	0	14	0	0	0	**84**	2	0	0	0	33	0	4	0	**63**	0	0	0	0	23	0	2	0	**71**	2	0	0	0
MCAS-MHR-2																																				
(w_1, w_2)	**(1, 0)**									**(0.9, 0,1)**									**(0.8, 0.2)**									**(0.7, 0.3)**								
Solution	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
$\gamma = 0$	**100**	0	0	0	0	0	0	0	0	23	0	0	0	**77**	0	0	0	0	3	0	0	0	**95**	2	0	0	0	23	0	2	0	**71**	2	0	2	0
$\gamma = 1$	**100**	0	0	0	0	0	0	0	0	9	0	0	0	**91**	0	0	0	0	0	0	0	0	**94**	6	0	0	0	16	0	2	0	**78**	2	0	2	0
$\gamma = 2$	**100**	0	0	0	0	0	0	0	0	12	0	0	0	**88**	0	0	0	0	15	0	2	0	**79**	2	0	2	0	4	0	2	0	**92**	2	0	0	0
$\gamma = 3$	**100**	0	0	0	0	0	0	0	0	18	0	0	0	**80**	2	0	0	0	20	0	0	0	**78**	0	0	2	0	14	0	2	0	**78**	6	0	0	0
$\gamma = 4$	**100**	0	0	0	0	0	0	0	0	6	0	3	0	**89**	0	0	0	0	8	0	0	0	**92**	0	0	0	0	6	0	2	0	**90**	2	0	0	0
$\gamma = 5$	**100**	0	0	0	0	0	0	0	0	3	0	0	0	**93**	2	0	2	0	5	0	2	0	**93**	0	0	0	0	18	0	0	0	**82**	0	0	0	0

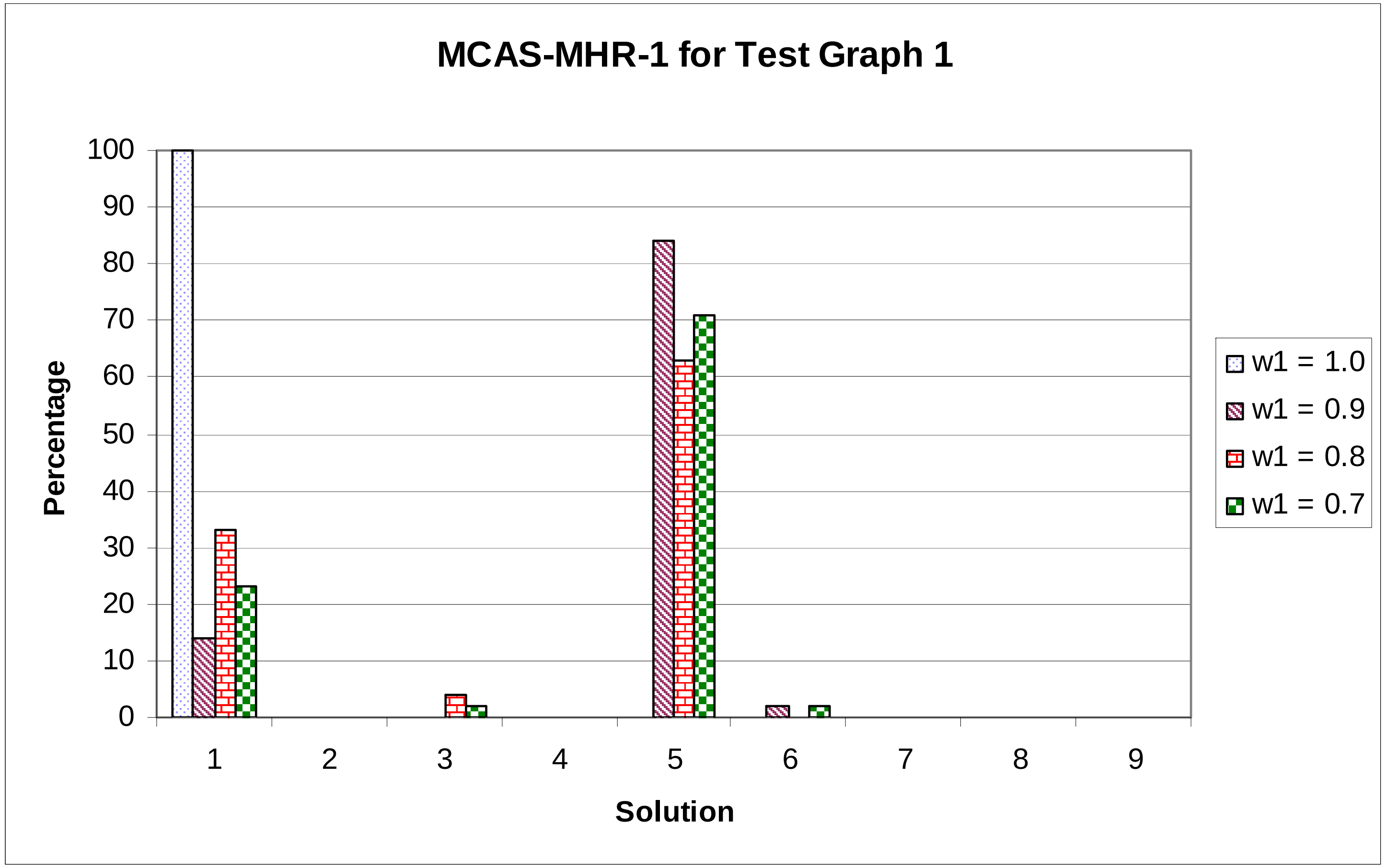


Fig. 6.51 Percentage of runs of MCAS-MHR-1 that obtain each solution of test graph 1

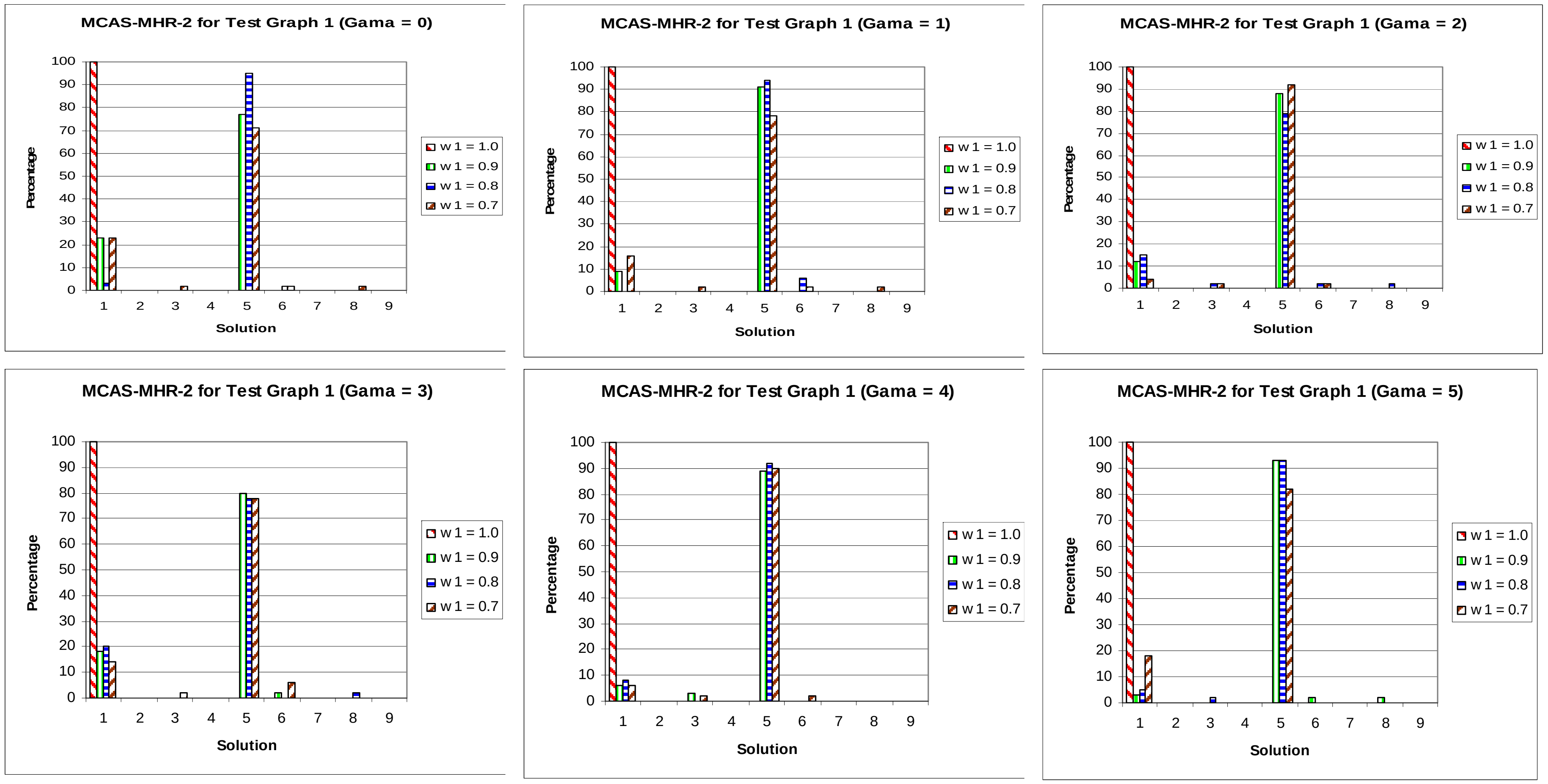


Fig. 6.52 Percentage of runs of MCAS-MHR-2 that obtain each solution of test graph 1

6.5.2 Experiment 2: Applying MCAS-MHR-1 and MCAS-MHR-2 to a complex 3D model

The purpose of this experiment is to apply the proposed two algorithms MCAS-MHR-1 and MCAS-MHR-2 (see section 5.8) to a complex 3D environment and to compare the results of the two algorithms.

Both algorithms were tested on the following Pro/Engineer 3D model (see Fig. 6.53) using different values of w_1, w_2 and γ (only for MCAS-MHR-2) for random networks of 200 points and 400 points. The other parameters of the algorithms are fixed as detailed at the beginning of this section. Both algorithms were used for finding the best shared paths of 4 commodities. All simulations were carried out for 100 cycles.

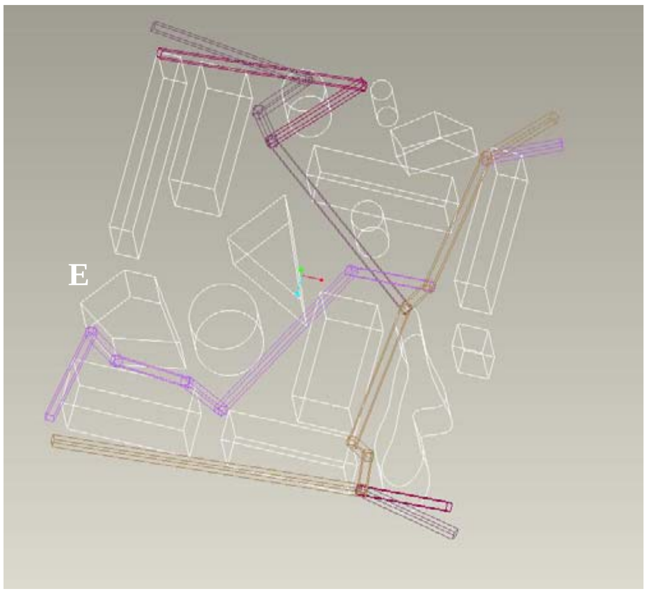


Fig. 6.53 Tested Pro/Engineer Model

For each trial, the final path length of each commodity, the connecting points of each path, the time spent on running the trial (in seconds), the total shared length of paths and strength_1 and strength_2 (defined in eqs. 5.39 and 5.40) were recorded in text files.

