65

AN AUTONOMOUS MISSION MANAGER FOR THE MULTIPLE VEHICLE SYSTEM

by

Tina H. Park

B.S., Aerospace Engineering
Georgia Institute of Technology (1995)

Submitted to the Department of Aeronautics and Astronautics

In Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 18, 1997

Signature of Author [illegible]
Department of Aeronautics and Astronautics
June 18, 1997

Approved by [illegible]
Dr. David S. Kang
Technical Supervisor

Approved by [illegible]
Robert A. Powers
Technical Advisor

Certified by [illegible]
Professor James D. Paduano
Thesis Supervisor

Accepted by [illegible]
Professor Jaime Peraire
Chairman, Committee on Graduate Students

An Autonomous Mission Manager for the Multiple Vehicle System

by
Tina H. Park

Submitted to the Department of Aeronautics and Astronautics on June 18, 1997
in partial fulfillment of the requirements for the degree of Master of Science
in Aeronautics and Astronautics at the Massachusetts Institute of Technology

Abstract

Clearing unexploded ordnance (UXO) is currently a dangerous and slow process that exposes personnel and equipment to considerable risk. A system using affordable, small, and autonomous robotic vehicles can be used to navigate to areas of indicated UXO, locate them, pick them up, and carry them away to an ordnance disposal area, or deploy charges on them to blow in place. This Small Autonomous Robotic Technician (SMART) system includes a control station with map building and path planning/re-planning capabilities. These capabilities enable the system to make and execute plans autonomously by means of an efficient road-building approach and an autonomous UXO-clearing approach.

Efficient road-building is an integral element of the SMART system. Considered in the context of optimization theory, road-building is equivalent to the vehicle(s) routing problem. The objective of this thesis is to solve the vehicle(s) routing problem in such a way that the time to build obstacle-free roads in a given area and to clear a given area, which contains UXO, is minimized without violating any constraints. A thorough study of possible routing algorithms was conducted. Algorithms for distribution of nodes to multiple vehicles were also considered. By combining one of two distribution algorithms (denoted "space filling curve" and "winner") with one of two routing algorithms ("mixed" and "greedy"), various solutions to the multiple-vehicle routing problem were developed.

This thesis also develops an autonomous planning system architecture, as well as management software capable of coordinating the multiple vehicle system. The mission manager software creates plans to carry out an Explosive Ordnance Disposal road-building mission that are non-optimal, but are reasonable and feasible. This software has been designed using heuristic algorithms to produce both the initial plans and re-planned plans. The algorithms have been implemented in C on the Linux operating system. The resulting mission planning system is tested in a simulated environment wherein commands are given to the vehicles to execute the mission plans. To close the loop, vehicles provide feedback to the mission manager. Simulated data using the mission manager is presented as a part of the results to evaluate the performance and effectiveness of the road-building approach for UXO clearance.

Technical Supervisor:	Dr. David S. Kang
Title:	Project Manager, Charles Stark Draper Laboratory
Technical Advisor:	Robert A. Powers
Title:	Technical Staff, Charles Stark Draper Laboratory
Thesis Supervisor:	Dr. James D. Paduano
Title:	Associate Professor, Department of Aeronautics and Astronautics

Acknowledgments

I must first thank my Draper supervisor Dr. David Kang for giving me the opportunity to work on the EOD project, my MIT advisor Professor James Paduano for providing guidance and support (even though I asked him at the last minute), my first Draper colleague Bob Powers for endless help, guidance, and advises throughout the research, and I also thank Michael Ricard for his help in the initial phase of the research.

I would like to thank the members of the EOD development team who have participated on the project, in particular to John Thele, for dynamic simulator and endless help in coding the algorithms; Bryan Koontz, for helpful resources and ideas; Ryan Norris, for the random test program; and Bill Kaliardos for helpful technical recommendations.

Special thanks go to The Charles Stark Draper Laboratory for providing generous financial support for my education and the EOD project.

Finally, I would like to thank my family and friends for their support and encouragement. I dedicate this work to mom and my boyfriend Tae, for their love, patience, and mental support.

Biographical Note

Tina Park was born in South Korea and raised in both South Korea and USA. She graduated from Fairfax Baptist Temple Academy in 1990 and completed Bachelor of Science Degree in Aerospace Engineering at Georgia Institute of Technology in June 1995. She completed her Master of Science Degree in Aeronautics and Astronautics at Massachusetts Institute of Technology in June 1997. She will start a career as a R & D engineer in control engineering department at Space Systems Loral in Palo Alto, CA.

Contents

List of Figures

List of Tables

Chapter One : Introduction

The application of robotics solutions to the problem of collecting and disposing of unexploded munitions offers the obvious benefit of reducing the risks to human life which the task would otherwise involve. To the extent that the robots can carry out their tasks autonomously, human workload is reduced, which is another benefit. If the target munitions are dispersed over a fairly wide area, then the use of multiple robots will reduce the time required to collect them. In this thesis and in Bryan Koontz's thesis [10], an approach to the problem based upon a centrally-controlled multi-vehicle autonomous robotic system is described. An introduction to the EOD (Explosive Ordnance Disposal) mission as currently performed is given below.

1.1 EOD (Explosive Ordnance Disposal) Mission

According to the *Boston Globe*, 110 million land mines reside in the world. The United States estimates that it will take more than 1,000 years to remove these mines at the current rate of clearance and will cost $33 billion to remove the mines that are already deployed. Furthermore, 2 to 5 million new land mines are deployed every year while only 100,000 mines are cleared [5]. Also, unexploded ordnance (UXO) reside in millions of acres of property in the United States due to the weapons system testing and troop training activities [14]. Furthermore, only 3,000 EOD technicians are available in the United States, and they must complete other tasks than clearing UXO [9].

Clearing UXO is currently a slow and dangerous process that exposes military personnel and expensive equipment to significant risk. The current manual UXO clearing process involves partitioning potential areas of having UXO into sectors with corners indicated by bright colored flags. A four-to-eight-man team is sent to visually sweep the area for UXO in each sector. Figure 1-1 depicts the terrain and natural obstacles that would present a challenge to robotic vehicles. All military personnel except the Marines carry out blow-in-place (BIP)[1] process when a UXO is detected. The Marines carry out the manual pickup and carry-away (PUCA) process to collect the UXO in an ordnance disposal area (ODA) to detonate later [9]. Figure 1-2 displays the variation in size, shape, and appearance of the UXO collected in a PUCA mission.

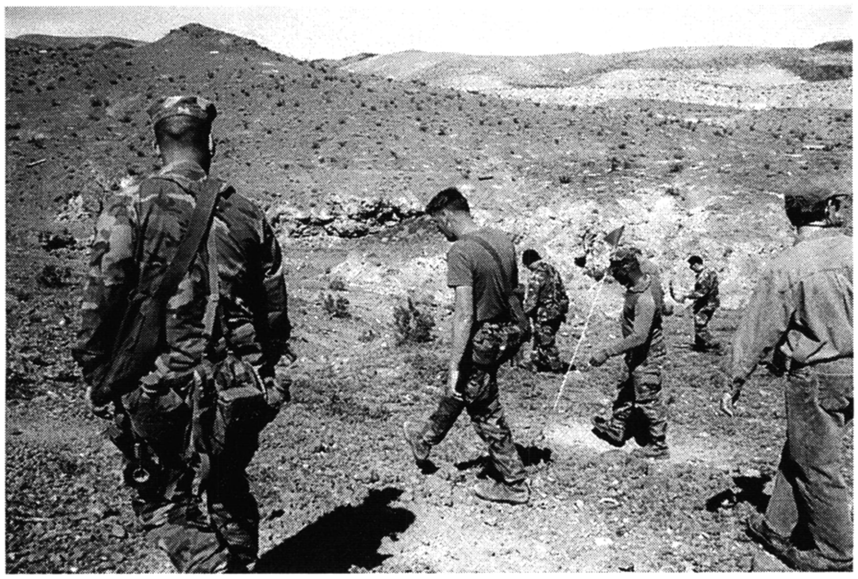


Figure 1-1: Sweep Team Looking for UXO

[1] Military personnel place a detonation charge, go away 1,000 yards, and return 30 minutes after the detonation.

Figure 1-2: UXO Gathered by the Sweep Team in a PUCA Mission

The risk is considered much greater in clearing the UXO than manually sweeping for the UXO. Three categories of UXO excavation technology are available for the military personnel. The oldest UXO excavation technology is the manual methods, which basically use shovels and other digging tools to excavate soil around the potential UXO. These methods present the most risk to the personnel. More upgraded methods use mechanized systems such as bulldozers, front-end loaders, and other heavy construction equipment. These methods produce faster and more efficient excavation than the manual methods. The most advanced excavation technology uses remote-controlled systems such as telerobotic and autonomous systems. These last methods provide the highest degree of safety to the personnel [14].

The UXO clearing process is not only labor intensive and dangerous, but also expensive. For example, DoD estimates to clear a 28,800 acres of property in Kaho'olawe Island, Hawaii is approximately $400 million, or about $14,000 per acre [14]. Therefore, the Research and Development Department of the Naval Explosive Ordnance Disposal Technology Division (NAVEODTECHDIV) is currently conducting a research and development program for small autonomous robots system to carry out PUCA and BIP processes.

The main objective of this thesis is derived directly from the functional requirements of the overall EOD mission, which is to clear UXO in an efficient manner. The main objective of the EOD mission is to provide affordable but autonomous robotics technology to clear UXO safely while reducing the number of personnel required. A multiple robotic vehicles system is an ideal selection to fulfill the overall EOD requirements. However, multiplicity and autonomous aspects require more advanced and efficient strategies for UXO retrieval. An autonomous mission manager provides a complex but efficient solution for UXO retrieval by a multiple vehicles system. Dividing the problem into two subsystems, namely planning and mapping, reduces the complexity of the mission manager system [9]. Furthermore, the management system is even more simplified by dividing into two distinct phases, which are the road-building, and clearing phases. These phases not only relieve the complexity of the mission manager but also present a more efficient way to carry out UXO retrieval processes.

1.2 Objectives

The objective of this thesis is to provide an efficient and effective mission manager to assign and control multiple vehicles to obtain obstacle-free roads, which assigns UXO nodes to multiple vehicles and creates obstacle-free roads along which they are visited. The manager must dynamically re-plan the vehicle paths as initially unknown obstacles are encountered. The objective is not to find the optimum paths, but to find feasible routes and re-plan efficiently.

1.3 Overview

Chapter 2 describes the initial mission planner in detail and addresses some additional challenges and issues. This chapter also includes some optimization background information related to the vehicle routing problem, including approaches for optimal, sub-optimal, and mixed solutions to the problem. Chapter 3 describes plan verification and re-planning, discusses challenges and issues, and describes a planner that re-plans for certain situations. Chapter 4 describes the mission manager and includes an extensive procedural steps along with the detailed explanations of the initial planner and verification manager procedures. Chapter 5 describes the implementation of the mission manager including the planning and managing architecture. The chapter briefly explains the simulation of the various tests to evaluate the mission manager. Chapter 6 includes several tests that have been run on the two implemented road planning algorithms to make comparisons between the algorithms. This chapter also describes the tests on different cases to evaluate the effectiveness of the management and coordination mechanisms, particularly for the road-building phase, and discusses the results of those tests. Chapter 7 concludes the thesis with a summary of the project, the merits of approaches to the road-building phase, and suggested areas for further research and development.

1.4 EOD System

The mission manager is part of an overall SMART system. To better understand the constraints of the problem and the information available, the solution sections describe the elements of the SMART system. These elements include the hardware, software, and the global positioning system. The configuration of the EOD robotic vehicles has evolved from current proven designs through both hardware and software testing. The baseline of the vehicle, the MITy-2, is a 6-wheel drive flexible frame micro-rover driven by battery power (Figure 1-3). Two new developments, which are the UXO local detection sensor and UXO manipulator system, are required for the EOD vehicles. These new developments are minimal compared to alternative totally new mobility concepts. The EOD project is one of the important robotic projects in IUVC (Intelligent Unmanned Vehicle Center). Refer to the Appendix A for the history of IUVC and descriptions of other on-going or past robotic projects.

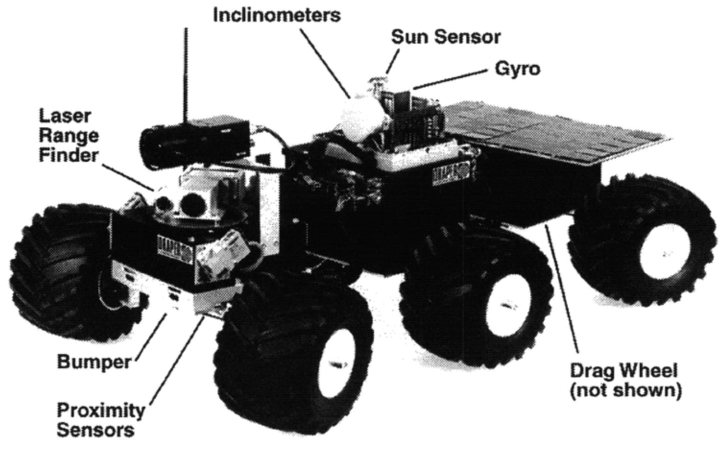


Figure 1-3: MITy-2 Micro-Rover

1.4.1 Hardware System

Both the hardware and software architectures of the EOD vehicle have been derived from the MITy-2. Two EOD vehicle prototypes (Figure 1-4) have been designed and built to test the feasibility of the EOD vehicles system for Mine Countermeasure operations. The baseline EOD vehicle is a 6-wheel drive flexible frame, which provides a high degree of maneuverability, driven by power generated by a small 12V DC motor. This six-wheel drive generates speed of the vehicle up to 6 fps or 1.8 mps [11].

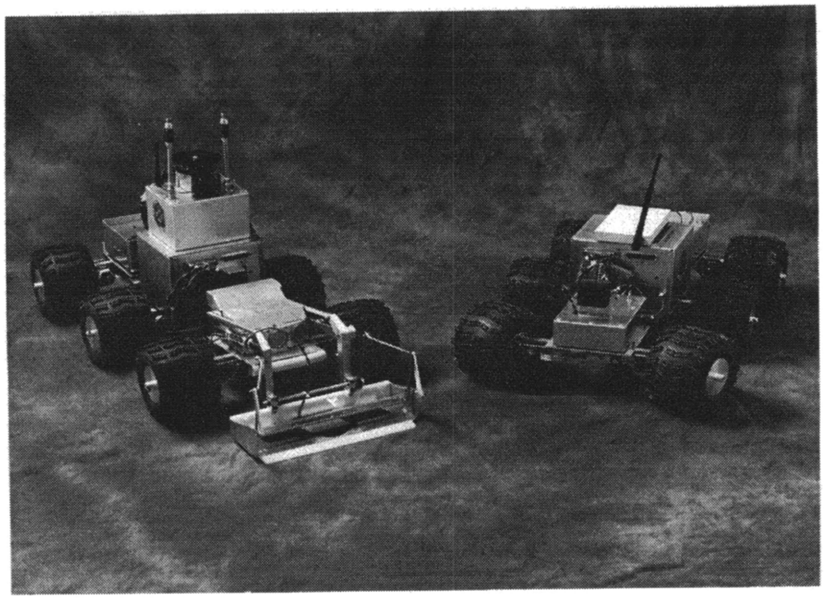


Figure 1-4: EOD SMART Vehicles

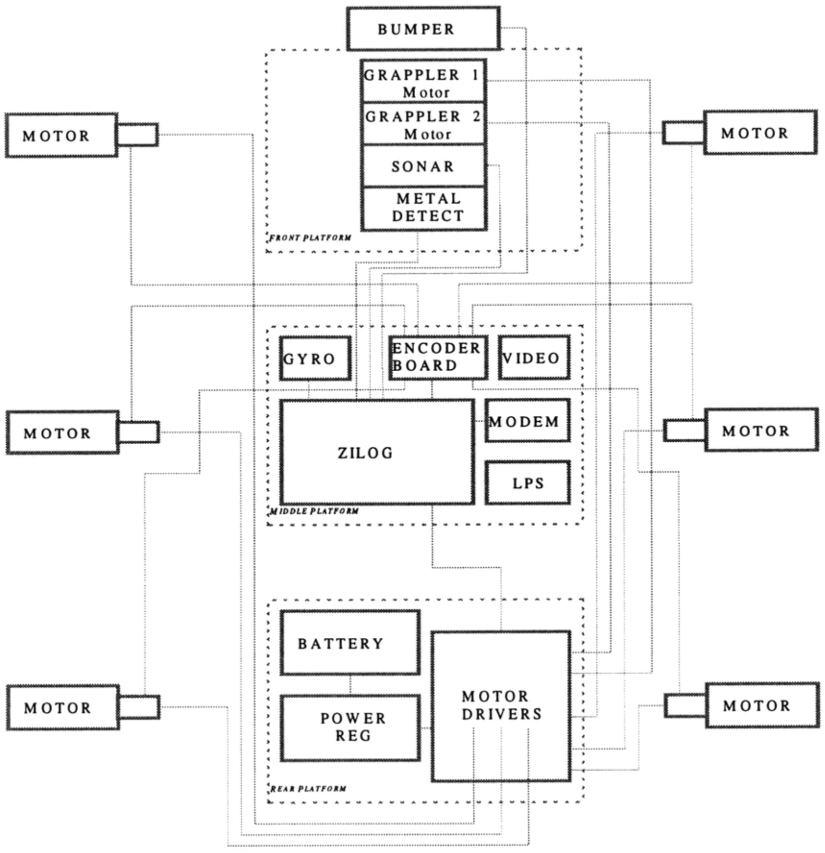


Figure 1-5: Schematic Showing Sensor and Electrical Hardware Placement

The flexible frame includes three individual platforms connected by flexible wires. Additionally, they are equipped with front and rear Ackerman steering mechanisms[2] and a modular chassis. The front platform includes a metal detecting unit, sonar electronics, a bumper, and the 2-DOF manipulator to acquire UXO with a metal detecting unit embedded in the base of the acrylic manipulator bed. The middle platform contains an onboard microprocessor, gyro, video camera and transmitter, serial modem, LPS (laser positioning system) transponder, and control circuitry. Lastly, the rear platform includes power regulation circuitry and battery packs (Figure 1-5) [11].

1.4.2 Software System

The software architecture for the EOD system is divided into two main areas: 1) onboard control software, and 2) control and ground station software. All software is developed under Linux. Onboard control software is programmed using C, then the software is compiled using a program "Dynamics C" to be downloaded to the processor on the vehicle. The ground station software is coded using C and the GUI (graphic-user-interface) is programmed using Motif. The ground station is located in a Pentium computer. The onboard control software is initially located in the same computer; then the control software is transferred to the processor on the vehicle.

1.4.2.1 Onboard Control

The onboard software for each of the vehicles is based on hierarchical design. This software is divided into layers, in such a way that the code for a layer depends only on the layers below itself. Thus, a lower layer would not be able to control a higher layer than itself. Any changes to a particular layer will thus only influence higher layers. The onboard control software includes code for handling high-level tasks, which let the vehicle complete a simple goal such as navigating to a point, initiating an area search for UXO, or picking up a detected UXO [11]. The control software can easily be expanded by adding layers or tasks. Most of the high-level task processing is done on the control or ground station since the onboard processor has limited memory.

1.4.2.2 Ground Station

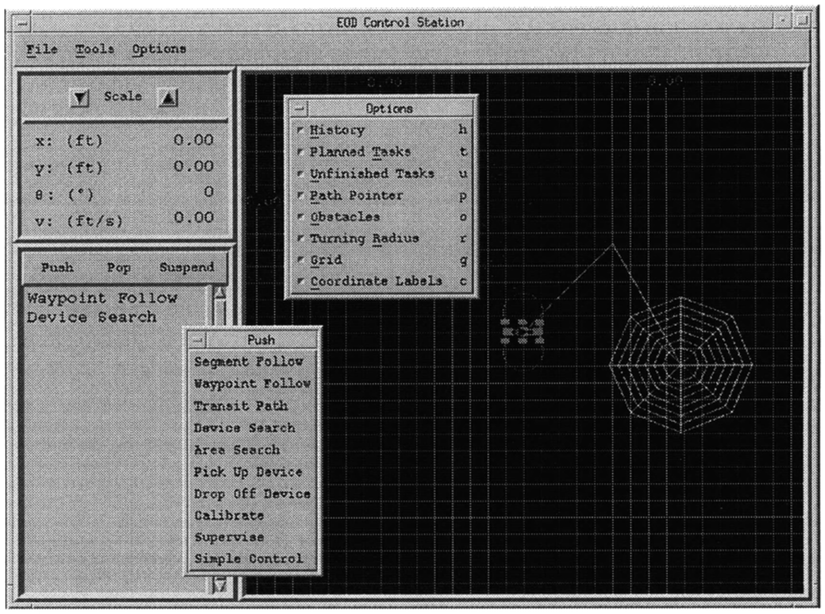
The ground station will communicate with each of the vehicles to receive telemetry and task-related data and to send task commands. The ground station carries out high-level mission control and management tasks. Figure 1-6 displays the control or operating station GUI. The mission control window interfaces the vehicle to the ground station (Figure 1-7). Some of the elements on this control window are the current assigned task, the current position, heading, and velocity of the vehicle. Multiple overlays displaying obstacles, sonar pings, path history, planned tasks, and a map grid can provide an overhead view of the vehicle but can be hidden if desired. Task parameters appear on the ground station display when a task needs to be created. Teleoperation or supervisory control of the vehicle is also supported to allow human operator interaction if needed [11].

The ground station for the EOD mission has two coupled tasks for path planning. The first task involves mapping with environmental information, to provide an updated map of the mission field at every cycle. This map contains the locations of UXO, known obstacles, and detected obstacles during the exploration phase. This structure combines global and local maps into one map to prevent confusion about the obstacle locations during the operation. This concept allows the dynamic growth of the map by providing speed and memory increases [18]. These capabilities will allow UXO clearance mission to be as autonomous as possible.

The second task (i.e. search) finds a path that minimizes a cost function between a start node and a final or goal node using the updated map. An A* algorithm that tries to find a path with the minimum cost has been developed [16]. Unlike Dijkstra's algorithm [15], which is a brute force method, the A* algorithm uses heuristics to yield a shortest path between two nodes in a reasonable time but not the optimal path.

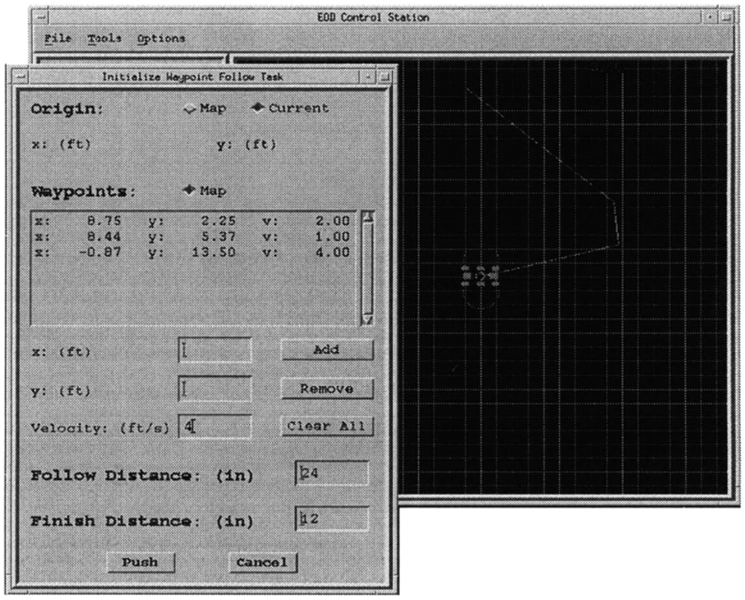
[2] This steering method is based on a design that terminates the center point of each wheel at a common point. The use of this type of steenng system reduces slippage during a turn, which reduces navigation errors.

This algorithm doesn't do an exhaustive search to minimize searching time [10]. Therefore, trades must be made between obtaining the optimal solution and generating the solution in the minimum time [3].



```
x: (ft)      0.00
y: (ft)      0.00
θ: (°)       0
v: (ft/s)    0.00
```

Figure 1-6: Control Station GUI Showing the Map Overlays and Available Mission Tasks



```
Origin:            Map   ◆ Current
x: (ft)                  y: (ft)

Waypoints:         ◆ Map
x:   8.75   y:   2.25   v:   2.00
x:   8.44   y:   5.37   v:   1.00
x:  -0.87   y:  13.50   v:   4.00

x: (ft)
y: (ft)
Velocity: (ft/s)   4

Follow Distance: (in)   24
Finish Distance: (in)   12
```

Figure 1-7: Control Station GUI Implementing a Waypoint-Follow Task

1.4.3 Global Positioning System

A laser positioning system (LPS) is used to correct navigational error from the dead reckoning system. This absolute positioning system provides a verification of vehicle's position using two synchronized laser beacons and onboard transponders (Figure 1-8). If the distance, l_1, between the two beacons is known then the angles θ_1 and θ_2 (calculated based on the timing sequence given by the transponder on the vehicle) can be used to compute the vehicle position [11].

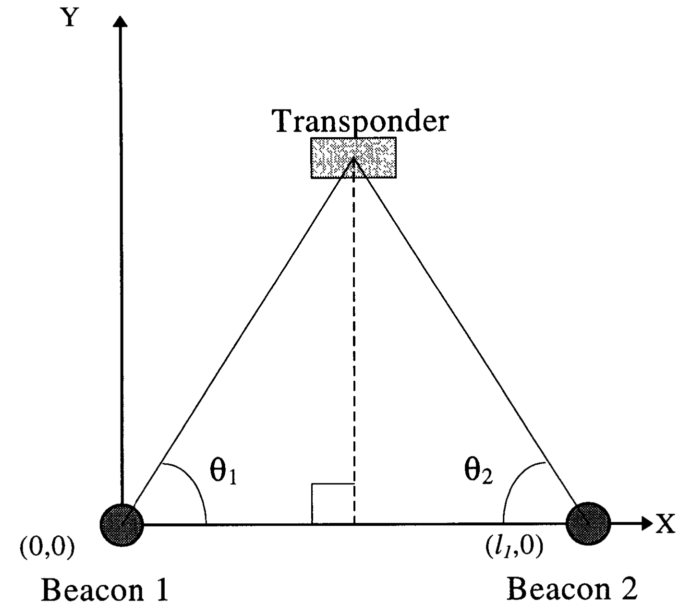


Figure 1-8: Laser Positioning System (LPS)

1.5 Requirements for the Autonomous Mission Manager

The locations of the UXO are known within the area of operation *a priori*[3]; however, the environmental conditions are not known *a priori*. The initial assumptions for the locations of the UXO are provided by a manual visual sweep, air or satellite reconnaissance, or an automated UXO survey vehicle [9]. In addition, the robotic vehicles should be able to do search task for the UXO within a given 1 m^2 area. Despite the small area to be searched, the vehicles must be able to pinpoint the exact location of the UXO. Since the terrain conditions are not assumed to be known initially, the robotic vehicles must also be able to avoid obstacles. A remotely located operator will monitor the multiple vehicles as well as coordinate activities for the PUCA operation. The operator station should be located at a safe distance from the area which contains potential UXO and the ODA. The use of the operator station will allow a single worker to safely accomplish the work that now requires and risks many ordnance disposal technicians.

In order to carry out the basic mission, multiple robotic vehicles must be coordinated and controlled. The vehicles are able to carry out simple tasks autonomously such as waypoint follow, obstacle or hazard avoidance, search, pick-up, drop-off, and others. The vehicles and the ground station will communicate by a two-way wireless link. However, the vehicle is not able to communicate to other vehicles in the field. Therefore, an operator or control station needs to coordinate the multiple vehicles. This operator station contains environmental information and vehicle information so that it can plan a series of commands and determine optimal routes for each of the vehicles. The hazard and location telemetry information from each of the vehicles is used to update the map on the operator station to do re-planning if required. Therefore, the key elements of an efficient autonomous vehicle(s) system are information fusion and vehicle management. The coordination of the vehicles can be accomplished more efficiently by an automated mission management system (Figure 1-9) on the ground station [9].

Using the planning and mapping capabilities, the mission manager can accomplish the UXO clearance mission by means of two distinct phases. The first phase is the "road-building" which tries to find obstacle-

[3] The locations of the UXO initially are assumed to be known within a 1-m^2 area.

free roads using the full capability of the sensors on the vehicles. Then, the "UXO-clearing" phase uses the network of obstacle-free roads to plan to actually clear the given field.

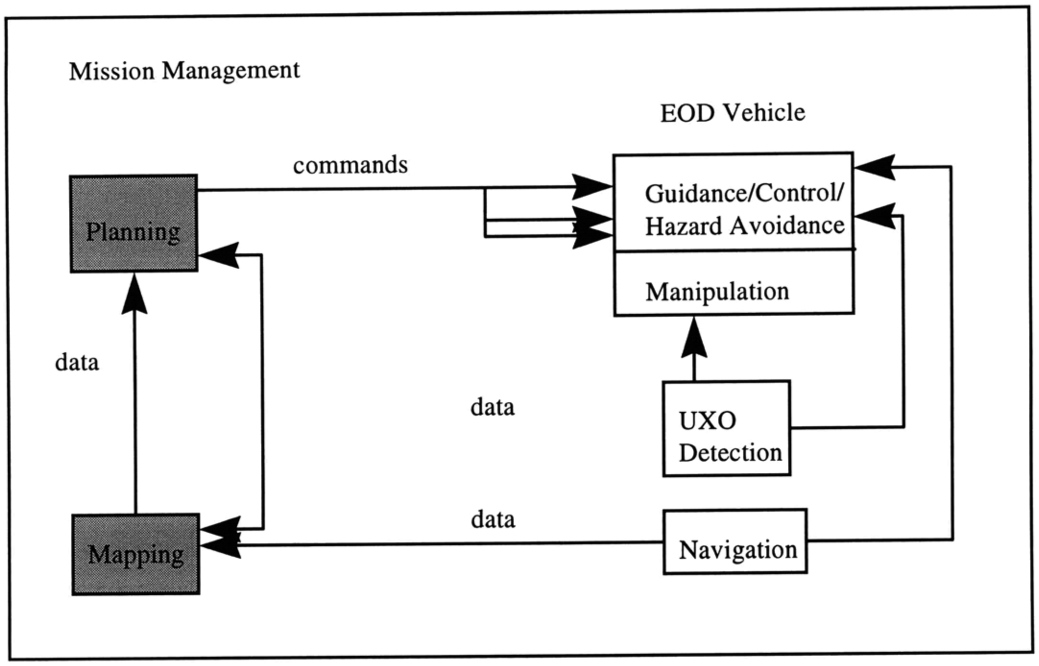


Figure 1-9: Automated Mission Management System

1.5.1 Road-building Phase

At the beginning of the UXO clearing mission, the EOD vehicle are assembled at an ODA located at one corner of an area that contains the potential UXO (Figure 1-10-I). The planning subsystem uses given locations of potential UXO to assign a sector to each of the available vehicles. Then, the manager generates a plan that assigns some UXO to each of the vehicles to visit using the slow speed mode[4]. This mode allows navigation through an unknown environment with obstacle avoidance. When the vehicle reaches the final UXO in its assignment, it picks up that UXO and returns to the ODA in a fast mode retracing its returning path. It then repeats this process for the second-to-last UXO (and so on) using the fast mode along the verified road. However, in the face of uncertainties such as detecting an obstacle in the assigned path or a detonated vehicle, several intermediate steps will occur during this road-building phase. Therefore, the mission management must be designed to deal effectively with the unexpected events such as detecting an unknown obstacle in the assigned road (Figure 1-10-III). If an obstacle interferes with a vehicle's plan so that one or more UXO do not get covered, then the mission manager needs to either re-assign the excluded UXO or re-plan at the moment of discovering an unexpected event during the mission (Figure 1-10-IV) [9]. The mapping subsystem stores the locations of detected obstacles in the UXO area as well as the established obstacle-free roads. The entire research of this thesis is devoted to create an automated solution to approach the road-building phase of the EOD mission. Thus, the remaining chapters will concentrate on the road-building aspect of the EOD mission.

[4] The slow speed mode is defined to be 1 fps and the fast speed mode is defined to be 6 fps.

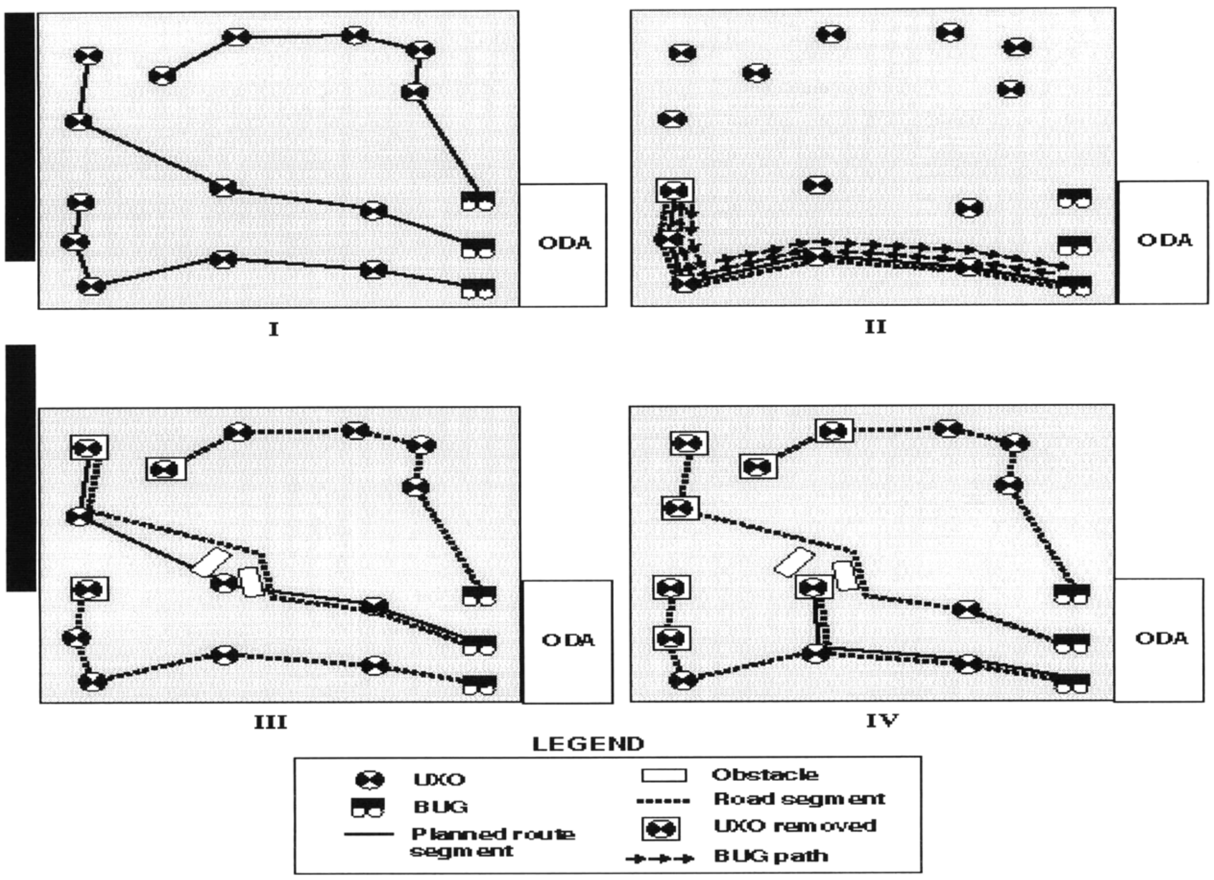


Figure 1-10: Illustration of the Road-Building Approach to UXO Clearance (I) Basic UXO Gatherers (BUGs) and Assigned Routes to UXO (II) The Slow Outward Path of One BUG and Its Fast Return with the Furthest UXO (III) One BUG Can Not Visit an Assigned UXO Location Due to Unforeseen Obstacles (IV) A New Plan Incorporating More Information

1.5.2 UXO-Clearing Phase

The mission manager uses the established obstacle-free roads to assign vehicles to proceed to each UXO at a much faster pace. The planner creates plans that optimize the tasks of clearing UXO on the established roads.

Bryan Koontz, another MIT graduate student, has developed an autonomous mission manager or planner for multiple vehicles system to clear UXO from a network of obstacle-free roadways. The approach to plan the UXO clearance borrows ideas from the fields of Game Theory, Graph Theory, Topology, and concepts in basic robot motion planning. In his thesis, the area to clear is treated as a simply connected and undirected network with the locations of the potential UXO as nodes [9]. The concept of competition among the individual vehicle is used throughout the algorithms. In other words, an individual vehicle competes for free space on the network with other vehicles to find an optimal path for its assigned route. On the other hand, the mission manager works to find an optimal path for each of the vehicles to reach a goal configuration on the network, which is the "road-map" such that no interference occurs between the vehicles [9]. Refer to the thesis "A Multiple Vehicle Mission Planner to Clear Unexploded Ordnance from a Network of Roadways" for more details on the UXO-clearing subject of the EOD mission.

1.6 Road-Building Manager

The road-building problem has two aspects. The first aspect, initial planning, involves creation of the initial pathways. The second aspect, verification, enables the system to re-plan or to adjust the current plan to meet new conditions. The initial planning and plan verifying solutions are coupled to form a *closed-loop* process in the multiple vehicle manager system (Figure 1-11). This closed-loop process allows the system to re-plan if conditions at the time of plan verification diverge from the conditions considered at the time of plan creation [6].