Consider

$$\{Q_i^{(j)} | i = 1, 2, 3, 4\}$$

to be the best solution produced in the jth trial ($j = 1, 2, ..., 100$). Here $Q_i^{(j)}$ means the path of the *i*th commodity of the best solution of the *j*th trial. Further, consider $\{R_i \mid i = 1, 2, 3, 4\}$ to be the overall best solution among the best solutions [$\{Q_i^{(j)} \mid i = 1, 2, 3, 4\}$, $j = 1, 2, ..., 100$] produced in *100* trials and t_i ($i = 1,2, ..., 100$) to be the computation time of the *i*th trial.

Tables 6.37 and 6.38 summarize the following descriptive statistics obtained for each of the algorithms over 100 trials after 100 cycles for different values of w_1, w_2 and γ (MCAS-MHR-2 only) for 200 point and 400 point networks.

Statistic	**Description**	**Mathematical Form**
$\overline{Q}$ - Avg. total length	Average total length of the paths of the best solutions over 100 trials	$\overline{Q} = \frac{\sum_{j=1}^{100} Q_j^{\Sigma}}{100}$ *where* $Q_j^{\Sigma} = \sum_{i=1}^{4} l(Q_i^{(j)})$
$SD(Q_j^{\Sigma})$ - SD total length	Standard deviation of the total lengths of the paths of the best solutions over 100 trials	$SD(Q_j^{\Sigma})$
$\overline{S}$ - Avg. shared length	Average shared length of the best solutions over 100 trials	$\overline{S} = \frac{\sum_{j=1}^{100} S_j^{\cap}}{100}$ *where* $S_j^{\cap} = \sum_{i=1}^{4} \sum_{\substack{k=1 \\ k \neq i}}^{4} l(Q_i^{(j)} \cap Q_k^{(j)})$
$SD(S_j^{\cap})$ - SD shared length	Standard deviation of shared lengths of the best solutions over 100 trials	$SD(S_j^{\cap})$
P_1 - % avg. shared length	Percentage of the average shared length to the average total length	$P_1 = \frac{\overline{S}}{\overline{Q}} \times 100$
R^{Σ} - Total length of the overall best solution	Total length of the paths of the overall best solution	$R^{\Sigma} = \sum_{i=1}^{4} l(R_i)$
$R^{\cap}$ - Total shared length of the overall best solution	Total shared length of the paths of the overall best solution	$R^{\cap} = \sum_{i=1}^{4} \sum_{\substack{k=1 \\ k \neq i}}^{4} l(R_i \cap R_k)$
P_2 - % shared length of the overall best solution	Percentage of the shared length of the paths of the overall best solution to the total length of the paths of the overall best solution	$P_2 = \frac{R^{\cap}}{R^{\Sigma}} \times 100$
$\overline{T}$ - Avg. Time	Average time per trial	$\overline{T} = \frac{\sum_{i=1}^{100} t_i}{100}$

The best solution out of 100 runs is selected using *strength_2* (see eq. 5.40) i.e., the solution with the highest value of *strength_2* is selected as the best solution. Here *l(.)* is the length of the argument and $R_i \cap R_k$ is the set of common edges between paths R_i and R_k.

Fig. 6.54 shows how the average total lengths and the average shared lengths of

MCAS-MHR-1 change for different values of weights. The average shared length increases as w_1 decreases. Similarly, the average total length also increases with lower values w_1.

Fig. 6.55 and Fig. 6.56 show how the average shared lengths and average total lengths of MCAS-MHR-2 differ for different values of w_1, w_2, and γ. As for MCAS-MHR-1, the average shared length and average total length increase as w_1 decreases. It can be noted that for each value of γ, the average shared length and the average total length increase as w_1 decreases.

As, for both algorithms the average total length increases as w_1 decreases, the designer of the algorithm must select the values for w_1 and w_2 carefully. Lower values of w_1 imply that both the total length and shared length of the solution increase and higher values of w_1 imply that both total length and shared length decrease. Hence the designer should select w_1 and w_2 such that they balance the optimality of both the total length and the shared length of the paths.

When comparing the average shared lengths for the different γ values for specific weights (see Fig. 6.55), it is noted that $\gamma = 2$ gives the highest average shared length or the closest to the highest average shared length in most of the cases of both the 200 point and the 400 point networks. Therefore, MCAS-MHR-2 with $\gamma = 2$ is selected for comparison with MCAS-MHR-1.

Fig. 6.57 compares the average shared lengths of MCAS-MHR-1 and MCAS-MHR-2 (with $\gamma = 2$) for different weights for the 200 and 400 point networks. For the 200 point network, it can be seen that MCAS-MHR-2 produces significantly larger average shared lengths for weights w_1 = 0.9999 and 0.999 when compared with the values of MCAS-MHR-1. However, there is no significant difference for the average shared lengths for MCAS-MHR-1 and MCAS-MHR-2 for the 400 point network. When comparing the average total lengths (see Fig. 6.58), there is no significant difference between the two versions.

The total lengths and shared lengths of the overall best solutions of MCAS-MHR-1 and MCAS-MHR-2 ($\gamma = 2$) are compared in Figs. 6.59 and 6.60. For both the 200 and 400 point networks, MCAS-MHR-1 produces higher shared values than MCAS-MHR-2 in 4 out of 6 cases. However, when comparing the total lengths, MCAS-MHR-2 produces smaller total length values in most cases.

When comparing the computational times of the two algorithms (see Fig. 6.61), the computation time of MCAS-MHR-1 is less for the more complicated network (400 point network).

When comparing memory requirements, MCAS-MHR-2 uses a single pheromone matrix for each colony and it needs more memory than MCAS-MHR-1. Indeed, the memory requirement of MCAS-MHR-2 grows with the number of commodities used in the algorithm.

Table 6.37 Descriptive statistics of MCAS-MHR-1 for the 200 point network

	$\overline{Q}$	$SD(Q_j^{\Sigma})$	$\overline{S}$	$SD(S_j^{\cap})$	P_1	R^{Σ}	$R^{\cap}$	P_2	$\overline{T}$ **(Sec.)**
***w*₁ = 1.0, *w*₂ = 0.0**	3568.9	30.6	119.6	192.5	3.35	3490.32	148.50	4.26	56.27
***W*₁ = .9999, *w*₂ = .0001**	3584.3	37.4	238.7	264.3	6.66	3595.12	887.32	24.68	57.03
***W*₁ = .999, *w*₂ = .001**	**3735.6**	126.4	588.7	230.4	15.76	3960.28	**1209.94**	30.55	63.66

Table 6.38 Descriptive statistics of MCAS-MHR-1 for the 400 point network

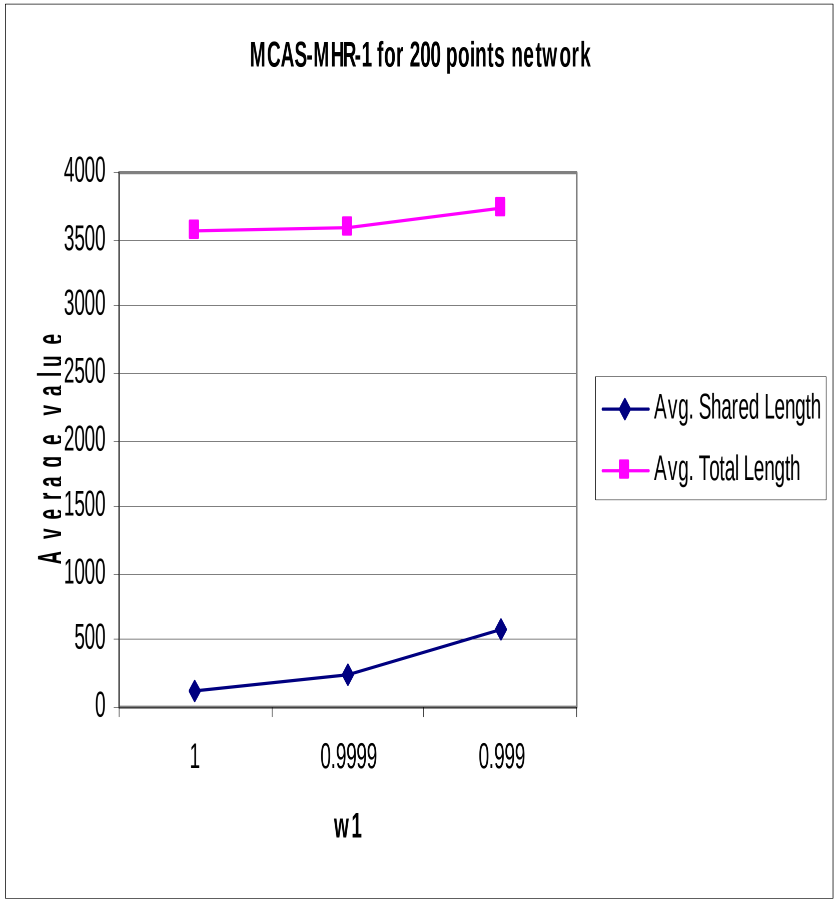
	$\overline{Q}$	$SD(Q_j^{\Sigma})$	$\overline{S}$	$SD(S_j^{\cap})$	P_1	R^{Σ}	$R^{\cap}$	P_2	$\overline{T}$ **(Sec.)**
***w*₁ = 1.0, *w*₂ = 0.0**	3342.36	36.38	58.23	110.85	1.74	3238.01	241.34	7.45	128.61
***W*₁ = .9999, *w*₂ = .0001**	3347.85	42.47	82.11	115.43	2.45	3238.44	241.34	7.45	129.52
***W*₁ = .999, *w*₂ = .001**	**3480.53**	96.66	338.30	195.07	9.72	3577.07	**828.75**	23.17	128.28

Table 6.39 Descriptive statistics of MCAS-MHR-2 for the 200 point network

Weights	γ	$\overline{Q}$	$SD(Q_j^{\Sigma})$	$\overline{S}$	$SD(S_j^{\cap})$	P_1	R^{Σ}	$R^{\cap}$	P_2	$\overline{T}$ **(Sec.)**
$w_1 = 1.0, w_2 = 0.0$	**0**	3567.02	30.70	133.25	202.09	3.74	3488.68	0.00	0.00	56.10
	1	3568.54	29.46	75.27	153.89	2.11	3489.08	518.03	14.85	56.20
	2	3572.95	31.52	92.53	166.40	2.59	3390.99	0.00	0.00	55.94
	3	3562.44	33.30	120.53	178.93	3.38	3460.98	104.40	3.02	55.56
	4	3573.30	28.56	148.12	215.85	4.15	3514.67	0.00	0.00	55.47
	5	3567.67	35.66	121.38	188.76	3.40	3461.55	43.69	1.26	55.21
$w_1 = .9999, w_2 = .0001$	**0**	3588.27	43.37	239.78	251.01	6.68	3493.48	800.54	22.92	56.26
	1	3587.12	45.09	259.04	233.30	7.22	3532.09	688.93	19.50	56.34
	2	3591.61	42.89	293.69	248.87	8.18	3574.13	910.93	25.49	56.11
	3	3587.27	43.32	282.10	247.06	7.86	3572.71	990.17	27.72	55.86
	4	3581.76	37.96	236.63	227.05	6.61	3554.92	740.03	20.82	55.73
	5	3593.44	38.78	310.84	243.90	8.65	3550.06	1004.85	28.31	55.56
$w_1 = .999, w_2 = .001$	**0**	3747.75	156.75	612.55	250.36	16.35	3812.51	1296.03	33.99	56.09
	1	3735.79	128.62	616.77	229.78	16.51	4065.98	1369.88	33.69	55.82
	2	3730.34	114.60	650.97	267.58	17.45	3609.02	1128.69	31.27	55.00
	3	3741.96	107.70	615.63	238.94	16.45	3774.27	1141.45	30.24	55.29
	4	3737.44	111.86	617.79	273.20	16.53	4084.38	1395.68	34.17	55.21
	5	3740.83	145.18	615.56	250.67	16.46	3548.82	1016.95	28.66	55.72

Table 6.40 Descriptive statistics of MCAS-MHR-2 for the 400 point network

Weights	γ	$\overline{Q}$	$SD(Q_j^{\Sigma})$	$\overline{S}$	$SD(S_j^{\cap})$	P_1	R^{Σ}	$R^{\cap}$	P_2	$\overline{T}$ **(Sec.)**
$w_1 = 1.0, w_2 = 0.0$	**0**	3339.39	38.59	34.80	81.36	1.04	3239.44	0.00	0.00	135.04
	1	3342.62	40.62	41.78	92.60	1.25	3186.64	241.34	7.57	133.79
	2	3345.17	35.52	24.50	70.98	0.73	3261.64	0.00	0.00	135.93
	3	3342.08	35.89	44.93	94.09	1.34	3261.90	0.00	0.00	134.79
	4	3340.79	37.23	53.81	96.21	1.61	3234.37	241.34	7.46	135.75
	5	3335.33	45.24	31.06	78.62	0.93	3197.35	412.80	12.91	135.47
$w_1 = .9999, w_2 = .0001$	0	3343.75	40.04	70.17	105.89	2.10	3264.22	380.03	11.64	135.97
	1	3355.58	48.88	101.50	142.10	3.03	3244.43	516.50	15.92	135.30
	2	3349.84	43.95	92.01	125.33	2.75	3267.76	402.77	12.33	134.09
	3	3353.93	45.21	85.96	138.72	2.56	3232.26	241.34	7.47	136.08
	4	3351.63	42.97	107.48	160.30	3.21	3334.68	456.43	13.69	136.87
	5	3346.46	45.10	82.42	146.13	2.46	3196.40	241.34	7.55	135.22
$w_1 = .999, w_2 = .001$	0	3466.34	102.51	289.11	194.28	8.34	3528.09	768.50	21.78	135.19
	1	3472.51	89.87	346.19	195.18	9.97	3542.69	857.55	24.21	134.47
	2	3471.20	92.79	348.81	185.79	10.05	3482.27	747.83	21.48	133.85
	3	3470.18	114.92	319.52	197.79	9.21	3532.38	1015.93	28.76	134.56
	4	3477.17	102.54	355.33	187.84	10.22	3612.18	988.48	27.37	135.05
	5	3477.33	98.62	349.55	196.05	10.05	3475.93	840.10	24.17	133.75



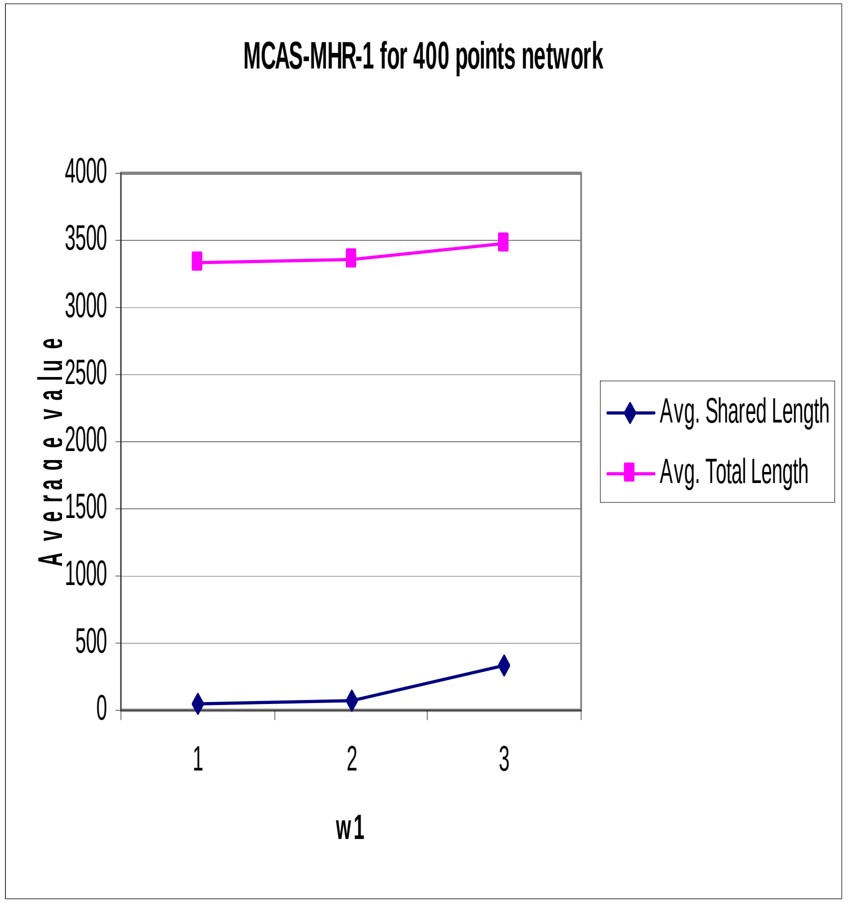
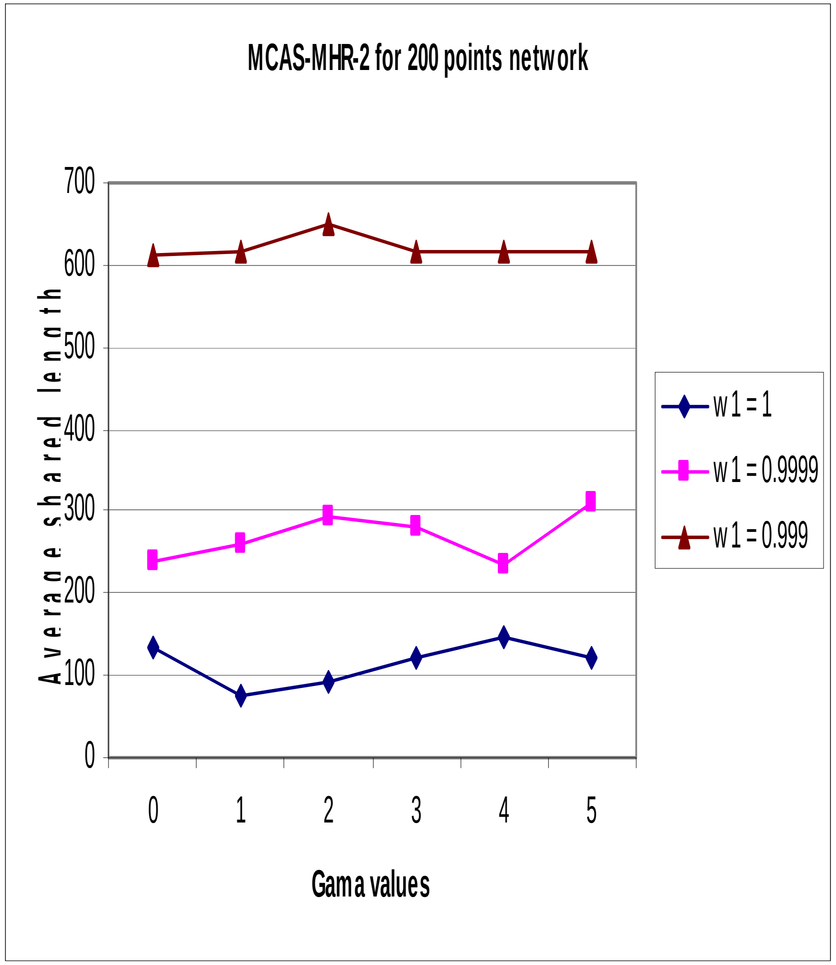


Fig. 6.54 Average values of total lengths and shared lengths of MCAS-MHR-1 for the 200 and 400 point networks against different values of weights



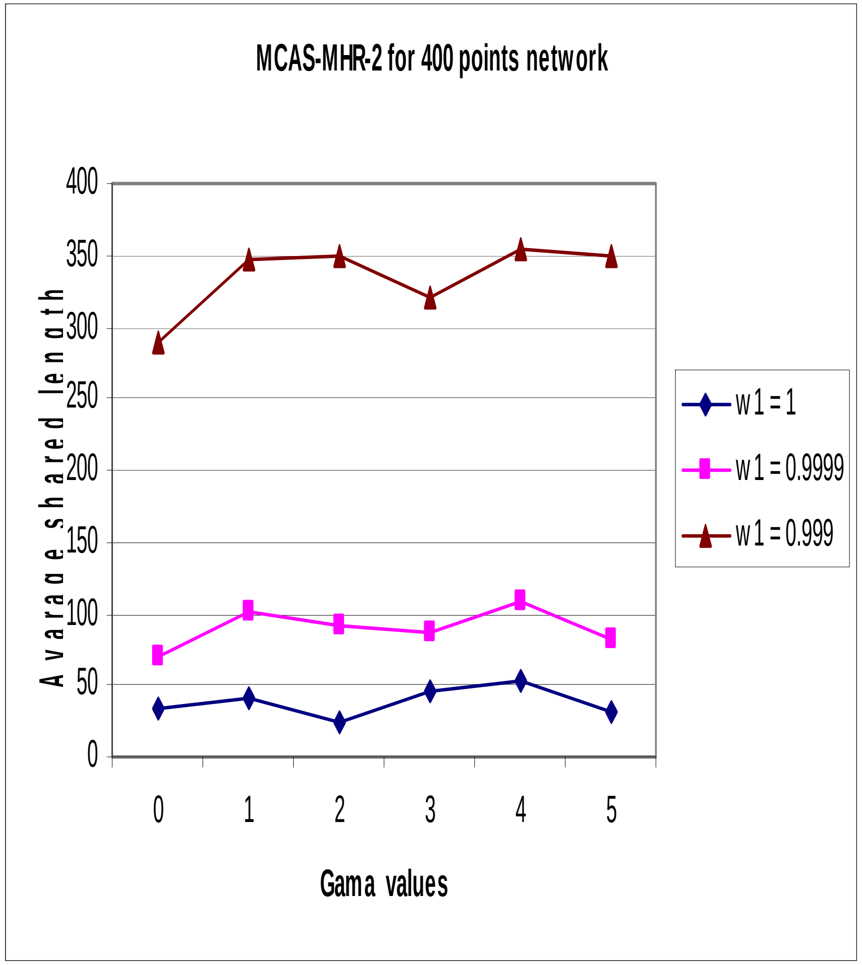
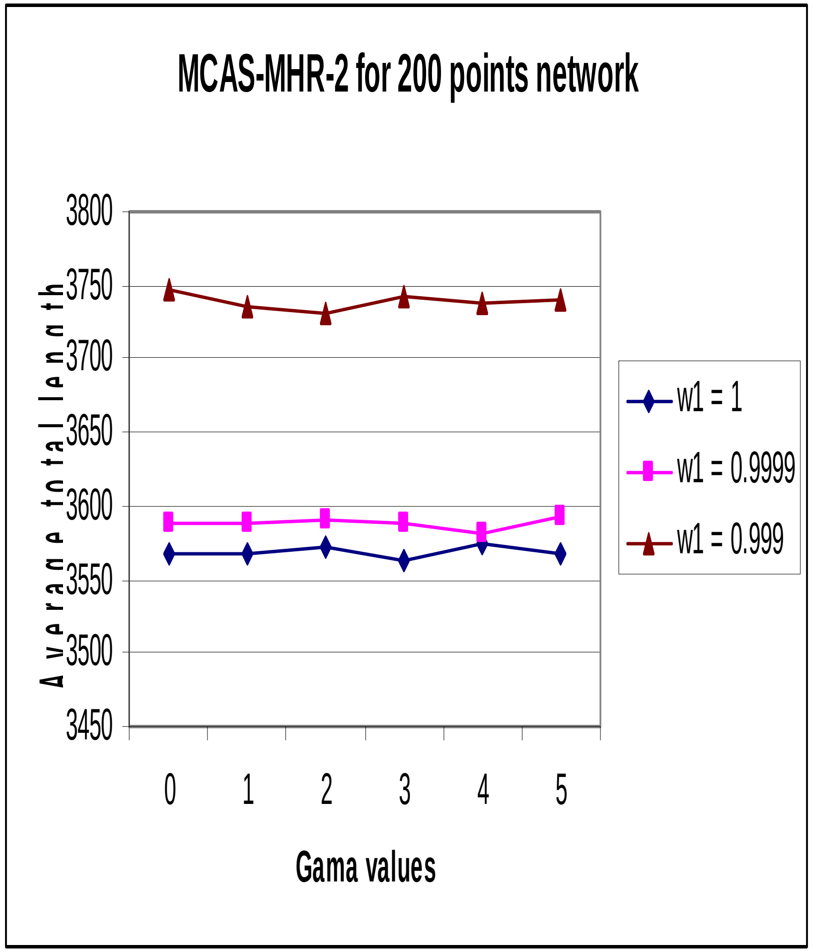