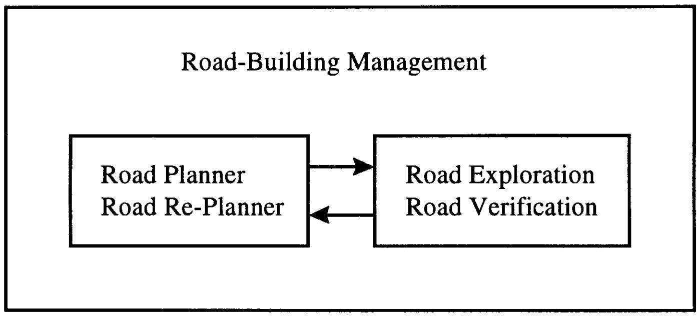


Figure 1-11: The Coupled Closed-Loop Process of the Road-Building Management System

The objective of optimizing the planning problem is to provide feasible plans and to meet the goals in such a way that the maximum planning optimization is acquired while minimum resources such as time, fuel, or money are utilized. In the road-planning case, the objective is to find a path that connects all the dispersed UXO with the minimum distance traveled by each of the vehicles or the minimum total time spent to find the path. The road-building phase initiates with an initial plan or a series of commands and follows the commands until conditions change in the environment. The initial plan is followed closely when possible and provides a basis for creating a new plan if the current plan is no longer feasible [6].

1.6.1 Initial Mission Planner

In the road-building phase of the EOD mission, the locations of UXO or nodes are predefined. Using some well-structured planning algorithms, a plan that connects the given nodes without violating any constraints is created. Although an initially planned route will likely be modified several times during the execution phase, an initial plan will definitely save time and difficulty in covering specific nodes. To enable the planning process, the nodes in the problem are classified as visited UXO, unvisited UXO, disposal location, home (service station), or obstacles. Refer to the Chapter 2 for more detailed descriptions of the planning algorithms to create an initial plan for the EOD road-building mission.

The initial mission planner creates road plans around any predefined obstacles for the vehicles to cover the least distance during the verification phase. This initial road plan should be identical to the verified road plan so long as the environment and constraints don't change during the verification phase. However, the initial plan will almost always change during the verification phase. Thus, some method of managing the problems that come up during the verification phase must be provided.

1.6.2 Verification Mission Manager

During initial planning, the verification of an initial road plan has not begun yet. Verification is actually accomplished by sending the vehicles along the initially planned roads to visit the nodes on the roads. During the verification phase, the vehicles must travel at low speed to allow time for the sensors to detect and avoid obstacles. At the verification stage, the vehicles visit the nodes and verify that the roads are free of obstacles. If the vehicles discover that the roads are not free of obstacles during the verification phase, the manager needs to re-plan. If the manager determines that a vehicle is lost (i.e. detonated), then the manager has an option to request a re-plan using the updated environmental information. Likewise, if a vehicle reports that it has completed its assigned task earlier than the other vehicles, then the manager has an option to request a re-plan.

Separate representation and storage for the planned and verified roads are required during the verification phase. Planned roads are assumed initially to be obstacle-free and the verified roads are actually obstacle-free roads. Thus, if any successful obstacle-avoidance maneuver takes place, then the segments that are added by the maneuver must be stored among the verified roads. On the other hand, if any unsuccessful obstacle-avoidance maneuver takes place then the segments that are blocked by the obstacle must be eliminated. The excluded node must be stored separately to plan to visit later in the road-building phase. Therefore, the verified obstacle-free roads must be stored in a separate storage for later use particularly to carry out UXO-clearing process.

The main objective of the verification mission manger is to execute, verify, and modify (correct) an initial plan that is passed from the initial planner. If an obstacle or problem arises during the verification process that will keep vehicles from further traveling, then the mission manager must provide re-plans. The mission manager usually tries to generate the modified plan that will most likely optimize the use of the remaining resources. However, modified plans may not be the optimal solution since this plan modifying process only optimizes immediate parts of the problem. Refer to the Chapter 3 for more detailed descriptions of the verification mission manager for the EOD road-building mission.

Chapter Two : Initial Planning

Given a list of the locations of UXO, an initial mission planner can generate an initial plan for any number of vehicles that connects all the given nodes in an area. The initial planner provides a connected path for each of the available vehicles. The number of UXO that must be visited are divided among the multiple vehicles to reduce the workload and time to cover a specified region. The main objective of the initial mission planner for road-building phase is to create a feasible plan in the shortest time possible. An initial plan is likely to change during the verification process due to the uncertainty in the terrain (i.e. obstacles). Thus, spending a long time on an initial road plan is inefficient and unnecessary.

2.1 Problem Definition

An optimal initial plan for the road-building phase is a route that connects all the given UXO nodes with the minimum time or distance. This problem basically becomes a node-covering one such as the traveling salesman problem (TSP) or vehicle routing problem (VRP) [5] if no obstacles are present. TSP and VRP are the most studied problems in the operations research and applied mathematics literature. Both the TSP and VRP are NP-complete problems. The TSP requires $(n-1)!$ different orderings of the points to be visited where, n is the number of nodes to be visited. In other words, there are $(n-1)!/2$ different solutions since each tour can be run in either of two directions. Thus, a TSP with 10 nodes yields 1,814,400 possible solutions.

The EOD vehicle routing problem is equivalent to a VRP rather than a TSP. The basic VRP problem involves the routing of a vehicle or vehicles from a central depot to a set of dispersed demand points to minimize the total travel costs or time spent. Similarly, the EOD vehicle routing problem involves the routing of a vehicle or vehicles from any location to pre-defined locations of UXO to minimize the total travel distance or time spent [2].

2.2 Options for the Road Planning Algorithm

An initial plan may be derived from a model of the local delivery or pick-up vehicle routing and scheduling problem. The VRP has both optimal or exact solutions and sub-optimal or estimated solutions. Finding exact solutions to a small (less than 50 nodes) VRP is a difficult and inefficient task. All of the known exact solution algorithms grow exponentially in computation time as nodes are added. Storage requirements can be reduced by constraint windows such as limitations on the vehicles, time windows, and/or quality-of-service constraints. However, the required storage still grows an exponentially for exact solutions. For instance, an exact algorithm based on the idea of dynamic programming is $O(n^2 2^n)$ where n is the number of nodes. This type of algorithm also requires computer memory storage equivalent to $n2^n$. Therefore, exact algorithms become impractical as the number of nodes gets larger (around 70 nodes) [12]. There may never be an efficient algorithm for exact optimal solution to the TSP and VRP.

On the other hand, heuristic algorithms lead to reasonable or good but not necessarily optimal solutions at reasonable computational costs. The computation time for sub-optimal or heuristic solution algorithms grow as low order polynomials. These heuristic algorithms have been experimentally proven to work well [4]. A mixed routing algorithm that is a combination of exact and heuristic methods also exists for the TSP and VRP [12].

Table 2-1 compares heuristic and exact procedures. Heuristic methods are compared to the exact algorithms, specifically dynamic programming and branch-and-bound methods. This Table illustrates that the exact algorithms are far less efficient than a general heuristic algorithm [7].

[5] Refer to the Appendix B for the detailed description and mathematical formulation of the TSP and VRP.

Table 2-1: Comparison of Heuristic and Exact Procedures

Number of Nodes	6	7	8	9	10	16
Number of Vehicles	2	2	2	2	3	2
Constraint[6]	13	3	9	5	7	7
General Heuristic Solution	26.0	119.0	16.8	56.0	17.2	19.8
Total Time[7]	0.048	0.066	0.063	0.075	0.107	0.189
Branch-and-bound Solution	26.0	114.0	16.8	54.0	17.2	19.2
Time[8]	0.5	0.2	0.8	245.0	1.31	80.1
Total Time[7]	74.5	NS/700[9]	NS/3000[9]	NS/700[9]	2000	NS[9]
Dynamic Programming Solution	26.0	114.0	16.8	53.0	17.2	NS[9]
Total Time[7]	74.5	NS/700[9]	NS/3000[9]	NS/700[9]	2000	NS[9]

2.2.1 Optimal Solutions

The largest vehicle routing problem of any complexity that has been solved exactly and completely is only 23 nodes. Four general algorithms exist for the exact TSP solutions. These algorithms are branch-and-bound, integer programming or ordinary linear programming, dynamic programming, and Lagrangian relaxation. All of the four algorithms for the exact TSP solutions can be modified slightly to solve the VRP as well. Both branch-and-bound and integer programming are types of general algorithms for integer linear programming (ILP). ILP is often referred to as the enumerative algorithm and is based on intelligent enumeration of all possible solutions. The enumerative algorithms use sub-tour[10] elimination methods. Thus, branch-and-bound and integer programming overlap to some extent [16]. Refer to the Appendix C for the details of the optimal solution algorithms.

2.2.1.1 Branch-and-Bound

The first technique is a branch-and-bound (Appendix C, C.1). The *branch* in a branch-and-bound refers to successive partitioning of the solution space. The *bound* refers to lower bounds that are used to construct a proof of optimality without exhaustive search. A branch-and-bound method is probably the most common type used for the TSP [15].

2.2.1.2 Integer Programming

The second technique is an integer programming or ordinary linear programming (Appendix C, C.2). This category of ILP is often referred to as the cutting-plane algorithm. The cutting-plane "cuts away" the non-integer parts of the feasible convex hull of the relaxed linear program by using hyper-planes. The hyper-planes represent constraints that have been generated in the linear programming calculations. These constraints are considered in such a way that no feasible integer solutions are ignored [1].

[6] The 6, 8, 10, and 16 node problems used a travel time constraint on each vehicle. The 7 and 9 node problems used a maximum number of nodes per vehicle constraint.
[7] Total CDC6500 CPU time (in seconds) to complete the calculations.
[8] CDC6500 CPU time (in seconds) to find the best solution.
[9] NS = Not solved. NS/700 = Not solved in 700 seconds of CDC6500 CPU time.
[10] A tour is a path around a network that includes each node once and starts and stops on the same node. A sub-tour is a tour that does not pass through all nodes of the network.

2.2.1.3 Dynamic Programming

The third technique is dynamic programming (Appendix C, C.3), which has been used less on the TSP. The main drawback of this method is the computer storage requirements. The combinations grow exponentially with the complexity of the problem. For instance, the core storage requirements to solve a 20 cities TSP exceeds 900,000 words. However, this algorithm is the only exact solution that appeared promising for finding the optimal solution of VRP [1]. Thus, dynamic programming can be extended to solve MVRP (Appendix E, E.1). Dynamic programming is similar to a branch-and-bound method in the sense that it considers all the feasible solutions. However, this method works backward from the last decisions made. The idea is to break down the problem into steps, make decisions at each step, and find a recurrence relation between a step and the previous step.

2.2.1.4 Lagrangian Relaxation.

As the problem size increases the lower bounds in a branch-and-bound algorithm increases due to the several repetitions of the sub-tours. The main constraints in the TSP are the requirements of the vehicle to visit each node exactly once. These constraining equations usually create complications in computing a tight lower bound. Rather than totally ignoring the constraints for simplicity, an alternative way such as a Lagrangian techniques (Appendix C, C.4) can be applied to relax the constraining equations. This technique is basically mathematical manipulations that penalize constraint violations and reward constraint satisfactions. In other words, a penalty term from the constraint violations or the reward term from the constraint satisfactions is added to the cost function [16].

2.2.2 Sub-Optimal Solutions

A general TSP that can be solved optimally or exactly is limited to about 100 nodes or less. The limit on time and computer storage drives the limitation of the nodes in a problem. Furthermore, this limitation is much more severe for the VRP due to the complexity in the computations. VRP is a subset of TSP (i.e. VRP is a constrained TSP). VRP is much more complex due to the vehicle constraints or limits on capacity and operation. The added constraints make the computations much more difficult. However, the same constraints eliminate some of the searches in the problem. Still, problems with a large number of nodes are impractical to solve optimally. However, many heuristic approaches that yield sub-optimal solutions are available to solve large problems [1].

Heuristic procedures are generally categorized into three broad classes: tour construction heuristics, tour improvement heuristics, and composite heuristics. Tour construction approaches always begin the process with a single node and add nodes to build a complete tour. Tour improvement approaches start with the initial tour obtained in the tour construction procedure and try to improve the tour. Lastly, composite approaches are a combination of the first two approaches. These procedures construct an initial tour by using the tour construction approaches and improve the initial tour by using the tour improvement approaches [4]. Refer to the Appendix D for the details of the sub-optimal solution algorithms.

2.2.2.1 Tour Construction Heuristics

Tour construction procedures create a complete tour that visits one node at a time by using the known distance or cost matrix. This procedure adds a node to the partially constructed tour in the VRP only if it will be included in the final solution. Experimental results show that tour construction heuristics generally don't produce good results. Nevertheless, many tour construction procedures are available to construct an initial tour for any given problem with or without constraints. Some of the heuristic methods are the nearest neighbor or greedy algorithm, the Clarke-Wright savings algorithm that usually solves the VRP, and pair insertion procedures [1].

Nearest Neighbor/Greedy Heuristics

A nearest neighbor or a greedy method (Appendix D, D1.1) is the simplest of all heuristic methods available. This method suggests beginning at the depot and always selects the nearest unvisited but feasible node to be the next visit until a complete tour is achieved. The advantage of the method is its simplicity of representation and implementation. A greedy heuristic is much simpler to implement than either a Clarke-Wright savings or a route insertion algorithm. However, this simple method usually doesn't create good tours for either the TSP or the VRP [1].

Clarke-Wright Savings Heuristics

Dantzig and Ramser were credited with developing the savings heuristics to solve the VRP specifically. Clarke and Wright also developed a savings heuristic (Appendix D, D.1.2), but with numerous modifications, to solve the VRP. As stated above, this procedure starts by choosing any node (usually the depot) as the origin of the tour. The procedure also assumes initially that every node in the problem is visited from the selected origin. Then the savings from combining two sub-tours into one node are calculated for all the nodes. The complete tour is constructed in the order of the decreasing savings that are computed without violating any constraint.

Pair Insertion Heuristics

An insertion procedure (Appendix D, D.1.3) basically produces a feasible tour by adding one node at a time to a sub-tour. This procedure applies a specific selection rule that selects an unvisited arbitrary node and suggests where to insert this node in the partial tour. The types of selection of the next node to be included in the final sub-optimal tour are the nearest, farthest, and random relative to any one of the nodes in the partial tour. The sub-optimal solutions discovered by the insertion procedure are within 3-5% of the optimal solution for a single VRP [1].

2.2.2.2 Tour Improvement heuristics

Tour improvement heuristics begin with a provided feasible tour to yield a better and improved tour. Two of the tour improvement heuristics have been applied to the TSP and VRP. The first method is the r-optimal heuristics (Appendix D, D2) that produce tours at least as good as those resulting from the Clarke-Wright savings method. The other method is the λ-optimal heuristics created by Lin and Kernighan. They both use the arc interchange or branch exchange technique to improve the current tour [1].

2.2.2.3 Composite Heuristics

The composite heuristics (Appendix D, D.3) are a combination of the tour construction and improvement heuristics. This procedure generates an initial tour by the tour construction method and improves the tour by the tour improvement method. The composite technique should always produce results at least as good as the tour from tour construction method without the improvement method [1].

2.2.3 Mixed Solutions

The mixed solution method is a combination of the optimal and sub-optimal algorithms (Appendix D, D.4). Mixed routing algorithms exist in many different forms. A particular mixed routing approach that was considered for EOD VRP is a combination of Lagrangian 1-tree and tour improvement heuristics. This algorithm is a combination of the exact and estimated solution method. The particular mixed routing algorithm that has been considered for EOD VRP involves three major steps. The fourth step is optional and may improve the solution further. The algorithm also results in a tour that is guaranteed to be less than 50% longer than the optimal tour. Section 2.3.1 explains the algorithm in detail and provides an example to demonstrate the steps in the algorithm.

2.2.4 Extended Solutions

The various algorithms that are explained previously solve a single VRP or single TSP. However, the EOD vehicle routing problem deals with multiple vehicles. Therefore, the algorithms that solve single VRP or TSP need to be extended to the multiple vehicles case. The multiple vehicles routing problem (MVRP) is harder to solve than the single vehicle routing problem whether an exact or heuristic algorithm is used. Most of the exact algorithms are ineffective to solve the MVRP primarily due to the complexity of the problem. Some of the heuristic algorithms are not efficient to solve the MVRP as well.

Any of the algorithms for the single VRP or TSP can be used for two vehicles without much modification. In fact, the same exact tour found using the previous algorithms can be applied to the two vehicles routing problem. The only difference is that the nodes in the tour are divided and assigned to two vehicles instead of just a single vehicle. The tour can be easily divided into two different paths: one in each direction from the depot or origin node. The two paths would be approximately equal in distance so that the two vehicles can cover roughly equal distance. However, the number of nodes may not be equally distributed along the two paths. Therefore, one vehicle may visit more nodes than the other vehicle. Thus, the two paths along the tour allow the two vehicles to build the routes simultaneously to save the overall time to cover the entire tour.

The algorithms that are mentioned in the previous section (details in Appendix C and D) fail if the number of vehicles increases from two. Thus, alternative algorithms or modified algorithms must be provided to solve the routing problem for more than two vehicles. The most obvious approach to the three or more vehicles routing problem is to use the idea "cluster first, route second". In other words, divide the network into smaller regions for each of the available vehicles, then apply one of the previous single VRP algorithms to each of the smaller regions. This approach may not yield a good solution to each of the vehicles in the problem; however, the entire tour is built by the multiple vehicles in an efficient manner and at a much faster pace than would be possible using a single vehicle.

Most of the tour construction heuristics discussed in the previous section are expensive in terms of computational effort. Furthermore, they performed relatively poorly on multiple VRP. Only a pair insertion and greedy heuristics can be extended for the multiple VRP. The tour improvement heuristics, especially the "3-optimal heuristic", generally produced a good solution to the single VRP and can be extended for the MVRP. Appendix E lists some of the discussed algorithms that are extended to solve MVRP and detailed examples to illustrate the modified algorithmic steps. A mixed routing method can be extended to solve MVRP as well. Sections 2.3.1 and 2.3.5 discuss the details of an extended mixed routing algorithm to solve MVRP. All of the algorithms that are developed to solve MVRP are compared to decide on the road-planning algorithm for EOD multiple vehicle routing problem.

2.3 Selected Road Planning Algorithms

Any exact algorithm to solve large problems is far way too expensive and often doesn't produce feasible solutions. Therefore, heuristic algorithms are the primary candidates to solve EOD VRP. Both the pair selection and greedy heuristics routing algorithms produce better solutions than the randomly assigning the order of nodes to the vehicles. A pair selection technique is effective for small problems[11]. The largest percentage error from the optimal solution is less than 5% for small problems. A pair selection procedure becomes ineffective once time constraints are present in the VRP. However, a greedy heuristic performs the same or slightly better than a pair selection algorithm for much larger problems with time constraints. EOD VRP does have constraints on the time for each of the vehicles. Each vehicle is operated by battery with limited lifetime but rechargeable. Therefore, the vehicles will have constraints on how long it can operate before the power dies. Furthermore, a greedy heuristic is easily understood and implemented into code. A pair selection algorithm is much more complex in implementation than a greedy routing method. However, as the problem size increases neither of the algorithms is superior to the other in terms of performance and computational effort. Refer to the section 2.3.2 for the detail of a greedy algorithm for a single or two EOD vehicles routing problem.

A mixed routing method [12] yields much more optimal routing solution for one or two vehicles than the tour greedy routing heuristic (i.e. tour construction). However, a greedy routing method [1] is much simpler to implement than a mixed routing method. Nevertheless, both a greedy and a mixed routing algorithm are not feasible for multiple vehicle routing problem since the algorithms produce a single tour. Therefore, only two paths are available for the vehicles. These methods can still be used to solve EOD

[11] Heuristic algorithms can usually solve a fairly large problem. However, the most exact algorithms can solve is approximately nine node problem. Thus, the heuristic solutions for more than ten nodes can't be verified to be a reasonable solution since no exact solution can be found.

MVRP using some heuristics to partition the area or distribute the nodes. In other words, a given area can be partitioned into smaller areas and a greedy or a mixed routing method can be applied to yield many single tours. Refer to the section 2.3.1 and 2.3.2 for the detail of the particular greedy and mixed routing methods to solve a single or two EOD vehicles routing problem respectively.

Although many heuristics exist for partitioning a given area or distributing nodes for MVRP, two algorithms have been derived and considered for EOD MVRP. The first method for distributing nodes to multiple vehicles is referred to as the space filling curve algorithm [2]. This algorithm groups the nodes according to their angles or swath width with respect to the initial position (i.e. home node) of the vehicles. The second method for distributing nodes to multiple vehicles is represented as the "Winner" algorithm. This algorithm selects the nodes that are reachable from the home node. Then, the algorithm groups each of the remaining nodes (i.e. nodes other than the selected nodes) into a cluster that contains one of the selected nodes that is the closest in distance. Refer to the section 2.3.3 and 2.3.4 for the detail of the two distributing algorithms to group nodes into clusters.

A greedy or a mixed routing algorithm can be combined with one of the two distributing algorithms to yield a complete planning algorithm for EOD MVRP. Thus, EOD MVRP would consider a combination of greedy or mixed routing method with a distributing algorithm for the initial road plans. Refer to the section 2.3.5 for the details of the combined algorithms to solve EOD MVRP.

2.3.1 Mixed Routing Algorithm

A mixed routing approach that is a combination of the Lagrangian 1-tree algorithm and some tour improvement heuristics is considered for EOD VRP.

The following steps represent the mixed routing algorithm in general terms [12]:

STEP 1: Find the minimum spanning tree that spans the n nodes[12]. Let this minimum spanning tree be T. Skip a node if it's blocked by a predefined obstacle. Once every node has been considered in the tree, connect the skipped node to a node that's already been connected in the tree (only if it becomes a feasible path).

STEP 2: Let n_O of the n nodes of T be odd-degree nodes where n_O is always an even number. Find a minimum-length pairwise matching of these n_O nodes using a matching algorithm. Let the graph that consists of the links contained in the optimal pairwise matching be denoted as M. Create a graph H that consists of the union of M and T.

STEP 3: The graph H is an Eulerian[13] graph since it contains no odd-degree nodes. Draw an Eulerian circuit on H that begins and ends at the starting node of the optimal tour if such starting node has been specified. This Eulerian circuit is the approximate solution to the TSP.

STEP 4 (Optional): Check for nodes of H that are visited more than once in the Eulerian tour and improve the obtained tour by taking advantage of the triangle inequality. For instance, if the possible solution has {A, B, C, D, B, E, ...} where each letter indicates a specific node. Then the tour can be improved by eliminating one of the repeated nodes. In other words, either use the sequence {A, C, D, B, E, ...} or {A, B, C, D, E, ...}.

Example 2.1 (Mixed Routing Algorithm for VRP)

Consider the nine UXO (referred to as the nodes) that the EOD vehicle has to visit starting from the node 1 (the home node) and ending at the same node. The vehicle should travel the shortest route to visit all the nine UXO. Figure 2-1 shows the location of the home node and the nine nodes numbered arbitrarily as 2 through 10. Also, Table 2-2 lists the Euclidean distances for all pairs of nodes [15].

[12] The Lagrangian 1-tree algorithm requires that the minimum spanning tree spans 2 through n nodes instead of all n nodes.

[13] Eulerian circuit is a tour that connects every node in the network at least once and traverses every edge on a graph exactly once (beginning and terminating at the same node). The algorithm seeks for an Eulerian circuit so that a tour is formed to solve a VRP. The algorithm also minimizes the distance in the tour by forming an Eulerian circuit with the least distance.

Table 2-2: Distance Matrix for Home and Nine Nodes

From\To	1	2	3	4	5	6	7	8	9	10
1	0	25	43	57	43	61	29	41	48	71
2	25	0	29	34	43	68	49	66	72	91
3	43	29	0	52	72	96	72	81	89	114
4	57	34	52	0	45	71	71	95	99	108
5	43	43	72	45	0	27	36	65	65	65
6	61	68	96	71	27	0	40	66	62	46
7	29	49	72	71	36	40	0	31	31	43
8	41	66	81	95	65	66	31	0	11	46
9	48	72	89	99	65	62	31	11	0	36
10	71	91	114	108	65	46	43	46	36	0

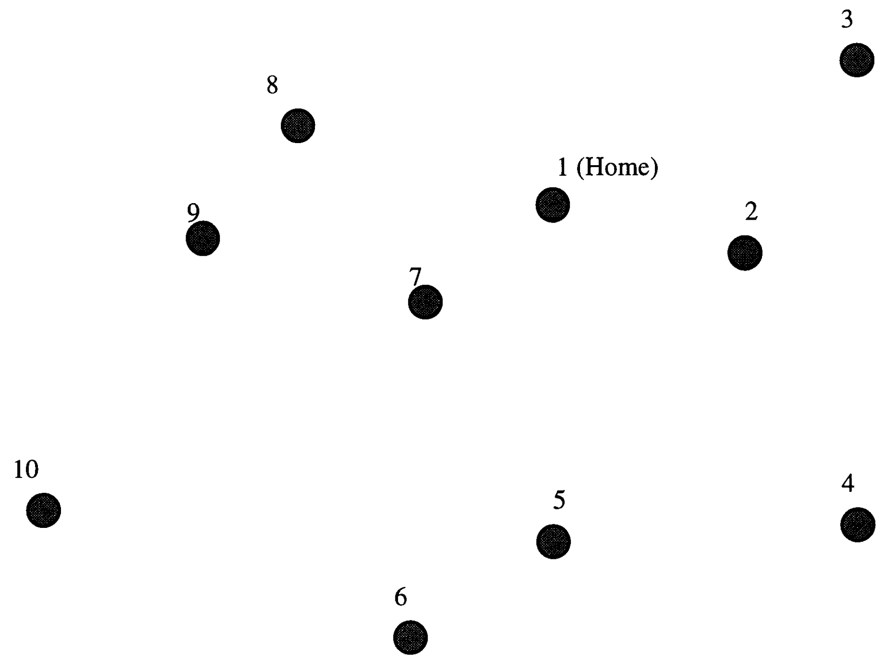


Figure 2-1: Home and Nine Nodes to be Visited

The minimum spanning tree called T for this problem is shown in the Figure 2-2 for this particular problem. From this Figure, six odd-degree nodes are detected on T: 2, 3, 4, 6, 7, and 10. These six nodes must be matched by pairwise method. The optimal pairwise matchings for the six nodes are 10-7, 2-3, and 6-4. The total length of these three links produce 143 units in graph M. Figure 2-3 shows the graph H that is the union of M and T. The graph H represents the Eulerian graph since it is obtained by following the path that begins at the starting node (i.e. home node) and traverses successively the edges of H exactly once until the path has returned to the starting node. The total length of H is 401 units and a tour order can be {1, 2, 3, 2, 4, 6, 5, 7, 10, 9, 8, 7, 1} or {1, 7, 8, 9, 10, 6, 7, 5, 6, 4, 2, 3, 2, 1} . If any of the paths in the minimum spanning tree has a predefined obstacle, the algorithm skips the node that has been obstructed. Once the process of the minimum spanning tree has been applied to every node in the problem, the algorithm tries to connect the obstructed node that has been skipped (if any) to the connected tree. Thus, every node in the problem is considered as long as it's not blocked all around.

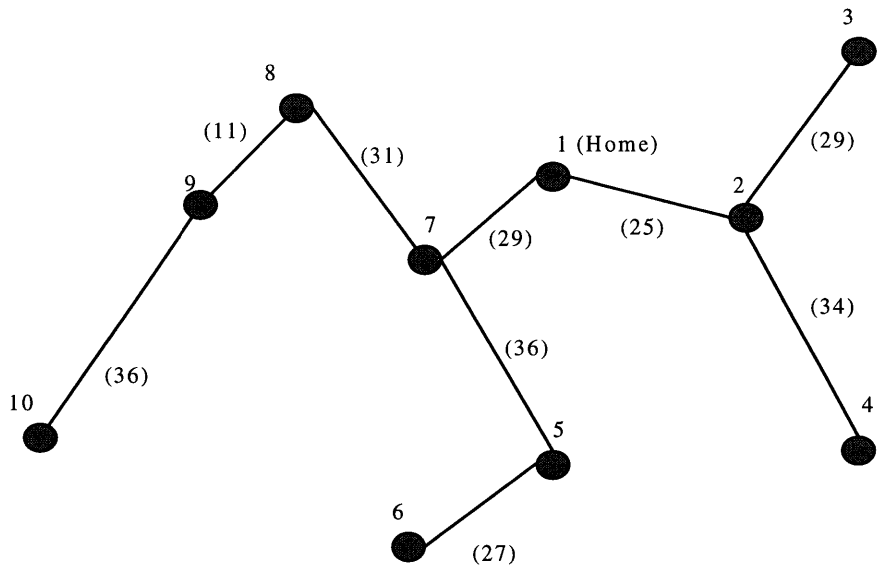


Figure 2-2: Minimum Weight Lagrangian 1-tree

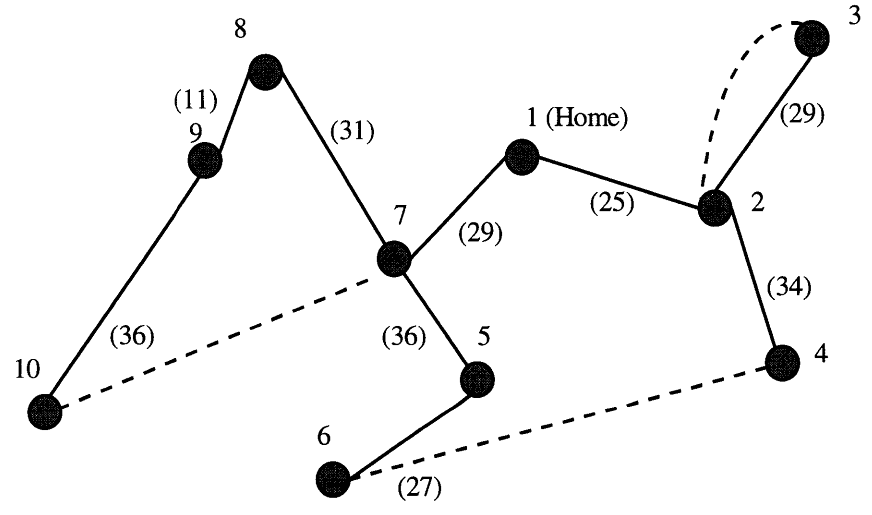


Figure 2-3: Ten-Node Problem after the Matching of Odd-Degree Nodes

Now, *STEP 4* can be applied to improve the solution further. Nodes 2 and 7 are visited twice in the graph H. Thus, the sequences {1, 3, 2, 4, ...} and {1, 2, 3, 4, ...} need to be compared to decide which one should be used. Likewise, the sequences {..., 5, 7, 10, 9, 8, 1} and {..., 5, 10, 9, 8, 7, 1} need to be compared to decide which should be used. The sequence with shorter length is chosen for inclusion in the final tour if it's feasible. For this example, the first two sequences for node 2 have the same length so the sequence {1, 3, 2, 4, ...} is chosen arbitrarily. Among the last two sequences for node 7, the sequence {..., 5, 7, 10, 9, 8, 1} is shorter and feasible. Thus, the final tour after *STEP 4* is {1, 3, 2, 4, 6, 5, 7, 10, 9, 8, 1}. Figure 2-4 displays the final tour solution for this problem with total length of 371 units, about 12% longer than the optimal tour. The true optimal tour is {1, 3, 2, 4, 5, 6, 10, 9, 8, 7, 1} and has the total length of 331 units. An additional improvement can be applied to the final solution by inspection. For instance, the sequence {..., 4, 6, 5, 7, ...} can be improved by switching two nodes. Thus, the improved sequence is {..., 4, 5, 6, 7, ...}. This improvement provides a tour with the length 347 units that is only 5% longer than the true optimal solution.

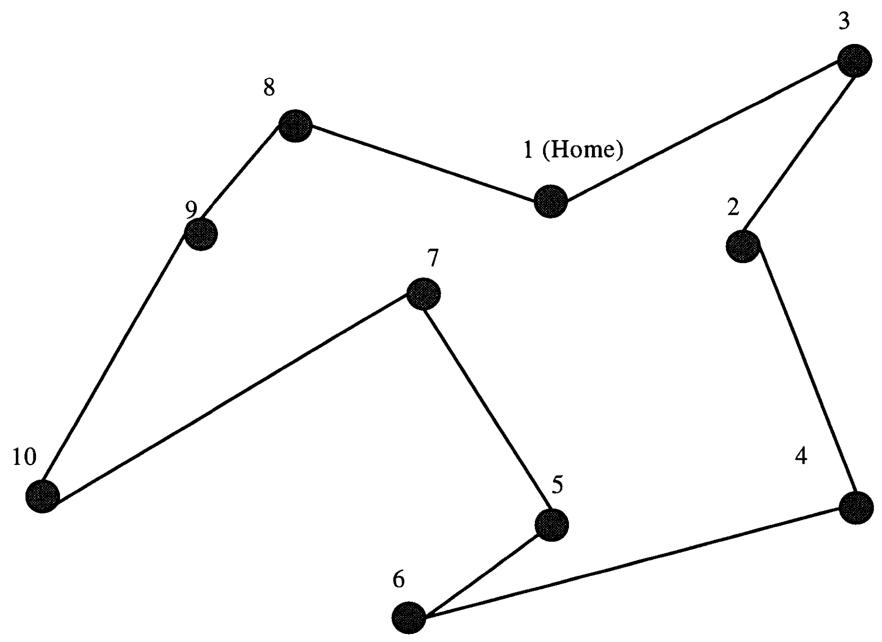


Figure 2-4: The Final Solution to the Mixed Routing Approach

Depending on the structure of the predefined environment, the mixed routing algorithm may exclude some nodes that may be obstructed. If an obstructed node can't be reached from any of the other nodes, then the node will be excluded from the final tour. A new or modified algorithm needs to be derived to take care of the excluded nodes initially. However, this thesis doesn't particularly address the nodes that have been excluded initially due to an obstacle.

Example 2.2 (Extended Mixed Routing Algorithm for VRP)

The proposed mixed routing method can be applied to solve the two-vehicle routing problem without any modification. Consider Example 2.1 but with the two available vehicles. The same final tour in the Figure 2-4 can be used to produce two individual routes for the two vehicles. Figure 2-5 shows the tour for two vehicles. The two routes are directed by arrows in the travel direction. Also, two vehicles are shown with their traveling directions represented by arrows. The last edge between node 5 and 6 is removed from the tour since all of the nodes have been covered at that point. The EOD VRP doesn't require the verification manager to find a path from the last node in the assigned path to the home node (i.e. vehicle's initial location) since the vehicle can travel back to the home node by tracing its covered path. The amount of workload for each vehicle is divided by two different methods. The workload between the two vehicles can be divided according to number of nodes. Thus, exactly half of the nodes will be assigned to each vehicle to cover. This method may be inefficient since the time to cover assigned nodes may differ greatly between the two vehicles. In other words, one vehicle may complete covering its assigned nodes and become idle for a long time until the other vehicle finishes its work. The alternative approach to distributing workload is to assign each vehicle approximately equal travel distance. In this case, the two paths have been optimized in terms of equalizing the travel distance rather than the number of nodes to be covered. The first method is extremely simple in implementation compared to the second method. However, the second method is much more efficient in terms of time spent in exploring roads.

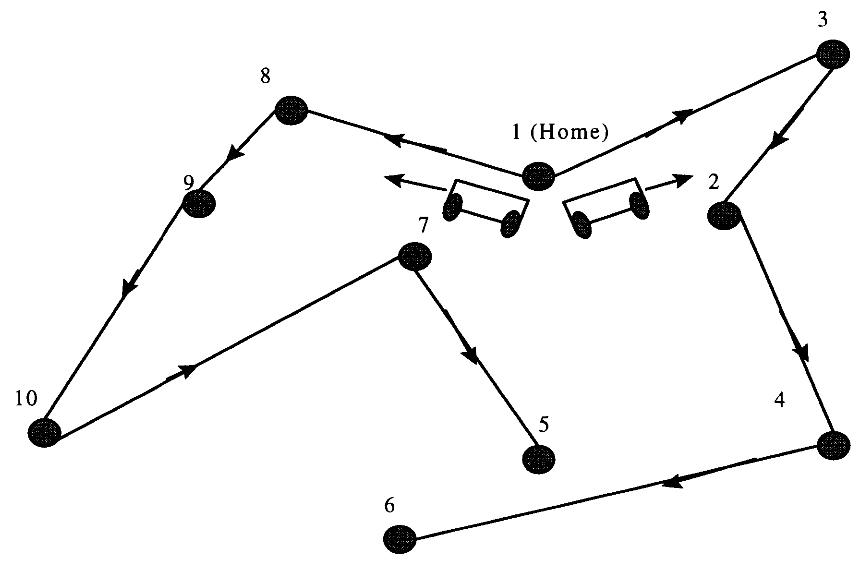


Figure 2-5: Two Routes for Two Vehicles for the Mixed Routing Approach

2.3.2 Greedy Routing Algorithm

The greedy heuristic is one of the tour construction heuristics that is much less expensive computationally. This greedy routing algorithm starts from any node and selects the nearest neighbor node to be included in the final path without considering where the path will go next. Thus, local optimization, rather than global optimization like the minimum spanning tree in the mixed routing method, is considered in a greedy routing algorithm. The process of selecting the nearest node is repeated until a final feasible path that includes all the reachable nodes is created. The particular greedy routing algorithm that has been considered is simple and efficient.

The following are the steps to a nearest neighbor or greedy heuristic routing algorithm in the general terms:

STEP 1: Select an initial node such as the home node to start the vehicle routing problem.

STEP 2: Find the node that is the closest to the last node that has been included in the final path and has not yet been included in the final path. The node can only be included in the final path if the path is feasible. Break ties, if any, arbitrarily.

STEP 3: If this node is found, then go to *STEP 2*. Otherwise, go to *STEP 4*.

STEP 4: If this path is obstructed, then ignore this node and go to *STEP 2*. Otherwise, go to *STEP 5*.

STEP 5: If any ignored node exists, then connect this node to the node that's already been connected to the final path. The ignored node can only be included in the final path if the path is feasible. Otherwise, *stop*, the final path has been found.

Example 2.3 (Greedy Algorithm for VRP)

Consider the same problem as the Example 2.1 (i.e. the nine UXO problem). The EOD vehicle has to visit starting from the node 1 or the home node but doesn't have to come back to the same node as long as all the other nodes have been visited. Also, the vehicle should travel the shortest route to visit all the nine UXO. Figure 2-1 shows the location of the home node or the node 1 and the nine nodes numbered arbitrarily as 2 through 10. Also, Table 2-2 lists the Euclidean distances for all pairs of nodes [15].

The home node or node 1 is selected to be the starting node of the greedy routing algorithm. Then, the greedy routing algorithm is applied to the rest of the nine nodes in the problem. Figure 2-6 shows the final tour of the nine-node problem using the greedy routing algorithm. The final path is {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} and has length of 336 units. This length includes the distance from the last node in the path back to the home node (i.e. from node 10 to node 1). The total length of the final tour is only 5 units off of the optimal solution, which has 331 total units. In fact, the total length of the final tour using the greedy routing algorithm is shorter than that using the mixed routing method.

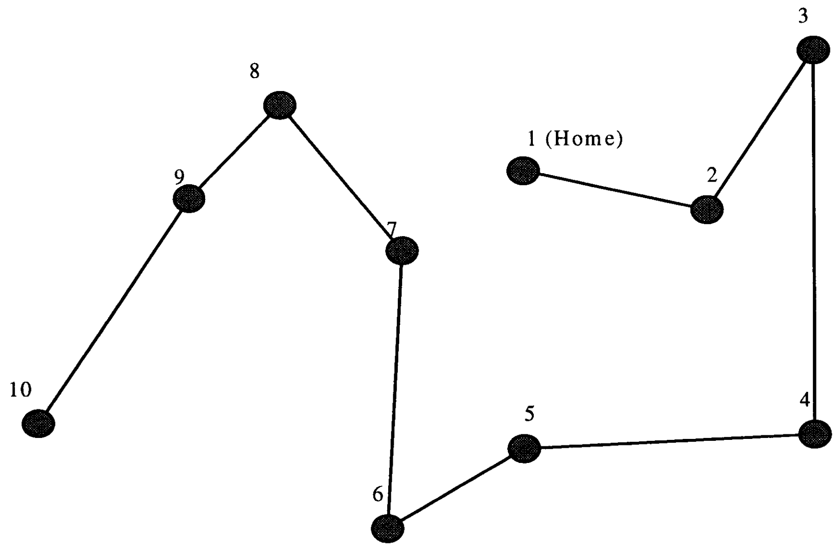


Figure 2-6: The Final Solution to Greedy Routing Algorithm

Depending on the known obstacle environment, the greedy routing algorithm may exclude some nodes that may be obstructed. If an obstructed node can't be reached from any of the other nodes, then the node will be excluded from the final path. A new or modified algorithm needs to be derived to take care of the excluded nodes initially. As stated in the previous section, this thesis doesn't particularly address the nodes that have been excluded initially due to an obstacle.

Although the two routing algorithms must be compared further, the greedy routing algorithm that is considered yields effective solutions to a single VRP. Refer to Chapter 6 for comparisons of the two routing algorithms.

2.3.3 Space Filling Curve Distributing Algorithm

Algorithms that distribute the nodes to multiple vehicles exist in many different forms. The particular distributing algorithms that have been considered and implemented can generate the same number of regions as the number of available vehicles. The space filling curve distributing (SFCD) algorithm calculates the angles of the nodes with respect to the home node (i.e. initial position of the vehicles) using x and y coordinates [2]. Then, the angle differences between any two nodes are computed. Depending on the number of available vehicles, the algorithm selects the first few largest angle differences to be the partitioning points. For instance, if three vehicle are available, then the three largest angle differences are selected. Wherever the selected angle differences lie, the nodes within the region between the two selected angle differences will be grouped into a cluster. This algorithm may create clusters that will have a highly uneven distribution of numbers of nodes. However, the algorithm tries to avoid planning paths through any area that has no UXO node to visit. Furthermore, this algorithm minimizes the complications in assigning the clusters of nodes to multiple vehicles. In other words, the vehicles will cover areas that are separated as much as possible.

The following are the steps to the SFCD algorithm in the general terms:

STEP 1: Calculate the angles for all the nodes with respect to the home node.

STEP 2: Compute the differences in angle between any two nodes.

STEP 3: Select the first few largest angle differences (i.e. equivalent to the number of vehicles).

STEP 4: Partition the area into smaller regions according to the area that falls between the two largest angle differences. If this region is found, then repeat the *STEP 4*. Otherwise, the area has been completely partitioned.

Example 2.4 (SFCD Algorithm MVRP)

Consider the Example 2.1 (i.e. nine UXO-node problem). Figure 2-1 shows the location of the home node or the node 1 and the nine nodes numbered arbitrarily as 2 through 10. Table 2-3 lists the angles of the nine nodes with respect to the home node (node 1). Now, suppose that this problem needs to be solved by three EOD vehicles. Thus, the three largest angle differences must be calculated among the nine nodes. Table 2-3 also lists the angle difference between any two nodes in the problem. The nodes are listed in the order of increasing angle with respect to the home node.

From the Table 2-3, the three largest angle differences are 80, 60, and 55 degrees. Therefore, the three divided regions are shown in the Figure 2-7. This Figure shows that the number of nodes are unevenly distributed to the three vehicles. However, the wide spaces that don't contain nodes are not included in the regions to be covered by the vehicles. Thus, in this sense the algorithm is minimizing the distances that the vehicles must travel in a given problem. Furthermore, the three vehicles are separated by the largest swath widths as possible. Therefore, they will less likely to run into each other during the verification process.

Once the area is divided into smaller regions, the greedy or mixed routing algorithms can be applied to each of the regions. This will create single paths for any number of available vehicles.

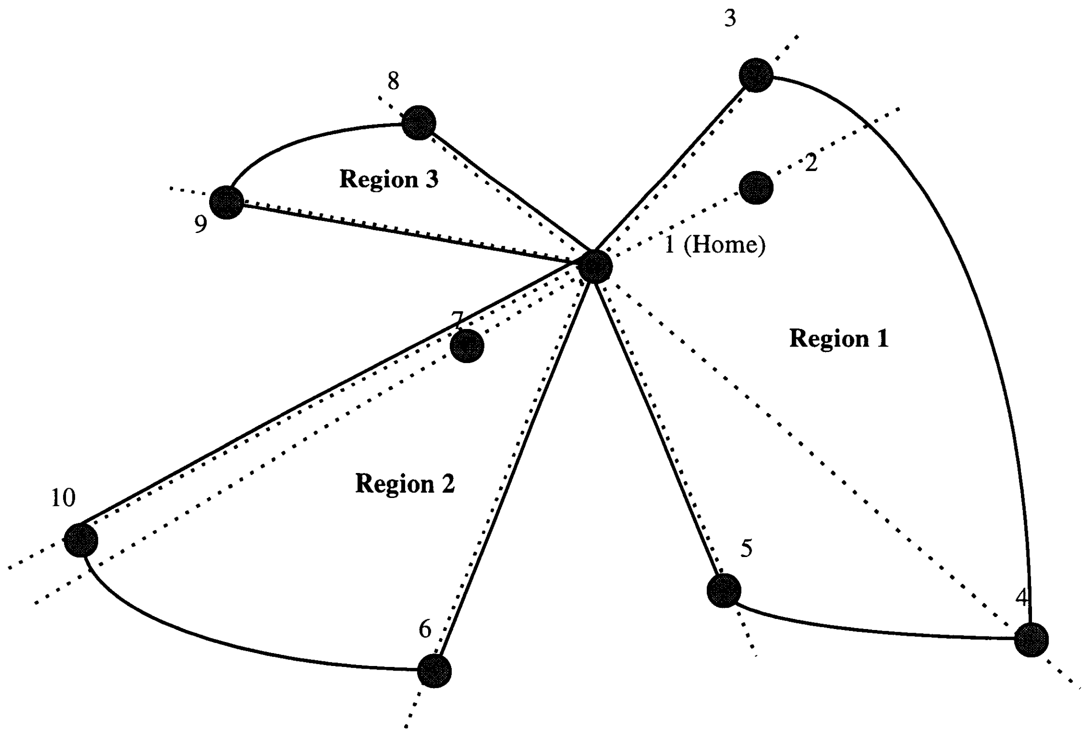


Figure 2-7: Divided Regions using SFCD Algorithm

Table 2-3: Angle Information for Nine Nodes for SFCD Algorithm

Node	Angle wrt Home Node (Degrees)	Angle Difference (Degrees)
3	40	80
2	65	25
4	110	45
5	130	20
6	190	60
7	240	50
10	245	5
9	300	55
8	320	20

2.3.4 Winner Distributing Algorithm

Another distributing algorithm that has been considered and implemented first selects all the possible nodes that can be reached by a straight-line path, given known obstacles, from the home node. From the list of possible nodes that can be reached, the algorithm assigns every node (i.e. the remaining nodes other than the possible nodes) in the problem to a "cluster" containing one of the reachable nodes. Then, the selected nodes in the clusters that contain the most nodes are selected to be the starting points of the paths for the vehicles. To simplify the problem, the number of selected nodes should be equal to the number of vehicles since each vehicle must be assigned to an individual path. The nodes that have been excluded from the selected possible nodes have to be reassigned to the group of selected possible nodes. Again, the excluded node is assigned to the closest possible node that has been selected.

The distributing algorithm that has been considered for node distribution is efficient for VRP with many nodes and known obstacles. This algorithm is not optimal in the cases with few nodes that must be covered and no known obstacles. For instance, if no obstacles are known initially and only a few nodes are present then all of the nodes will become possible initial nodes. Thus, the algorithm will randomly select the first few possible nodes (according to the number of vehicles) without a way to optimize the selection. This distributing algorithm is simple in implementation and fast in computation effort. Refer to the Chapter 6 for comparisons between the two proposed distributing algorithms.

The following are the steps to the "winner" distributing algorithm in the general terms:

STEP 1: List all the nodes that are reachable from the home node using a straight line path. Assign every remaining node in the problem to a cluster that contains a reachable node.

STEP 2: Select the M clusters containing the most nodes, where M is the number of vehicles. Select the reachable node in that cluster as the starting point.

STEP 3: Assign each of the remaining nodes to the cluster that it is closest to.

STEP 4: Apply one of the routing algorithms to each of the final clusters.

Example 2.5 (Winner Distributing Algorithm for MVRP)

Consider the routing problem of seven UXO nodes with eight obstacles and various numbers of available vehicles. Figure 2-8 shows the locations of the seven nodes (UXO) denoted by X_i, home node denoted by a square box, and eight predefined obstacles with their locations and sizes. Table 2-4 lists the coordinates of the various nodes and the type of the nodes. From this Table, the relative distance between any two nodes or obstacles can be derived.

Table 2-4: Coordinate Matrix for the Winner Algorithm Example

Type of node	X-coordinate	Y-coordinate	Size of Node
Home	0	0	-4
UXO	-10	0	-1
UXO	10	0	-1
UXO	10	5	-1
UXO	3	3	-1
UXO	5	5	-1
UXO	-10	-3	-1
UXO	1	1	-1
Obstacle	-5	5	Radius of 6
Obstacle	-1	-8	Radius of 3
Obstacle	-10	-6	Radius of 1
Obstacle	13	3	Radius of 3.5
Obstacle	5	7	Radius of 2
Obstacle	7	5	Radius of 1
Obstacle	-1	3	Radius of 2.5
Obstacle	3.5	2.5	Radius of 0.5

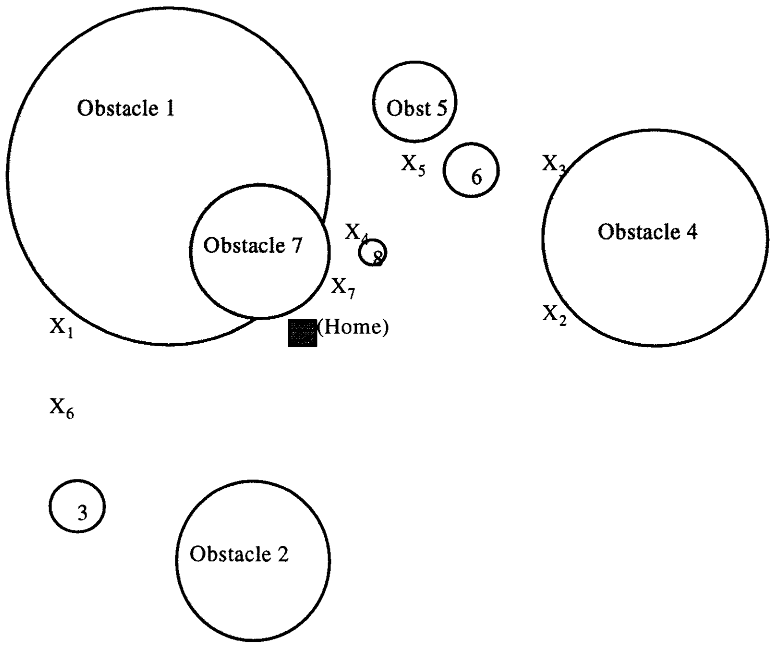


Figure 2-8: Seven UXO and Eight Obstacles Problem for Winner Algorithm

From Figure 2-8, the number of nodes that are reachable from the home node along straight line is six. All of the nodes are reachable from the home node except the first node X_1. Therefore, six different groups of nodes will be formed. Table 2-5 lists the six groups of nodes for the six possible nodes. The number "0" represents the home node.

Depending on the number of paths (i.e. the number of vehicles), several possible nodes can be selected to be the starting point of individual paths. For instance, consider the problem with one available vehicle. For one available vehicle, node 6 is selected to be the starting node since it's assigned to more nodes than other possible nodes. Therefore, excluded nodes 2, 3, 4, 5, and 7 have to be reassigned to the group that contains the selected node 6.

Table 2-5: Group of Nodes for Winner Algorithm

Possible Node	Group of Nodes
2	0, 2
3	0, 3
4	0, 4
5	0, 5
6	0, 1, 6
7	0, 7

Once the area is divided into smaller regions, the greedy or mixed routing algorithms can be applied to each of the regions. This will create many single paths for a number of available vehicles. Suppose the greedy routing algorithm is applied to find the final path for one vehicle. The final path is {0, 7, 4, 5, 2, 6, 1} and is shown in the Figure 2-9. Note that the node 3 is excluded in the final path due to the obstacles. This is one of the cases where the greedy routing algorithm becomes less effective.

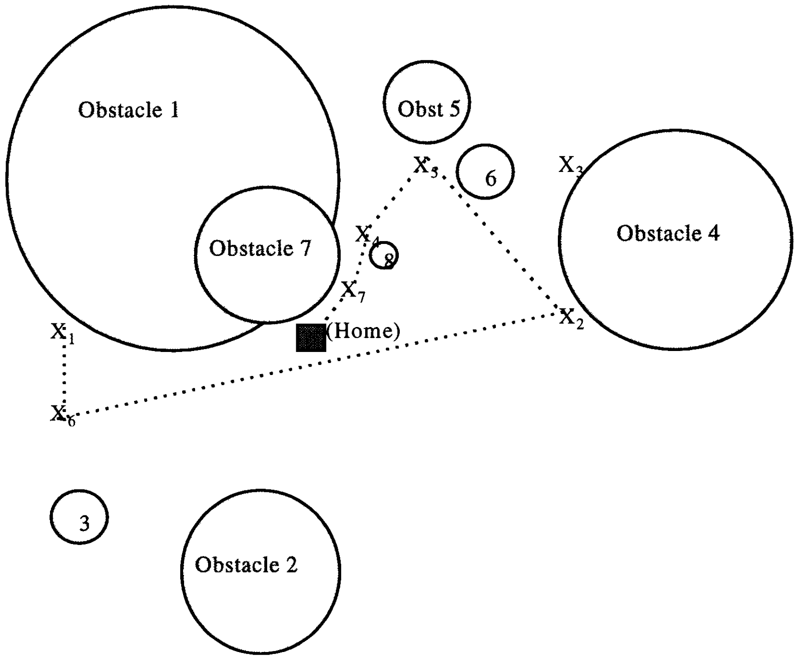


Figure 2-9: Winner Algorithm's Final Path for One Vehicle

The winner algorithm finds any number of regions using similar approach as one vehicle case. The only drawback of the algorithm is that it excludes some nodes that can't be reached from the initial position (i.e. home node). The algorithm also generates only the number of regions that are equivalent to the number of nodes that can be reached from the home node. For instance, Example 2.5 has six nodes that can be reached from the home node. Thus, the algorithm can't generate more than six different regions. If more than six vehicles are available, then some vehicles will be idle for this particular mission. Chapter 6 shows several cases with different number of available vehicles and more detailed illustrations of the algorithm.

Any of the two distributing algorithms can be combined with either of the two routing methods discussed in the previous sections to provide effective solution for MVRP. Thus, once all the initial nodes have been established for each of the group, either of the two routing algorithms can be applied to create several paths for multiple vehicles. Thus, different routes (i.e. equal to the number of vehicles) are generated almost simultaneously.

2.3.5 Combined Algorithm

The routing algorithms (sections 2.3.1 and 2.3.2) and the distributing algorithms (sections 2.3.3 and 2.3.4) can be combined easily. The mixed routing algorithm or the greedy algorithm is used to find a single tour for one or two vehicles. As discussed in the sections 2.3.1 and 2.3.2, the routing methods must be extended to solve the multiple vehicle routing problem. The two proposed distributing algorithms are efficient at grouping the nodes into clusters. But they becomes useless if no routing algorithm exists to actually route each cluster of nodes. Therefore, the distributing algorithms can group nodes into several clusters. Then, the mixed solution or greedy routing method for a single tour can connect the nodes within the clusters to generate several connected paths for multiple vehicles. Chapter 6 illustrates the different combined algorithms for various cases in detail.

The following are the steps to the combined algorithm in the general terms:

STEP 1: Select one of the two proposed distributing algorithms to partition the area into clusters.

STEP 2: Select one of the two considered routing algorithms to generate route or path for a group of nodes.

STEP 3: If any cluster of nodes exists without a planned route, then go to *STEP 2.* Otherwise, *stop,* the final routes have been found for each group of nodes.

Example 2.6 (Combined Algorithm for MVRP)

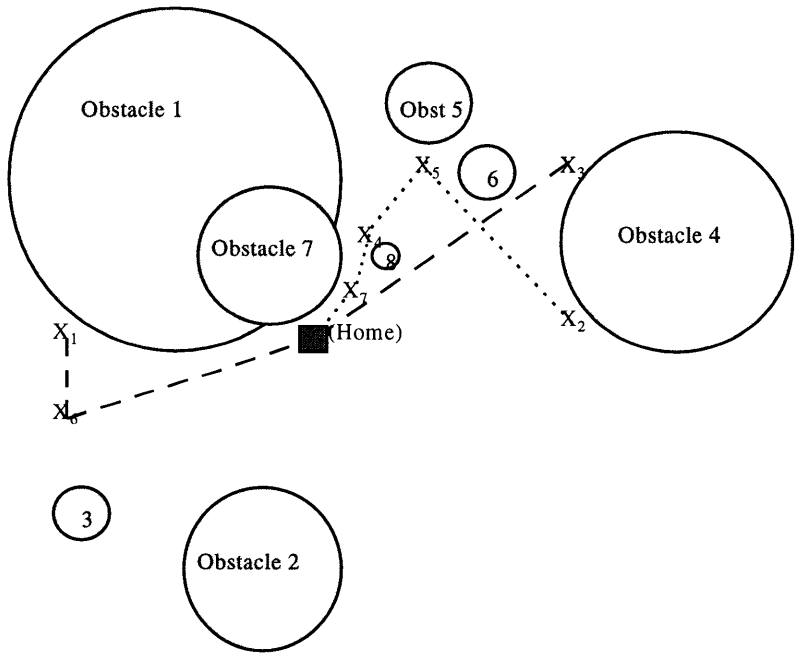


Figure 2-10: Two Minimum Spanning Trees for Combined Algorithm

Consider the same example as Example 2.5 except for 2 vehicles. From the Table 2-5, the two clusters that contain the most nodes are the node 6 and any of the other possible nodes. If the second possible node is selected to be the node 2, then this group will include the nodes 4, 5, and 7. The node 3 can be assigned to either the node 6 group or the node 2 group, since this node can only be reached from the home node. If the node 3 is assigned to the node 6 group, then the following are the groups of the two possible nodes: {0, 1, 3, 6} and {0, 2, 4, 5, 7}. After these two groups of nodes are formed, a routing algorithm can be applied to each of the groups to find two final paths for the two available vehicles. Arbitrarily, the mixed routing method has been applied to each of the two groups of nodes. Figure 2-10 shows the two independent minimum spanning trees. One of the minimum spanning trees is represented by the dashed lines and the other is represented by the dotted line.

Both of the minimum spanning trees have two odd degree nodes. However, the odd degree nodes for the dashed minimum spanning tree shown can't be paired due to the obstacles. Thus, the vehicle will have to travel back and forth on the same path since no other path is available to get to the next destination (Figure 2-11). The odd degree nodes for the dotted minimum spanning tree can be paired together. Figure 2-11 shows the two minimum spanning trees with their paired odd degree nodes.

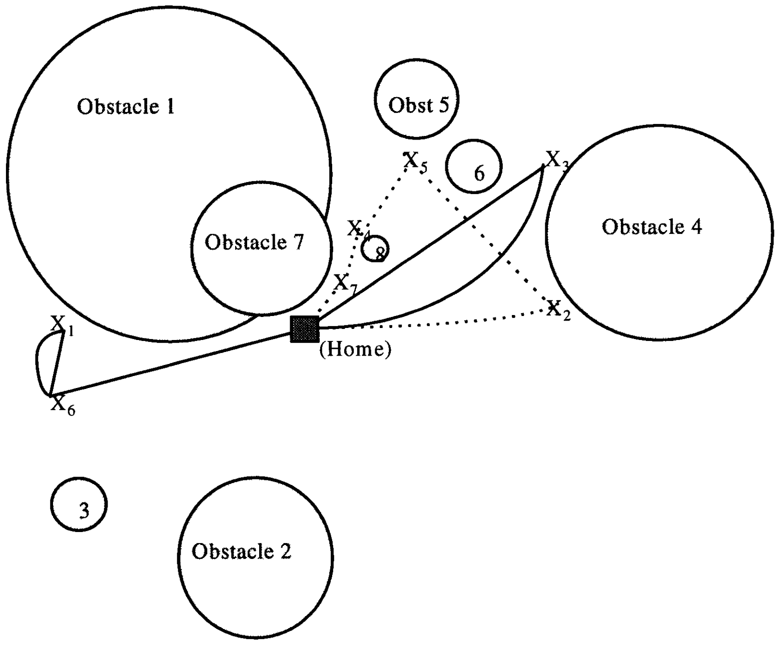


Figure 2-11: Odd Degree Nodes for Combined Algorithm

The final paths for the two groups of nodes are {0, 7, 4, 5, 2} and {0, 6, 1, 6, 0, 3, 0}. The optional step of the mixed routing algorithm doesn't apply to this particular case. The final path {0, 7, 4, 5, 2} doesn't need any more tour improvement. However, the second final path {0, 6, 1, 6, 0, 3, 0} has two repeated nodes. The repeated nodes in the second path can't be modified due to the obstacles. In other words, the route becomes infeasible if any of the repeated nodes is removed from the final path. Therefore, Figure 2-11 shows the two final paths for the two vehicles.

Example 2.7 (Combined Algorithm for MVRP)

Example 2.2 shows a complete tour for one or two vehicles. However, no additional route is available for the third vehicle on the tour produced in Figure 2-4 or 2-5. Thus, the mixed routing algorithm needs to be combined with one of the distributing algorithms to solve for the MVRP with more than two vehicles. Consider Example 2.1 with three available vehicles to illustrate a combined algorithm for MVRP. The given network (Figure 2-1) can be divided into three regions using the SFCD algorithm. Figure 2-7 shows the network being divided into three smaller regions by the largest angle-difference concept. Figure 2-12 shows the minimum spanning trees for each of the three regions.

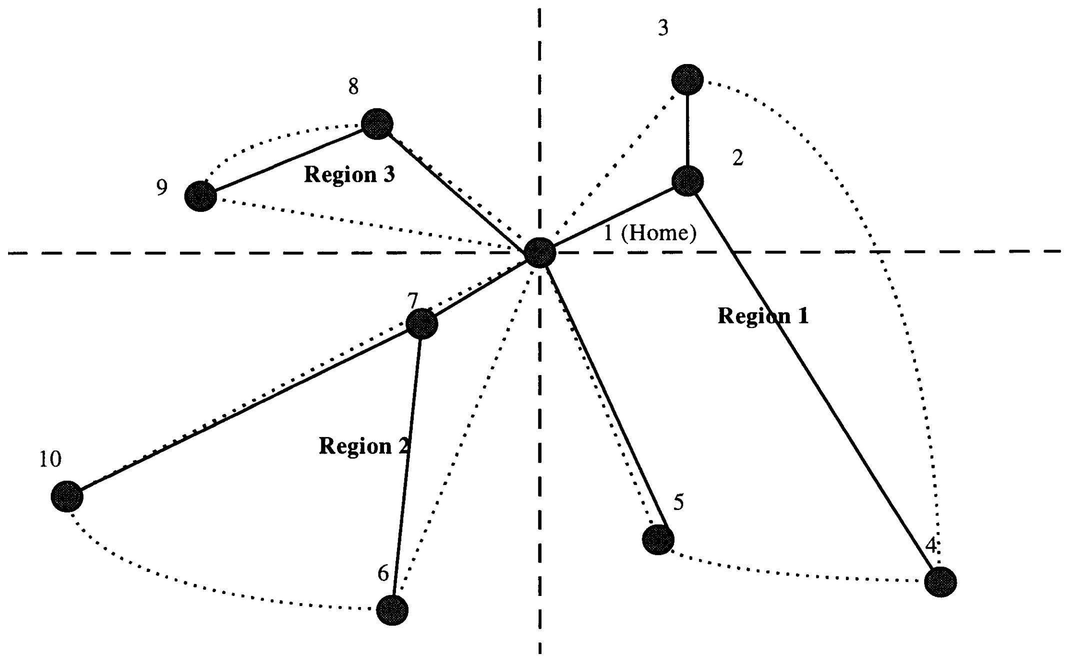


Figure 2-12: Three Minimum Spanning Trees for Combined Algorithm

Once the nodes have been grouped into three clusters, the mixed routing algorithm is applied to yield the three individual routes for the three vehicles. Dividing the network into three equal areas simplifies the implementation. However, the nodes may not be evenly distributed among the areas. Thus, one vehicle may have to travel a longer path than the other vehicles. Figure 2-13 shows the three divided regions and the three individual routes for the three vehicles.

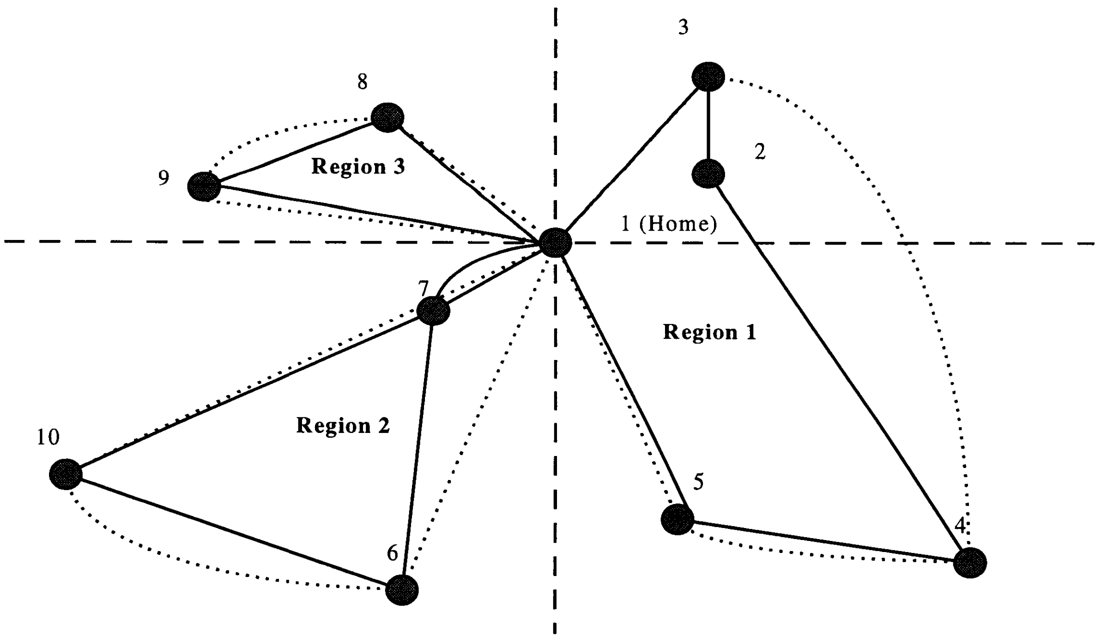


Figure 2-13: The Final Tours for Combined Algorithm

The particular division shown in the Figure 2-7 is not the optimal solution but a way to divide the network into three smaller regions. The three minimum spanning trees can't be formulated simultaneously.

However, the program should run fast enough to create the three initial plans sequentially. Thus, they are theoretically generated almost simultaneously. The tour in the first region is {1, 3, 2, 4, 5, 1}. Note that the original tour before the tour improvement step contained a repeated node (node 2). The tour in the second region is {1, 7, 6, 10, 7, 1} with no improvement applied in this region. The tour in the third region is {1, 8, 9, 1} with no improvement in this region as well.

The combined algorithm that has been discussed above is not the only way to combine the routing and distributing algorithms. The particular combined algorithm in this section is discussed to show how the routing and distributing algorithms can be combined to solve the MVRP. Other combinations of the routing and distributing algorithms can be derived to solve the MVRP. Refer to the Chapter 6 for the comparisons between the two combined algorithms that have been implemented and coded.

2.4 Addressing Additional Challenges and Issues

A real-time planning problem for multiple vehicles involves many different initial planning aspects. In this section, several realistic issues are chosen to be discussed in detail for the EOD road-building mission. Some of the issues that have been considered include too many UXO need to be visited, multiple home nodes (i.e. multiple origins), and other important issues that may come up during initial phase of the mission. The chosen issues will be defined in detail and some possible approaches will be proposed.

2.4.1 Too Many UXO (Unexploded Ordnance)

The main difficulty that needs to be resolved in the initial road plan is the complexity of the problem. In other words, the number of computational requirements and verifications causes the storage requirements to grow rapidly as the number of nodes or UXO increases. For instance, a planning problem with about 100 nodes or more is indeed realistic but impractical to create an initial road plan due to a substantial amount of computational effort. The real-time mission manager usually spends on the order of several minutes to create an initial plan. However, if more than 100 nodes are considered in the initial plan then the manager will reach its computational limit. In order to deal with this problem, a number of nodes denoted by N that each vehicle can handle needs to be specified or derived from its constraints. The number N is used to partition the large problem into several N-node problems. Thus, N or a multiple of N nodes are considered as the complete planning problem at a time. Example 2.8 illustrates a planning problem with too many nodes.

Example 2.8 (Too many nodes)

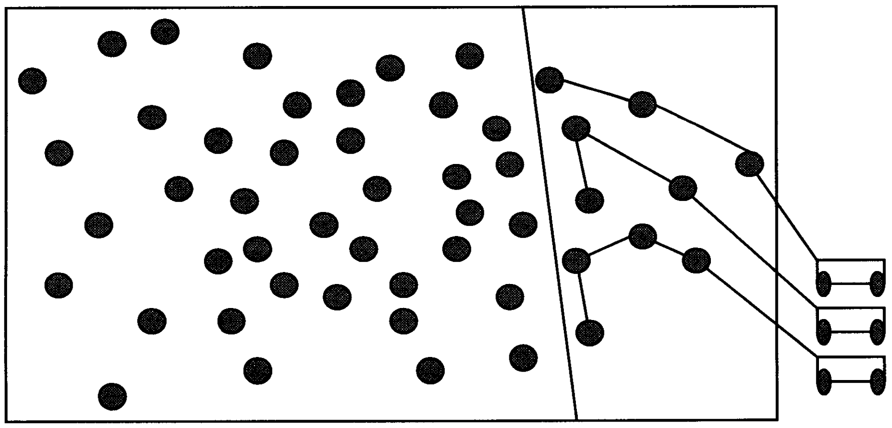


Figure 2-14: First Area Route Plan for Too Many UXO Problem

Consider a planning problem with about 50 nodes to be visited and three available vehicles. The network is partitioned among the three vehicles using a heuristic method. The first divided area contains about 10 nodes is considered for the route planning. Thus, $N = 10$ and the nodes in the first area are the first 10 closest nodes to the initial position of the vehicles. Figure 2-14 depicts the 50-node planning problem. The first area is treated as the entire planning problem first and other regions in the problem are ignored until the nodes in the first area are completely visited. Then the plans to cover the second area are created and prosecuted and so forth. Figure 2-14 also shows an initially planned route using one of the combined algorithms for each of the three vehicles in the first area. The last nodes that the vehicles visit in the first area become the initial positions for the second area of interest.

The second area of the 50-node problem is then chosen next to be visited by the vehicles. The second area will contain the next 10 closest nodes from the initial location of the vehicles (i.e. home node). The 10 nodes that have been covered in the first area are eliminated from the network if all of the nodes are visited successfully. If any of the first 10 nodes has not been visited due to obstacles, that particular node is saved for the future visit. Uncovered nodes or UXO are mission management issues (refer to the Chapter 3 for this example). Figure 2-15 shows the second area route plans for the next 10 closest nodes in the network.

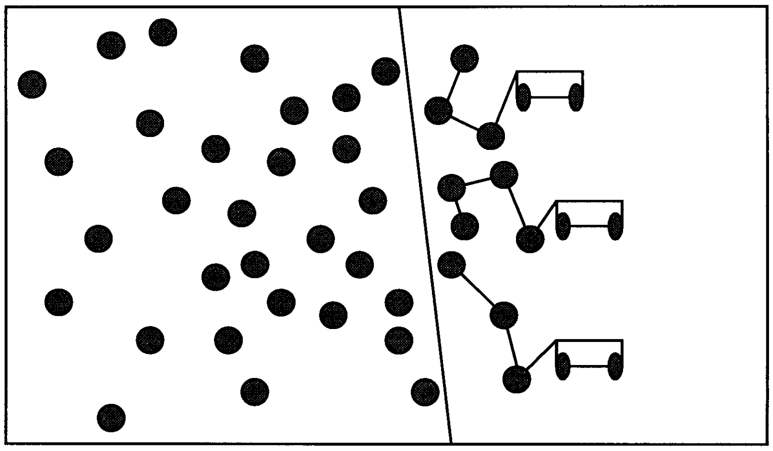


Figure 2-15: Second Area Routes for Too Many UXO Problem

The third, fourth, and fifth areas are planned in a similar way to that described above. The verified road-building plan for the entire mission is then constructed as a composite of the five individual plans. Figure 2-16 shows the composite plan for the entire network. Piecewise route planning for a large problem may not lead to the optimal solution, but may result in a reasonable and feasible approximate solution.

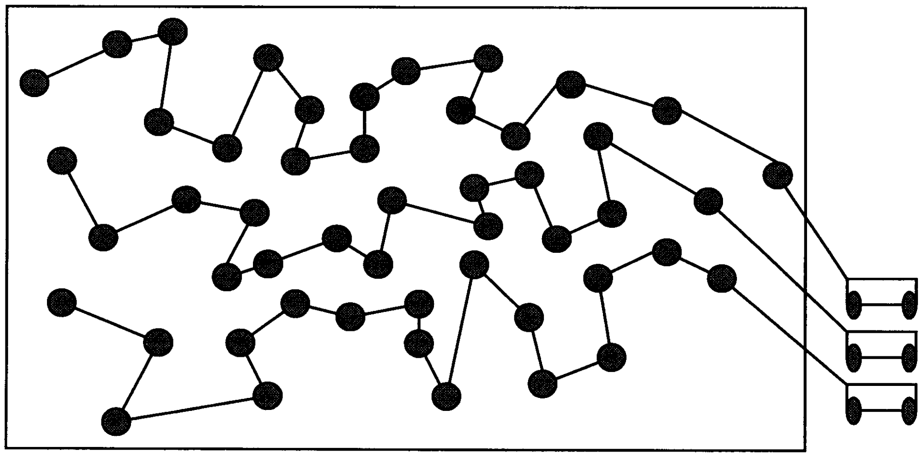


Figure 2-16: Composite Plan for Too Many UXO Problem

Another approach can be applied to the problem with too many UXO or nodes. The entire network in Example 2.8 can be divided into three regions that contain an approximately equal number of nodes. Then, each vehicle can be assigned to each of the three smaller regions to find a feasible route that connects all of the nodes within the regions. Thus, the concept of "cluster first, route second" is applied to the large routing problem as well. Therefore, once the network is divided into regions and vehicles are assigned to each of the regions then the problem reduces down to three single VRPs. If the individually reduced problem is still too large to solve then the problem should be further reduced. Thus, the network is divided into more than three regions to simplify the large problem. Many other heuristic approaches may exist for a large vehicle routing problem. The particular solution discussed here may not be the optimal or the minimum-road-distance solution. However, this heuristic solution does provide an answer and may even be a good solution to the large vehicle routing problem.