Fig. 6.55 Average shared lengths of MCAS-MHR-2 for the 200 and 400 point networks against different values of γ



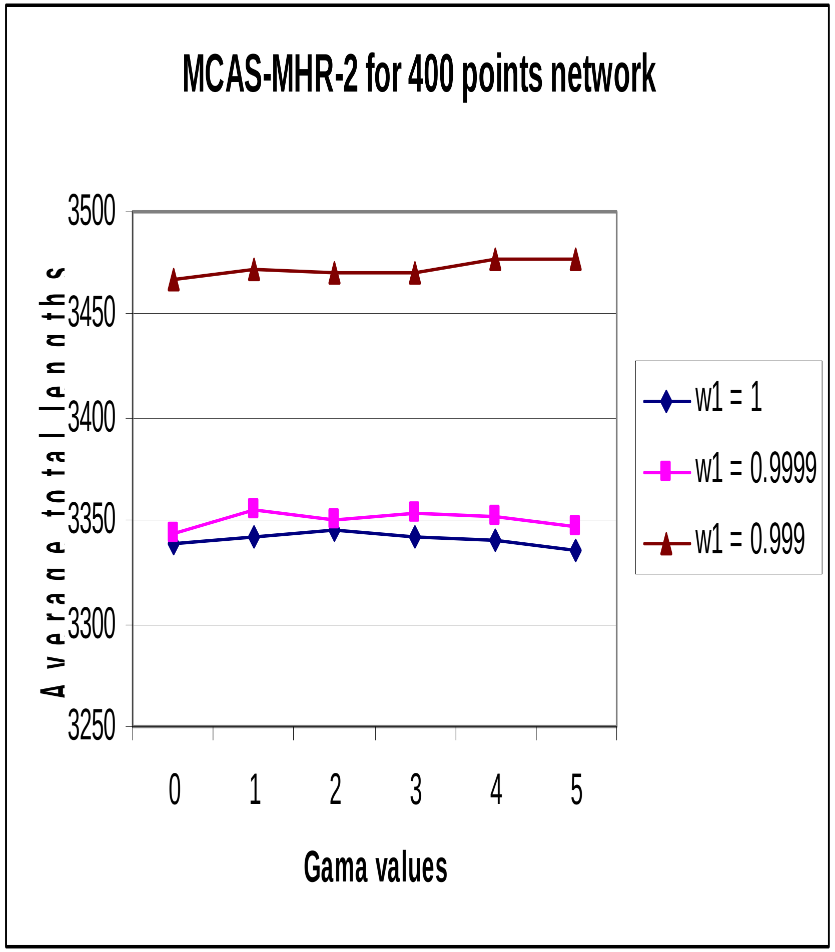
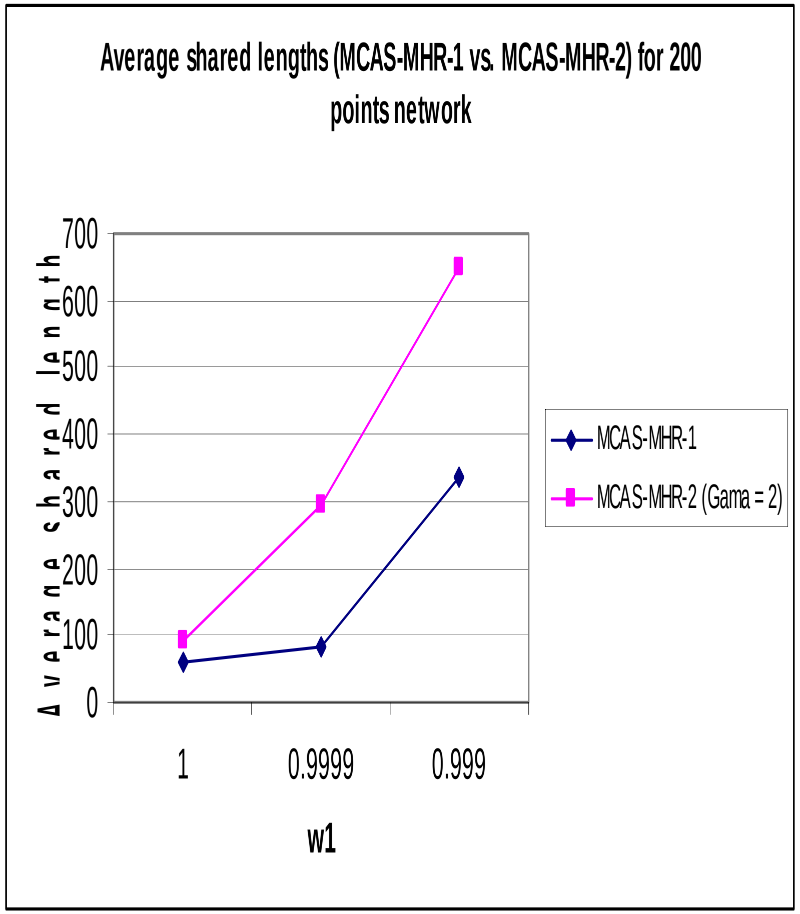


Fig. 6.56 Average total lengths of MCAS-MHR-2 for the 200 and 400 point networks against different values of γ



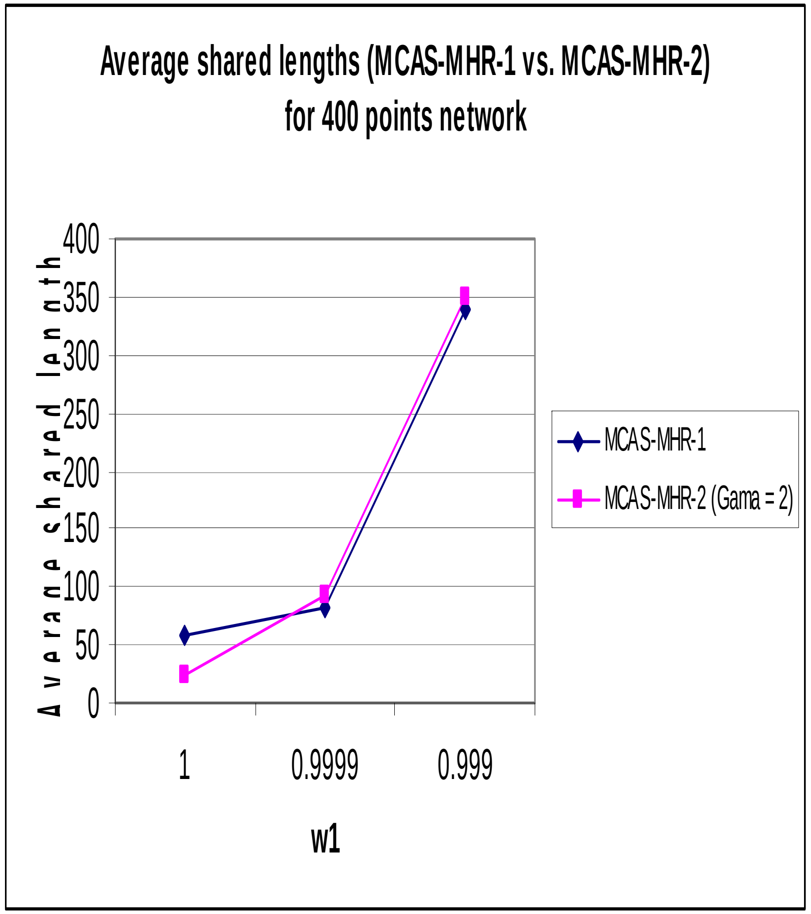
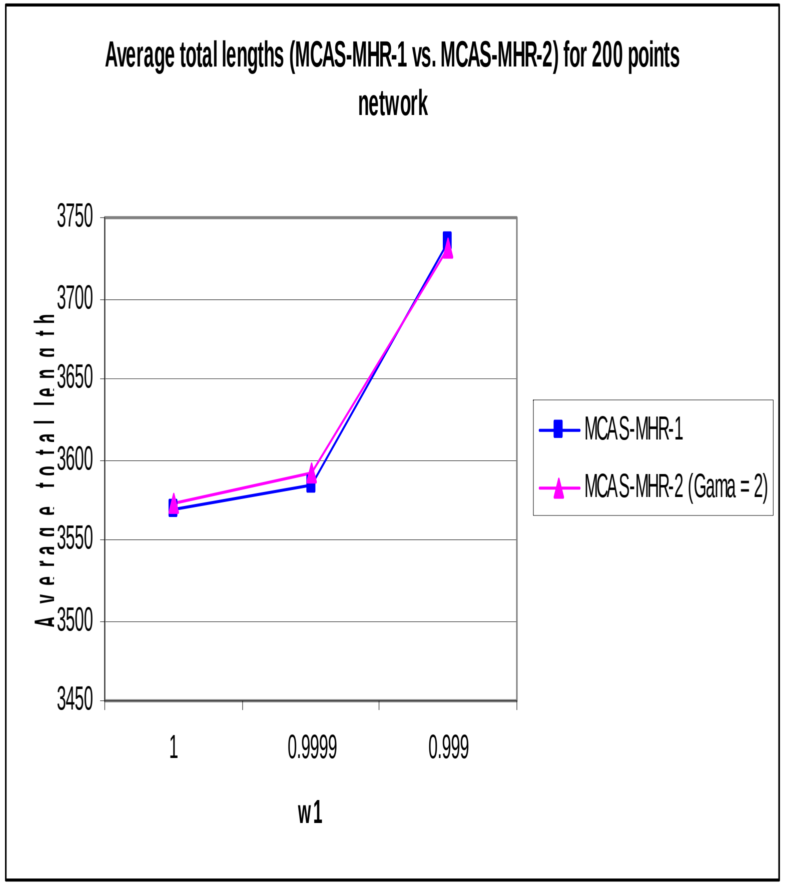


Fig. 6.57 Comparison of average shared lengths of MCAS-MHR-1 and MCAS-MHR-2 ($\gamma = 2$) of the 200 and 400 point networks against different values of weights



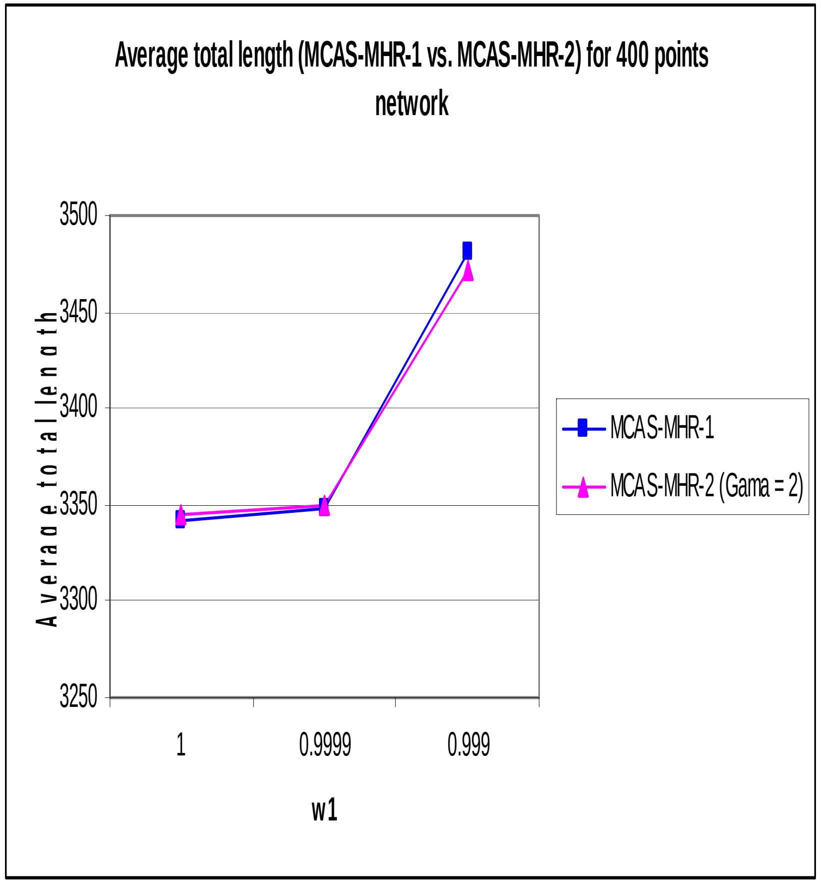
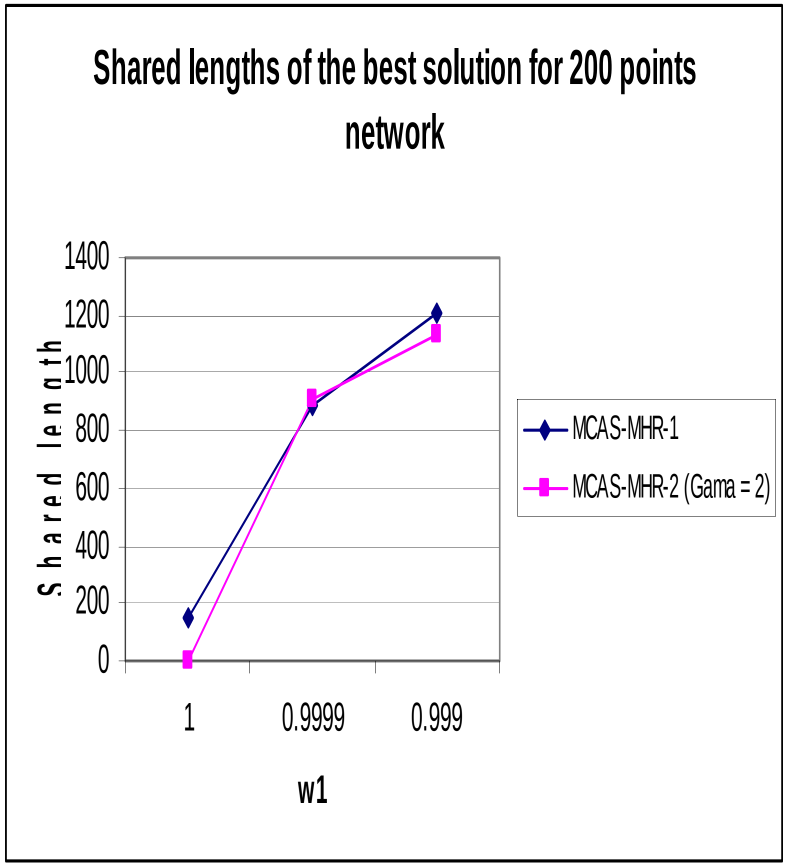


Fig. 6.58 Comparison of average total lengths of MCAS-MHR-1 and MCAS-MHR-2 ($\gamma = 2$) of the 200 and 400 point networks against different values of weights



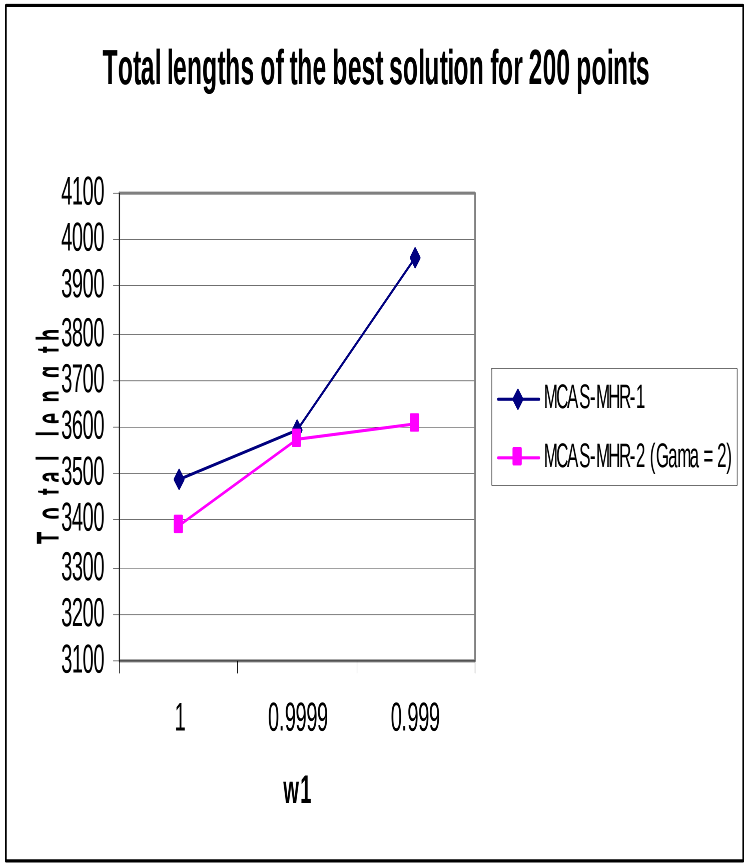
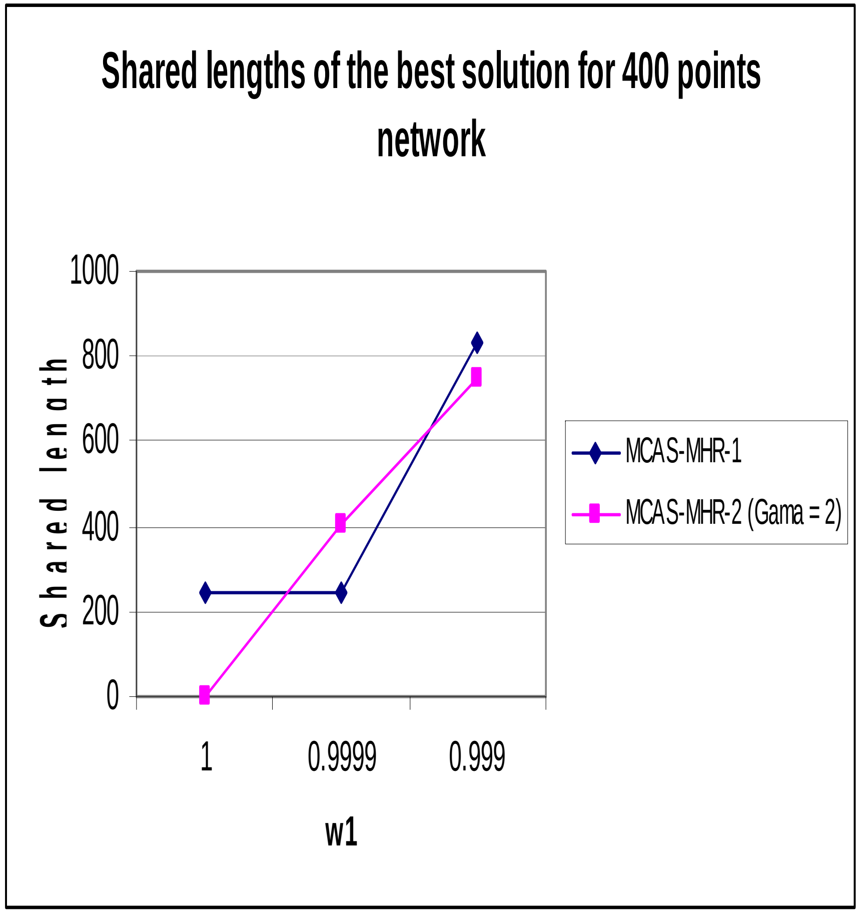


Fig. 6.59 Comparison of shared lengths and total lengths of the best solutions of MCAS-MHR-1 and MCAS-MHR-2 ($\gamma = 2$) for the 200 point network against different values of weights



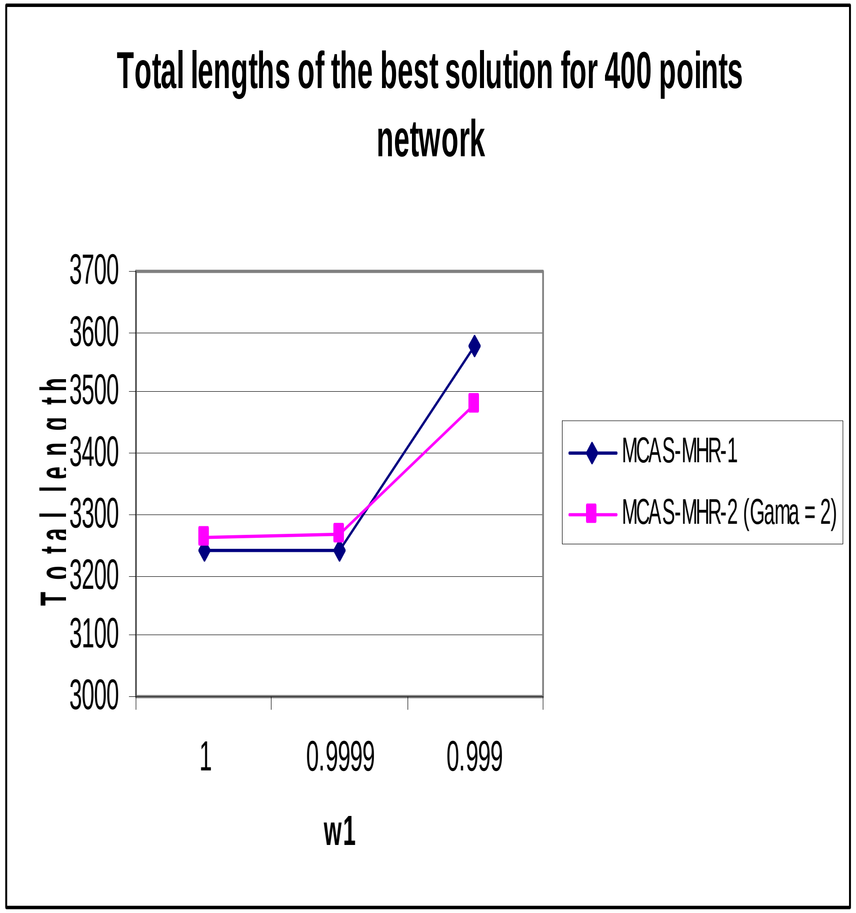
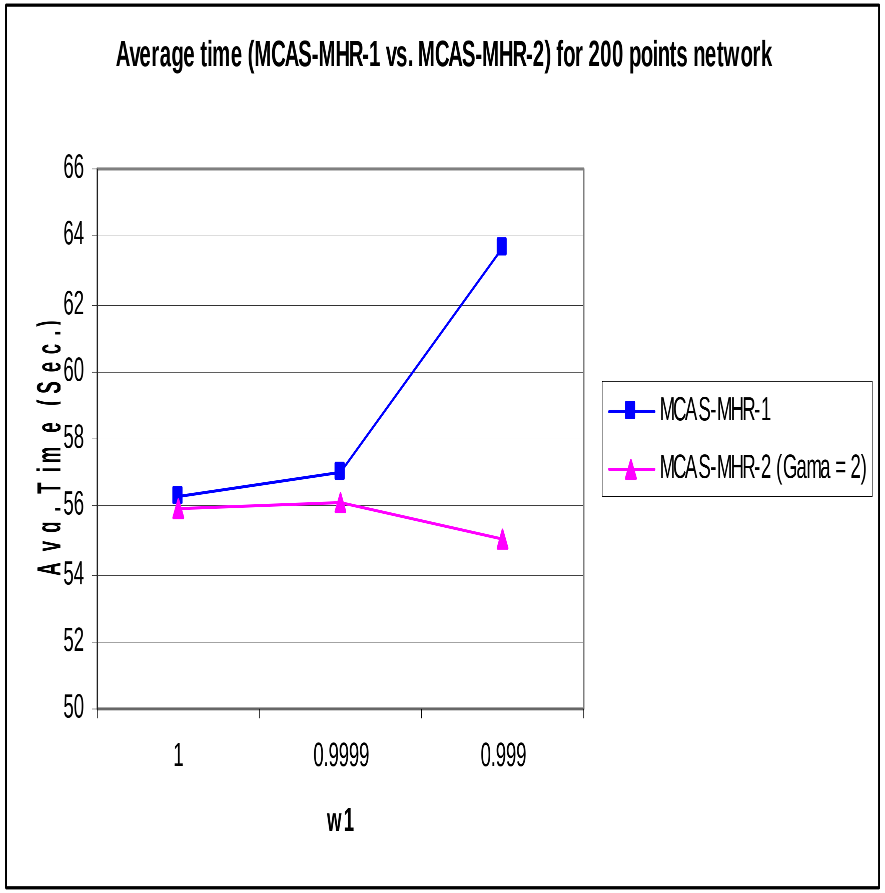