2.4.2 Multiple Home Nodes (Origins)

Another important initial planner issue is considering multiple home nodes or origins for the EOD mission. The existence of multiple home nodes introduces the additional requirement of assigning UXO or nodes to specific home nodes. Thus, the new requirement further increases the computational complexity of the problem. However, the existence of multiple home nodes for the EOD mission gives new "degrees of freedom" in the routing and clearing problem. For instance, the distances the vehicles have to explore for the obstacle-free routes may be reduced by a great amount due to the flexibility of the home locations. Furthermore, the vehicles are not restricted to start their exploration from a single home node. The usual approach to VRP with multiple home nodes has been using the concept "cluster first, route second". The nodes are assigned to a home node, then either single or multiple VRPs are solved for each home node.

One of the heuristic approaches in assigning the nodes to home nodes is listed as the following steps [13].

STEP 1: For each node *i*, compute the quantity in the Equation (2.1) where $d'(i)$ and $d''(i)$ are the distances from *i* to the nearest and second nearest of its home node respectively.

$$r(i) = \frac{d'(i)}{d''(i)} \tag{2.1}$$

STEP 2: Specify a threshold value δ such that $0 < \delta < 1$ and compare to each calculated $r(i)$. If $r(i)$ is less than or equal to δ, then the node *i* is immediately assigned to its nearest home node. However, if $r(i)$ is greater than δ, then the node *i* is reserved for more careful consideration.

STEP 3: After all nodes *i* such that $r(i)$ is less than or equal to δ have been assigned to a home node, nodes with $r(i)$ greater than δ are processed again. If two nodes *j* and *k* have already been assigned to a given home node O_s, then inserting node *i* between the two nodes *j* and *k* on a route originating and terminating at O_s increases the length of that route by the Equation (2.2). Therefore, node *i* is assigned to the home node associated with the minimum of the quantities calculated by the Equation (2.2) for all pairs of nodes (*j*, *k*) already assigned to a home node.

$$d_{jk}(i) = d(j,i) + d(i,k) - d(j,k) \tag{2.2}$$

STEP 4: After all the nodes have been assigned to a home node in the manner described in the *STEP 1* through *STEP 3*, then a single-home node VRP algorithm can be applied to plan initial routes.

Example 2.9 (Multiple Home Nodes)

Consider a nine-node problem with three available home nodes to be solved using either two or three available vehicles. Figure 2-17 shows the nine nodes and three available home nodes. Since the multiple home node VRP will eventually be reduced down to single VRPs for each home node, a single vehicle can easily be assigned to each home node. Thus, first consider the case with three vehicles for the three available home nodes. The relative distances for each of the node to other nodes are same as the distance matrix in the Table 2-2. Home node 1 represents the node 1 in the distance matrix. However, the two extra home nodes 2 and 3 are added to the problem. To simplify the problem, home nodes 2 and 3 are positioned approximately same as the nodes 8 and 10 respectively. Furthermore, each of the three available vehicles will be positioned at one of the home nodes (Figure 2-17).

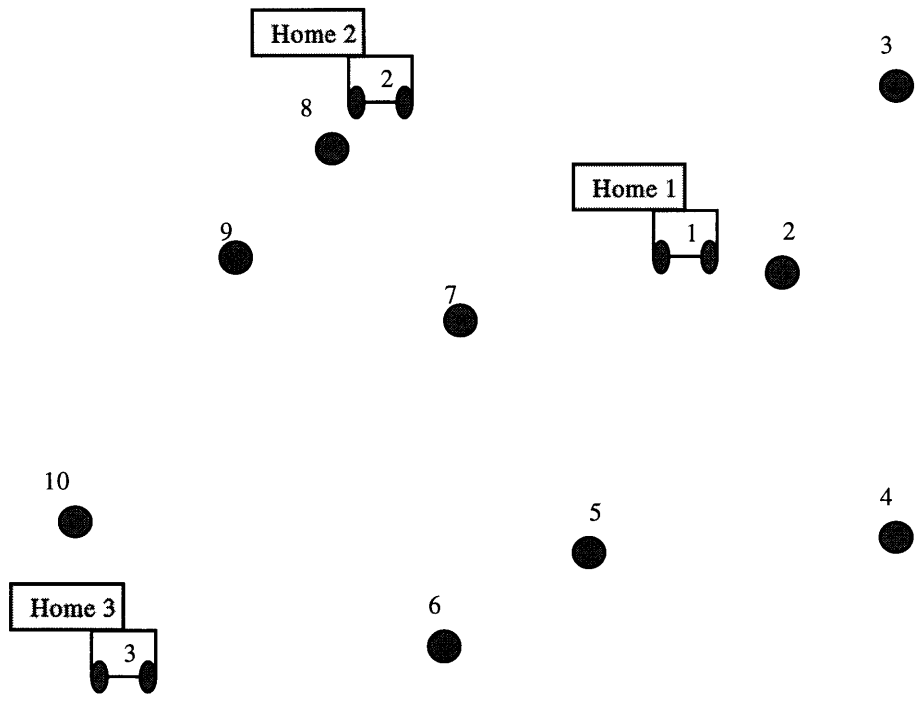


Figure 2-17: Nine Nodes with Three Home Nodes Problem

First, the value $r(i)$ needs to be calculated for all of the nodes in the problem to assign each node to a home node. This value can be calculated by using the Equation (2.1). Table 2-6 lists the values for all of the nodes 2 through 10. Note that $r(i)$ values range between 0 and 1 for all of the nodes. Therefore, if the threshold value is assumed to be 1, then all of the nodes can be assigned to a home node immediately without going through *STEP 2* through *STEP 4*. Thus, the threshold value δ is assumed to be 1 for the home node assignments made in the Table 2-6.

Table 2-6: Quantity $r(i)$ for Multiple Home Node Problem

Node	r(i) Value	Assigned Home
2	0.38	1
3	0.53	1
4	0.60	1
5	0.66	1
6	0.75	3
7	0.94	1
8	0.00	2
9	0.31	2
10	0.00	3

Figure 2-18 shows the nodes with their home node assignments and the three routes for each of the available vehicles. The distances that each vehicle covers in their routes are not evenly distributed. The first and third vehicles are assigned approximately equal distance, but the second vehicle is assigned twice as much distance as the other vehicles due to too many assigned nodes to the first home node. Thus, assigning each vehicle to cover the nodes associated with its respective home node may not be effective in some cases such as this example. However, this is the starting point of the routing problem with multiple available home nodes (i.e. origins). Some heuristics can be applied to the multiple home node algorithm to improve the routes for the three available vehicles.

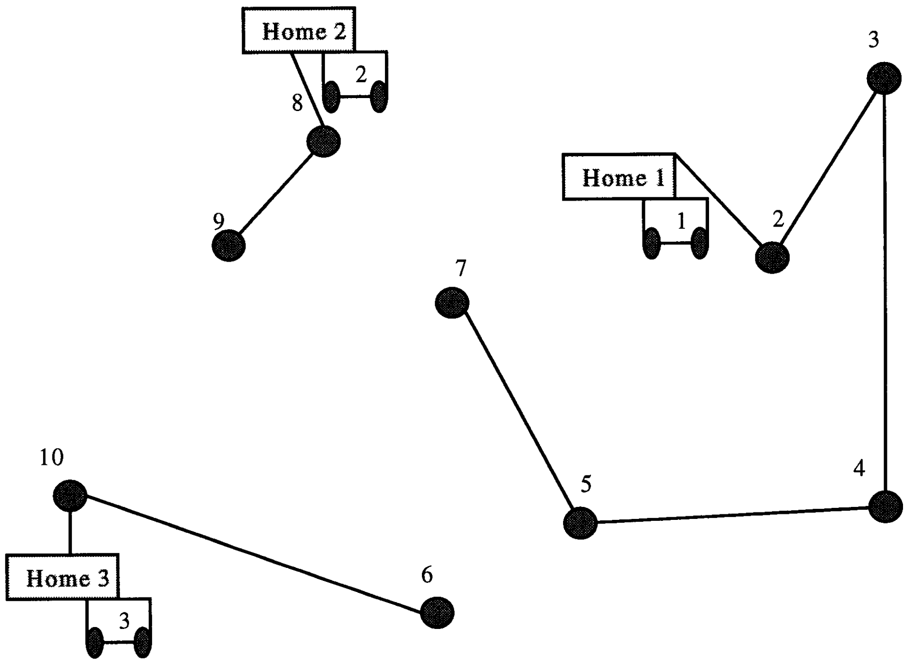


Figure 2-18: Multiple Home Node Assigned to Nodes for $\delta = 1$

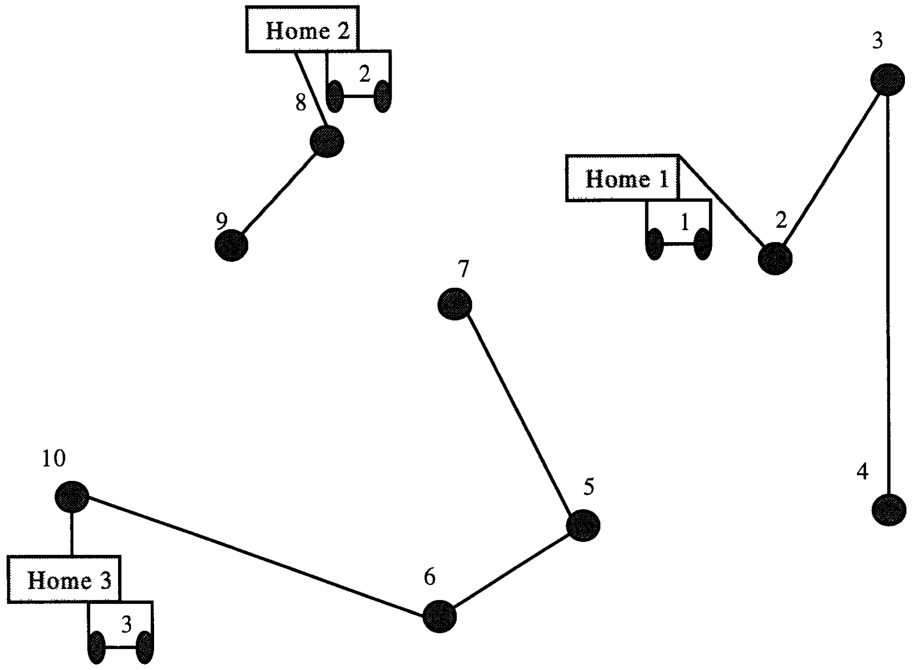


Figure 2-19: Multiple Home Node Assigned to Nodes for $\delta = 0.5$

The threshold value is defined from 0 to 1. If this value is chosen to be other than 1, then the home node assignments will change. For instance, consider the threshold value of 0.5 for the problem presented in the Example 2.9. Now, only 4 of the 9 nodes will get assigned to a home node immediately since the other 5 have the *r*(*i*) value greater than the threshold value of 0.5. Nodes 3, 4, 5, 6, and 7 have to be assigned to a home node by more careful analysis. *STEP 3* needs to be carried out to complete the home node assignments of the five remaining nodes. Figure 2.19 shows the new assignment of a home node for the five nodes that didn't meet the threshold requirement. Two of the three routes are approximately equal in the distance, but the third one is still longer than the other two routes. However, the three routes are much more evenly distributed than the solution with the threshold value of 1. Therefore, a much more effective solution can result with a smaller threshold value for the multiple home node problem.

For three available vehicles, the route assignments are easy for the three home nodes problem. However, if only two vehicles are available, then the routes have to be distributed differently than simply assigning a route associated with each home node. Thus, consider the same problem but with two available vehicles to be assigned to the three home nodes. Figure 2-20 shows the two planned routes for two vehicles with three home nodes. The simple strategy is to ignore one of the home nodes and treat it as one of the UXO node that must be visited. In other words, assign the nodes including the ignored home node to the two considered home nodes. Then, the two vehicles can start from the two considered home nodes. The planned routes for the two vehicles shown in the Figure 2-20 are not the optimal solution but rather a simply found solution that is reasonable and feasible for the EOD mission. Other heuristics can be applied to divide the workload of the clusters of nodes between the two vehicles.

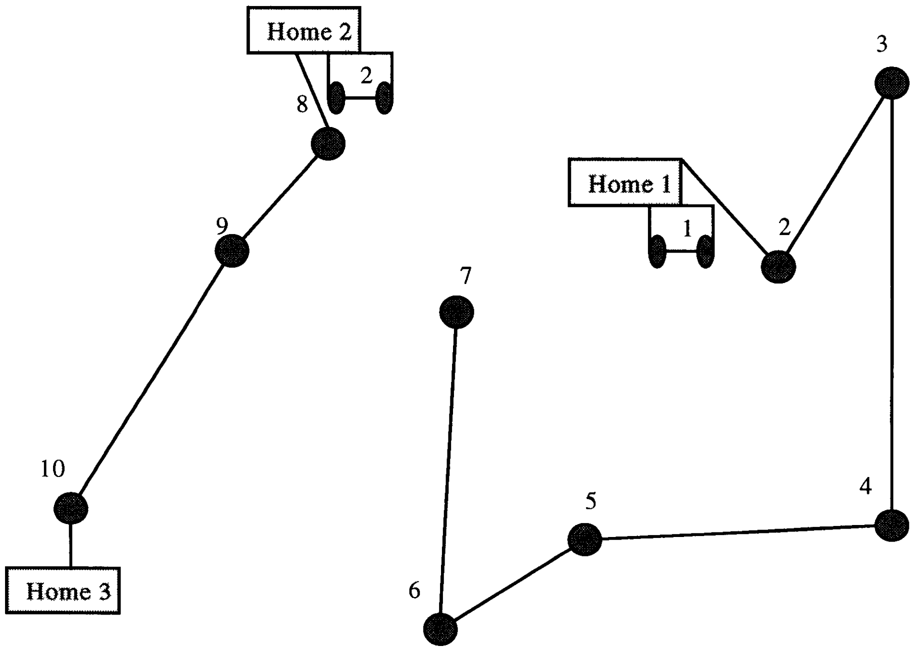


Figure 2-20: Multiple Home Node Assigned to Two Vehicles with $\delta = 1$

Chapter Three : Plan Verification and Re-Planning

Chapter 2 explained some of the possible algorithms to create an initial road plan for the EOD MVRP. As discussed in Chapter 1, the verification phase follows the initial road plan phase. The initially planned roads must be verified to be clear of obstacles before the route plan can be passed to the UXO clearing mission planner or manager. In a real-time routing problem, uncertainty almost always exists in the mission. Therefore, a verification manager must be provided to solve problems that arise in real time. Just as the initial mission planner exists in many different forms, many different methods exist for verification. Depending on the type of the mission and its requirements, the verification manager can be selected appropriately from many options.

3.1 Problem Definition

The EOD road-building problem assumes that the locations of UXO within the area of operation are known *a priori*, but that the terrain conditions are not. Therefore, this problem is deterministic and static in terms of the locations of the UXO or nodes. But the problem becomes dynamic and stochastic in terms of the conditions of the terrain or environment, such as probability of loss or added vehicles or unknown obstacles present in the area of operation. The manager must be able to re-optimize during the verification process to complete the mission [2].

3.2 Dynamic Event-Driven Management

The mission manager issues waypoint following commands to the multiple vehicles using the initial road plan from the initial planner. As the vehicles visit the assigned UXO nodes, the vehicles report to the manager that the road segments leading to an UXO node is obstacle-free. The manager stores these verified road segments in the verified road plan for the future. The manager also responds to the various events the vehicles run into during the verification process.

The mission manager for the EOD road-building phase will essentially manage two main events. It intervenes when a vehicle runs into an unknown obstacle that is impassable or when a vehicle completes its assigned task (i.e. usually waypoint following) earlier than the other vehicles. If a vehicle runs into an obstacle, it automatically initiates an obstacle-avoidance maneuver. The vehicle backs out a couple of feet from the point where the vehicle detected an obstacle. Depending on which side of the vehicle is hit by an obstacle, the vehicle turns towards a planned goal point. In other words, if the vehicle's right side of the bumper is hit by an obstacle, then the vehicle turns left towards a defined goal point. The vehicle may or may not be able to go around the detected obstacle using this maneuver (i.e. depends on the size of the obstacle). If the obstacle-avoidance maneuver succeeds, the manager stores the verified road segments, including the segments from the maneuver, in the verified road plan. Example 3.1 shows a successful obstacle-avoidance maneuver.

Example 3.1 (Obstacle avoidance)

Consider a problem with nine nodes and two vehicles. Figure 2-1 displays the nine nodes and home node where the three vehicles start the mission. Also, Table 2-2 lists the relative distances of the nodes to each other. Assume that the initial mission planner generated the initial road plan for the two vehicles as shown in the Figure 2-5. The mission manager commands the two vehicles to complete the assigned waypoints. Suppose one of the vehicles runs into an obstacle in the assigned path (Figure 3-1). The vehicle will automatically initiate obstacle avoidance.

Figure 3-2 shows the points used to go around the obstacle to the originally assigned node. The two segments that connect the three points which have been added due to the obstacle avoidance process will be added to the list of the verified segments. Therefore, the final tour is {1, 8, 9, 9.1, 9.2, 9.3, 10, 7, 5} instead of {1, 8, 9, 10, 7, 5}.

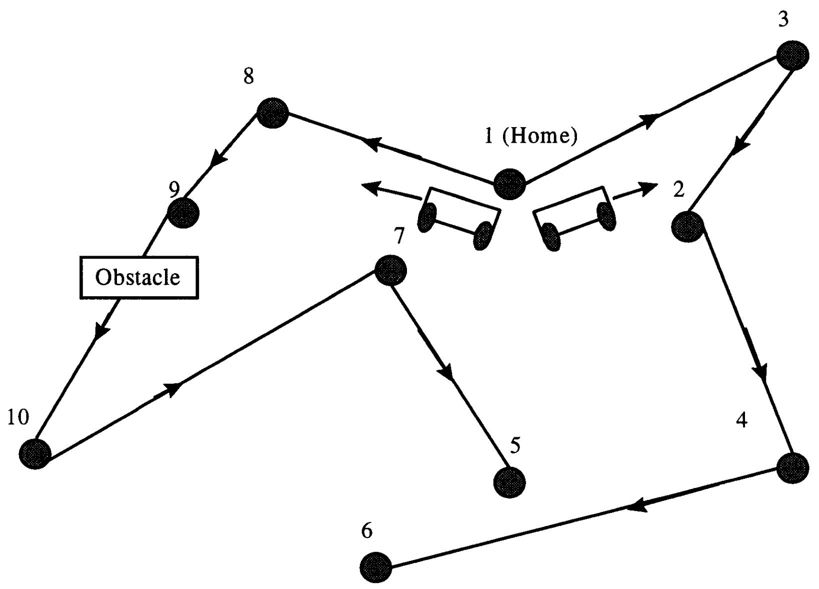


Figure 3-1: Obstructed Vehicle for Nine Nodes Problem

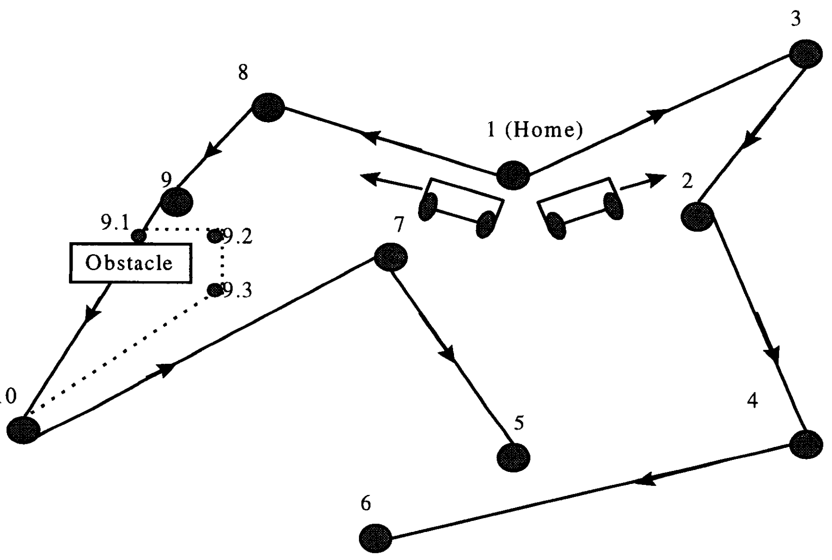


Figure 3-2: Obstacle Avoidance for Nine Nodes Problem

If the obstacle-avoidance maneuver fails, the verification manager commands the obstructed vehicle to by-pass the obstructed node. The manager stores the skipped node into an excluded-node file to visit later. The manager calls a re-planner to provide a plan to cover the excluded nodes. Refer to the section 3.3.1 for more detail on the re-planner to cover the excluded nodes. Example 3.2 shows a vehicle by-passing an obstructed UXO node.

Example 3.2 (By-Pass Obstructed Node)

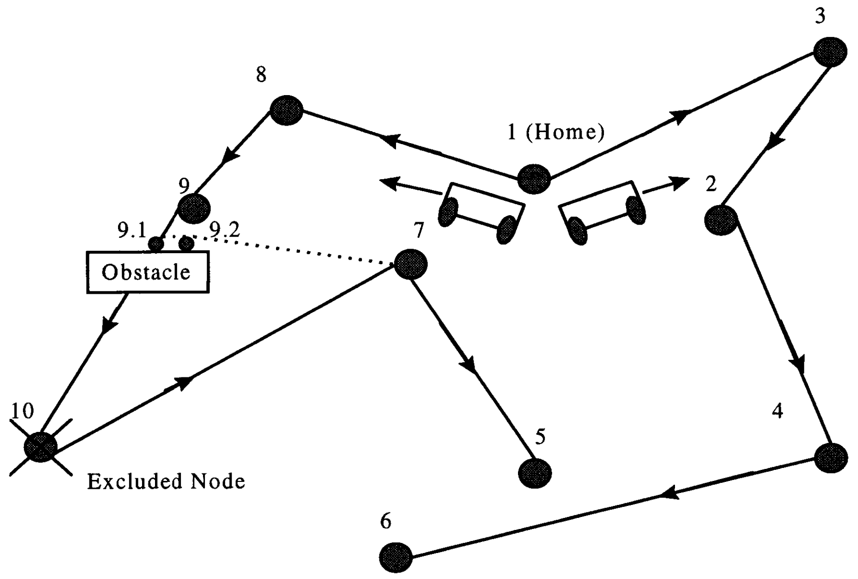


Figure 3-3: Obstacle Avoidance Failure for Nine Nodes Problem

If the obstacle avoidance failed, then the manager will command the vehicle to go to the next node in the originally assigned path (Figure 3-3). Note that the segment that connects the points 9.1 and 9.2 are included in the final verified list of segments. Thus, the final tour is {1, 8, 9, 9.1, 9.2, 7, 5} and the node 10 is saved in the list of excluded nodes. The manager will assign a vehicle to visit the excluded nodes after the completion of assigned task. Suppose node 7 is also unreachable from the point 9.2, then the mission manager commands the vehicle to go to the next one (i.e. node 5) in the path. This process will continue until the vehicles runs out of the nodes to visit or finds a node that it can visit.

Before considering a re-plan for a vehicle that has completed its task, the mission manager will usually command the idle vehicle to visit any of the existing excluded nodes if reachable or wait for the other vehicles to finish. Consider the same Example 3.1 with two vehicles to cover the complete tour. Suppose one of the vehicles completes its assigned path earlier than the other vehicle. Then, the vehicle will report back to the manager that it became idle. The manager will command the idle vehicle to visit any of the excluded nodes that have been stored so far (i.e. if the vehicle is able to reach the node). If no excluded nodes have been stored in the list, then the manager either lets the vehicle remain idle or calls a re-planner for a new plan. This re-planner redistributes the remaining workload to the available vehicles. To minimize the re-planning due to a vehicle running out of tasks, the initial mission planner should plan the initial roads to be approximately equal in distance for all of the vehicles.

3.3 Mission Re-Planner

Two different types of the re-planner exist for the EOD road-building mission. The first approach, referred to as the single-goal re-planner, is efficient for missions with few excluded nodes. This re-planner usually generates a separate road plan to visit each of the excluded nodes from the initial road plan. The second approach, referred to as the multiple-goal re-planner, is efficient for missions with many excluded nodes. This re-planner usually generates a single new road plan to visit the remaining uncovered nodes using the updated data such as new home nodes[14], new known obstacle nodes, and others. Thus, the multiple-goal re-planner is similar to the initial planner that has been discussed in the Chapter 2. Essentially, the multiple-goal re-planner computes a road plan using multiple home nodes, but the initial planner computes a road plan using a single home node.

[14] The position where a vehicle has stopped when the re-planning takes place is considered to be a new home node since this is where the vehicle will start the re-planned path.

3.3.1 Single-Goal Re-Planner

At the completion of all the vehicles' tasks, the manager checks to see if any nodes have been excluded in the mission. If so, and if there are relatively few excluded nodes, the verification mission manager calls the single-goal re-planner to reassign the excluded nodes using the verified paths. These single-goal paths will be assigned to the available vehicles to perform the verification process again. The re-planner tries to distribute the new paths of the excluded nodes to the vehicles as evenly as possible using a heuristic method. The even distribution of the excluded nodes among the vehicles will optimize the road-building mission process. Once the re-planner computes the new paths to cover the excluded nodes, the verification manager is called again to generate and issue the new waypoint following tasks to the vehicles. Thus, the iterative process between the verification manager and the re-planner continues until either all the excluded nodes have been connected or the possible paths to the excluded nodes have been exhausted. Example 3.3 shows the re-planner creating a single-goal re-plan to visit the excluded nodes.

Example 3.3 (Single-Goal Re-Planning)

Suppose some nodes have been excluded during the verification phase due to a second obstacle (Figure 3-4). The manager calls the single-goal re-planner. Suppose vehicle 2 completes its assigned task and vehicle 1 stops at the node 9.2 due to the unknown obstacles. At the end of the waypoint following task for both vehicles, the verification manager checks the excluded-node file. This file should contain the nodes 5, 7, and 10. The verification manager calls the single-goal re-planner to generate paths to the three excluded nodes. The mission manager inputs excluded nodes 5, 7, and 10, the new discovered obstacles, and the verified edges 1-3, 3-2, 2-4, 4-6, 1-8, 8-9, 9-9.1 and 9.1-9.2 into the mission re-planner.

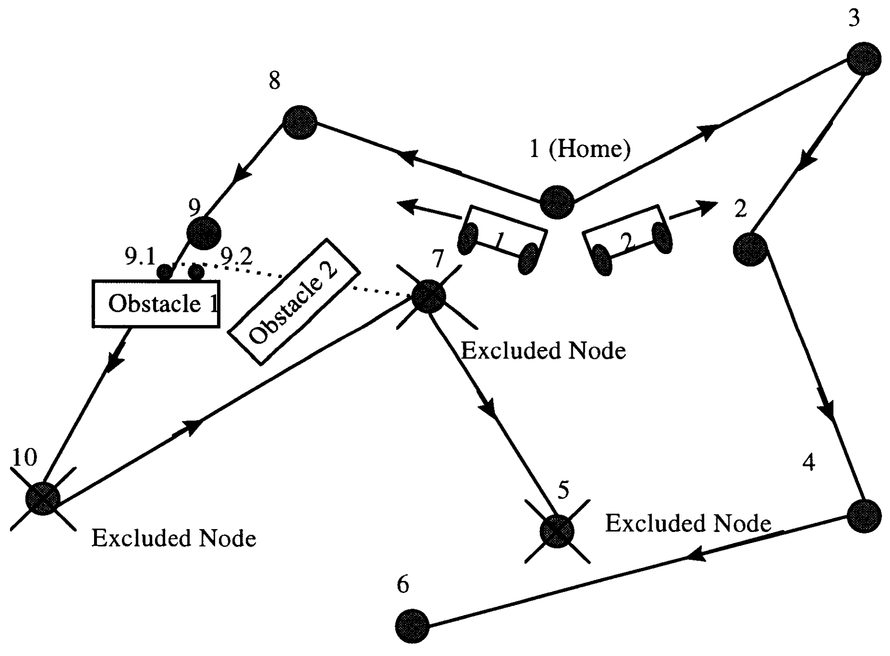


Figure 3-4: By-Pass Obstructed Node Failure for Nine Nodes Problem

Once the re-planner generates a new plan such as shown in the Figure 3-5, then the verification manager commands the vehicles to travel to the uncovered nodes using the connected paths at the fastest speed. In the Figure 3-5, the solid lines represent the verified path (i.e. free from obstacles). The dashed lines represent the closest path segments for each of the excluded nodes. The vehicles can travel as fast as possible on the solid lines and should move slowly along the dashed line segments.

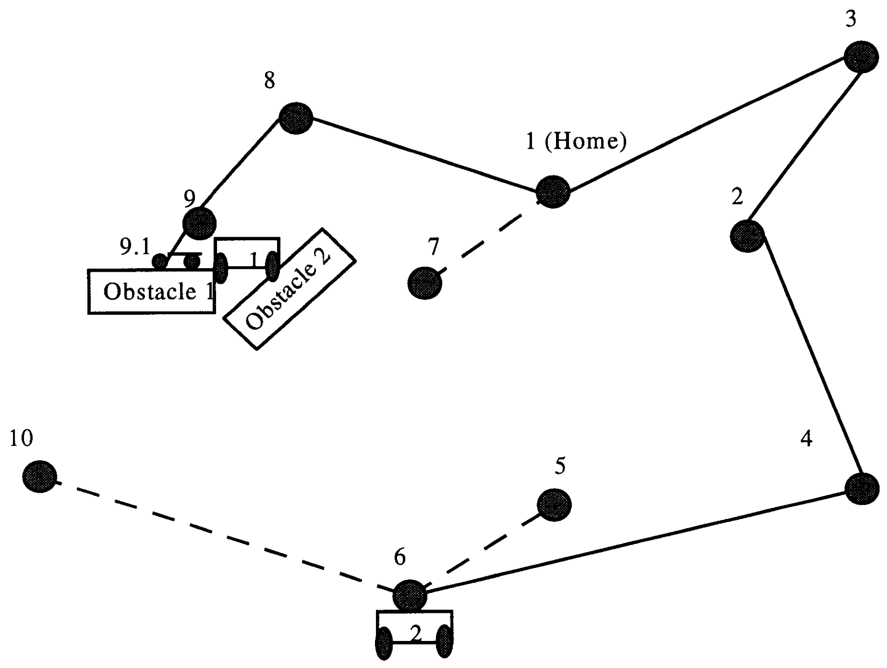


Figure 3-5: Single-Goal Re-Plan for Nine Nodes Problem

Since two vehicles are available for this problem, the workload can be divided between the two vehicles to cover the excluded nodes in the shortest possible time. The re-planner distributes the excluded nodes to the vehicles according to the distance between the vehicles and the re-planned paths. Therefore, the re-planner assigns the excluded node 7 to vehicle 1 and the excluded nodes 5 and 10 to vehicle 2. The verification manager commands vehicle 1 located at node 9.2 to cover node 7. Therefore, the sequence of the re-planned road for this vehicle is {9.2, 9.1, 9, 8, 1, 7}. Vehicle 2 located at node 6 is commanded to cover the nodes 5 and 10. The sequence of the re-planned road for this vehicle is {6, 5, 6, 10}. If either of the vehicles skip their assigned excluded node due to an unknown obstacle, then the whole process repeats again.

3.3.2 Multiple-Goal Re-Planner

If the number of excluded nodes is much greater than the number of covered nodes, then the multiple-goal re-planner can be considered rather than the single-goal re-planner. If too many excluded nodes exist at the end of the vehicles' tasks, the single-goal re-planner will not provide an efficient routing solution. If the planner tries to connect all the uncovered nodes to only a few connected nodes, then it will create a lot of individual edges rather than complete roadways. Therefore, the multiple-goal re-planner should be used to create a new initial plan. This alternative re-planning can be applied to solve for a more efficient routing solution. This re-planner creates a new road plan using multiple home nodes rather than a single home node. The multiple home nodes are the locations where each of the vehicles have stopped when the manager called the re-planner. Thus, the re-planner acts like the initial planner except it takes into account the multiple home nodes (i.e. the vehicles' initial position is not same).

The verification manager has to update the environmental data such as the added home nodes, the visited UXO nodes, the uncovered nodes, detected obstacles, and others. The multiple home nodes VRP has been discussed in the Chapter 2. The basic idea is to assign the uncovered UXO nodes to a home node using a heuristic method. Then, the single-vehicle road planning algorithm can be applied to the nodes associated with each home node. Refer to the section 3.4.2 for more detail of the multiple-goal re-planning concept.

3.4 Addressing Additional Challenges and Issues

Some of the issues that have been considered include excluded or obstructed UXO need to be visited, replacing the detonated vehicles, and other important issues that may come up during the verification phase. The chosen issues will be defined in detail and solutions proposed. Basically, the mission manager must deal with the issues that arise during verification.

3.4.1 Excluded UXO

Example 3.4 (Excluded UXO)

One of the most important mission management issues is dealing with excluded UXO or node(s) in the planned road. When planned nodes are abandoned due to newly discovered obstacles, the manager must mark the planned paths to the abandoned nodes to be unreachable so that the vehicle will not attempt to visit them later using the same paths. Also, the abandoned nodes must be saved so that the manager will remember to cover the nodes in the future. Figure 3-6 shows a case where an obstacle blocks the path between two planned nodes in a network. The rectangular box represents an obstacle that blocks a node in the initial plan. The two dotted lines represent the initially planned segments that connect three nodes but become obsolete when the obstacle is detected along the segments. Suppose an attempted obstacle avoidance for this particular obstacle has failed. Furthermore, the obstructed vehicle can't visit the rest of the nodes in the path. The node that is blocked by an obstacle and the nodes in the remaining path will be stored as excluded nodes until the manager provides a new or modified plan to visit the same excluded nodes. The manager also stores the two dotted lines (i.e. initially planned segments) in the obstructed-path file so that they will not be considered in re-planning. Therefore, the point where the vehicle stopped due to an obstacle is the last node that has been stored in the verified road plan. This point will be used as the home node for the re-planning and re-plan verification processes. The last three nodes are stored in the excluded-node file.

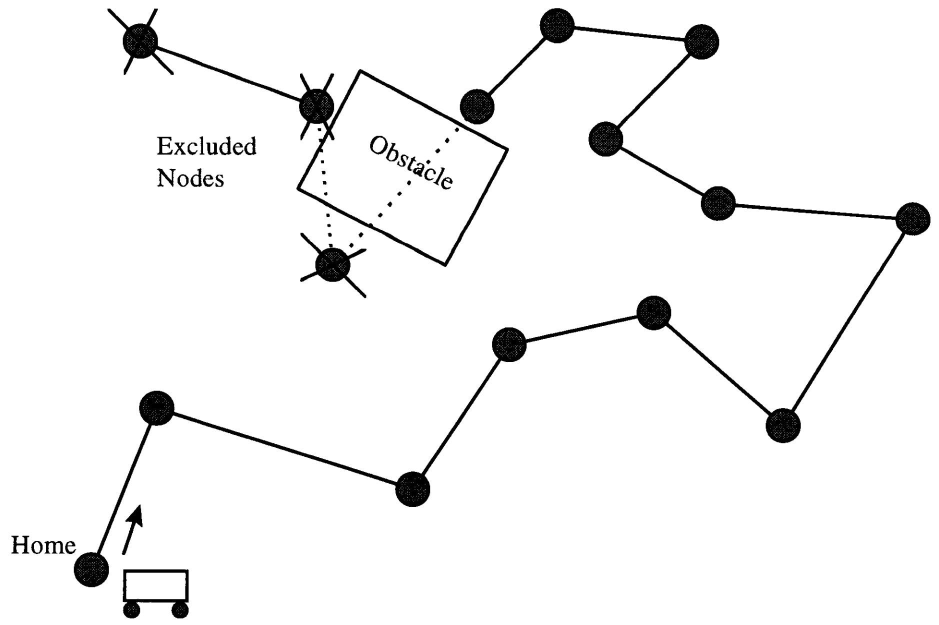


Figure 3-6: Excluded Nodes by an Obstacle

Many different strategies exist to modify the initial plan to visit the excluded nodes due to an obstacle on the way. The EOD mission manager calls the single-goal re-planner to re-plan paths to cover the excluded nodes. The re-planner creates a re-plan that covers the excluded nodes with the new information about the connected nodes and detected obstacles. Figure 3-7 illustrates the closest paths from the excluded nodes to the connected paths in the Figure 3-6. The solid line of segments represent the verified road segments. The dashed lines represent the candidate paths (i.e. the closest paths) a vehicle can follow to visit the excluded nodes. The short dotted lines still represent the segments removed due to the obstacle in the initial path. Once this re-plan is available, the verification manager selects a vehicle that is closest to the re-planned road segments. Then, it generates and issues the waypoint following command to that vehicle.

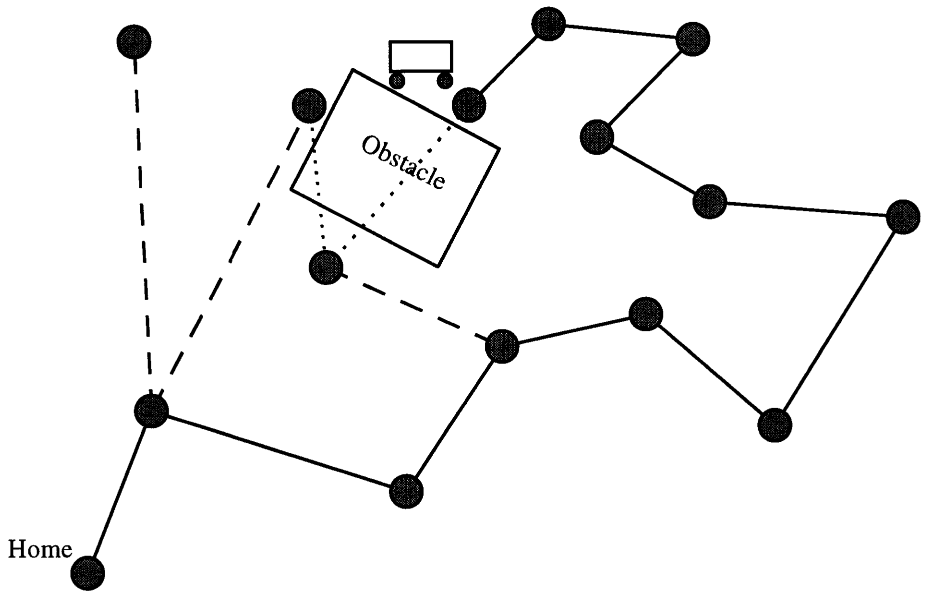


Figure 3-7: Candidate Connecting Paths to Connect Excluded Nodes

The manager doesn't simply store the excluded nodes to the verified road plan when the segments are in fact verified to be impassable. The manager needs to identify the possible connecting segments and decide the best one to connect to each of the excluded nodes based on some criteria such as the shortest length, and existence of other paths that have been verified and included in the plan. If the chosen candidate segment has an obstacle along the path, then the next best segment must be explored. This process continues until one candidate segment has been verified to be obstacle-free or the choice of the candidate segments has been exhausted. Once a segment has been verified and included in the plan, the remaining candidate segments are ignored. In this case, the verified list of plans and the initial list of plans will be different due to the removed and newly added paths.