Fig. 6.60 Comparison of shared lengths and total lengths of the best solutions of MCAS-MHR-1 and MCAS-MHR-2 ($\gamma = 2$) for the 400 point network against different values of weights



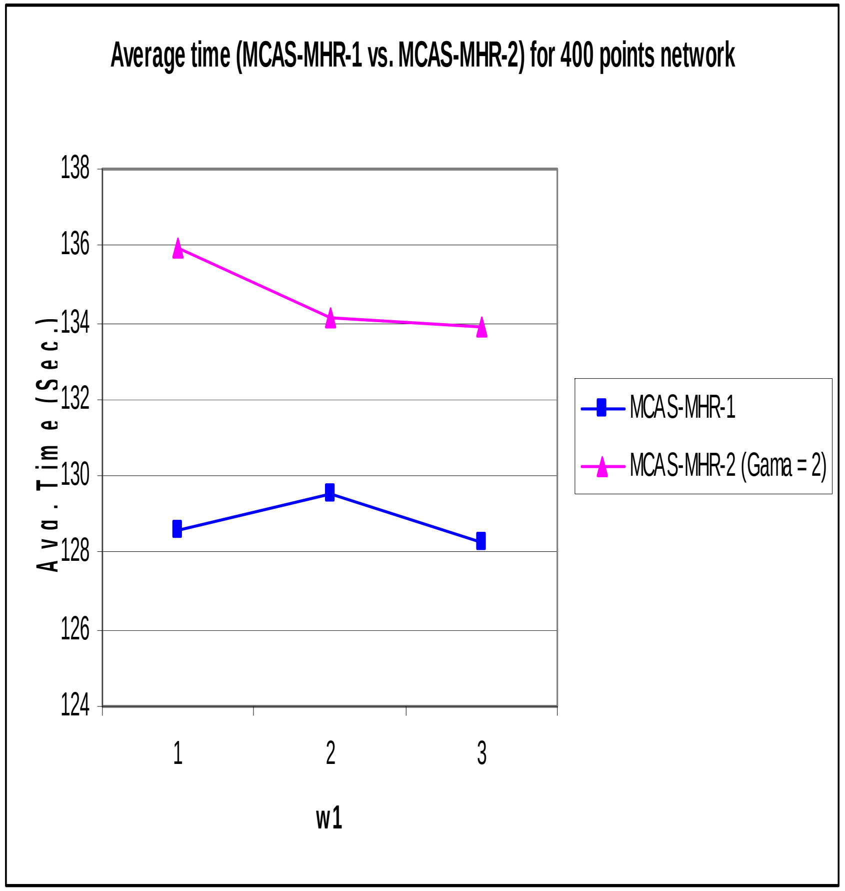


Fig. 6.61 Average time per trial of MCAS-MHR-1 and MCAS-MHR-2 ($\gamma = 2$)
for the 200 and 400 point networks against different values of weights

6.6 Summary of the Chapter

This chapter presented experimental results obtained for automatic hose/pipe routing of 3D models using ant colony optimization. Initially ant colony optimization was restricted to a single objective (length of the hose) and was tested on two virtual networks, grid and random, drawn in the CAD model. According to the initial experiments (see section 6.1), the results showed that ant colony optimization based on randomly generated networks was able to produce a reasonable solution in a reasonable time. Further, the failure rate of the ACO based on random networks is low compared to that for grid networks. The results of the two modifications MAS and N_MAS (sections 6.2 and 6.3) showed that they have improved the performance of the ant system (AS). In section 6.4, the results of the multi-objective ant colony optimization were presented. According to these results, the proposed multi-objective ant colony optimization algorithm PSACO generates better solutions compared to the currently best MOACO for compromised (trade-off) solutions. Finally, the results of the two multi-colony ant systems (MCAS-MHR-1 and MCAS-MHR-2) for multi-hose routing were presented in section 6.5. According to these results, there is no significant difference in the quality of the solutions of the two algorithms. However, MCAS-MHR-1 takes less computation time, has a smaller number of parameters and requires lower memory capacity.

7 CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

The research described in this thesis has covered the application of ant colony optimization to finding the optimum layout of hose/pipe routing with several objectives and parallel multi-hose routing.

Initially, two versions of an ant colony algorithm based on networks generated from grid points and random points have been proposed for automatic 3D hose routing. These two versions were restricted to minimizing the total length of pipes and avoiding the obstacles. According to the results obtained it is clear that ant colony algorithms executed on random-based networks were able to find good solutions in a competitive time. For avoiding obstacles when generating the pipes, the C++ library RAPID was incorporated into the program and the algorithms were able to handle complex models and any shape that can be generated using a CAD package.

Two modifications (see sections 5.4 and 5.5) were introduced to the ant colony optimization algorithm and were empirically shown to significantly improve the ant colony algorithm, ant system (AS).

7.1.1 Pareto Strength Ant Colony Optimization for Multi-Objective Hose Routing

In Section 5.6, a general purpose Pareto strength ant colony optimization algorithm (PSACO) has been introduced and applied to automatic multi-objective hose routing in 3D space. A single pheromone matrix has been used for each of the objectives and the algorithm updates its pheromones based on the domination concept.

The results were compared with the Pareto-based ant colony optimization algorithm P-ACO. P-ACO is considered as the best Pareto based ACO algorithm that generates good solutions at the central part of the Pareto front. This algorithm also uses separate pheromone matrices for each of the objectives.

The algorithms optimize three objectives: total lengths of the hoses, total number of bends and the angles of bends. Further, a modification has been introduced to the random proportional rule of both of the algorithms to attract ants towards the edges that make bend angles closer to catalogue angles. In addition, the set of neighbour nodes that remains to be visited by an ant has been modified such that the edges make bend angles with the previous edge close to one of the catalogue angles.

As in the initial experiments, the tessellated format (STL format) of the original objects and the C++ library, RAPID, have been used for collision detection. As a result of these, the proposed algorithm can handle free-form obstacles and is not restricted to a particular CAD package. Algorithms do not need to call the collision detection algorithm during the execution, as the random network is created at the beginning of the algorithm. As a result of this, Pareto ant colony algorithms can reduce the computation time required.

Algorithms have been compared graphically (scatter-plot matrix, value path, and bar chart) and in terms of metrics of performances (using M_2^*, M_3^* and C metrics) and computation time. Graphically, there is not much difference between the two algorithms for the first tested model - Model 1 (see Fig. 6.32). But the difference is significant for Model 2 (see Fig. 6.32). According to the metrics M_2^* and M_3^*, PSACO generates better distributions of non-dominated solutions and better maxima in each dimension of the non-dominated solutions. From the *C*-metric, it can be concluded that PSACO is very competitive compared to P-ACO. For Model 1, non-dominated solutions obtained by the two algorithms do not show much difference in terms of the *C*-metric while the former (PSACO) offers a good set of non-dominated solutions for Model 2, which in most cases dominate the solutions returned by P-ACO. In addition, the non-dominated solutions obtained by P-ACO for Model 2 have not dominated the non-dominate solutions of PSACO.

In terms of computation time, the results showed that PSACO takes less computation time for both of the models tested.

Since PSACO uses a single pheromone matrix, it needs only a fixed amount of main memory for all of the objectives whilst P-ACO's memory requirement increases with the number of objectives in the problem.

From a theoretical point of view, PSACO has a sound theoretical background in terms of the current state of the art in multi-objective optimization. Furthermore, it can be used as a general purpose multi-objective ant colony algorithm for other problems as well.

Since a multi-objective optimization algorithm obtains more than one solution, three metrics: *l_p-metric, Tchebycheff metric* and *Pseudo-Weight Vector Approach* were used for choosing a solution out of the non-dominated solutions returned by the algorithm. These final results also showed that PSACO's solution has not been dominated by the corresponding solution of P-ACO.

The refinement algorithm was applied to the final solution obtained for the second tested model - Model 2 (using *lp-metric* and *Tchebycheff-metric*). The results obtained after refinement improved the former. Moreover, the computation time for a run of the refinement algorithm is low compared to that for searching the entire search space. Hence, it is recommended to run the refinement algorithm to improve the solution obtained by using the entire search space.

7.1.2 Proposed Ant Colony Algorithms for Multi-Hose Routing

In Section 5.8, two versions of multi-colony ant systems (called MCAS-MHR-1 and MCAS-MHR-2 respectively) have been introduced for routing multiple hoses/pipes in parallel. The two versions use a separate colony for each commodity (pairs of start and end points). The first version uses a single pheromone matrix for all colonies whilst the second version uses a separate pheromone matrix for each colony. Thus, ants in the second version (MACS_MHR_2) were able to smell different pheromones individually laid by ants in other colonies. Ants in the first version cannot recognize pheromones individually as all pheromones laid by ants from different colonies on an edge are summed up to a single value. A further modification was introduced to the random propositional rule in MCAS-MHR-2 to attract ants towards edges that were used by ants of other colonies. When pheromone updating, both methods evaluate the quality of a solution according to not only the total lengths of paths but also the shared length of paths.

An objective of this work is to apply these methods to multi-hose routing with maximum use of common edges. Initially, the two methods were tested on a simple test graph (see section 6.5.1) using two commodities. In Experiment 2 (see section 6.5.2), the two methods have been applied to multi-hose routing in a complex 3D CAD model using two randomly generated networks of size 200 points and 400 points and 4 commodities.

In both algorithms, the relative importance of the total length and the shared length must be identified; accordingly respective weights (w_1 and w_2) must be selected. It is difficult to select pre-defined values for w_1 and w_2 for all routing problems. The best way to obtain a better result is to run the algorithms with different values of w_1 and w_2 and to select the appropriate solution from solutions produced over different values of w_1 and w_2.

When comparing the overall best solutions, MCAS-MHR-1 gives greater shared lengths, whilst MCAS-MHR-2 produces greater total lengths in most cases.

When comparing the average shared lengths, MCAS-MHR-2 performs slightly better than MCAS-MHR-1. Obviously this is a result of the use of foreign pheromones in the random propositional rule used in MCAS-MHR-2. However, MCAS-MHR-2 uses an additional parameter (γ) and needs a separate pheromone matrix for each of the commodities; this increases the memory requirement of the algorithm as the number of commodities increases.