In the discussion above, only a single vehicle is assumed to be available to visit the candidate connecting paths to cover the excluded nodes. However, the EOD mission is assumed to have multiple vehicles available to visit the nodes in the network. Consider three available vehicles. Each vehicle can be assigned to cover an excluded node since three nodes have been excluded in the previous problem. Furthermore, the candidate connecting segments don't have to be considered one at a time. Since three vehicles are available, one of them can travel along the first chosen candidate path. The second vehicle can take the second considered candidate path and so forth. Once one of the vehicles completes and verifies that its assigned candidate path is free from obstacles, the manager commands the other two vehicles to stop their exploration. Furthermore, other heuristic methods can resolve the problem with a planned node or nodes being excluded due to an obstacle.

3.4.2 Detonated Vehicle

Although the locations of UXO are known *a priori*, they are known only within few feet. During the verification of the obstacle-free roads, the vehicles may get blown up accidentally by the UXO. Since the locations of UXO are not known exactly, the initially planned roads may cross the exact points or neighbor areas of the UXO. The vehicles do carry a metal detector to sense the possible UXO, but the sensors aren't activated during the road-building phase. The metal detecting sensors are turned on during the UXO-clearing phase where the actual searching, acquiring, and picking-up tasks are performed. Thus, the vehicles have only the obstacle detector such as a sonar and a bumper in the active mode during road-building phase. Therefore, the manager must consider the cases where the vehicles may be detonated by UXO during the verification process.

The simplest plan modification due to a detonated vehicle is to command a new vehicle at the home node to replace the lost vehicle. However, if no additional vehicle is available, then the manager can call the multiple-goal re-planner to compute a new road plan to cover the remaining nodes.

Example 3.5 (Detonated Vehicle)

An approach to deal with detonated or removed vehicles is to either re-plan entirely or do a partial re-plan that only modifies the initial plan rather than providing a completely new plan. Consider a case of three vehicles prosecuting an initially planned route that connects all of the nodes in a given network. Figure 3-8 illustrates the three initially planned routes for the three available vehicles at the home node. These routes are not necessarily optimal but reasonable and feasible routes.

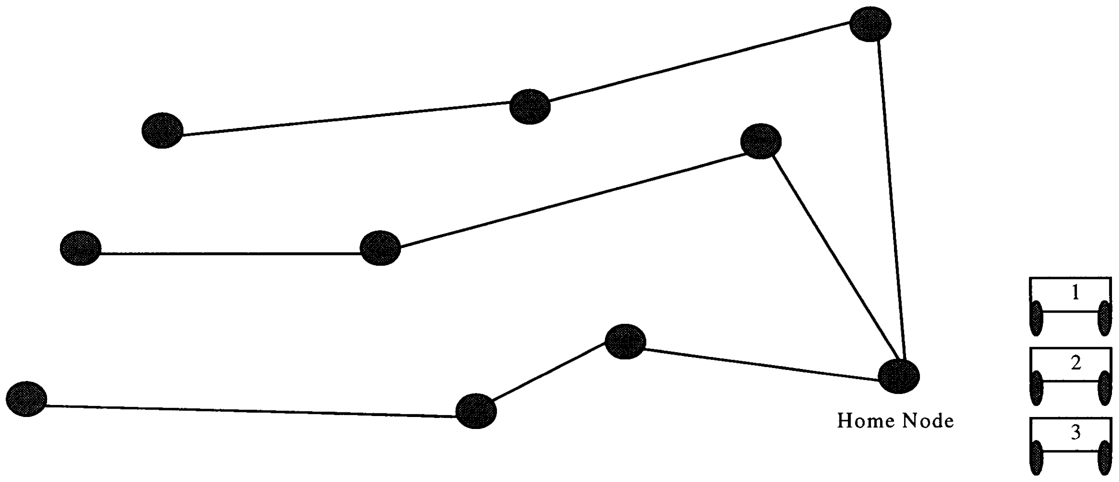


Figure 3-8: Three Initial Routes for Three Vehicles for Detonated Vehicle Example

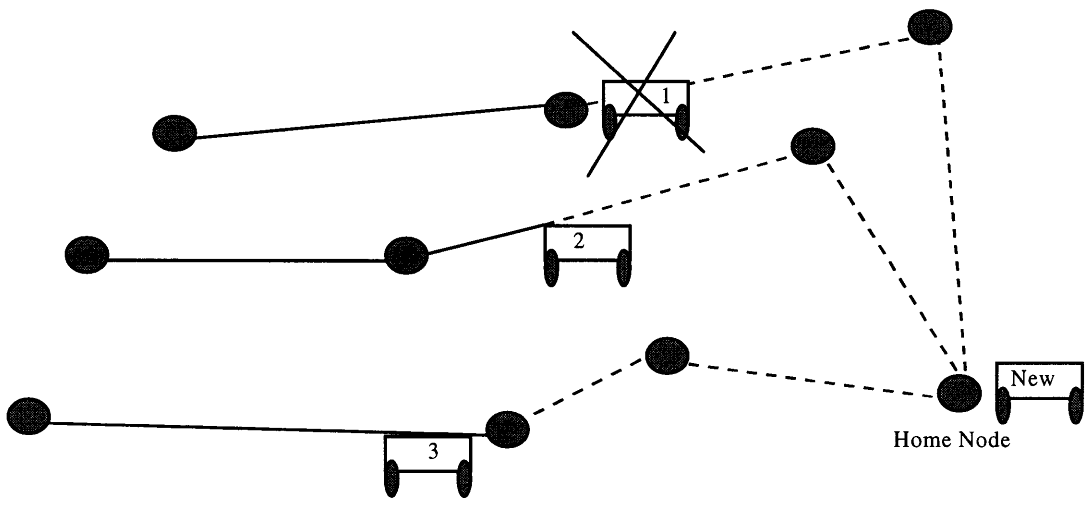


Figure 3-9: Example of a Detonated Vehicle

Figure 3-9 shows the situation where one of the three vehicles is detonated by an UXO during the verification process. The dashed lines represent the road segments that have already been verified to be obstacle-free paths. The big "x" on the first vehicle indicates that the vehicle has been detonated by an UXO. The solid lines represent initially planned roads yet to be explored.

One approach is to simply replace the detonated vehicle with a new vehicle positioned at the home node as shown in the Figure 3-9. The new vehicle can travel the verified segments of road at the fast speed. Once the new vehicle arrives at the point where the previous vehicle is detonated, it resumes the verification process to verify the rest of the route. The manager doesn't have to re-plan for the other two vehicles since a re-plan is done for just the vehicle that has been blown-up. However, if no new vehicle is available to replace the detonated vehicle then some other method of re-plan must be provided by the manager.

If no additional vehicle is available other than the three vehicles, then the mission manager must re-plan to redistribute the assigned and uncovered nodes. As soon as the mission manager decides to re-plan, the remaining two vehicles must halt further verification of UXO. The mission manager calls the multiple-goal re-planner to reassign the remaining uncovered nodes to the other two vehicles. Figure 3-10 shows a re-plan by the multiple-goal re-planner. The second vehicle covers the rest of the nodes in the path assigned to the detonated vehicle. Likewise, the third vehicle covers the last node of the path assigned to the second vehicle.

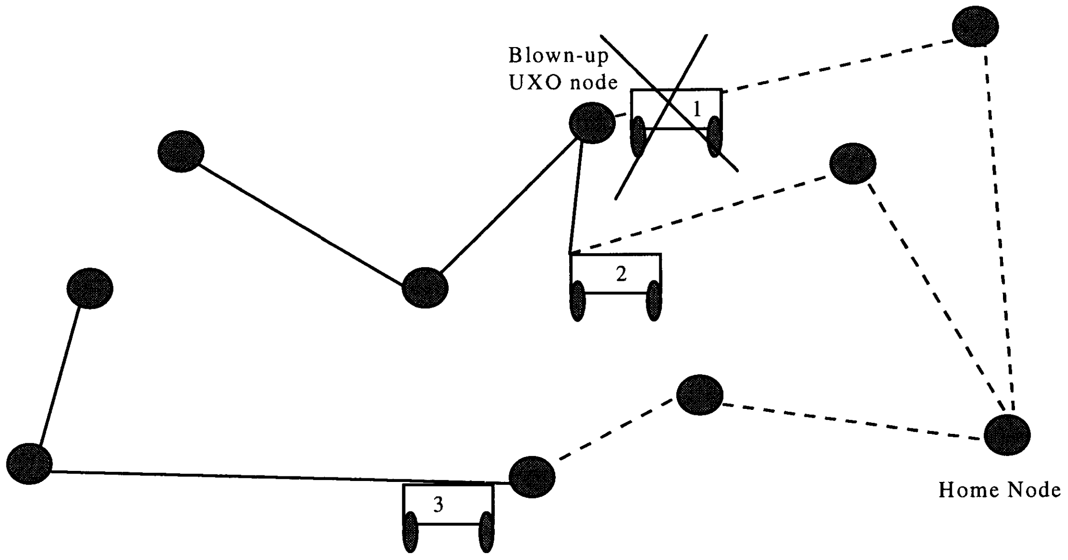


Figure 3-10: Reassigning a Node for Detonated Vehicle

Chapter Four : Mission Manager Software Functional Description

A detailed description of the mission manager is presented to show the basic features of the mission management system. The implementation of some software modules within the overall mission manager architecture are described in the Chapter 5.

4.1 Mission Manager Architecture

An overview of the mission manager is introduced for the user to understand the "big" picture of an EOD road-building process using an automated mission coordinating solution. The mission manager is divided into three main functions (Figure 4-1). The first is referred to as the initial mission planner, which establishes an initial road plan for a defined mission problem. The second major function is referred to as the verification mission manager. This function manages and coordinates the multiple vehicles during the exploration phase of road-building mission.

The last main function is referred to as the re-planning function. Two different forms of re-planner exist, depending on the level of complexity in the updated environmental information. The first type of the re-planner, the single-goal re-planner, provides road plans that will be efficient only for few excluded nodes. This re-planner creates a plan to cover the excluded nodes using the verified paths. The second type of the re-planner, the multiple-goal re-planner, provides a new road plan that will be efficient for many excluded nodes. This re-planner creates a new road plan for the excluded nodes using the updated environmental information (including new home nodes).

The last two functions (i.e. verification manager and re-planner) are closely coupled in any given mission problem. The mission managing function is called at least once during the mission and may become an iterative process depending on the mission problems. However, re-planning function may not be called any number of time depending on whether obstacles are encountered during exploration.

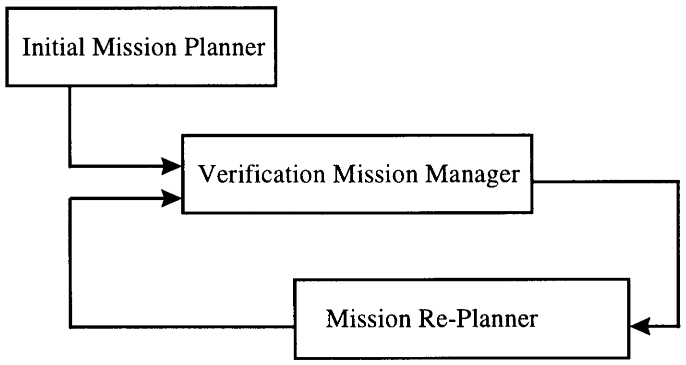


Figure 4-1: Three Main Functions of the Mission Manager

To illustrate the full capability of the mission manager, both the initial planning and managing functions are exemplified in a step-by-step procedure in the section 4.3.

4.2 Assumptions in the Mission manager

The following are the major assumptions of the mission that have been established to develop the baseline mission manager:

- The locations of the UXO nodes are known a priori and each represented as a point in the network. The approximate locations of the UXO nodes can be known through various methods such as an autonomous helicopter, satellite reconnaissance, autonomous sweeping vehicle, and others.
- Vehicles are assumed to follow a straight line of path.

- Vehicles are also assumed to be capable of turning through any angle sharply and instantaneously. In other words, the vehicles can turn in a place (like a tank).
- Vehicles are assumed to transition from zero (fps) to slow (1 fps) to fast (6 fps) speed instantaneously. This assumption simplifies the calculations of the total distance that the vehicles have traveled at the end of the mission.
- Vehicles keep track of their exact location at all times.
- Vehicles are assumed to have unlimited range. This assumption allows simplification in the vehicle routing problem.
- Vehicles are assumed to be at the home node at the start of the verification phase.
- The commands are assigned to the vehicles instantaneously. In other words, the time delay in the transmission of data via a two-way radio modem is ignored in the problem. The communication issue is outside of the scope of this thesis.
- Vehicles are assumed to be capable of detecting and avoiding obstacles.

4.3 Planning and Managing Procedures

The three major functions of the mission manager have been described in the Chapter 2 and 3. These two chapters have discussed the algorithms associated with the selected initial planner, verification manager, and re-planner. However, the integration of the three main functions haven't been discussed in the detail. The road-building mission manager that includes initial planner and re-planner is described in the following sequence of steps.

4.3.1 Initial Mission Planner Procedure

STEP 1: Read an input data file that contains the known environmental information provided by a user.

Table 4-1 shows a typical input data file that contains the *x* and *y* coordinates and the types of the various nodes to the initial mission planner.

Table 4-1: An Input Data File to the Initial mission planner

Node	X-Coordinate	Y-Coordinate	Type
0	0	0	-4 (Home)
1	-10	3	-1 (UXO)
2	7	2	-1 (UXO)
3	5	5	-1 (UXO)
4	2	8	-1 (UXO)
5	8	8	-1 (UXO)
6	-5	5	2 (Obstacle)
7	5	7	1 (Obstacle)
8	10	10	-3 (ODA)

STEP 2: If the mission is a large mission (i.e. 100s of nodes), then partition the problem.

From the input file (*STEP 1*), the verification manager checks to see whether the given mission contains too many UXO nodes. If the number of UXO nodes exceeds a user-defined value *N*, then the planner executes a partitioning function. The value *N* can be derived from the computational limits on the initial planner. For instance, the planner may not be able to handle more than 70 UXO nodes. Then, the value *N* is set to 70 nodes. The partitioning function simply divides the number of nodes by the predefined value *N* and generates several smaller sub-problems. If the value *N* is defined as 70 nodes, then the first divided sub-problem will contain the first 70 UXO nodes that are closest to the home node. The second sub-problem will have the next 70 UXO nodes that are closest to the home node and so on. After the large mission is partitioned this way, *STEP 4* through *STEP 10* can be applied to each sub-problem separately.

Note that each of the smaller problems contains the exact same predefined obstacle nodes, home node, and ODA nodes.

STEP 3: If the mission contains multiple home nodes, then distribute the UXO nodes in such a way that each UXO is assigned to a single home node.

The proposed algorithm in section 2.4.2 for multiple home nodes can be used to distribute the UXO nodes into groups, each corresponding to a single home node. Once every UXO node has been assigned to a home node using Equation (2.1) and (2.2), then *STEP 4* through *STEP 10* can be applied to each group. Having multiple home nodes becomes a critical planning issue in any multiple vehicle routing problems. For instance, suppose three vehicles are available to cover the UXO nodes and two home nodes. The planner generates three paths for the three vehicles to cover the nodes. However, if one path contains both the home nodes and the other two paths contain just the UXO nodes, then the vehicles will have no roadway leading to a home node in the clearing process. Therefore, the planner should create multiple paths but include at least one home node in each path.

STEP 4: Create distance and connectivity matrices using the information in the input data file.

The distance between any two nodes is easily calculated using their coordinates. Likewise, the connection between any two nodes can be determined easily using the coordinates of the obstacle nodes. If no obstacle is present in the path between any two nodes, then the two nodes are assumed to be connected. If any two nodes are connected, then the connectivity is represented as the number "1". Otherwise, the connectivity is represented as the number "0". Table 4-2 and 4-3 show examples of distance and connectivity information obtained from the input data file in Table 4-1. Note that the connectivity matrix is symmetric. This indicates that the network is undirected (i.e. the vehicles are allowed to travel in any direction on a path). Figure 4-2 shows a graphical representation of the input information using the coordinates of the nodes. Note that the two predefined obstacles are drawn as circles with the radius given in the Table 4-1.

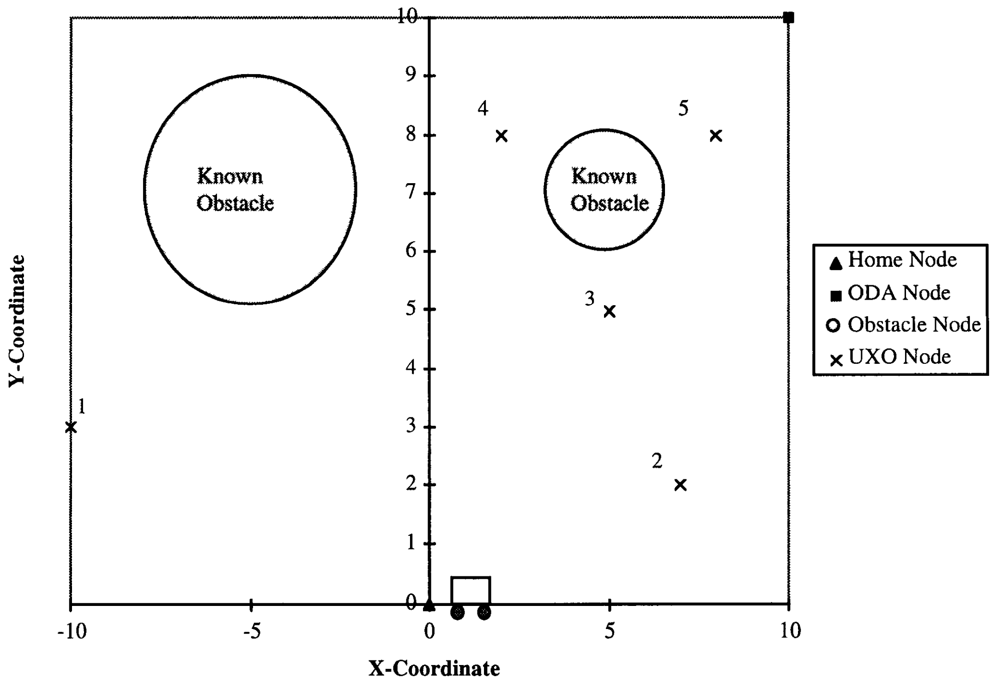


Figure 4-2: A Graphical Representation of the Input in Table 4-1

Table 4-2: Distance Matrix of the Example Input Information

From/To	0	1	2	3	4	5	6	7	8
0	0	10.4	7.3	7.1	8.2	11.3	7.1	8.6	14.1
1	10.4	0	17.0	15.1	13.0	18.7	5.4	15.5	7.0
2	7.3	17.0	0	3.6	7.8	6.1	12.4	5.4	8.5
3	7.1	15.1	3.6	0	4.2	4.2	10.0	2.0	7.1
4	8.2	13.0	7.8	4.2	0	6.0	7.6	3.2	8.2
5	11.3	18.7	6.1	4.2	6.0	0	13.3	3.2	2.8
6	7.1	5.4	12.4	10.0	7.6	13.3	0	10.2	15.8
7	8.6	15.5	5.4	2.0	3.2	3.2	10.2	0	5.8
8	14.1	7.0	8.5	7.1	8.2	2.8	15.8	5.8	0

Table 4-3: Connectivity Matrix of the Example Input Information

From/To	0	1	2	3	4	5	6	7	8
0	0	1	1	1	1	1	0	0	1
1	1	0	1	1	0	0	0	0	0
2	1	1	0	1	1	1	0	0	1
3	1	1	1	0	0	1	0	0	1
4	1	0	1	0	0	0	0	0	1
5	1	0	1	1	1	0	0	0	1
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	1	0	1	1	1	1	0	0	0

STEP 5: Plan an initial road-plan for the number of available vehicles.

```
************
Initial Plan
************

Robot 1:
         0.0,    0.0
         2.0,    8.0
        10.0,   10.0
         8.0,    8.0
         5.0,    5.0
         7.0,    2.0
       -10.0,    3.0
         0.0,    0.0    Distance: 54.64

************************************ Time: 854 µs
```

Figure 4-3: An Example of Initial Plan Output Data for a Single Vehicle

The user inputs the number of available vehicles at the command line. Likewise, the user inputs the type of road planning algorithm to use to generate an initial road plan for the given problem at the command line. The various proposed road planning algorithms have been fully discussed in the Chapter 2. Once the planner computes an initial road plan, it will store the created initial road plan in an output file. Figure 4-3 shows an example output file of an initial plan[15] for the previous sample problem. The initial plan in this Figure is a road plan for a single vehicle to cover the five UXO nodes and an ODA node in the problem.

If multiple vehicles are available, then the planner can create multiple paths to distribute the UXO nodes among the vehicles. The time that is shown in the output is the time that the planner took to generate the initial plan. Note that the initial plan time is on the order of milli seconds (i.e. extremely fast). However this initial plan time doesn't include the time to read the input data file. Also, the planner computes the total distance that the vehicle will have to travel to cover the initially assigned nodes. The road plan is represented by the coordinates of the nodes in the order of the visit.

Figure 4-4 displays the graphical representation of the initial plan in the Figure 4-3. Note that the vehicle starts from the home node and doesn't return to the home node. All the vehicles will stop at the last assigned node but the manager has an option to command the vehicle to return to the home node by using the verified roadway that they have provided.

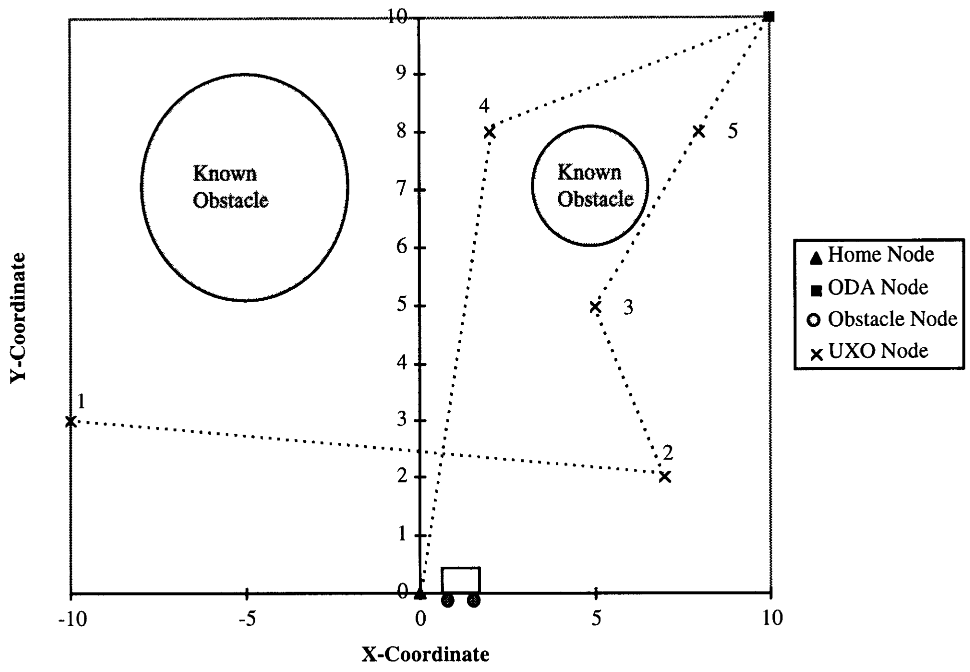


Figure 4-4: Graphical Representation of an Initial Plan for a Single Vehicle

4.2.2 The Verification mission manager Procedure

STEP 6: Generate the waypoint following commands to each of the available vehicles.

These task commands can easily be generated using the initial road plan. The node coordinates in the initial plan represent waypoints that must be followed by the vehicles. A waypoint following task requires a certain type of data structure. The manager must fill the data structure with the initially planned waypoints (i.e. initial road plan) and select an appropriate speed for the vehicles. If multiple paths are available in the initial plan, then the manager must generate task commands using the multiple paths for the multiple vehicles.

For the example problem, the waypoints that must be explored by a single available vehicle are the nodes 0, 4, 8, 5, 2, and 1. The vehicle should be commanded to execute these waypoints at a slow speed (1 fps). The terrain environment is unknown to the vehicle so the vehicle must be guided by detecting sensors

[15] The displayed initial road plan for this example is generated using the combined algorithm of the mixed routing method and the SFC distributing method.

to avoid any hidden obstacles. The obstacle detecting sensors require a long time to scan an area completely. Therefore, the vehicle must travel at the slowest speed to allow the sensors to scan.

STEP 7: Respond to any detected event (*i.e. obstacle, competed task, and detonated vehicle*) *and update the verified segment file.*

The manager must keep track of the nodes that a vehicle visits successfully during verification. When a vehicle travels along a segment or "edge" (i.e. path between any two nodes) without any detected event, then the manager stores the path between the two visited nodes in a verified path file. From the example problem, suppose the vehicle is able to visit the assigned nodes without any detected event. The vehicle reports to the manager that it has competed the waypoint following task. The manager stores the verified plan, which is the initial plan in this case, to an output file. Figure 4-5 shows a sample output file that contains the verified road plan for the example problem. The time listed in Figure 4-5 is the total time to build this roadway using a single vehicle. Note that the time to manage the data was negligible compared to the vehicle's travel time. The output file also contains the coordinates of excluded UXO nodes (i.e. nodes which have been eliminated due to the newly detected obstacles). Furthermore, the total number of vehicles used and detonated in the mission are given in the output file.

```
************
Actual Path
************

Robot 1:
      0.0,    0.0

      2.0,    8.0
     10.0,   10.0
      8.0,    8.0
      5.0,    5.0
      7.0,    2.0
    -10.0,    3.0
       0.0,    0.0    Distance: 54.64

************************************ Time: 54.64 s
```

Figure 4-5: A Sample Output File of the Mission Manager

However, if a vehicle is blocked during the verification, then the manager must generate a plan modification or a re-plan. The manager usually considers modification to the initial plan before calling the mission re-planner. At this stage, the manager has several options to execute depending on the event. Three main events can stop a vehicle from further exploration of its assigned UXO nodes. The first and the most common event is a vehicle being stopped by an obstacle that it can't avoid to reach the assigned node. Refer to *STEP 7.1* for further details of the manager's response to an impassable obstacle. Before the execution of the modified plan, the manager must store the additional nodes that are involved in the vehicle's obstacle-avoidance maneuver. The second event is a vehicle being idle due to completion of its assigned task. Refer to *STEP 7.2* for further details of the manager's response to an idle vehicle. The last (hopefully uncommon) event is detonation of a vehicle during the verification process. Refer to *STEP 7.3* for further details of the manager's response to a detonated vehicle. Before execution of a re-plan, the manager must store the location where the vehicle has been detonated (i.e. this location is approximately where UXO has blown-up and this node no longer needs to be serviced).

STEP 7.1: Execute by-pass node command for an obstructed vehicle.

If a vehicle reports to the manager that it has failed an obstacle avoidance, the manager must provide a new command. The manager usually commands an obstructed vehicle to proceed to the next node in the assigned path. The manager must remove the node that an obstructed vehicle ignored from the initial road plan. Furthermore, the manager stores the removed node in an excluded-node file for later visit.

From the example problem, suppose a vehicle covers the nodes in the sequence of 0, 4, 8, and 5 before an unknown obstacle is detected (Figure 4-6). The vehicle tries to do obstacle-avoidance maneuver, but reports to the manager that it has failed. The manager stores the two nodes from the obstacle-avoidance maneuver in the verified road plan (solid lines represent verified roadway). The dotted lines of path represent the initial roadway.

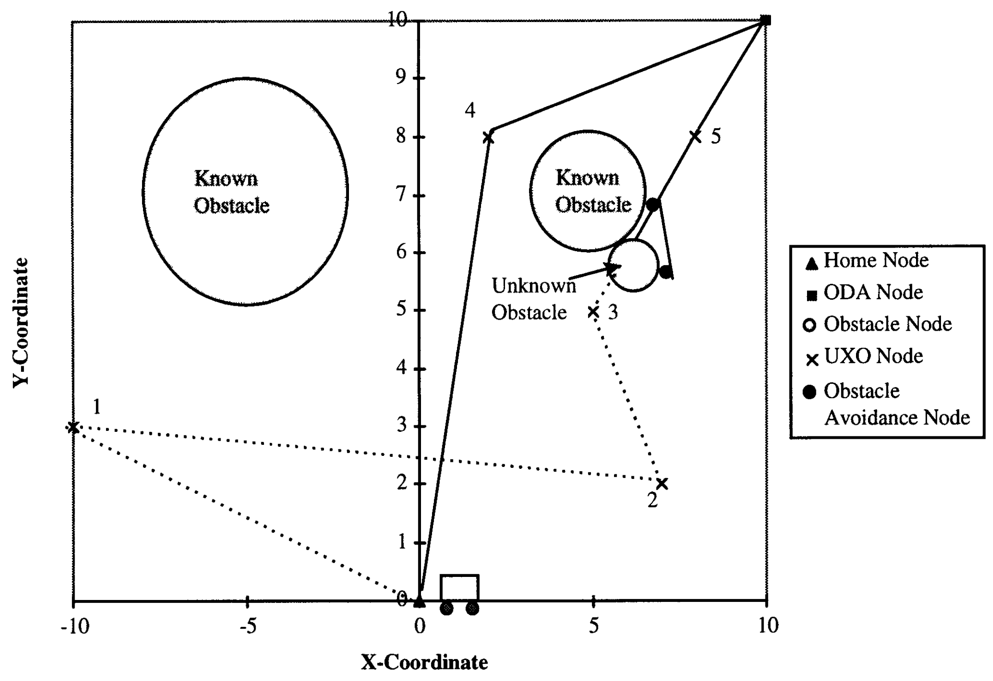


Figure 4-6: An Unknown Obstacle Detected During Verification Process

Next, the manager responds to the obstructed vehicle by commanding it to proceed to the next node in the path, which is node 2 (Figure 4-7). The dashed lines represent the re-planned roadway.

STEP 7.1 is repeated throughout the verification phase if any vehicle reports obstacle avoidance failure to the manager. When all of the vehicles complete their assigned task, the excluded nodes (if any) will be re-planned by the single-goal re-planner. Also, the verified roadways are stored into an output file similar to the one shown in the Figure 4-5. Refer to section 4.3 for the details of the re-planner.

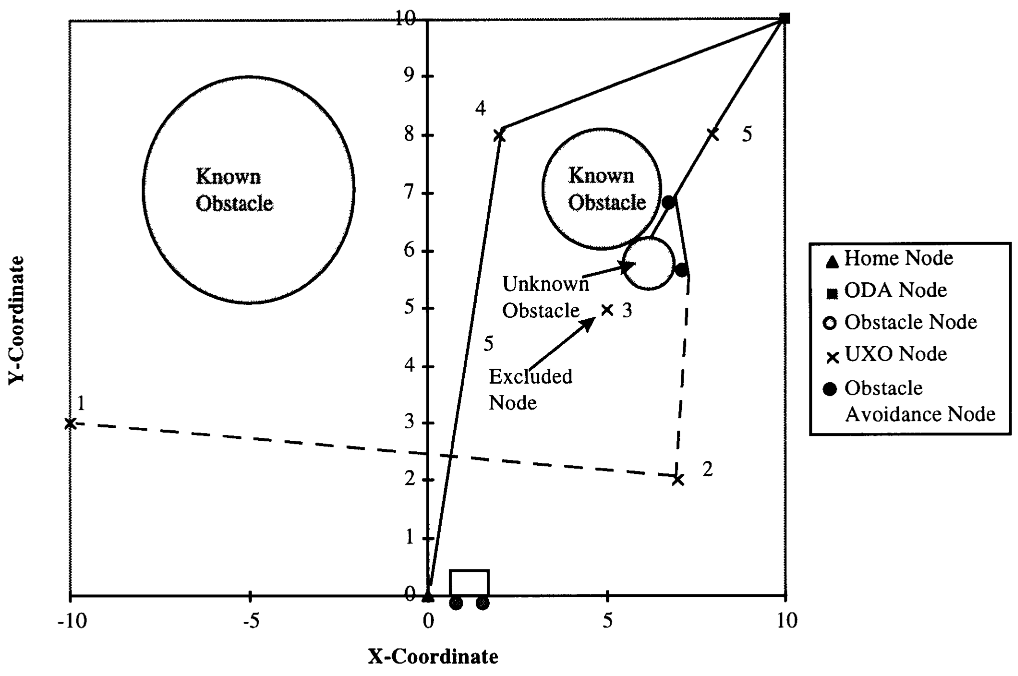


Figure 4-7: By-Pass Node for Obstructed Vehicle

STEP 7.2: Execute modified re-plan for an idle vehicle.

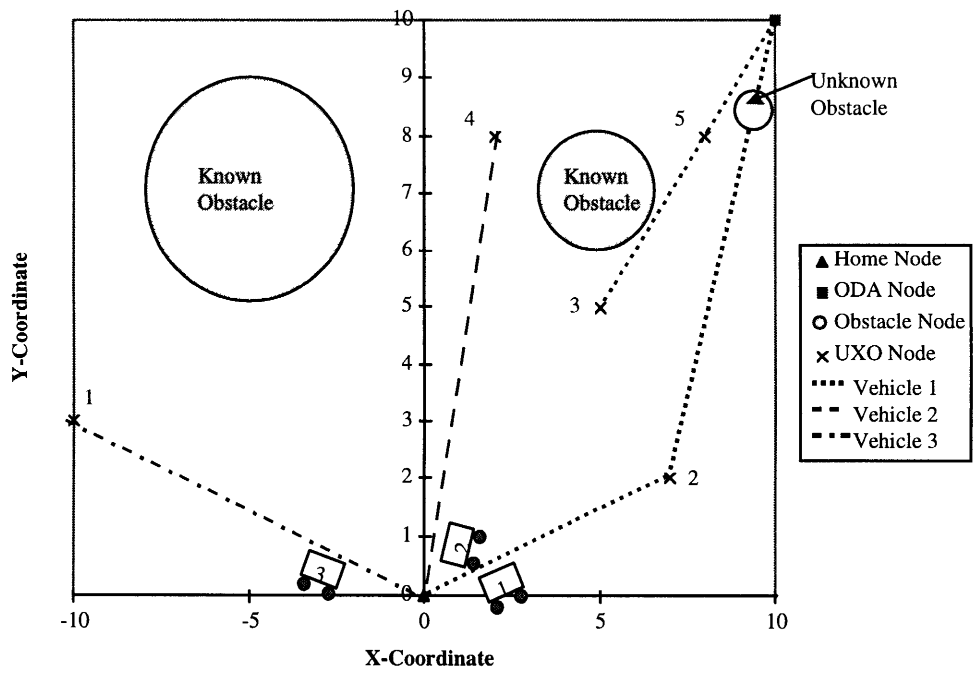


Figure 4-8: Representation of an Initial Plan for Three Vehicles

If a vehicle reports to the manager that it has completed its assigned task, then the manager needs to generate new waypoints for the idle vehicle. Modifying the plan for an idle vehicle is extremely difficult since the rest of the vehicles are still in the middle of the verification phase. Furthermore, no other UXO nodes are available to be assigned to the idle vehicle. A simple plan modification for an idle vehicle is to assign an excluded node (if any are reachable) to the vehicle. If either no excluded node exists in the file so far or none of the excluded nodes is reachable (i.e. obstructed by a known obstacle), then the manager allows the idle vehicle to wait for other vehicles to finish their tasks. However, the manager does have an option to re-plan to redistribute the uncovered UXO nodes using new knowledge of the environment with or without the verified paths. Refer to the section 4.3 for details of the re-planner procedure.

In the example problem, consider three available vehicles rather than a single vehicle for building the obstacle-free roadways. Figure 4-8 displays the graphical representation of the three initially planned routes for the three vehicles. Suppose the first vehicle detects an unknown obstacle in its path and that obstacle avoidance has failed. The ODA node (i.e. node 8) is stored into an excluded node file. When the second vehicle completes its assigned path, the vehicle reports to the manager that it became idle. Then, the manager executes the modified plan by commanding the idle vehicle to proceed to the ODA node if the path is not obstructed with known obstacles.

Assuming the second vehicle doesn't detect any unknown obstacle on the path to ODA node, it is able to visit ODA node. Once all the vehicles complete their task, the manager stores the verified roadways to an output file similar to the Figure 4-5. Figure 4-9 shows the re-planned routes for the idle and the obstructed vehicles (i.e. vehicle 1 and 2). Note that vehicle 3 is unaffected by this plan modification since it's a re-plan only for vehicle 1 and 2..

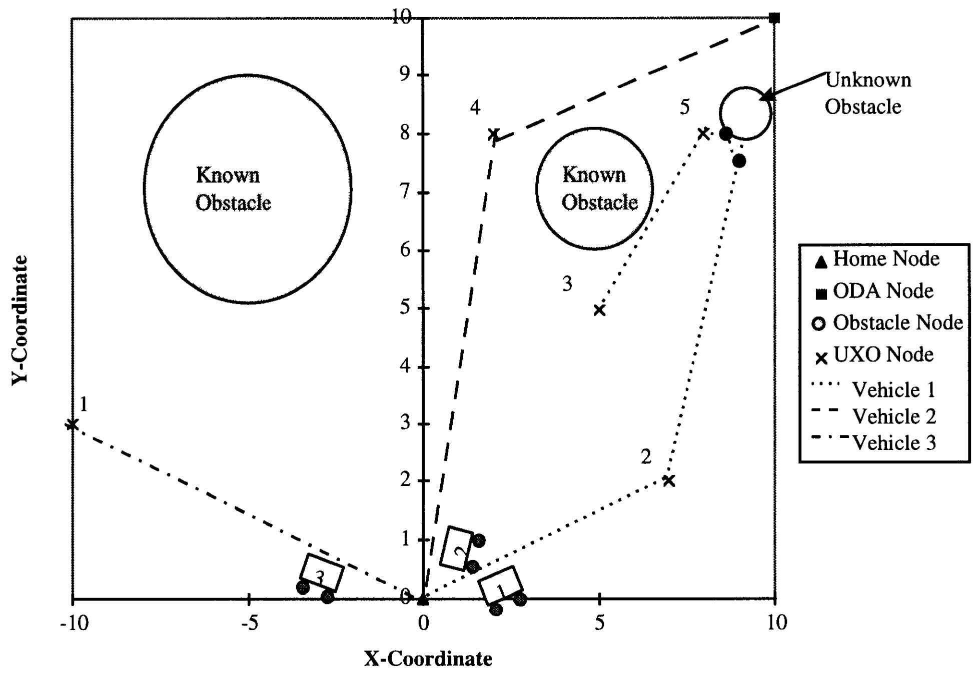


Figure 4-9: Re-Plan for Idle Vehicle

STEP 7.2 is repeated throughout the verification phase if any vehicle reports that it has completed its assigned task to the manager. When all of the vehicles complete their assigned task, any remaining excluded nodes will be re-planned by one of the two re-planners. Also, the verified roadways are stored into an output file similar to the one shown in Figure 4-5. Refer to section 4.3 for the detailed sequence of steps of the re-planner procedure.

STEP 7.3: Execute modified re-plan for a detonated vehicle.

If a vehicle stops reporting a verified segment of its path to the manager, the manager assumes that the vehicle is unavailable (i.e. lost or dead). Distributing the remaining uncovered UXO nodes from the path assigned to the detonated vehicle is extremely difficult. The other vehicles are still in the process of executing their assigned roadways. However, if any additional vehicle is available at the home node, then a simple plan modification can be achieved by simply commanding the new vehicle to cover the unassigned UXO nodes. If this is the case, the mission is executed as initially planned. However, if no vehicle is available at the home node, then the manager calls the multiple-goal re-planner to redistribute the UXO nodes among the remaining vehicles.

When all of the remaining vehicles complete their assigned task, any remaining excluded nodes will be re-planned by the single-goal re-planner. Also, the verified roadways are stored into an output file similar to the one shown in the Figure 4-5. Refer to section 4.3 for the detailed sequence of steps of the re-planner procedure.

4.2.3 The Mission Re-Planner Procedure

The re-planner is mainly responsible for planning to cover the excluded UXO node(s). The manager calls the re-planner in the following cases: re-plan to cover the excluded nodes when all of the vehicles have completed their assigned task, re-plan if a vehicle completes its assigned task much earlier than the other vehicles, and re-plan if a vehicle is lost during its verification phase.

STEP 8: Execute the single-goal re-planning only if any excluded node exists.

The manager calls the re-planner only if a re-planning must be done to cover the excluded UXO nodes. The single-goal re-planner is usually called to cover only few excluded UXO nodes. The re-planner finds possible paths to visit the excluded nodes using verified paths as much as possible. Chapter 3 has described a way to handle this type of re-planning in detail.

STEP 9: Execute the multiple-goal re-planning only if requested by the manager.

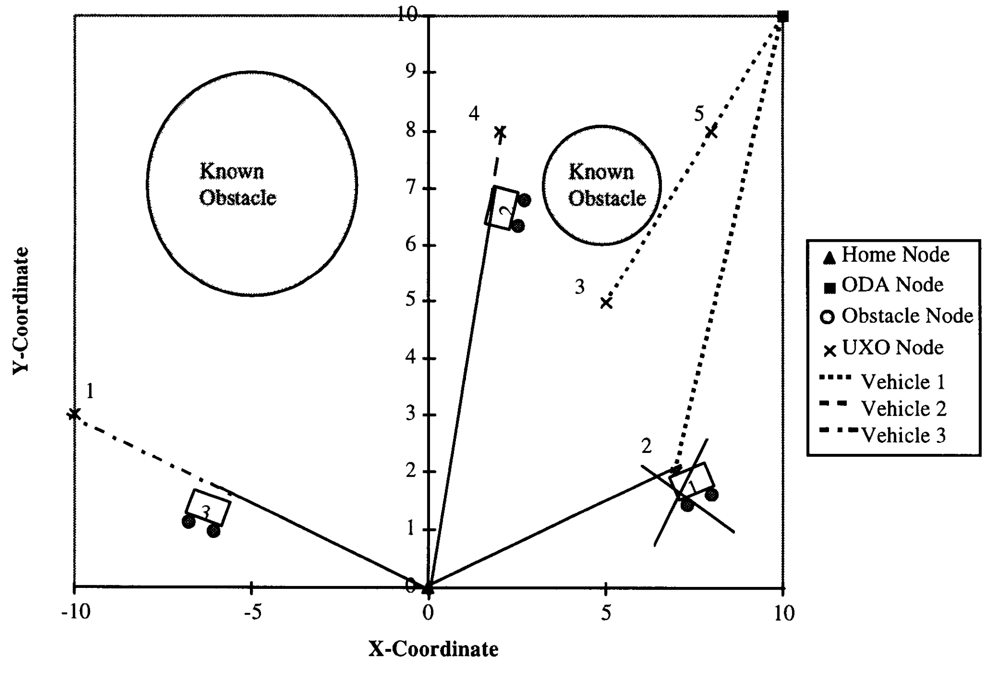


Figure 4-10: Detonated Vehicle During the Verification Phase

The manager can call this re-planner to create a new road plan based on the updated environmental data. This re-planner is usually considered in the case where many UXO nodes have been excluded during the verification process. Furthermore, the manager considers this re-planning if a vehicle gets detonated or completes its task early in the verification phase. The single-goal re-planning is not the optimal planner to handle these type of events (i.e. many UXO nodes may be left when the re-planner is called).

Consider the example problem with three vehicles (i.e. three paths). Suppose vehicle 1 explodes when it gets to its first assigned node 2. Also, suppose that vehicle 2 and 3 haven't reached their assigned node when vehicle 1 blew up (Figure 4-10). If the manager calls the multiple-goal re-planner for a new plan, then the rest of the five nodes that must be visited will be considered to be the excluded nodes. The manager stores nodes 1, 3, 4, 5, and 8 as excluded nodes when vehicle 1 is detonated. The two known obstacles remain the same.

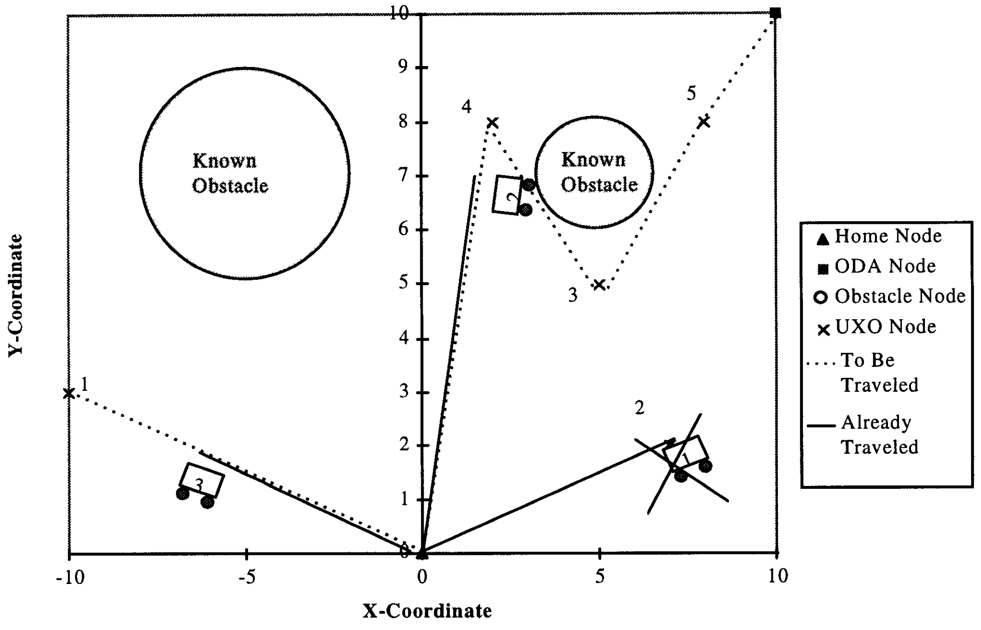


Figure 4-11: Re-Plan Using the Multiple-Goal Re-Planner

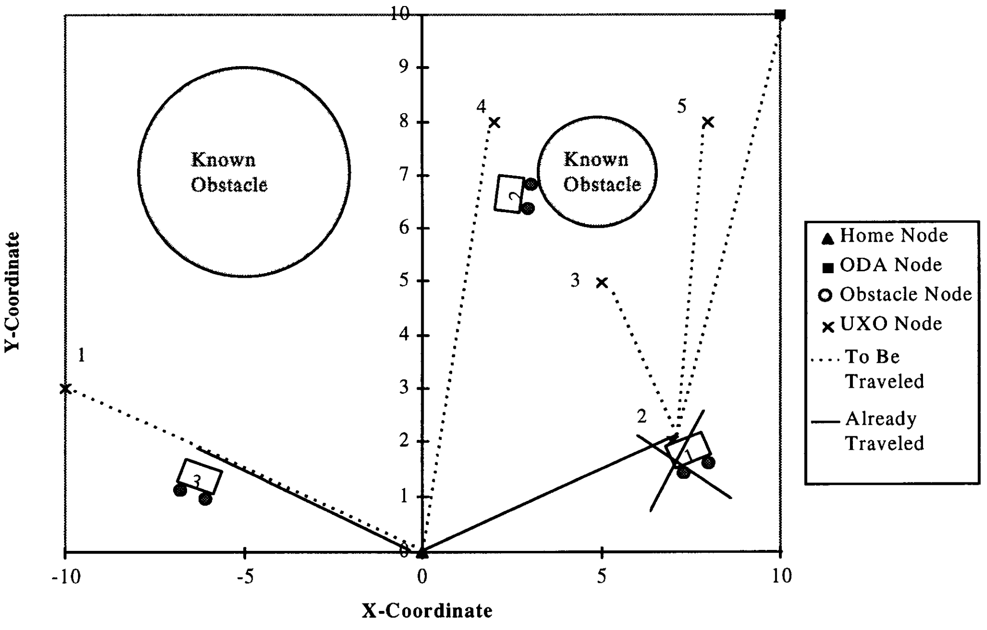


Figure 4-12: Re-Plan Using the Single-Goal Re-Planner

Figure 4-11 shows the re-planned road using the multiple-goal re-planner. Node 1 and 4 have been used as the home nodes in this re-plan. Figure 4-12 shows the re-planned road segments using the single-goal re-planner. This re-planner simply created 5 separate segments that are all originating from the home node. Therefore, this becomes inefficient since the three original routes become 5 shorter routes. Therefore, the vehicles will have to travel a lot further than they originally were required to do. The solid lines still represent the verified road segments for both graphical representations.

STEP 10 and *STEP 12* are included in the verification mission manager procedure rather than the mission re-planner. Thus, only *STEP 8* and *9* fall under the mission re-planner procedure.

STEP 10: Repeat STEP 6 through 9 until either all of the roadways have been verified to be obstacle-free or no more road plan is available for the remaining uncovered UXO nodes.

The verification manager and the re-planner may iterate several times depending on the complexity of the mission. Theoretically, unless an UXO node is completely surrounded by obstacles, the vehicle should be able to visit any of the excluded nodes with the single-goal re-planner.

STEP 11 (optional): Command all the vehicles to travel back to the home node.

At the end of the mission, the manager should bring all the vehicles back to the home node. This will get the vehicles ready for the next mission.

STEP 12: Output a summary of the road-building mission.

At the end of the mission, the manager should generate a summary of the mission that contains the various performance data. This can be used to evaluate the performance of any particular road-building mission. Figure 4-13 shows a typical summary of a road-building mission. Furthermore, the manager should create an input file for the UXO-clearing manager to prosecute the nodes. This input file contains the connectivity information (i.e. the obstacle-free paths), *x* and *y* coordinates, and the types of the nodes (Table 4-4).

```
*******************************
Road-Building Mission Summary
*******************************
Total Nodes:     9
UXO Nodes:       5
ODA Nodes:       1
Obstacle Nodes:  2
Home Node:       1
Work Space:      10x10 (ft^2)
# of Vehicles:   3
Total Distance:  54.64 (ft)
Total Time:      54.640854 s
```

Figure 4-13: A Sample Summary of Road-Building Mission

Table 4-4: A Sample Input File for the UXO-Clearing Planner

From/To	0	1	2	3	4	5	6	7	8	X-Coordinate	Y-Coordinate	Type
0	0	0	0	0	1	0	0	0	0	0	0	-4
1	0	0	1	0	0	0	0	0	0	-10	3	-1
2	0	1	0	1	0	0	0	0	0	7	2	-1
3	0	0	1	0	0	1	0	0	0	5	5	-1
4	1	0	0	0	0	0	0	0	1	2	8	-1
5	0	0	0	1	0	0	0	0	1	8	8	-1
6	0	0	0	0	0	0	0	0	0	-5	5	2
7	0	0	0	0	0	0	0	0	0	5	7	1
8	0	0	0	0	1	1	0	0	0	10	10	-3

4.3 Selection Criteria for Road Planning Algorithm

The initial planner calls a road planning function to compute the initial roadways for any number of vehicles. Many different road planning algorithms exist for the MVRP. Thus, the initial planner can call a few different road planning algorithms for a given mission. These calculated initial plans can be compared to select the best plan for the mission before the plan verification phase .

Choosing the best road planning algorithm for a mission requires a set of rules to make the comparisons. The most important criterion is the variation in the distributed workload of the multiple vehicles. If the vehicles can complete their task almost simultaneously, then the manager would be less likely to run into an event of a vehicle being idle for a period of time.

Another important criterion to choose the best planning algorithm is the shortest distance. A road planning algorithm that yields the least total travel distance for the vehicles is the prime candidate for the planner to use to generate an initial plan. Therefore, the road planning algorithm that provides an evenly distributed workload among the vehicles and the minimum total distance of the routes should be selected to create an initial plan. Many other criteria exist for selecting a road planning algorithm to generate initial roadways for the multiple vehicles.

4.4 The Order of Managing Events

Although multiple events may happen simultaneously, the manager can only consider one event at a time. Therefore, the order of managing events needs to be established for the efficiency of the mission manager. Some general guidelines for the order of managing events that the road-building mission manager should follow are listed in the following.

1) Respond to the event that arrived first (i.e. "first come first serve").
2) Respond to the event that requires plan modification.
3) Respond to the event that requires the re-planning.

4.5 Selection Criteria for Re-Planner

As mentioned in the previous section, two different aspects of the re-planning exist for the road-building mission. The first aspect is equivalent to the initial planner (i.e. route planning without using the connected paths but updated environmental data). This corresponds to the multiple-goal re-planner. The second aspect is the re-planner that connects the excluded nodes using the connected paths. This corresponds to the single-goal re-planner. The following are a possible set of rules that the mission manager can use to select the type of re-planner to use.

1) If few excluded nodes exist, use the re-planner that uses the connected paths (i.e. the single-goal re-planner).
2) If many excluded nodes exist, use the re-planner that creates a new road plan equivalent to the initial plan (i.e. multiple-goal re-planner).

Chapter Five : Mission Manager Implementation

The implementation of a mission manager for the EOD multiple vehicle system considers the issues of the initial road planning, re-planning, managing data, and simulated testing. The mission planning and control software is implemented in C in the Linux operating system on a Pentium computer. The simulator has been implemented in the same way as the mission planning and managing software and simulated tests run on the same computer as well.

5.1 Mission Manager Software

An overview of the three main software modules of the mission manager and other related software modules is given in Figure 5-1. The three main modules are the initial mission planner, the verification mission manager, and the mission re-planner. The main program of the mission manager initiates the three major modules and controls the program operation. Figure 5-1 shows the three main modules and the data flow throughout the mission managing process.

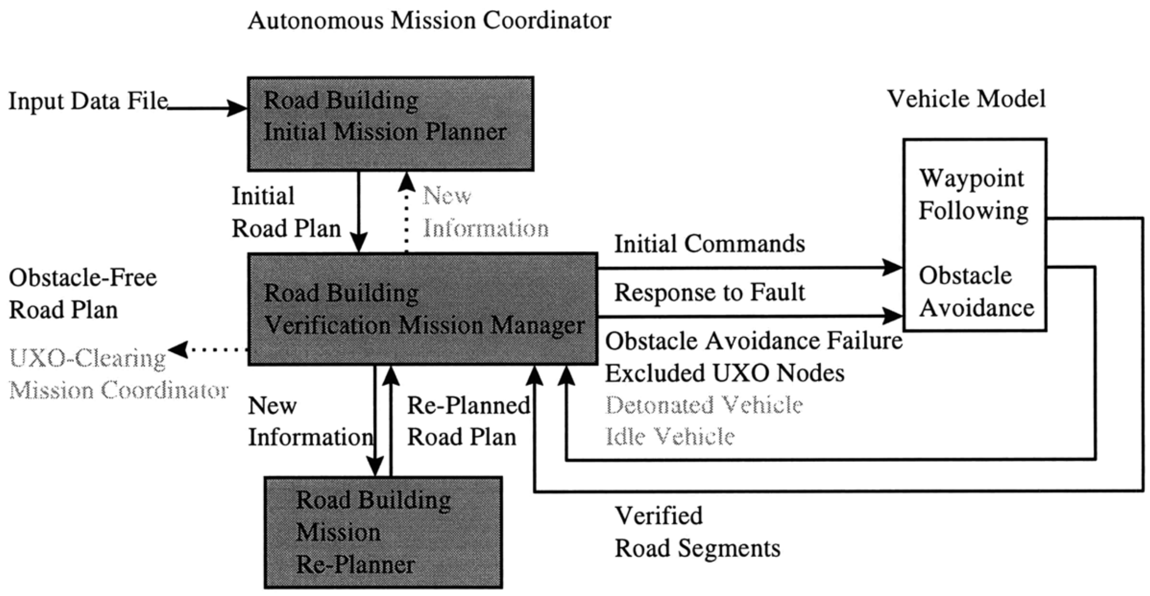


Figure 5-1 Interactions between Mission Coordinating System and System Software

The initial mission planner reads an input data file that contains the locations of the UXO nodes, home node, ODA node, and known obstacle nodes. From the input data, the planner computes an initial road plan for a given mission using a road-planning algorithm. Refer to section 5.2 for more detail on the implementation of the initial mission planner. The verification mission manager uses the initial road plan to generate waypoint following commands to the vehicles. Refer to section 5.3 for more detail on the implementation of the verification mission manager. The vehicles execute the command and report the verified road segments and detected events to the manager. The various events are listed in the Figure 5-1. However, note that the manager's responses to only the obstacle avoidance failure and excluded UXO have been implemented. Although the concept of the manager's response to a detonated or idle vehicle has been described, it has not been implemented to obtain simulated testing. Thus, the shaded texts in the Figure 5-1 represent the software modules that either have not been implemented or are not part of the road-building mission manager software (i.e. UXO-clearing mission coordinator).

If the vehicles verify the initial road segments to be obstacle-free roads, then the road-building mission is done. However, suppose the vehicles run into many unknown obstacles in their assigned paths. Then, the mission manager will have to do plan modification, re-planning, or both depending on the effectiveness of the obstacle-avoidance maneuver. Furthermore, the verified road plan will differ from the initial plan due to the added nodes from the obstacle detection and avoidance. If the manager finds any excluded UXO node at the end of the plan verification process, then it calls the mission re-planner. Refer to section 5.4 for more detail on the mission re-planner. This planner reads an updated (i.e. new information) map that contains the excluded UXO nodes, connected nodes, and detected unknown obstacle nodes and computes a new road plan. Thus, the mission managing process starts all over again. This process is repeated until either all the excluded nodes have been covered or the possible paths that will lead to the excluded nodes have been exhausted.

If the manager at any point in the verification phase decides to do re-planning, then the multiple-goal re-planner is called to re-plan the entire network with more detailed environmental information. However, the multiple-goal re-planner has not been implemented to test its functionality. The re-planner that takes care of the few excluded UXO nodes has been implemented to satisfy the road-building mission. Once the re-planned routes are calculated, the mission managing process starts all over again. The manager stores the final verified road segments in an output file that will be passed to the UXO-clearing mission manager.

5.2 Initial mission planner Architecture

An input data file for the EOD road-building mission contains the *x* and *y* coordinates and the types of the nodes (i.e. a graph). Any road planning algorithm needs this known environmental information to be stored in an arrays for computational purpose. This graph is implemented using an *adjacency list*. An adjacency list uses an array of memory address pointers to linked-lists containing the structure of the various nodes. Three elements are defined in the structure of any node: the locations of the node in terms of the *x* and *y* distances from an origin and the type of the node. Table 5-1 lists the various types of nodes and their definitions. Note that any positive number for node-type indicates an obstacle node. An obstacle node is represented as a circle with the positive number being the radius of the circle. The other nodes are represented as a point on a graph (i.e. no defined size).

Table 5-1: Node Type Definition in an Input Data File

Node Type	Definition	Symbol or Size
-1	UXO Node	"x" - No Size
-4	Home Node	"Δ" - No Size
-2	ODA Node	"■" - No Size
> 0	Obstacle Node	Circle with Defined Radius

The initial mission planner reads the input file *known.sim* that contains the information for UXO (locations), predefined obstacles (locations and sizes), and home nodes. The number of available vehicles for the mission is defined by the user on-line. From the input file the mission planner decides whether the mission is too large. If the mission is too large, then the *partitioning* function is called to divide the network into smaller regions. The planner chooses the first *N* nodes that are closest to the home node to be the first region. Then, one of the road planning algorithms is applied to each of the regions. Then, the mission planner checks if multiple home nodes are defined in the input file. If so, the planner calls the *dividing* function to assign nodes to appropriate home node. Then, one of the road planning algorithms is applied to the nodes associated with a single home node. Otherwise, the planner applies one of the road planning algorithms to the problem directly. However, both the partitioning and the dividing functions have not been implemented to test their functionality in the road-building mission.

Four different combinations of the routing and distributing algorithms exist so that they can be compared by running different tests on them. The combined algorithm of mixed routing and SFC distributing is consider to be the most optimal and effective algorithm. On the other hand, the combined algorithm of greedy routing and winner distributing is assumed to be the worst case of the four combined algorithms. The other two combined algorithms should fall somewhere between these two. The two extremes of the combined algorithms have been implemented to compare the test results.

The mission planner can call any of the algorithms to generate an initial plan. The planner saves an initial road plan for a problem to an output file, which will be used as an input file to the verification manager program. Figure 5-2 shows a typical mission planner flow diagram. Note that the partitioning, dividing, and two of the combined functions have not been implemented (i.e. the shaded functions in the Figure 5-2).

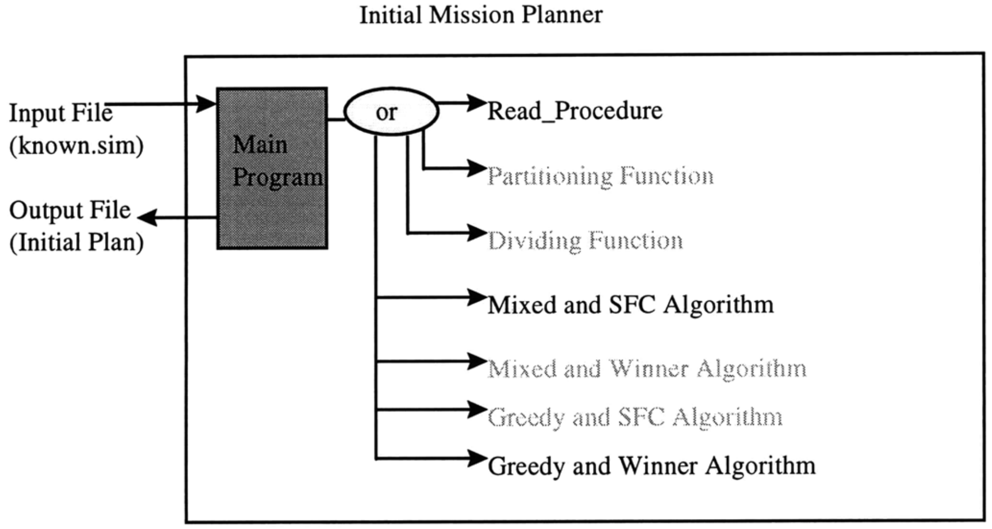


Figure 5-2: Initial mission planner Flow Diagram

5.2.1 Main Program

The main program calls the *read_procedure* to read an input file (*known.sim*) that contains various environmental information (i.e. mainly obstacles and UXO). Depending on the conditions, one of the following functions will be called by the main program:

- If too many UXO nodes, then call *partitioning* function.
- If multiple home nodes, then call *dividing* function.
- Otherwise, call one of the two route planning algorithm functions.

Once an initial road plan is generated and returned to the main program, it is saved into an output file. This output file will be used as an input file for the verification mission manager.

5.2.2 Route Planning Algorithm Programs

The route planning algorithm programs call the *connectivity* function that provides the node connectivity and predefined obstacle information. In other words, this function reads through the input file and creates an obstacle map (i.e. locations and sizes) and connectivity map (i.e. whether any path between two UXO nodes is blocked from a known obstacle).

The first combined algorithm is the combination of the mixed routing and the SFC distributing algorithms. This algorithm should yield the most optimal solution, since the mixed routing (i.e. mixture of the optimal and sub-optimal solutions) is used. Furthermore, the SFC distributing algorithm doesn't exclude any node in grouping the nodes into clusters. The second combined algorithm is the combination of the mixed routing and the "winner" distributing algorithms. This should be less effective than the first combined algorithm. The winner algorithm tends to exclude several nodes from the groups of nodes depending on the known obstacle map during the planning stage. Although the mixed routing algorithm should result in optimal routes, the winner distributing algorithm doesn't guarantee to include all the nodes.

The third combined algorithm is the combination of the greedy routing and the SFC distributing algorithms. This should still be less effective than the first combined algorithm due to the limitation of the greedy routing algorithm. Although the SFC distributing algorithm groups all the nodes, the greedy routing algorithm tends to exclude some nodes due to the known obstacles. The greedy algorithm starts from an arbitrary node and connects it to the nearest reachable node. If a node is not reachable, then it is simply skipped until it can be reached from a connected node. If the skipped node can't be reached from any connected node, then this node is stored in the exclude-node file. The fourth combined algorithm is the greedy routing and "winner" distributing algorithm. This may be the least effective algorithm in creating an initial road plan. Both the routing and distributing algorithms are based on the greedy concept. The greedy concept considers the immediate optimization rather than the overall optimization.

Once the connectivity map is created, either the *SFC* or *winner distributing* functions are called to group the nodes into clusters. Then, either the *mixed routing* or *greedy routing* function is called to create a single road plan. The *mixed routing* program calls the *minimum spanning tree* function. Then, the *odd-degree connecting* function is called to pair the odd-degree nodes. A last function is called if any repeated node exists in the initial road plan. This function removes any repeated node in the plan if the final road is still feasible. Otherwise, the route planning program exits with the initial road plan written to an output file. However, the *greedy routing* program runs by itself and exits with the initial road plan saved to an output file.

5.3 Verification Mission Manager Architecture

The mission manager reads an input file (i.e. the output file of the initial mission planner) to generate waypoint following commands to the vehicles. The EOD mission manager responses to the events that occur during the verification phase. Depending on the events, the verification manager decides to do either plan modification or re-planning to complete the given mission. Once the mission is complete, the mission manager stores the verified road plan to an output file. This output file will eventually be used by the UXO-clearing mission manager. Figure 4-3 shows a typical verification mission manager.

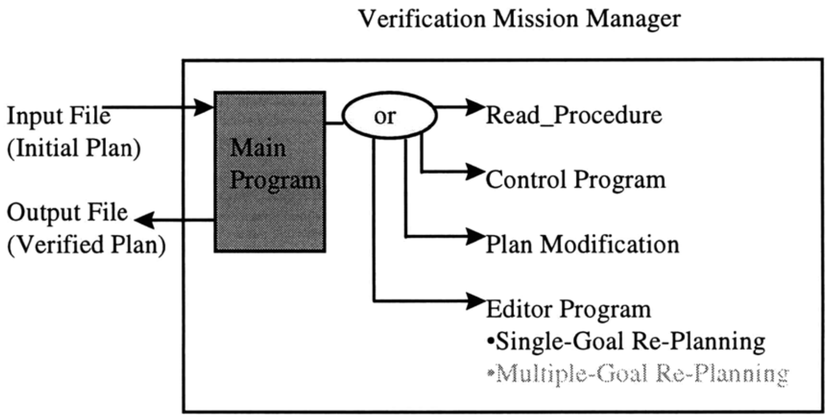

Verification Mission Manager
Input File (Initial Plan)
Output File (Verified Plan)
Main Program
or
Read_Procedure
Control Program
Plan Modification
Editor Program
•Single-Goal Re-Planning
•Multiple-Goal Re-Planning

Figure 5-3: Verification Mission Manager Flow Diagram

5.3.1 Main Program

The main program calls the *read_procedure* to read the input file, which is the output file generated by the initial mission planner. Then, the main program calls the *control* program to generate waypoint commands using the road plan from the input file. Depending on the events during the verification process, the main program calls different functions to respond to the events. If a vehicle runs into an obstacle, the *obstacle avoidance* is automatically performed by the vehicle. If the obstacle-avoidance maneuver fails, then the vehicle by-passes the excluded node and goes to the next assigned node. The main program stores the excluded node, the verified road segments, and detected obstacle information at the end of the plan verification process.

Then the main program calls the *editor* procedure with a specific re-planning method (i.e. single-goal or multiple-goal re-planner) if any excluded node exists. The main program calls the *single-goal re-planning* function to cover few excluded nodes. If a vehicle completes its task earlier than other vehicles, then the main program also calls the *editor* procedure (i.e. usually the *multiple-goal re-planning* function) in the same manner. Likewise, if a vehicle is detonated then the main program calls the *editor* routine. The re-planning process is repeated until the mission is done or the possible re-plan has been exhausted. Once the iterative re-planning process is done, the manager outputs an EOD road-building mission summary.

5.3.2 Control Program

The mission manager calls the control function to actually generate the waypoint following commands for the various available vehicles. The waypoint commands are derived from the input file, which contains locations of UXO nodes. Each of the waypoint commands include *begin* and *end* point, *speed* (usually 1 fps), *offset* bound, and *finish* bound. Two different speeds will be used during the verification phase. For instance, if a vehicle needs to explore an area with uncertainty then the vehicle is commanded to travel at a slow speed (i.e. around 1 fps). However, if a vehicle needs to travel through a verified road then the vehicle is commanded to travel at a fast speed (i.e. around 6 fps). The offset bound indicates the distance that the vehicle can go off before the correction must be done by the controller. Likewise, the finish bound indicates the distance that the vehicle can miss before the vehicle considers itself to be done with the task. The mission manager calls the control function to issue other commands such as stop the currently assigned task, and others.

5.3.3 Editor Program

The editor program is basically the re-planning function. In the re-planning function, the verification manager tries to respond to an event without affecting the other vehicles. For instance, if an obstacle avoidance has failed, then the manager will command the obstructed vehicle to go to the next node in the assigned path if possible. Once the vehicles cover their last reachable node, then the manager checks the excluded node file. The manager calls the *single-goal re-planning* function to take care of the excluded nodes (if any). If few nodes have been excluded, then the single-goal re-planning function tries to connect each excluded node to the connected node using the verified paths. However, if too many nodes have been excluded, then the manager calls the *multiple-goal re-planning* function that acts like an initial planner. In other words, this function creates a new road plan using the new environmental information rather than connecting one excluded node at a time. This particular re-planning function has not been implemented.

Likewise, if a vehicle completes its assigned road, then the mission manager tries to do simple plan modification. In this case, the managing function may assign the excluded UXO nodes (if any) to the idle vehicle. If the idle vehicle is not able to cover any of the excluded UXO nodes, then the mission manager may either allow that vehicle remain idle or call the multiple-goal re-planning function that creates a new road plan.

If a vehicle is detonated, then the mission manager calls the editor program to either do plan modification or re-planning. If any additional vehicle is available at the home node, then the editor assigns the rest of the detonated vehicle's road to the new vehicle. However, if no additional vehicle is available, then the editor program calls the multiple-goal re-planning function. The re-planning function will eventually generate a new road plan with the updated environmental map.

5.4 Road-Building Simulation

Although two prototypes of the EOD vehicle have been built, they aren't developed to the point that they can be integrated into the high-level software of the mission manager. Furthermore, only two vehicles have been built so no more than two physical vehicles are available to form a multiple vehicles system. Therefore, no field testing was performed on the current system to verify the feasibility of the mission coordination software. However, particular case studies have been simulated to support and verify the feasibility of the mission manager software. Furthermore, test programs have been implemented to generate random test cases for the simulation of the multiple vehicle system to verify the feasibility and efficiency of the mission manager software. Chapter 6 explains in detail a evaluation of the mission manager software. In Chapter 6, the mission planning and control software is used to plan a simulated EOD multiple vehicle mission.

5.4.1 The Implementation Environment

The simulation is divided into the environment, obstacle mapping and avoidance, and road planning/re-planning and verification. The mission planning and managing software interfaces with the simulated EOD vehicles through the various simulations listed above. Each of the simulations feeds the mission manager software information and data that are comparable to the real tested performance of the ground vehicle in a real mine field. Furthermore, simulated tests can verify the feasibility of a mission manager by providing both the time to create an initial plan and time to complete the verification of the initial plan including the re-planning process [6]. Thus, the simulation acts as a quantitative analysis tool in addition to being a qualitative observer [13].

Once the vehicles start to execute the waypoint following tasks, the manager initiates the "clock". The EOD road-building mission is based on an event-paced simulations. In other words, the clock is advanced only to those instants in time when certain important *events* take place. Some of the important events include a vehicle bumping into an unknown obstacle during the verification process or a vehicle completing its waypoint following task much earlier than the other vehicles. In event-paced simulations, time is advanced in irregular intervals.

5.4.1.1 Environment Simulation

In the road-building phase, the mission manager will only be concerned about the geometric hazards[16]. Thus, the environment simulation provides information related to the obstacles, UXO, and depots to the other simulations such as the vehicle dynamics simulation. The simulated environment information that is in the *unknown.sim* input file can be changed or replaced by an operator or user to plan for different mission. However, the input file can't be modified or changed on-line. The input file also contains information about the probability of an obstacle avoidance success for the given obstacle[17]. The terrain is represented by a coordinate system with grid. Therefore, all of the simulated objects are in terms of x, y, and heading angle with respect to the coordinated system [6].

To simplify the problem, obstacles are generally displayed as circles with defined radii. If a rectangular obstacle needs to be displayed, then a cluster of circles with defined radii can be used. The obstacles can be positioned, scaled, and oriented by the user in any part of the environment. UXO are usually represented by an "x" and can be positioned anywhere in the environment. UXO are modeled to be same size so they can't be scaled or oriented for different UXO. Likewise, the ODAs and depots are usually represented by a point (a square object) and can also be located in any part of the environment. The home node is usually represented by a point (a triangular object) and located at the origin (i.e. (0, 0)). The ODA and home node models can't be scaled or oriented for different ODAs and homes.

5.4.1.2 Obstacle Mapping and Avoidance Simulation

The ground station is required to store geographical information. The geographical locations of the obstacles, excluded UXO, and the planned nodes are represented in map stored in computer's memory. This stored map includes the predefined and detected obstacles' locations and sizes. Likewise, the map includes the locations of visited UXO and planned UXO.

The mission manager uses the map data to plan feasible paths around any obstacles that have been detected. A typical obstacle avoidance algorithm creates another goal position slightly beyond the point where the vehicle bumped into an obstacle. Then, the vehicle tries to visit the originally assigned UXO node. Thus, the manager adds the two new points (i.e. the point where the vehicle has bumped into an obstacle and the point where the vehicle pivots to avoid the obstacle and resume its progress to the next node) to the verified road map. The mission manager keeps track of the added nodes due to the obstacle-avoidance maneuver.

[16] Other non-geometric hazards such as extremely weak surfaces do exist in a typical mine field.

[17] This thesis doesn't deal with the details of the complex obstacle avoidance. Therefore, for the testing purpose only the probability of the obstacle avoidance success is considered rather than the actual obstacle avoidance procedure.

5.4.2 The Testing Program

A test program has been implemented to generate random cases of the MVRP problem. The test program reads input information given by the user on-line. This program requires the user to input the number of total nodes and of this total nodes the number of UXO, home, and ODA nodes. It also requests the area of the network and the distances relative to each of the nodes. The program uses the provided information to generate random *x* and *y* coordinates and assigns the type of the nodes to the randomly coordinated nodes. These randomly generated coordinates are stored into an output file that can be accessed by any other program.