Computational times for the two algorithms do not show much difference for the simple network (200 points network). However, MCAS-MHR-1 takes less computational time for the 400 points network as it uses less memory access and fewer computations.

According to the results found in both versions, there is no significant difference in the quality of the solutions between the two versions: MCAS-MHR-1 and MCAS-MHR-2. Thus, MCAS-MHR-1 is recommended for this type of problem as it takes less computation time, requires a smaller number of parameters and has low memory requirements.

7.2 Recommendations for Future Work

The following suggestions are put forward for future investigations.

1. When creating the contender node list $J_i^*(r)$ of MAS (see section 5.4), the current best value is obtained from previous cycles or generations. But this could be further improved, if the best solutions generated in the current cycle are also taken into consideration.
2. When introducing two types of ants into MAS (see section 5.5), the pheromone values $\tau(r,u)$ in the state transition rule of the explorer ants have been taken to the power -1 ($1/\tau(r,u)$). Other negative power values (integers and real values) could be tested for the pheromone values, thereby controlling how much the explorer ants are repulsed by the pheromone values.
3. In this thesis, the final solution of the Pareto ant colony optimization in multi-objective hose routing was selected using the methods described in section 3.4 (see also section 6.4.7). However, when selecting the final solution in this way, the algorithm does not take into account problem-specific higher-level information (non-technical, qualitative and experience-driven) that cannot be incorporated into the model (see also Fig. 3.1).

   Instead of relying on the experience of a skilled engineer for selecting the final solution, an intelligent system could be designed which incorporates higher-level information relating to hose/pipe routing and this system could then suggest the final solution from non-dominated solutions obtained by Pareto ant colony optimization algorithms (see Fig. 7.1).

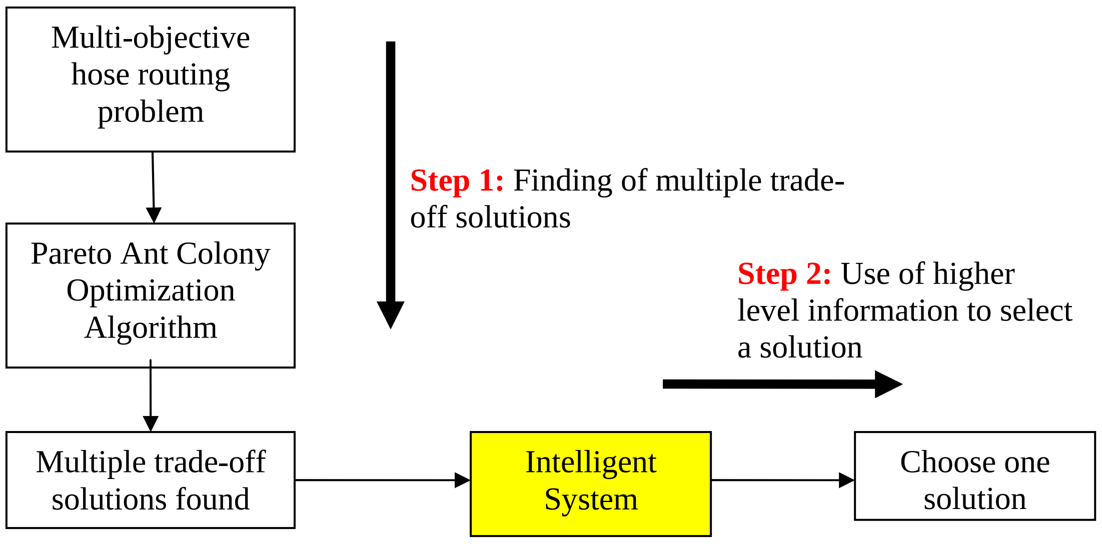

Multi-objective hose routing problem
Pareto Ant Colony Optimization Algorithm
Multiple trade-off solutions found
Step 1: Finding of multiple trade-off solutions
Step 2: Use of higher level information to select a solution
Intelligent System
Choose one solution

Fig. 7.1 Intelligent system which incorporates higher-level information for selecting a solution

4. The multi-colony algorithms (MCAS-MHR-1 and MCAS-MHR-2) proposed in section 5.8 were run on a single PC. The next step is to implement each colony on different PCs (grid computing) and speed up the algorithms.
5. Finally, these two versions use the classical approach of multi-objective optimization (weighted sum approach) and thus another possibility for improvement would be to use Pareto optimization techniques.

ACRONYMS

ACO	Ant Colony Optimization
ACS	Ant Colony System
AEDG	Alternative Energy Distributed Generation
AGSS	Airport Ground Service Scheduling
AI	Artifitial Intelligence
AS	Ant System
ASCII	American Standard Code for Information Interchange
CAD	Computer Aided Design
CD	Cell Decomposition
CG	Cell Generation
COMPETants	Competing Ant Colonies
CSP	Constraint Satisfaction Problem
EDD	Earliest Due Date First
GA	Genetic Algorithms
GAPRUS	Genetic Algorithms based Pipe Routing Using Tessellated Objects
GBI	GA-Based Inspiration
HC	Hill Climbing
JSP	Job Shop Scheduling
MACS	Multiple Ant Colony System
MACS-VRPTW	Multiple Colony System for Vehicle Routing Problems with Time Windows
MAS	The Proposed Reduced Sized Search Space for Ant Colony Optimization
MCAS	Multi Colony Ant System
MCAS-MHR-1	Multi Colony Ant System for Multi Hose Routing (Version 1)
MCAS-MHR-2	Multi Colony Ant System for Multi Hose Routing (Version 2)
M-MMAS	Max-Min Ant System
MOACO	Multi Objective Ant Colony Optimization
MOAQ	Multi Objective Ant-Q
MOGA	Multi Objective Genetic Algorithms
MONACO	Multi-objective Network Ant Colony Optimization
MOOP	Multi Objective Optimization Problem
N_MAS	The Proposed Explorer Ants for Avoiding Stagnation
ND	Non Deteministic
NP	Non-deterministic Polynomial
NSGA	Non-dominated Sorting Genetic Algorithm
P-ACO	Pareto Ant Colony Optimization
PC	Personal Computer
PRM	Probabilistic Road Map
PSACO	Pareto Strength Ant Colony Optimization
PSO	Particle Swarm Optimization
QAP	Quadratic Assignment Problem
RAPID	Robust and Accurate Polygon Interference Detection System
RBI	Rule-Based Inference
SA	Simulated Annealing
SCM	Supply Chain Management
SPEA	Strength Pareto Evolutionary Algorithm
SPRD	Ship Pipe Route Design

STL	Stereo Lithography
TC	Tree of Combination
TSP	Travelling Salesman Problem
VRPTSR	Vehicle Routing Problems with Tight time windows, Short travel time and Re-used Vehicles

REFERENCES

[1] Zhu, D. & Latombe, J.C. (1991). Pipe Routing = Path Planning (With Many Constraints). In Proc. IEEE, International Conference on Robotics and Automation, Sacramento, California – April, 1996, pp. 1940 – 1947.

[2] Brooks, R.A. & Lozano-Perez, T. (1982). A Subdivision Algorithm in Configuration Space for Findpath with Rotation. A1 Memo 684, A1 Lab., MIT, 1982.

[3] Zhu, D. & Latombe, J.C. (1989). New Heuristic Algorithms for Efficient Hierarchical Path Planning. Rep. STAN-CS-89-1279, Computer Science Dept., Stanford, Aug. 1989.

[4] Conru, A.B. (1994). A Genetic Approach to the Cable Harness Routing Problem. In Proc. Of the IEEE World Congress on Computational Intelligence, 27-29, June, 1994, pp. 200-205.

[5] Dijkstra, E.W. (1959). A Note on Two Problems in Connexion with Graphs. Numerical Mathematic 1, pp. 269-271.

[6] Park, H., Lee, H., & Cutkosky, M.R. (1991). Computational Support for Concurrent Engineering of Cable Harnesses. Computers in Engineering, Proceedings of the International Computers in Engineering Conference and Exhibit, Vol. 1, No. 1, San Francisco, USA, 1992, pp. 261-268.

[7] Petrie, C.J., Webster, T.A., & Cutkosky, M.R. (1995). Using Pareto Optimality to Coordinate Distributed Agents. Artificial Intelligence Engineering Design Anal Manufacturing 9(4), 1995, pp. 269-281.

[8] Cerezuela, C., Cauvin, A. Boucher, X., & Kieffer, J.P. (1998). A Decision Support System for a Concurrent Design of Cable Harnessed: Conceptual Approach and Implementation, Concurrent Eng. Res. Appl. 6(1), 1998, pp. 43-52.

[9] Ng, F.M., Ritchie, J.M., Simmons, J.E.L., & Dewar, R.G. (2000). Designing Cable Harness Assemblies in Virtual Environments. Journal of Materials Processing Technology 107 (2000), pp. 37-43.

[10] Kabul, I., Gayle, R. & Lin, M.C. (2007). Cable Route Planning in Complex Environments Using Constrained Sampling. ACM Symposium on Solid and Physical Modelling, Proceedings of the 2007 ACM symposium on Solid and physical modelling, Beijing, China, pp. 395-402.

[11] Sandurkar, S., & Chen, W. (1998). GAPRUS – Genetic algorithms based pipe routing using tessellated objects. The Journal of Computers in Industry.

[12] Thantulage, G., Kalganova, T. & Fernando, W.A.C. (2006). A Grid-based Ant Colony Algorithm for Automatic 3D Hose Routing. IEEE Congress on Evolutionary Computation, CEC 2006, Vancouver, Canada, Jul., 2006. pp. 48 – 55.

[13] Thantulage, G., Kalganova, T. & Wilson, M. (2006). Grid Based and Random Based Ant Colony Algorithms for Automatic Hose Routing in 3D Space. Transactions on Engineering, Computing and Technology, Volume 14, International Journal of Applied Science, Engineering and Technology (IJASET), Enformatika, ISBN 1503-5313, ISBN 975-00803-3-5, Aug., 2006. pp. 144 – 150.

[14] Drumheller, M. (2002). Constraint-Based Design of Optimal Transport Elements. ACM Symposium on Solid and Physical Modelling, Proceedings of the seventh ACM symposium on Solid modeling and applications, Saarbrücken, Germany: SESSION: Engineering Applications, ISBN:1-58113-506-8, pp. 401-412.

[15] Lee, C.Y. (1961). An Algorithm for Path Connections and its Applications. IRE Trans Electr Comp, 1961, pp. 346-365.

[16] Mitsuta, T., Kobayashi, Y., Wada, Y., & Kiguchi, T. (1986). A Knowledge-Based Approach to Routing Problems in Industrial Plant Design. In Proceedings of the 6th International Workshop on Expert Ststems & Their Applications, 28-30, April, Avignon, France, 1986, pp. 237-256.

[17] Kim, D.G., & Corne, D. (1996). Industrial Plant Pipe-Route Optimization with Genetic Algorithms. Parallel problem solving from nature IV, Berlin, Springer, 1996, pp. 1012-1021.

[18] Hesser, J., Maenner, R., and Stucky, O. (1989). Optimization of Steiner Trees Using Genetic Algorithms. In D. Schaffer, Editor, Proceedings of the Third International Conference on Genetic Algorithms, 1989, pp. 231-236.

[19] Ito, T. (1999). A Genetic Algorithm Approach to Piping Route Path Planning. Journal of Intelligent Manufacturing (1999) 10, pp. 103-114.

[20] Ahuja, N. and Hwang Y.K. (1992). Gross Motion Planning - A Survey. ACM Computing Surveys, 24(3), 219-291.

[21] Aurenhammer, F. (1991). Voronoi Diagrams - A Survey of Fundamental Geometric Data Structure. ACM Computing Survey, 23(3) (Sept.), pp. 345-405.