Three different test files are required to carry out the simulated tests of the mission manager. The first test file is referred to as *known.sim*. This file contains the known environmental information of a problem. For instance, the file will contain randomly generated *x* and *y* coordinates and the types of the nodes that are known *a priori*. These nodes generally include the UXO nodes to be visited, some predefined obstacles, the home node (service station), and ODA to dispose the UXO.

The second test file is referred to as *unknown.sim*. The environmental information in this test file is known only by the simulator (totally unknown by the vehicles or the manager). The structure of the unknown test file is similar to the known test file. However, the unknown test file contains the unknown obstacles. The probability of an obstacle avoidance success for a certain obstacle is defined within the unknown test file. The probability of an obstacle avoidance success is used to simulate the likelihood of a vehicle skipping a node. This simulation will show how the manager reacts to the situations where an obstacle avoidance fails. The manager will either do plan modification or re-planning.

The third test file is referred to as *test.sim*. This test file includes both the known and unknown test files and other things such as the number of vehicles. Lastly, the user requests the manager to store the verified final road plan to an output file (*final.out*). This output file will be used by the UXO-clearing mission manager to plan the clearing process. The test program also stores the various simulated times involved during the mission. For instance, the simulated time to complete the mission planning is stored into another output file (*statistics.out*). The simulated time to complete an obstacle avoidance, the re-planning process, and travel to the assigned nodes should be stored in the same output file for the statistical analysis.

5.4.3 Road Planning/Re-Planning and Verification Simulation

The initial mission planner or re-planner doesn't require any simulations to test the implemented algorithms. The initial planner or re-planner program runs to compute an initial or re-planned road plan given a problem. Thus, no simulated testing is required for the initial planner or re-planner. However, the verification mission manager must run through simulated testing to verify the feasibility of the managing algorithm. During the verification process, the manager must interact with the vehicles to complete the mission. For instance, suppose the manager commands the vehicles to follow waypoints. Furthermore, consider a vehicle running into an obstacle in the assigned path during the execution of the waypoints following task. Before the vehicle reports to the manager that it has bumped into an obstacle, it automatically performs an obstacle-avoidance maneuver. If the maneuver fails, then the vehicle reports to the manager and waits for a new command. The manager will receive this request and generate a new command for the waiting vehicle. Therefore, interaction is necessary during the verification process. The interaction between the manager and the vehicles is an iterative process until the mission is completed.

Chapter Six : Evaluation Tests

An illustration of a road-building mission using the developed mission manager is presented in section 6.1. This illustration supports the feasibility of the mission manager to build obstacle-free roads given an area that contains UXO. A series of randomly generated tests have been run using the developed mission manager to determine the performance and limitations in section 6.2.

6.1 An Annotated Example Test of the Mission Manager

A detailed example of a road-building mission using an autonomous mission manager is presented to demonstrate the efficiency of the manager in controlling the multiple vehicles system. Computer-generated outputs and graphical representations of the road plans for the various stages of the mission manager are given to illustrate the progression of the road-building mission. The example mission is generated using the random test-generating program. In the example test, three vehicles are considered to build obstacle-free road segments among 20 UXO nodes.

6.1.1 Initial Road Planning Stage

STEP 1: Reading an input file that contains the known environmental information.

The following shows the input data file called *known.sim* (randomly generated for the initial planner) that contains the *x* and *y* coordinates and the type of the various nodes. The first column doesn't exist in the actual *known.sim* input file. This column represents arbitrary numbers marked for each UXO node to match the numbering system on the graphical representation. The home node is always marked as 0, the UXO nodes are marked as 1 through 20 (in this case), and the known obstacle nodes don't need to be marked with numbers. The first column is added to clarify the graphical representation of the road-map. The next two columns represent the *x* and *y* coordinates in feet. The last column represents the type of the nodes. The -4 node that is at the origin (0, 0) is the home node for this mission. The -1 nodes at the various locations are the UXO nodes. The last four nodes are the known obstacles with the number given as the radius in feet.

```
 0        0.00    0.00    -4
 5       84.86   18.18    -1
10       41.01   48.75    -1
13       91.27   64.90    -1
 8       44.28   31.92    -1
11       47.59   64.75    -1
 7       36.63   23.44    -1
 9        7.93   47.64    -1
16       95.06   79.33    -1
20       62.30   95.90    -1
14       12.04   82.67    -1
17        7.61   97.78    -1
12       66.18   58.28    -1
 1        8.03   14.87    -1
19       52.12   92.89    -1
18       33.05   93.13    -1
 2       56.24    5.62    -1
 3       73.73    1.39    -1
 6       94.00    2.43    -1
15       60.71   78.85    -1
 4       75.05   12.83    -1
obst1    32.43   63.98     9
obst2    99.30   88.67     3
obst3    37.27   12.69     1
obst4    34.22   41.13     7
```

Figure 6-1 depicts the input information using the coordinates of the nodes above. The workspace is 100 feet by 100 feet and contains 20 UXO nodes (i.e. the square) that are numbered arbitrarily from 1 to 20, 4 known obstacles (i.e. the circles), and a home node (i.e. the triangle) that is numbered as 0 on the network. Three simulated vehicles are initially positioned at the home node.

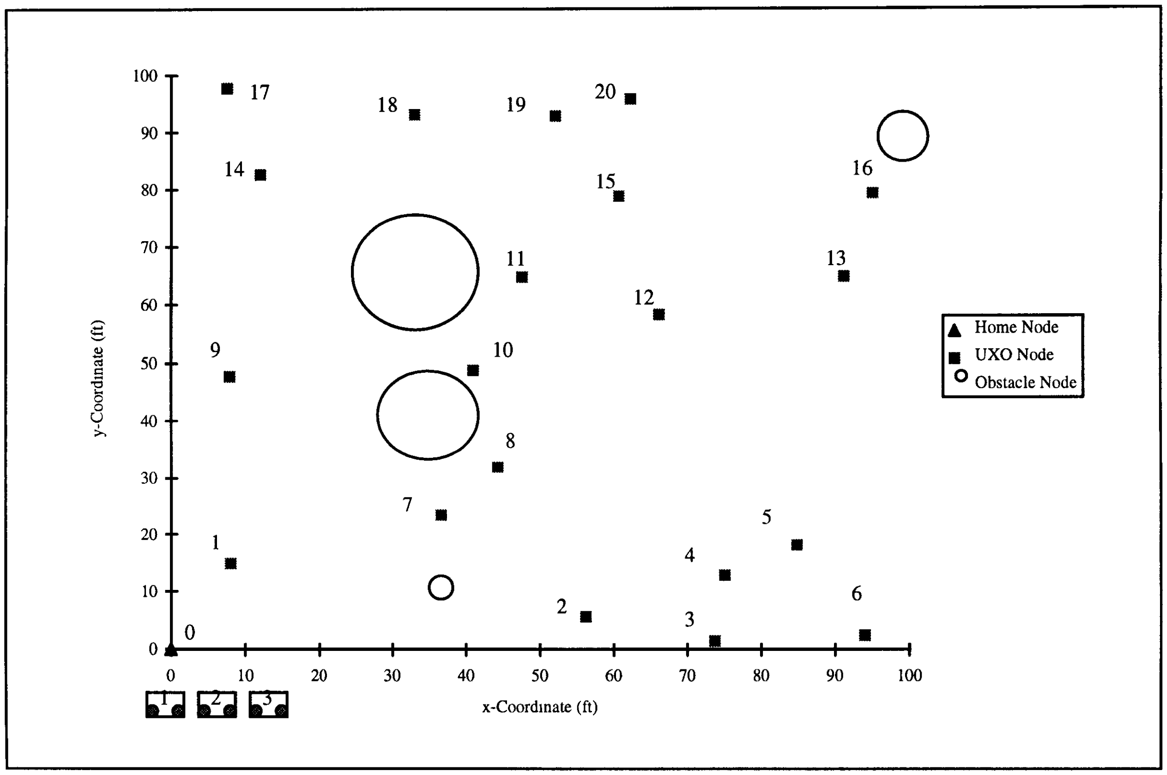


Figure 6-1: Graphical Representation of the *known.obst* Input

STEP 2: Checking whether the mission is too large (i.e. more than 100 UXO nodes).

From the *known.sim* input file, the mission manager determines that this is not a large mission. The example mission considers only 20 UXO nodes. Therefore, the *partitioning* function is skipped in this case.

STEP 3: Checking whether the mission contains multiple home nodes.

The mission manager determines that this only has one home node. Thus, the *dividing* function is skipped in this case as well. Furthermore, the *dividing* function has not been implemented so that the manager will simply ignore the multiple home nodes. Although the home node can be positioned at any point, the manager can always assume the location of the home node to be at the origin (i.e. (0, 0))[18]. Since a single home node has been assumed, the three available vehicles will all begin the road-building verification process from the origin node (i.e. home node).

STEP 4: Creating the distance and connectivity matrices using the known.obst input data file.

The distance between any two nodes is calculated using the *x* and *y* coordinates. This distance matrix is used to compute the initial road plan. The connectivity matrix shows whether a path between any two nodes is free of known obstacles. This connectivity matrix is used to select the best path for a vehicle to travel between any two nodes.

[18] This is to simplify the problem and also increase the efficiency of the SFC distributing algorithm in the initial road plan.

STEP 5: Planning an initial road-plan for the three vehicles to cover 20 UXO nodes using a road planning algorithm.

The combined algorithms are used to create an initial road-plan for the three vehicles. Note that only two combined algorithms have been implemented to compare the initial road plans. The first one that has been implemented and used to compute the initial plan is the mixed routing and the SFC distributing algorithms. The second combined algorithm that has been implemented and used is the greedy routing and "winner" distributing algorithms. The following shows the initial road plan (i.e. initial planner's output file) for the example using the mixed and the SFC algorithms. The first vehicle is assigned to three UXO nodes and the visiting sequence is {0, 9, 14, 17}. The second vehicle is assigned to twelve UXO nodes and the visiting sequence is {0, 1, 7, 8, 10, 15, 20, 16, 13, 12, 11, 19, 18}. The third vehicle is assigned to five UXO nodes and the visiting sequence is {0, 2, 3, 4, 5, 6}. The time that is listed following initial plan represents the time to compute the initial road-plan for the three vehicles. The initial planning time is not even a second for the three vehicles to cover 20 UXO nodes. Figure 6-2 presents the initial road plan for the three vehicles in the graphical representation. Note that the workload among the vehicles is not evenly distributed in this case. Vehicle 2 is assigned to cover the most distance in the workspace. The implemented road-planning program doesn't consider any optimization to improve the distribution of the workload for either combined algorithm. Although the SFC distributing algorithm does poorly dividing the workload among the three vehicles, the mixed routing algorithm optimizes individual road-plan using the minimum spanning tree concept.

```
************
Initial Plan
************

Robot  1:
        0.00,    0.00
        7.93,   47.64
       12.04,   82.67
        7.61,   97.78    Distance:    99.31 ft

Robot  2:
        0.00,    0.00
        8.03,   14.87
       36.63,   23.44
       44.28,   31.92
       41.01,   48.75
       60.71,   78.85
       62.30,   95.90
       95.06,   79.33
       91.27,   64.90
       66.18,   58.28
       47.59,   64.75
       52.12,   92.89
       33.05,   93.13    Distance:   273.26 ft

Robot  3:
        0.00,    0.00
       56.24,    5.62
       73.73,    1.39
       75.05,   12.83
       84.86,   18.18
       94.00,    2.43    Distance:   115.41 ft

******************************************** Time: 6246 µs
```

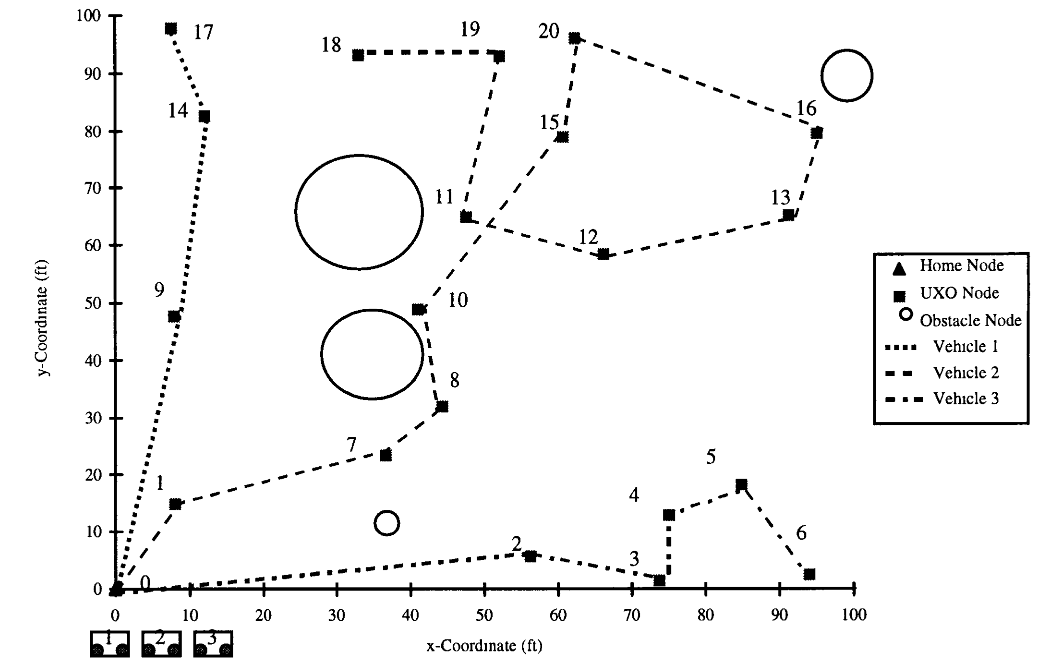


Figure 6-2: Initial Road-Plan Using the Mixed and SFC Combined Algorithm

The greedy and winner algorithm generates similar initial road-plan for this mission. This algorithm usually yields less optimal solution than the mixed and SFC algorithm. Thus, the vehicles will have to travel further during the verification process (i.e. the overall mission time using the greedy and winner algorithm is higher than the mixed and SFC algorithm). Refer to the section 6.2 for the statistical results that will show comparisons between the two implemented road planning algorithms.

6.1.2 Initial Plan Verification Stage

STEP 6: Generating the waypoint following commands for each of the three vehicles using the initial road-plan.

The following are the data structures of the waypoint following tasks that have been generated by the manager using the initial road-plan. B*egin.x*, and *begin.y* (in feet) are simply the coordinates of the starting point of a segment. Likewise, *end.x* and *end.y* (in feet) are the coordinates ending point of a segment. The *velocity* will be 1 fps if the segment hasn't been verified to be obstacle-free. Otherwise, the *velocity* will be 6 fps for the vehicle to travel to the destination. The following waypoint following tasks are generated using the initial road-plan of the mixed and the SFC combined algorithm. The three vehicles are commanded to do the waypoint following tasks listed below to initiate the verification process.

```
**********************
Waypoint Following Task
**********************

Robot  1:
Waypoint 1:       begin.x  =  0.0        begin.y =  0.0
                  end.x    =  7.93       end.y   = 47.64
                  velocity =  1
Waypoint 2:       begin.x  =  7.93       begin.y = 47.64
                  end.x    = 12.04       end.y   = 82.67
```

```
                   velocity =  1
Waypoint 3:        begin.x  = 12.04       begin.y = 82.67
                   end.x    =  7.61       end.y   = 97.78
                   velocity =  1

Robot  2:
Waypoint 1:        begin.x  =  0.0        begin.y =  0.0
                   end.x    =  8.03       end.y   = 14.87
                   velocity =  1
Waypoint 2:        begin.x  =  8.03       begin.y = 14.87
                   end.x    = 36.63       end.y   = 23.44
                   velocity =  1
Waypoint 3:        begin.x  = 36.63       begin.y = 23.44
                   end.x    = 44.28       end.y   = 31.92
                   velocity =  1
Waypoint 4:        begin.x  = 44.28       begin.y = 31.92
                   end.x    = 41.01       end.y   = 48.75
                   velocity =  1
Waypoint 5:        begin.x  = 41.01       begin.y = 48.75
                   end.x    = 60.71       end.y   = 78.85
                   velocity =  1
Waypoint 6:        begin.x  = 60.71       begin.y = 78.85
                   end.x    = 62.30       end.y   = 95.90
                   velocity =  1
Waypoint 7:        begin.x  = 62.30       begin.y = 95.90
                   end.x    = 95.06       end.y   = 79.33
                   velocity =  1
Waypoint 8:        begin.x  = 95.06       begin.y = 79.33
                   end.x    = 91.27       end.y   = 64.90
                   velocity =  1
Waypoint 9:        begin.x  = 91.27       begin.y = 64.90
                   end.x    = 66.18       end.y   = 58.28
                   velocity =  1
Waypoint 10:       begin.x  = 66.18       begin.y = 58.28
                   end.x    = 47.59       end.y   = 64.75
                   velocity =  1
Waypoint 11:       begin.x  = 47.59       begin.y = 64.75
                   end.x    = 52.12       end.y   = 92.89
                   velocity =  1
Waypoint 12:       begin.x  = 52.12       begin.y = 92.89
                   end.x    = 33.05       end.y   = 93.13
                   velocity =  1

Robot  3:
Waypoint 1:        begin.x  =  0.0        begin.y =  0.0
                   end.x    = 56.24       end.y   =  5.62
                   velocity =  1
Waypoint 2:        begin.x  = 56.24       begin.y =  5.62
                   end.x    = 73.73       end.y   =  1.39
                   velocity =  1
Waypoint 3:        begin.x  = 73.73       begin.y =  1.39
                   end.x    = 75.05       end.y   = 12.83
                   velocity =  1
Waypoint 4:        begin.x  = 75.05       begin.y = 12.83
                   end.x    = 84.86       end.y   = 18.18
                   velocity =  1
Waypoint 5:        begin.x  = 84.86       begin.y = 18.18
                   end.x    = 94.00       end.y   =  2.43
                   velocity =  1
```

Before the vehicles actually start to travel on the planned road segments, the true condition of the terrain is discussed. The simulator arranges the unknown obstacles in the workspace using the *unknown.sim* input file. The unknown obstacles are known only by the simulator. Neither the manager nor the vehicles are aware of the unknown obstacles until they get detected by the vehicles. The following shows the *unknown.sim* input file. The first two columns represent the *x* and *y* coordinates in feet. The third column represents the radius of the obstacle in feet. The last column represents the probability that a particular obstacle will be avoided by an obstacle avoidance maneuver.

```
94.57   34.88  6   0.82
80.78   62.43  1   0.79
52.98   50.38  6   0.05
83.21   49.67  8   0.16
90.80   98.96  4   0.48
 0.68   30.86  2   0.65
83.81    3.05  5   0.37
81.82   32.55  8   0.37
 1.06   59.84  2   0.98
18.88   55.31  3   0.46
72.79   88.73  7   0.06
 0.63   89.22  6   0.60
17.34   29.34  9   0.57
75.36   98.68  2   0.64
29.92   75.88  1   0.39
60.13   21.08  7   0.97
```

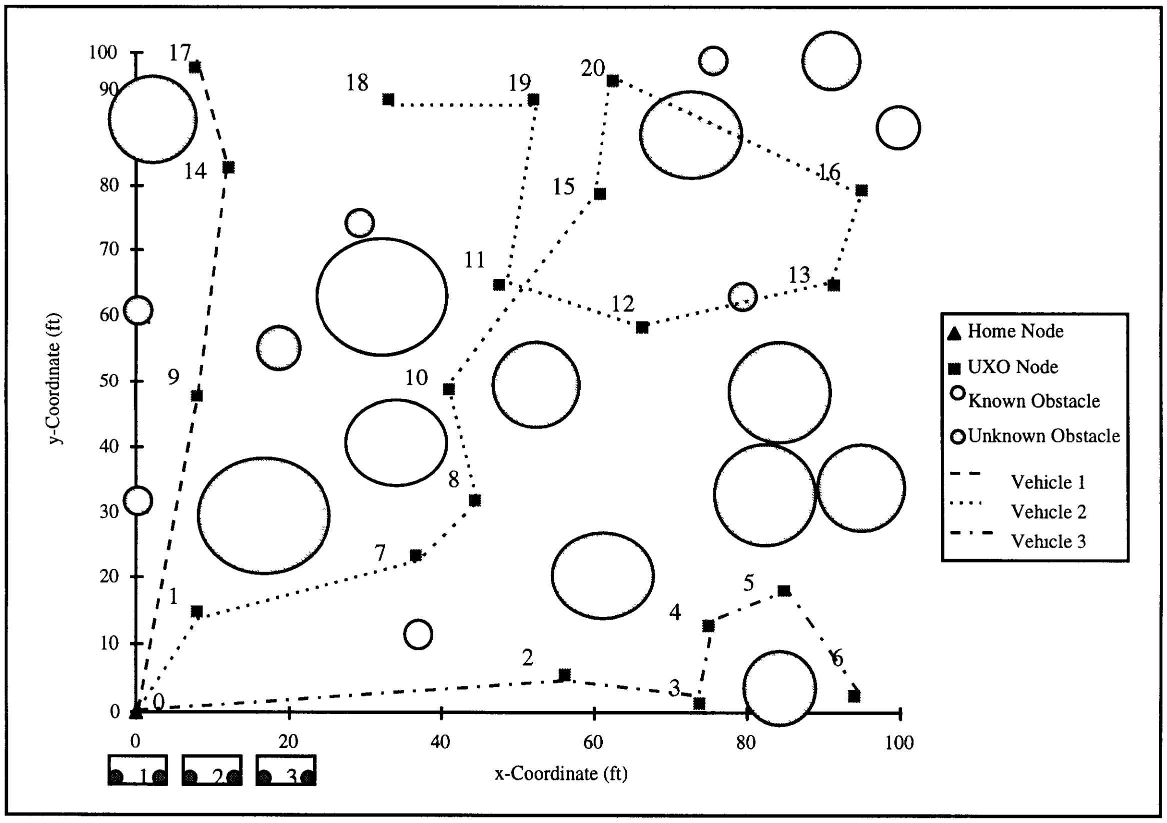


Figure 6-3: Graphical Representation of the *known.obst* and *unknown.obst* Input

Figure 6-3 shows the input information using the coordinates of the nodes above. The workspace is 100 feet by 100 feet and contains 20 UXO nodes (i.e. the square objects) that are numbered arbitrarily from 1 to 20, 4 known obstacles (i.e. the circle objects), 16 unknown obstacles (i.e. also circle objects), and a home node (i.e. the triangle object) that is numbered as 0 on the network. The white circles represent the known obstacles and the gray circles represent the unknown obstacles . Three simulated vehicles are initially positioned at the home node. Figure 6-3 also displays the initial road plan for the three vehicles that have been generated in *STEP 5*.

From the Figure 6-3, the planned segment from the node 20 to 16 is seen to be blocked by an unknown obstacle. Likewise, the planned segment from the node 13 to 12 is seen to be blocked by another unknown obstacle. Thus, at least two obstacle-avoidance maneuvers are expected to happen during the verification phase.

STEP 7: Responding to any detected event such as obstacle and completed task.

After the vehicles start exploring the initially planned road segments, the vehicles report to the manager if they have successfully visited a node (i.e. if they have verified a road segment to be obstacle-free). The manager stores the verified road segments to an output file (the compiled output file will be given at the end of the verification process). The following represents the coordinates of the nodes that the vehicles actually visited (i.e. in the order of visit) during the verification phase using the mixed and SFC algorithm. The distance that is listed is the actual distance of the path that the vehicles explored to cover as many assigned nodes as possible. The coordinates of the excluded nodes have been listed as well. The time that is listed is the time from the start of the verification process to the completion of the exploration of the actual path with the most distance. The time for this case is shown as 265 seconds. Using the mixed and SFC algorithm, vehicle 2 has taken the most time to complete its task. The verified road segments also include additional nodes if the vehicle has attempted any obstacle-avoidance maneuver to avoid detected obstacle in its assigned path. These extra nodes (generated by an obstacle-avoidance maneuver) are printed in the boldface in the following output.

```
***********
Actual Path
***********

Robot  1:
        0.00,     0.00
        7.93,    47.64
       12.04,    82.67
        7.61,    97.78     Distance:     99.31 ft

Robot  2:
        0.00,     0.00
        8.03,    14.87
       36.63,    23.44
       44.28,    31.92
       41.01,    48.75
       60.71,    78.85
       62.30,    95.90
       61.23,    96.44
       66.52,    76.33
       95.06,    79.33
       91.27,    64.90
       82.75,    62.65
       82.74,    62.65
       52.12,    92.89
       33.05,    93.13     Distance:    264.97 ft

Robot  3:
        0.00,     0.00
       56.24,     5.62
       73.73,     1.39
       75.05,    12.83
       84.86,    18.18
```

```
      94.00,     2.43     Distance:    115.41 ft

***************************************** Time: 264.976246 s
Excluded Nodes
**************

      66.18,    58.28
      47.59,    64.75
```

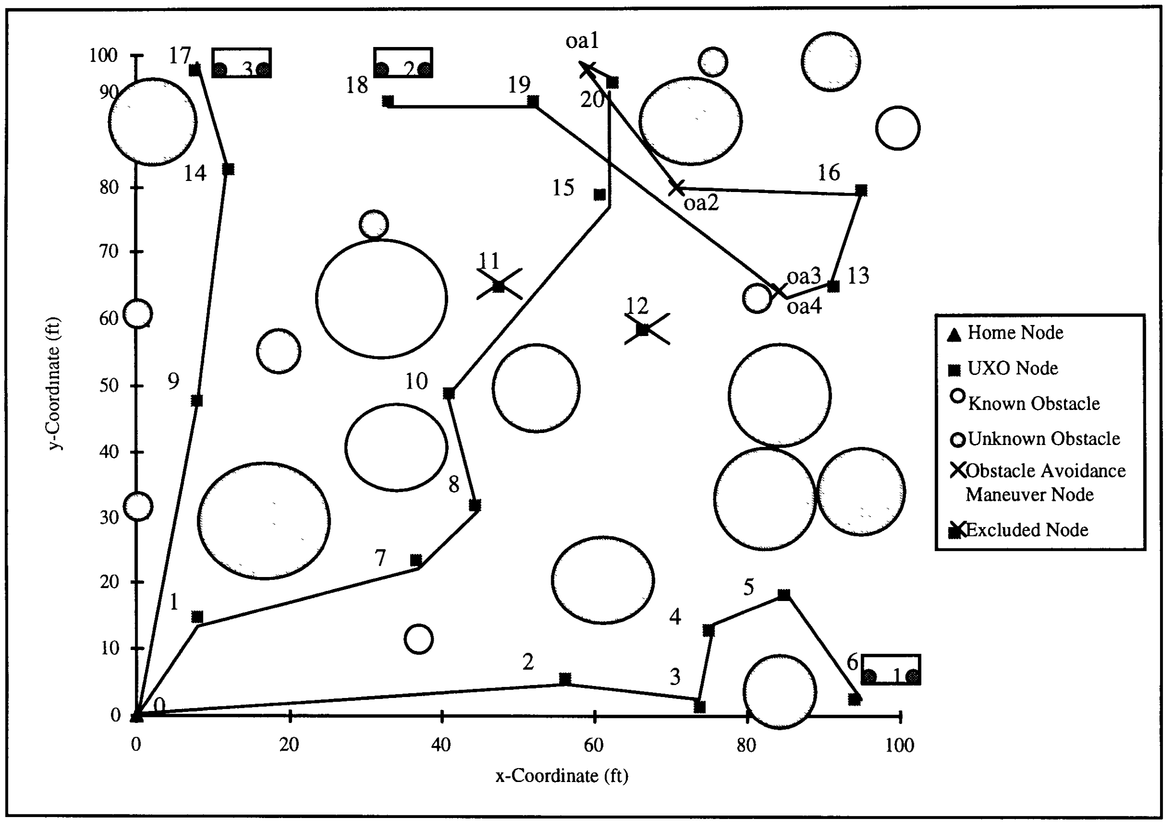


Figure 6-4: Graphical Representation of the Actual Path Using the Mixed and SFC Algorithm

Vehicle 1 and 3 successfully visit all the initially assigned nodes without any modification or re-plan. Vehicle 2 has completed its assigned task with some modifications in the initial plan. Four extra nodes denoted by "oa1" through "oa4" have been added to the verified road segments due to two obstacle-avoidance maneuvers (this will be discussed in detail later). Note that the node oa3 and oa4 are right next to each other so that it looks like only one node exists in the map. Furthermore, vehicle 2 fails to visit two nodes due to an unknown obstacle in the planned path. Thus, two nodes have been excluded from the initial road-plan. The manager will call the single-goal re-planner to create a new plan to cover the two excluded nodes (this will be discussed in detail later). Figure 6-4 displays the graphical representation of the actual path that the vehicles traveled to cover all the nodes except the two excluded nodes. Thus, this graphical representation depicts the actual obstacle-free roadways that the vehicles have built with some modifications but no re-planning. The re-planning will be shown later in the section.

Dynamic Event-Driven Management

The first event that happens during the verification phase is that the vehicle 1 completes its task after 99 seconds. The vehicle reports to the manager that it has finished the task and requests the next command. The manager needs to either modify the current plan to provide more tasks for the idle vehicle or call the multiple-goal re-planner to provide a new road plan for all the vehicles. However, neither the modification nor the multiple-goal re-planning functions have been implemented. Thus, the manager simply ignores the idle vehicle. The second event that happens is that vehicle 3 completes its task after 115 seconds.

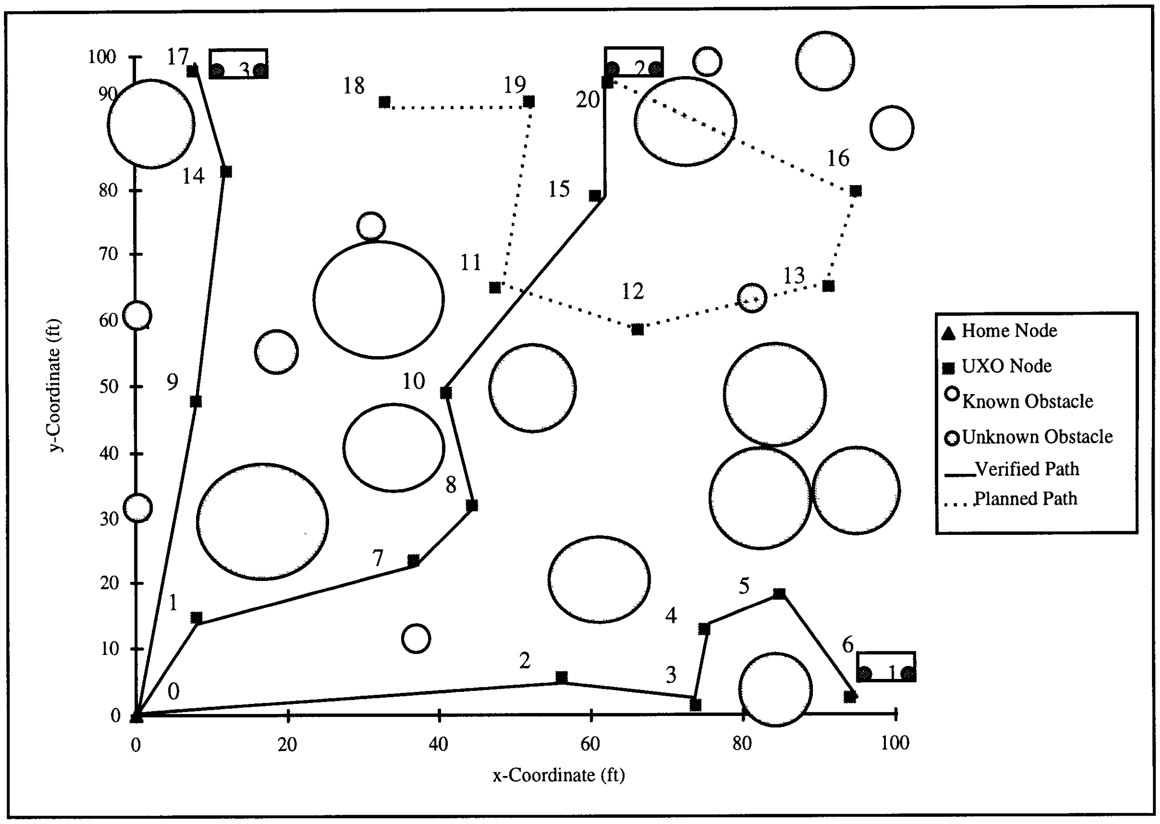


Figure 6-5: Vehicle 2 Blocked by an Unknown Obstacle

The next event is that vehicle 2 runs into an unknown obstacle at 128 seconds (Figure 6-5). However, the vehicle doesn't report to the manager that it has detected an obstacle. Instead, it automatically executes an obstacle-avoidance maneuver to get around the detected obstacle to the destination (i.e. the assigned node 16). Thus, it tries to avoid the obstacle by backing out few feet and turning towards right side of the obstacle.

Figure 6-6 shows the successful obstacle-avoidance between nodes 20 and 16 (Figure 6-6). Two new nodes created by the obstacle-avoidance maneuver have been added to the verified road plan. Thus, the vehicle travels an extra 14 feet to visit node 16 from node 20 around the detected obstacle in the planned path. Since the vehicle travels at 1 fps, the mission time is delayed 14 seconds due to the obstacle-avoidance maneuver.

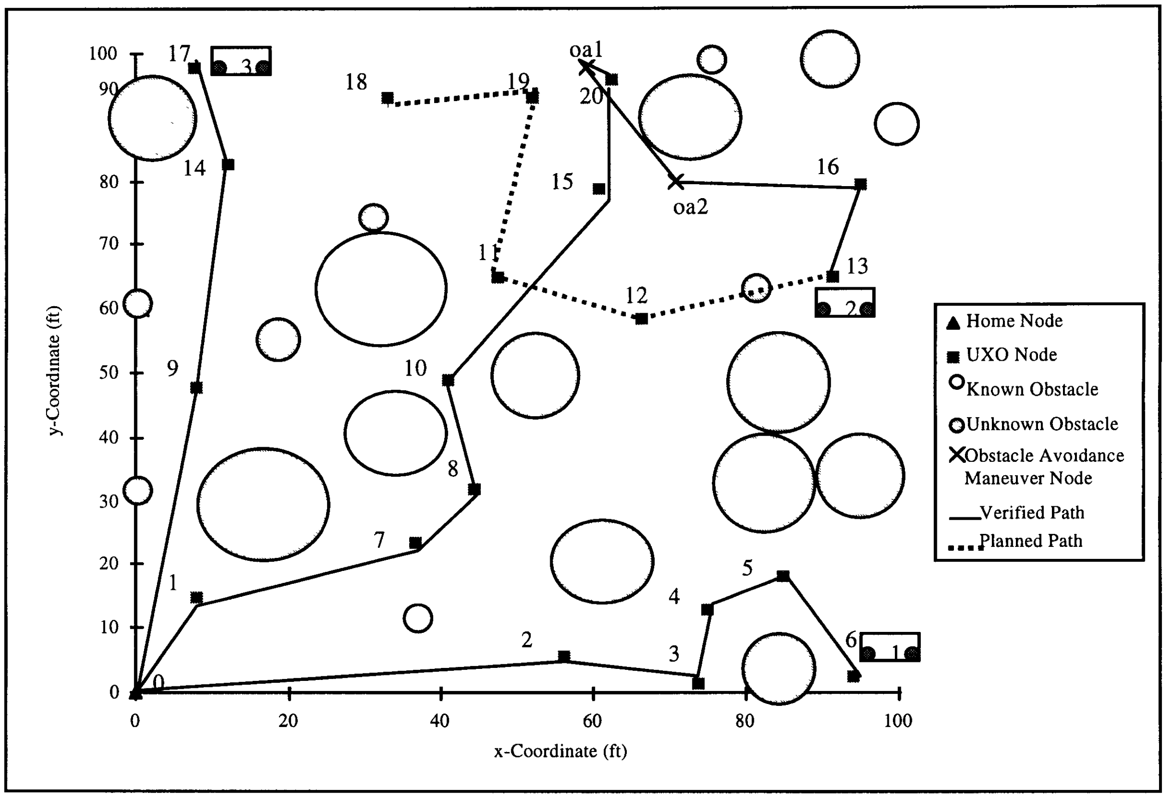


Figure 6-6: A Successful Obstacle-Avoidance Maneuver

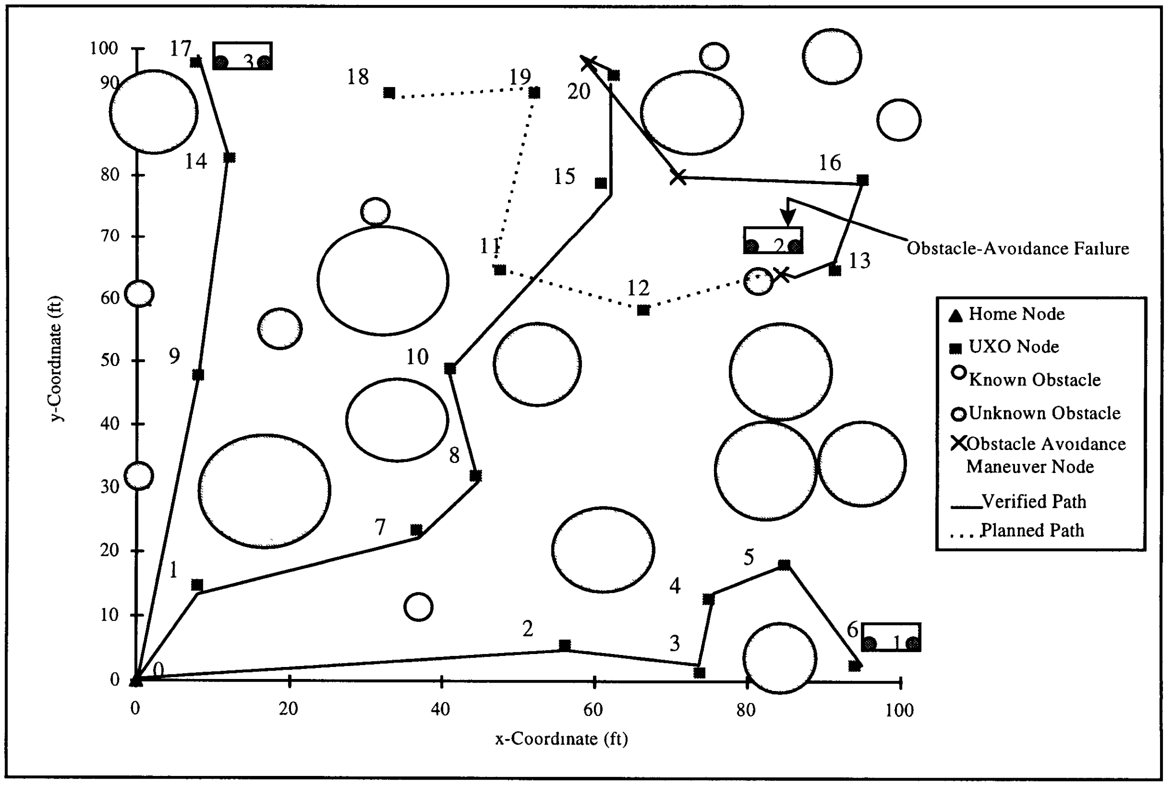


Figure 6-7: An Obstacle-Avoidance Maneuver Failure

The next event is that vehicle 2 runs into another unknown obstacle in the path between nodes 13 and 12 at 194 seconds (Figure 6-7). The vehicle automatically executes an obstacle-avoidance maneuver to go around the obstacle to the destination (node 12). However, the obstacle-avoidance fails in this case and the vehicle reports its failure to the manager.