[22] Koditschek, D. E. (1989). Robot Planning and Control via potential functions. Robotics Review, Vol 1, MIT Press, Cambridge, Mass.

[23] Kang, S.S., Myung, S.H., & Han, S.H. (1999). A Design Expert System for Auto-routing of Ship Pipes. J Ship Prod 1999: 15(1), pp. 1-9.

[24] Park, J.H. & Storch, R.L. (2002). Pipe-Routing Algorithm Development: Case Study of A Ship Engine Room Design. Expert Systems with Applications Vol. 23, Issue 3, Oct 1, 2002, Science Direct, pp. 299-309.

[25] Ito, T. & Fuduka, S. (1998). Hybrid Approach to Piping Route Path Design Using GA-Based Inspiration and Rule-Based Inference. Volume 6, Number 4, Sage Publications, 1998, pp. 323-332.

[26] Dorigo, M. & Stutzle, T. (2004). Ant Colony Optimization, MIT Press, Cambridge, MA.

[27] Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, Vol. 26, No. 1, pp. 1-13.

[28] Colorni, A., Dorigo, M., & Maniezzo, V. (1991). Distributed Optimization by Ant Colonies. In proceedings of ECAL91 – European Conference on Artificial Life, Elsevier Publishing, pp. 134-142.

[29] Stützle, T. & Hoos, H. (1997). MAX-MIN Ant System and Local Search for the Travelling Salesman Problem, IEEE International Conference on Evolutionary Computation, pp. 309- 314

[30] Maniezzo, V., Dorigo, M., & Colorni, A. (1994). The Ant System Applied to the Quadratic Assignment Problem. Technical Report IRIDIA/94-28, Universit Libre de Bruxelles, Belgium.

[31] Dorigo, M, Gambardella, L. (1997). Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem, IEEE Transactions on Evolutionary Computation, 1:1. pp. 53-66.

[32] Cordon, O., Fernandez de Viana, I. & Herrera, F. (2002). Analysis of the Best-Worst Ant System and Its Variants on the TSP. Mathware & Soft Computing, 9:2-3. pp. 177-192.

[33] Bullnheimer, B. Kotsis, G. & Struss, C. (1999). A New Rank-based Version of the Ant System: A Computational Study. Central European Journal for Operations Research and Econimics, 7:1. pp. 25-38.`

[34] Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). From Natural to Artificial Swarm Intelligence. New York: Oxford University Press.

[35] Corne, D. Dorigo, M., & Glover, F. (Eds.). (1999). New Ideas in Optimization. Maidnhead, UK: McGraw-Hill.

[36] Gambardella, L.M., & Dorigo, M. (1999). Ant Algorithms for Discrete Optimization. Artificial Life 5: Massachusetts Institute of Technology. pp. 137-172.

[37] Engelbrecht, A.P. (2005). Fundamentals of Computational Swarm Intelligence. ISBN: 978-0- 470-09191-3.

[38] Deb, K. (2003). Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons Inc., ISBN: 0-471-87339-X.

[39] Miettinen, K. (1999). Nonlinear Multiobjective Optimization. Boston: Kluwer.

[40] Meisel, W.L. (1973). Tradeoff Decision in Multiple Criteria Decision Making. In J.L. Cochrane and M. Zeleny (Eds), Multiple Criteria Decision Making. Columbia, SC: University of South Carolina Press. pp. 461-476.

[41] Cleveland, W.S. (1994). Elements of Graphing Data. Murray Hill, N J: AT & T Bell Laboratories.

[42] Geoffrion, A. M., Dyer, J.S. & Feinberg, A. (1972). An Interactive Approach for Multi-criterion Optimization with an Application to the Operation of an Academic Department. Management Science 19(4), pp. 357-368.

[43] Zitzler, E., Deb, L. & Thiele, L. (2000). Comparison of Multi-objective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, 8:2, 2000. pp. 173-195.

[44] Coello, C.A., Van Veldhuizen, D.A. & Lamant, G.B. (2002). Evolutionary Algorithms for Solving Multi-objective Problems. Kluwer.

[45] Zeleny, M. (1973). Compromise Programming. In J.L. Cochrane and M. Zeleny (Eds.), Multiple Criteria Decision Making, Columbia: SC: University of South Carolina Press. pp. 262-301.

[46] Baran, B. & Schaerer, M. (2003). A Multi-objective Ant Colony System for Vehicle Routing Problem with Time Windows, Proc. Twenty first IASTED International Conference on Applied Informatics, Insbruck, Austria, Feb. 10-13, 2003. pp. 97-102.

[47] Cardoso, P., Jesus, M. & Marquez, M. (2003). Multi-objective Network Optimization Based on an ACO. Proc. X Encuentros de Geometria Computational, Seville, Spain, Jun. 16-17, 2003.

[48] Doerner, K., Hartl, R.F. & Teimann, M. (2003). Are COMPETants More Competent for Problem Solving? – The Case of Full Truckload Transportation, Central European Journal of Operation Research (CEJOR), 11:2, 2003. pp. 115-141.

[49] Doerner, K., Gutjahr, W.J. & Hartl, R.F. (2004). Pareto Ant Colony Optimization: A Meta-heuristic Approach to Multi-objective Portfolio Selection. Annals of Operational Research, 2004. pp. 79-99.

[50] Gambardella, L., Tailard, E. & Agazzi, G. (1999). MACS-VRPTW: A Multiple ACS for Vehicle Routing Problems with Time Windows. In: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, 1999. pp. 73-76.

[51] Iredi, S. Merkle, M. & Middendorf, M. (2001). Bi-Criterion Optimization with Multi-colony Ant Algorithms. Proc. First International Conference on Evolutionary Multi-criterion Optimization (EMO'01), LNCS 1993, 2001. pp. 359-372.

[52] Mariano, C.E. & Morales, E. (1999). A Multiple Objective Ant-Q Algorithm for the Design of Water Distribution Irrigation Networks. Technical Report HC-9904, Instituto Mexicano de Tecnologia, Jun., 1999.

[53] Garcia-Martinez, C., Cordon, O. & Herrera, F. (2004). An Empirical Analysis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-criteria TSP. LNCS, Volume 3172/2004, ISBN: 978-3-540-22672-7, Publisher: Springer Berlin/Heidelberg, Nov., 2004. pp. 61-72.

[54] Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, 6:2, 2002. pp. 182-197.

[55] Zitzler, E., Laumanns, M. & Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In K. Giannakoglou et al. (Eds.) EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Athens, Greece, September 2001. pp. 12-21.

[56] Silverman, B.W. (1986). Density Estimation for Statistics and Data Analysis. Chapman and Hall, London.

[57] Nowe, A., Verbeeck, K. & Vrancx, P. (2004). Multi-type Ant Colony: The Edge Disjoint Paths Problem. Ants 2004, LNCS 3172, Springer-Verlag Berlin Heidelberg, 2004. pp. 202-213.

[58] Sim, K.M., & Sun, W.H. (2003). Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions. IEEE Transactions, MAN, and CYBERNETICS – Part A: Systems and Humans, Vol. 33, No 5, Sep. 2003. pp. 560-572.

[59] Gottschalk, S., Lin, M.C., & Manocha, D. RAPID (Robust and Accurate Polygon Interface Detection).

[60] RAPID User Manual, <http://www.cs.sunysb.edu/~algorith/implement/RAPID/implement.shtml>

[61] A* Search Algorithm < http://en.wikipedia.org/wiki/A*_search_algorithm >

[62] Middendorf, M., Reischle, F. & Schmeck, H. (2002). Multi Colony Ant Algorithms. Journal of Heuristics, 8: 305–320, 2002, Kluwer Academic Publishers. Manufactured in The Netherlands. pp. 305 – 320.

[63] Goal Programming < http://en.wikipedia.org/wiki/Goal_Programming >

[64] Weise, T. (2007). Global Optimization Algorithms: Theory and Application, http://www.it-weise.de/

[65] Alam, S., Bui, L.T., Abbass, H.A., & Barlow, M. (2006). Pareto Meta-Heuristics for Generating Safe Flight Trajectories Under Weather Hazards. 6th International Conference on Simulated Evolution and Learning (SEAL'06) , LNCS 4247, Hefei, China. pp. 829-836.

[66] Alaya, I., Solnon, C., & Ghedira, K. (2007). Ant Colony Optimization for Multi-Objective Optimization Problems. Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence - Volume 01 (ICTAI – 2007). pp. 450-457.

[67] Pinto, D. & Baran, B. (2005). Solving multiobjective multicast routing problem with a new ant colony optimization approach. Applications, Technologies, Architectures, and Protocols for Computer Communication archive Proceedings of the 3rd international IFIP/ACM Latin American conference on Networking table of contents. Cali, Columbia SESSION: Multicast table of contents. pp. 11 – 19.

[68] Bui, L.T., Whitacre, J.M., & Abbass, H.A. (2008). Performance analysis of elitism in multi-objective ant colony optimization algorithms. IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence), Hong Kong. pp. 1633-1640.

[69] Chaharsooghi, S.K. & Meimand Kermani, A.H. (2008). An intelligent multi-colony multi-objective ant colony optimization (ACO) for the 0–1 knapsack problem. IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence), Hong Kong. pp. 1195-1202.

[70] Yagmahana, B. & Yanisey, M.M. (2007). Ant Colony Optimization for Multi-Objective Flow Shop Scheduling Problem. Computers & Industrial Engineering, Science Direct, Volume 54, Issue 3, April 2008, pp. 411-420.

[71] Sun, R., Wang, X, & Zhao, G (2008). An Ant Colony Optimization Approach to Multi-Objective Supply Chain Model. Second International Conference on Secure System Integration and Reliability Improvement, 2008. SSIRI '08, IEEE Computer Sociery. pp. 193-194.

[72] Chaharsooghi, S.K. & Meimand Kermani, A.H. (2008). An Effective Ant Colony Optimization Algorithm (ACO) for Multi-Objective Resource Allocation Problem (MORAP). Applied Mathematics and Computation, Volume 200, Issue 1, 15 June 2008, pp. 167-177.

[73] Panahi, H., Rabbani, M., Tavakkoli-Moghaddam, R. (2008). Solving an Open Shop Scheduling Problem by a Novel Hybrid Multi-Objective Ant Colony Optimization. Eighth International Conference on Hybrid Intelligent Systems, HIS '08, 10-12 Sept. pp. 320-325.

[74] Colson, C.M., Nehrir, M.H. & Wang, C. (2009). Ant colony optimization for microgrid multi-objective power management. Power Systems Conference and Exposition, 2009. PES '09. IEEE/PES, March 2009. pp. 1-7.

[75] Du, Y, Zhang, Q. & Chen, Q. (2008). ACO-IH: An improved ant colony optimization algorithm for Airport Ground Service Scheduling. IEEE International Conference on Industrial Technology, 2008. ICIT 2008, 21-24 April 2008. pp. 1-6.

[76] Fan, X., Lin, Y., & Ji, Z. (2006). The Ant Colony Optimization for Ship Pipe Route Design in 3D Space. The Sixth World Congress on Intelligent Control and Automation, 2006. WCICA 2006. pp. 3103-3108.

[77] Ma, X., Iida, K., Xie, M., Nishino, J., Odaka, T. & Ogura, H. (2006). A Genetic Algorithm for the Optimization of Cable Routing. Systems and Computers in Japan, Vol. 37, No. 7, 2006. pp. 21-30

[78] Liu, Q. & Wang, C. (2008). A Modified Particle Swarm Optimizer for Pipe Route Design. 11th IEEE International Conference on Computational Science and Engineering Workshops, 2008. CSEWORKSHOPS '08. pp. 157 – 161.

[79] Huibiao, L., Zhefu, Y. & Peiting, S. (2008). Hanging Bridge Algorithm for Pipe-Routing Design in Ship Engine Room. International Conference on Computer Science and Software Engineering, 2008. pp. 153-155.

.