The manager stores node 12 in the excluded-node file and commands the vehicle to by-pass the blocked node and visit the next assigned node (node 11) if it's not blocked from any known obstacle. The manager also stores the point (node oa3) where the vehicle has stopped. The next possible node to visit from node oa3 is node 11. Thus, the vehicle by-passes node 12 and travels from node oa3 to node 11 (Figure 6-4).

The vehicle detects the same obstacle going from node oa3 to node 11. Thus, the vehicle automatically executes another obstacle-avoidance maneuver. However, the maneuver fails again and the vehicle reports its failure. The manager stores node 11 in the excluded-node file and commands the vehicle to by-pass the blocked node to visit the next assigned node (node 19) in the plan if it's not blocked by any known obstacle. The manager also stores the point (node oa4) where the vehicle has stopped. The next possible node to visit from node oa4 is node 19. Thus, the vehicle by-passes node 11 and travels from node oa4 to node 19 (Figure 6-4). After this event, the vehicle is able to complete the remaining assigned tasks. Figure 6-4 displays the completed waypoint following tasks with some plan modifications in the second vehicle route.

Other events such as an idle or detonated vehicle may cause the manager to modify the initial plan to achieve the goal of the mission in the most optimal way. Simple modification concepts for these events have been proposed in the Chapters 3 and 4. However, these haven't been implemented to test their functionality.

6.1.3 Re-Planning Stage

Although simple modifications are used to deal with an unexpected event, the manager needs a way to plan to cover the excluded nodes. Furthermore, the manager should have an option to call a re-planner (i.e. multiple-goal re-planner) whenever it decides to abandon its initial plan for a more optimized plan. The multiple-goal re-planner is usually called when the vehicle either gets lost or added (i.e. equivalent to a vehicle that completes its task earlier than the other vehicles) in the mission. A proposed concept for this re-planner is presented in the Chapters 3 and 4. However, the multiple-goal re-planner hasn't been implemented.

STEP 8: Executing the single-goal re-planning since two excluded nodes exist.

Since the manager detects two excluded UXO nodes, it calls the single-goal re-planner to create a plan to cover the two nodes. The following shows the first re-plan to cover the two excluded nodes (11 and 12). The re-planner usually creates one plan per excluded node using the verified road segment information. Once the re-planner finds a segment to connect the excluded node to a verified road segment, it assigns the node to the closest vehicle that is available. Both excluded nodes are assigned to vehicle 2 since it is the closest available vehicle. The "High Speed Distance" represents the distance that the vehicle should travel at 6 fps since the road segments have been verified to be obstacle-free. The "Search Speed Distance" represents the distance that the vehicle should travel at 1 fps since the road segments have not been verified to be obstacle-free. The re-plan to cover node 12 is in the sequence of {18, 19, oa4, 12} and the re-plan for node 11 is in the sequence of {12, oa4, oa3, 13, 16, oa2, oa1, 20, 15, 10, 11}. The paths from oa4 to 12 and 10 to 11 are the only two road segments that have not been verified. Thus, the vehicle should travel slow on these paths. Otherwise, the vehicle can travel fast. Note that the second re-plan for node 11 depends on the re-plan for the first excluded node 12 in the planning stage. In other words, the re-plan for node 11 is created assuming that the re-plan for node 12 has worked. The time listed below is the mission time so far (i.e. including the initial planning and verification time).

```
*********
Replan  1
*********

Robot  2:
    ____Planned Path____
      33.05,    93.13
      52.12,    92.89
      82.74,    62.65    High Speed Distance:      62.10 ft
      **66.18,    58.28**    Search Speed Distance:    17.13 ft

Robot  2:
    ____Planned Path____
      66.18,    58.28
      82.74,    62.65
      82.75,    62.65
      91.27,    64.90
      95.06,    79.33
      66.52,    76.33
      61.23,    96.44
      62.30,    95.90
      60.71,    78.85
      41.01,    48.75    High Speed Distance:     144.67 ft
      **47.59,    64.75**    Search Speed Distance:    17.30 ft

***************************************** Time: 264.979785 s
```

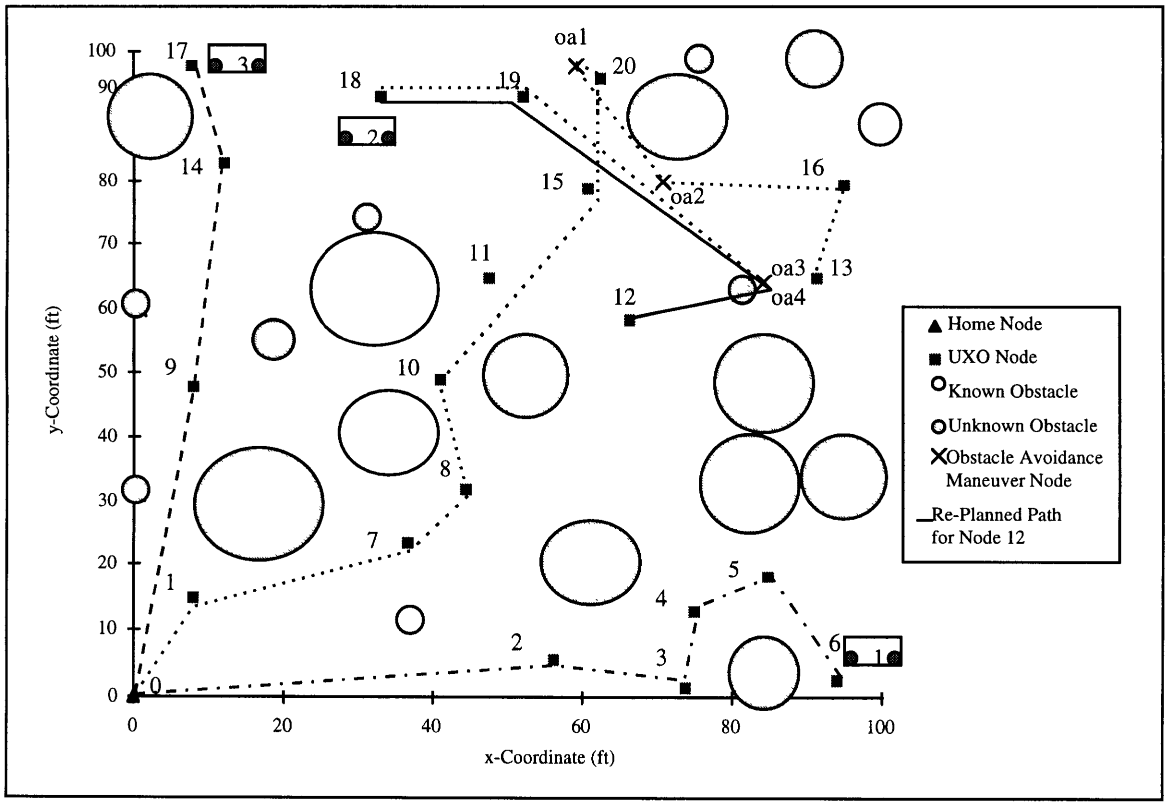


Figure 6-8: Graphical Representation of the Single-Goal Re-Plan for Node 12

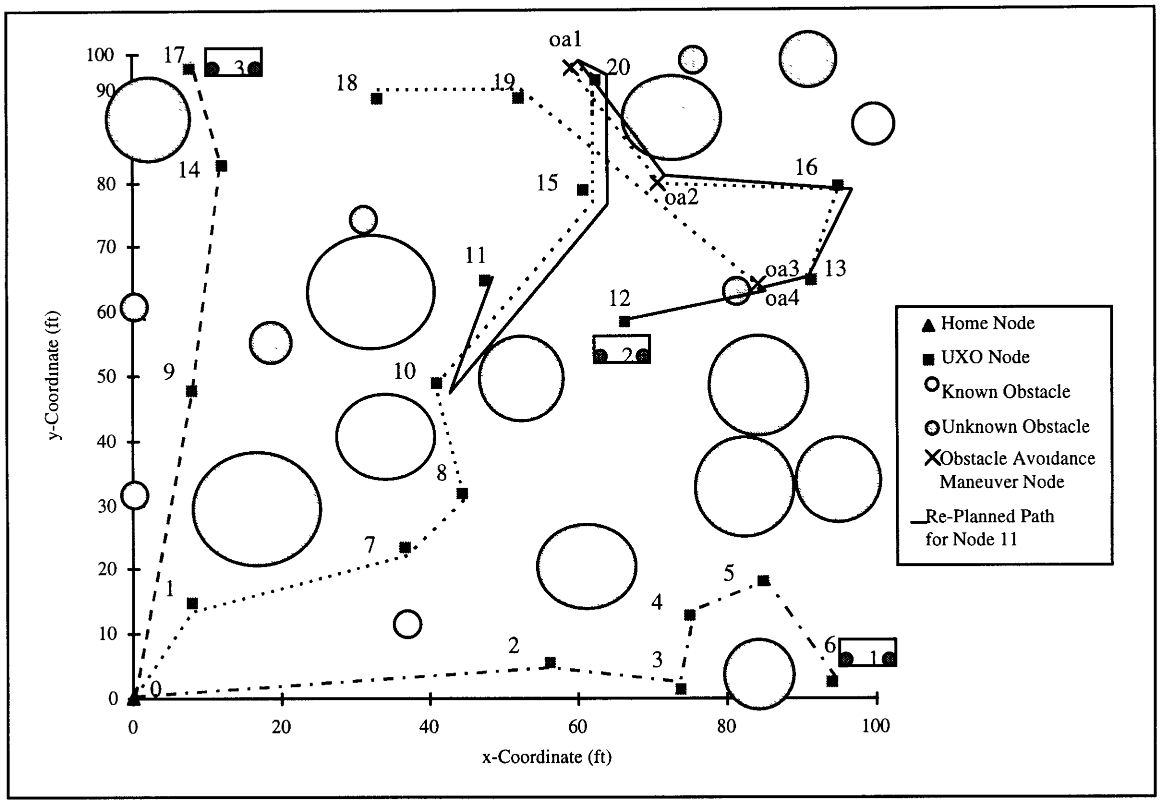


Figure 6-9: Graphical Representation of the Single-Goal Re-Plan for Node 11

Figure 6-8 and 6-9 show the re-plans to cover the two excluded UXO nodes. The two re-plans have been illustrated separately to clarify the explanation of each re-plan. Vehicle 2 is indeed the closest vehicle to verify the re-planned road segments to cover both node 11 and 12.

6.1.4 Re-Plan Verification Stage

STEP 6: Generating the waypoint following commands for the vehicle 2 using the re-planned road segments.

The following are the data structures of the waypoint following tasks that have been generated by the manager using the re-planned road segments. B*egin.x*, and *begin.y* (in feet) are simply the coordinate for the starting point of a segment. Likewise, *end.x* and *end.y* (in feet) are the coordinate for the ending point of a segment. The *velocity* will be 1 fps if the segment hasn't been verified to be obstacle-free. Otherwise, the *velocity* will be 6 fps for the vehicle to travel to the destination. The waypoint following tasks are generated using the re-planned road segments of the mixed and the SFC combined algorithm. The vehicle is commanded to do the waypoint following tasks listed below to initiate the verification process again.

```
***********************
Waypoint Following Task
***********************

Robot  2:
Waypoint 1:       begin.x  = 33.05       begin.y = 93.13
                  end.x    = 52.12       end.y   = 92.89
                  velocity = 6
Waypoint 2:       begin.x  = 52.12       begin.y = 92.89
                  end.x    = 82.74       end.y   = 62.65
                  velocity = 6
```

```
Waypoint 3:        begin.x  = 82.74       begin.y = 62.65
                   end.x    = 66.18       end.y   = 58.28
                   velocity = 1
Waypoint 4:        begin.x  = 66.18       begin.y = 58.28
                   end.x    = 82.74       end.y   = 62.65
                   velocity = 6
Waypoint 5:        begin.x  = 82.74       begin.y = 62.65
                   end.x    = 82.75       end.y   = 62.65
                   velocity = 6
Waypoint 6:        begin.x  = 82.75       begin.y = 62.65
                   end.x    = 91.27       end.y   = 64.90
                   velocity = 6
Waypoint 7:        begin.x  = 91.27       begin.y = 64.90
                   end.x    = 95.06       end.y   = 79.33
                   velocity = 6
Waypoint 8:        begin.x  = 95.06       begin.y = 79.33
                   end.x    = 66.52       end.y   = 76.33
                   velocity = 6
Waypoint 9:        begin.x  = 66.52       begin.y = 76.33
                   end.x    = 61.23       end.y   = 96.44
                   velocity = 6
Waypoint 10:       begin.x  = 61.23       begin.y = 96.44
                   end.x    = 62.30       end.y   = 95.90
                   velocity = 6
Waypoint 11:       begin.x  = 62.30       begin.y = 95.90
                   end.x    = 60.71       end.y   = 78.85
                   velocity = 6
Waypoint 12:       begin.x  = 60.71       begin.y = 78.85
                   end.x    = 41.01       end.y   = 48.75
                   velocity = 6
Waypoint 13:       begin.x  = 41.01       begin.y = 48.75
                   end.x    = 47.59       end.y   = 64.75
                   velocity = 1
```

STEP 7: Responding to any detected event such as obstacle and completed task.

After the vehicle 2 starts exploring the re-planned road segments, the vehicle reports to the manager if it has successfully visited a node (i.e. if it has verified a road segment to be obstacle-free). The manager stores the verified road segments to an output file (the compiled output file will be given at the end of the verification process). The following represents the coordinates of the nodes that the vehicles actually visited (i.e. in the order of visit) during the verification phase for the re-planning using the mixed and SFC algorithm. The time that is listed is the overall mission time so far (i.e. including all the stages).

```
*********
Replan  1
*********

Robot  2:
____Actual  Path____
      33.05,   93.13
      52.12,   92.89
      82.74,   62.65   High Speed Distance:     62.10 ft
      82.75,   62.65   Search Speed Distance:    0.01 ft

Robot  2:
____Actual  Path____
      82.75,   62.65
      82.74,   62.65
      82.75,   62.65
      91.27,   64.90
```

```
    95.06,    79.33
    66.52,    76.33
    61.23,    96.44
    62.30,    95.90
    60.71,    78.85
    41.01,    48.75    High Speed Distance:     127.55 ft
    47.59,    64.75    Search Speed Distance:    17.30 ft

******************************************* Time: 313.8881183 s
```

Dynamic Event-Driven Management

The first event that happens during the verification phase for the re-planning is that the vehicle runs into an obstacle in the path between the node oa4 and 12. The vehicle automatically executes an obstacle-avoidance maneuver to go around the detected obstacle to the destination (the node 12). However, the obstacle-avoidance fails so that the vehicle reports to the manager. The node 12 is the last assigned node to the vehicle so that it can't by-pass the node to the next assigned node. The manager stores the node 12 in the excluded-node file and commands the vehicle to execute the next waypoint following task (i.e. to cover the node 11). The manager also stores the point (i.e. oa5) where the vehicle has been stopped by the detected obstacle. The point oa5 is approximately same as oa3 so that oa5 is referred to be oa3 in the remaining section. Figure 6-10 shows the actual path that the vehicle traveled in attempt to cover the assigned excluded node 12.

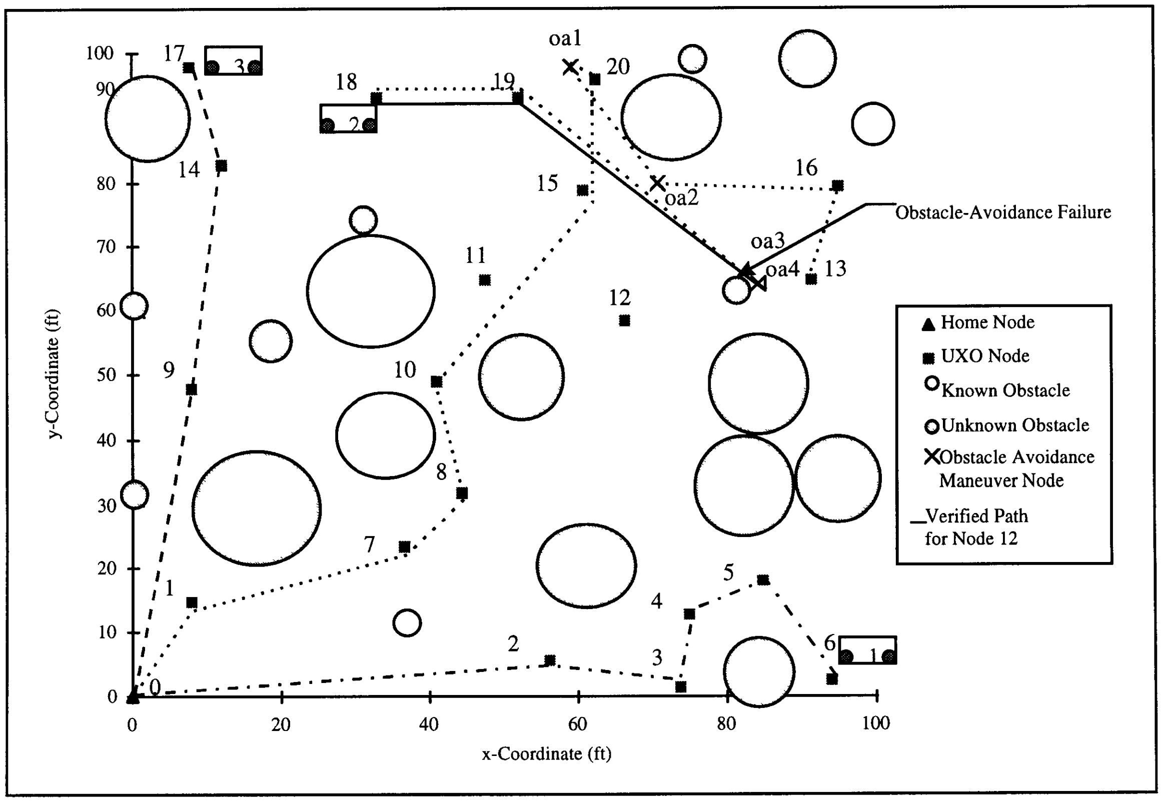


Figure 6-10: Graphical Representation of the Actual Path Using First Re-Plan for Node 12

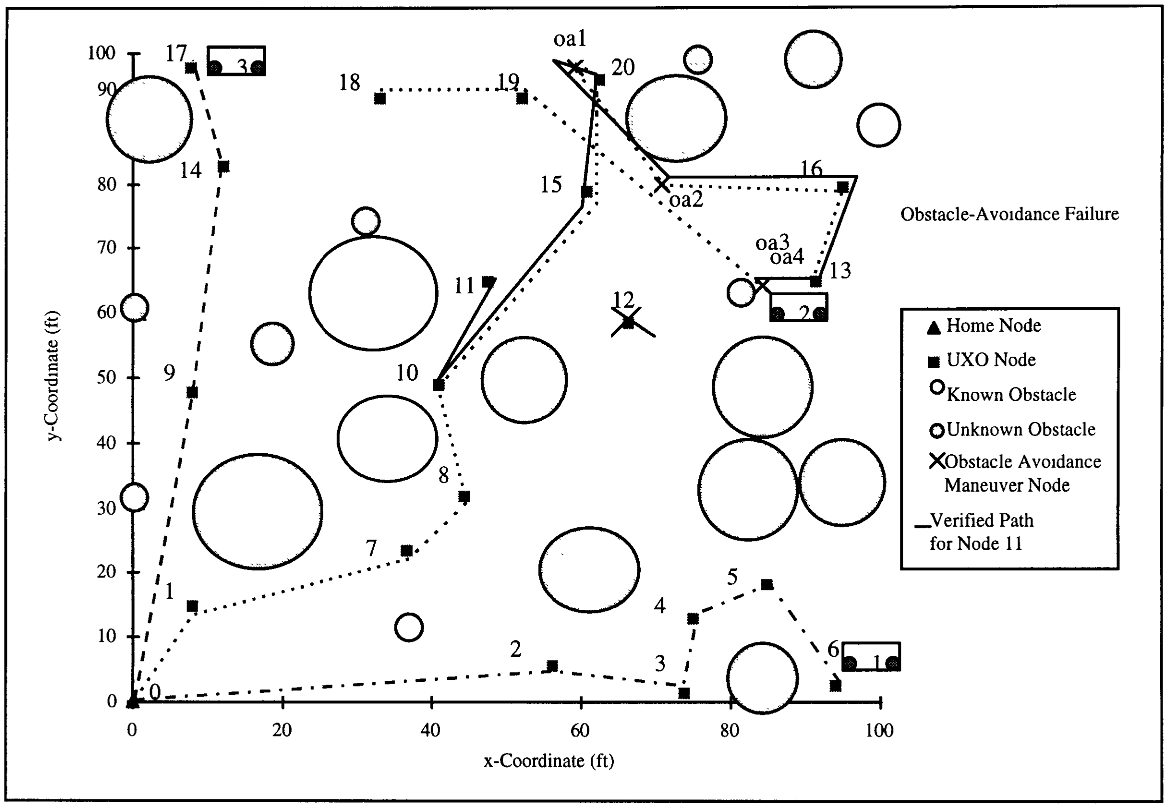


Figure 6-11: Graphical Representation of the Actual Path Using First Re-Plan for Node 11

The vehicle successfully covers node 11 with a modification to the re-planned road segments. Since the vehicle never made it to the node 12, the vehicle can't execute the next planned road segments from node 12 just as in the original re-plan. Thus, the modified plan to cover node 11 is same as the original re-plan except the first node 12 is left out. Thus, the vehicle starts from the node oa4 and travels through the assigned paths to cover the node 11. Figure 6-10 shows the actual paths that the vehicle traveled to try to cover node 12. Figure 6-11 shows the actual paths that the vehicle traveled to cover node 11.

Second Time Re-Planning Stage

STEP 10: Repeating STEP 6 through STEP 8 until either all of the roadways have been verified to be obstacle-free or no more road plan is available for the remaining excluded UXO nodes.

Since the manager still finds another excluded node (i.e. the node 12), the manager calls the single-goal re-planner again to create the second re-plan to cover the excluded node 12. Thus, the re-planning and verification of the re-plan repeats again.

STEP 8: Executing the single-goal re-planning again since one excluded node exists.

The following shows the second time re-plan to cover the one excluded node (i.e. the node 12). The excluded node is assigned to the vehicle 2 since it is the closest available vehicle. The re-plan to cover the node 12 is in the sequence of {11, 10, 15, 20, oa1, oa2, 12}. The paths from oa2 to 12 is the only road segment that has not been verified. Thus, the vehicle should travel slow on this path. Otherwise, the vehicle can travel at the fast speed mode. The time listed below is the mission time so far (i.e. including all the stages).

```
*********
Replan  2
*********
Robot  2:
    ____Planned Path____
       47.59,   64.75
       41.01,   48.75
       60.71,   78.85
       62.30,   95.90
       61.23,   96.44
       66.52,   76.33    High Speed Distance:      92.40 ft
       66.18,   58.28    Search Speed Distance:    18.05 ft

***************************************** Time:  313.8889513 s
```

Figure 6-12 shows the graphical representation that displays the second time re-plan to cover the one excluded UXO node. Vehicle 2 is indeed the closest vehicle to verify the re-planned road segments to cover node 12.

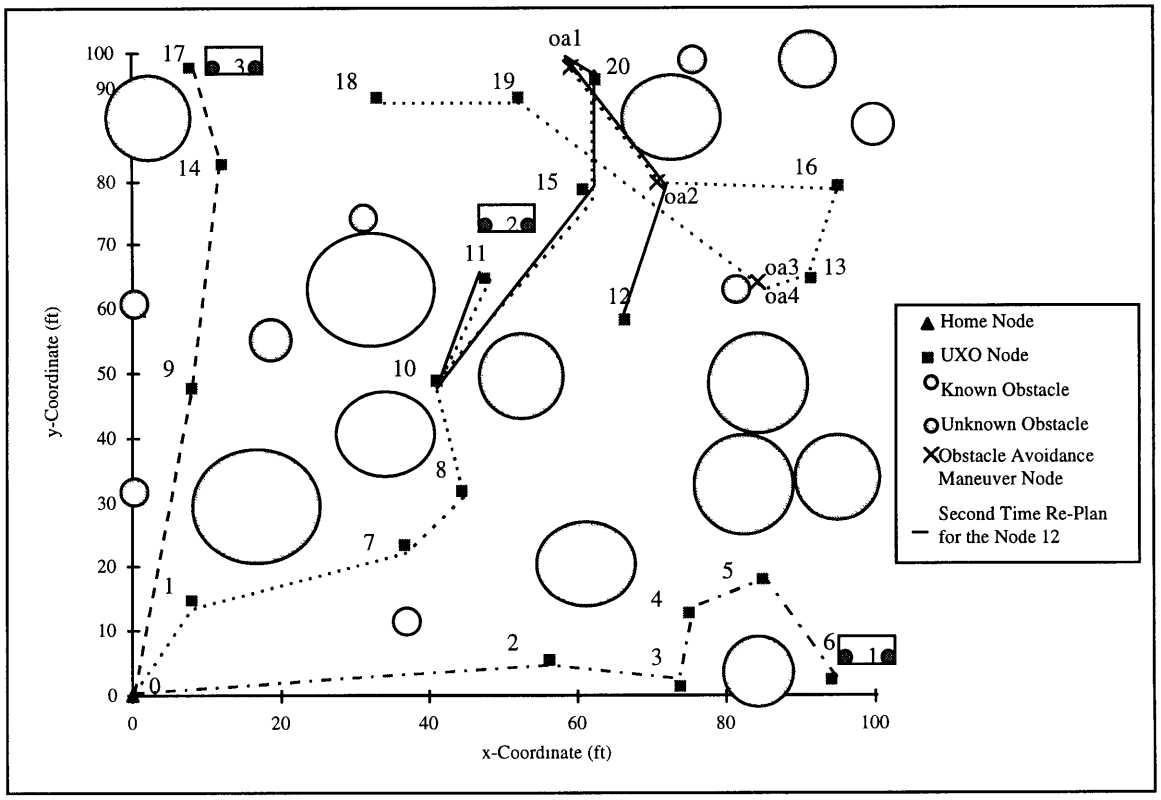


Figure 6-12: Graphical Representation of the Second Time Re-Plan

Second Time Re-Plan Verification Stage

STEP 6: Generating the waypoint following commands for the vehicle 2 using the second time re-planned road segments.

The following are the data structures of the waypoint following tasks that have been generated by the manager using the second time re-planned road segments. B*egin.x*, and *begin.y* (in feet) are simply the coordinate of the starting point of a segment. Likewise, *end.x* and *end.y* (in feet) are the coordinate of the ending point of a segment. *Velocity* will be 1 fps if the segment hasn't been verified to be obstacle-free.

Otherwise, *velocity* will be 6 fps for the vehicle to travel to the destination. The following waypoint following tasks are generated using the second time re-planned road segments of the mixed and the SFC combined algorithm. The vehicle is commanded to do the waypoint following task listed below to initiate the verification process again.

```
***********************
Waypoint Following Task
***********************

Robot  2:
Waypoint 1:        begin.x  = 47.59       begin.y = 64.75
                   end.x    = 41.01       end.y   = 48.75
                   velocity = 6
Waypoint 2:        begin.x  = 41.01       begin.y = 48.75
                   end.x    = 60.71       end.y   = 78.85
                   velocity = 6
Waypoint 3:        begin.x  = 60.71       begin.y = 78.85
                   end.x    = 62.30       end.y   = 95.90
                   velocity = 6
Waypoint 4:        begin.x  = 62.30       begin.y = 95.90
                   end.x    = 61.23       end.y   = 96.44
                   velocity = 6
Waypoint 5:        begin.x  = 61.23       begin.y = 96.44
                   end.x    = 66.52       end.y   = 76.33
                   velocity = 6
Waypoint 6:        begin.x  = 66.52       begin.y = 76.33
                   end.x    = 66.18       end.y   = 58.28
                   velocity = 1
```

STEP 7: Responding to any detected event such as obstacle and completed task.

After the vehicle 2 starts exploring the re-planned road segments, the vehicle reports to the manager if it has successfully visited a node (i.e. if it has verified a road segment to be obstacle-free). The manager stores the verified road segments to an output file (the compiled output file will be given at the end of the verification process). The following represents the coordinates of the nodes that the vehicles actually visited (i.e. in the order of visit) during the verification phase for the re-planning using the mixed and SFC algorithm. The time that is listed is the overall mission time so far (i.e. including all the stages).

```
*********
Replan  2
*********

Robot  2:
____Actual  Path____
      47.59,   64.75
      41.01,   48.75
      60.71,   78.85
      62.30,   95.90
      61.23,   96.44
      66.52,   76.33    High Speed Distance:     92.40 ft
      66.18,   58.28    Search Speed Distance:   18.05 ft

****************************************** Time: 347.3381183 s
```

Dynamic Event-Driven Management

Fortunately, the vehicle executes the verification process without detecting any more events. Therefore, the second time re-planned road segments are the verified road segments to cover node 12.

Thus, the graphical representation of the actual paths of the second time re-plan is the same as in Figure 6-12. Figure 6-13 shows the three obstacle-free routes that can be used in the UXO clearing process.

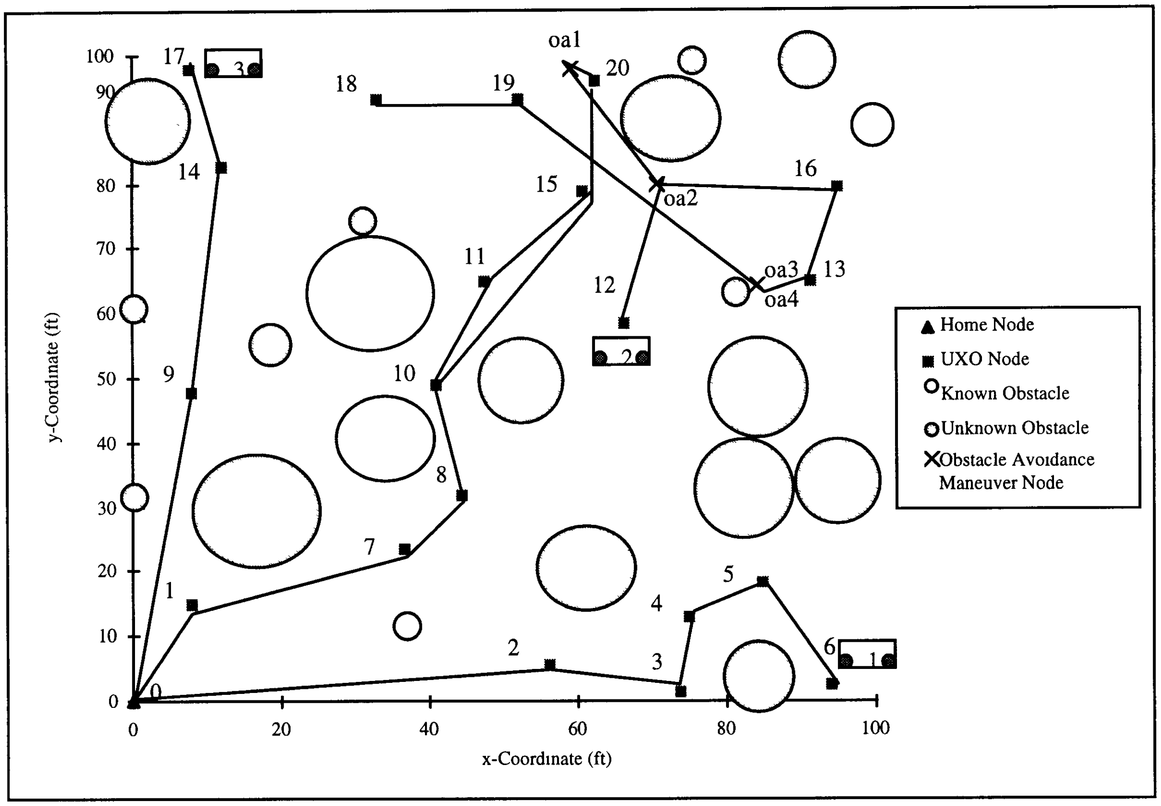


Figure 6-13: The Final Verified Routes for the 20-Node Mission Problem

6.1.5 Data Processing Stage

STEP 12: Outputting the summary of the road-building mission.

The following presents the final summary of the road-building mission that contains the performance data. From the final report, this mission can be evaluated by some of the performance data. The overall time to build the obstacle-free roads is approximately 6 minutes to cover the 20 UXO nodes using only 3 vehicles. No excluded node exists after a series of the re-planning processes. The three vehicle cover approximately 800 feet in distance to visit all 20 UXO nodes.

```
************
Final Report
************

# of UXO Node:                20
# of ODA Node:                 0
# of Known Obstacle Node:      4
# of Unknown Obstacle Node:   16
# of Home Node:                1
# of Excluded Node:            0

Work Space Length:           100 ft
Work Space Width:            100 ft

# of Vehicles:                 3
```

```
High Speed Distance:
     Robot  1:                  0.00 ft
     Robot  2:                282.06 ft
     Robot  3:                  0.00 ft
     All:                     282.06 ft

Search Speed Distance:
     Robot  1:                 99.31 ft
     Robot  2:                300.33 ft
     Robot  3:                115.41 ft
     All:                     515.05 ft

Total Distance:               797.11 ft
Total Time:                   347.3381183 s
```

6.2 Random Tests of the Mission Manager

Several random tests of the mission manager have been run to determine the performance and limitations of the manager described in the previous chapters. The main objective of these tests is to show the manager's effectiveness in performing the road-building mission using multiple robotic vehicles. Several simulation parameters have been varied to obtain the marginal performance results and perhaps limitations of the manager (i.e. particularly the initial planner). It is assumed that there is a limit on the number of vehicles which can provide a significant improvement in the performance of the road-building mission. Likewise, it is assumed that there is a limit on the number of UXO that the manager can plan/re-plan. The number of known and unknown obstacles in the mission also affects the performance of the mission manager in planning and re-planning processes. In fact, the unknown obstacles are assumed to be the biggest cost driver in terms of the total mission time including the re-planning time. The simulated test results can also be used to compare the efficiency of the two proposed and implemented road planning algorithms. The compared results between the two initial planning algorithms should show the important issues that must be considered in formulating planning algorithms. All simulations have been run on a Pentium computer in the Linux operating system.

Before the various simulated tests can be presented, important assumptions of the simulation must be addressed. The manager is able to simulate the vehicles to travel at one of the two constant speed modes (i.e. slow and fast) as discussed in the Chapter 1. The manager can also simulate and manage parallel task achievement. For instance, if multiple vehicles are used in building obstacle-free roads simultaneously, then the total mission time is derived from the longest exploration time (i.e. time that the vehicle with the most workload takes). Therefore, the total mission time includes the initial planning, all the re-planning, and the longest exploration time. Another assumption that has been incorporated into the simulations is that all the vehicles start the verification process at the home node which is always at the origin (i.e. (0, 0)).

6.2.1 Consistency Test of the Mission Manager

To test the consistency of the mission manager in planning and re-planning, the following series of tests have been run, based on the same mission. Multiple tests have been run based on the same workspace, number of vehicles, number and position of UXO nodes, and number and position of obstacles. Several runs of the mission manager on this mission have been obtained using both implemented planning algorithms. Table 6-1 lists the key simulation parameters used for the tests to evaluate the consistency of the mission manager. The "separation" in the Table represents the separated distance in feet between any two nodes. The Figure 6-14 shows the simulated test results. The "Greedy" in Figure 6-14 refers to the greedy routing and "winner" distributing combined algorithm. Likewise, "Mixed" refers to the mixed routing and SFC distributing combined algorithm.

Table 6-1: Simulation Parameters for Consistency Test

Parameter	Parameter Value
# of UXO Node	50
# of Known Obstacle Node	10
# of Unknown Obstacle Node	40
# of Home Node	1
# of ODA Node	0
Workspace	150 x 150 ft^2
Separation	10 ft
# of Vehicle	3

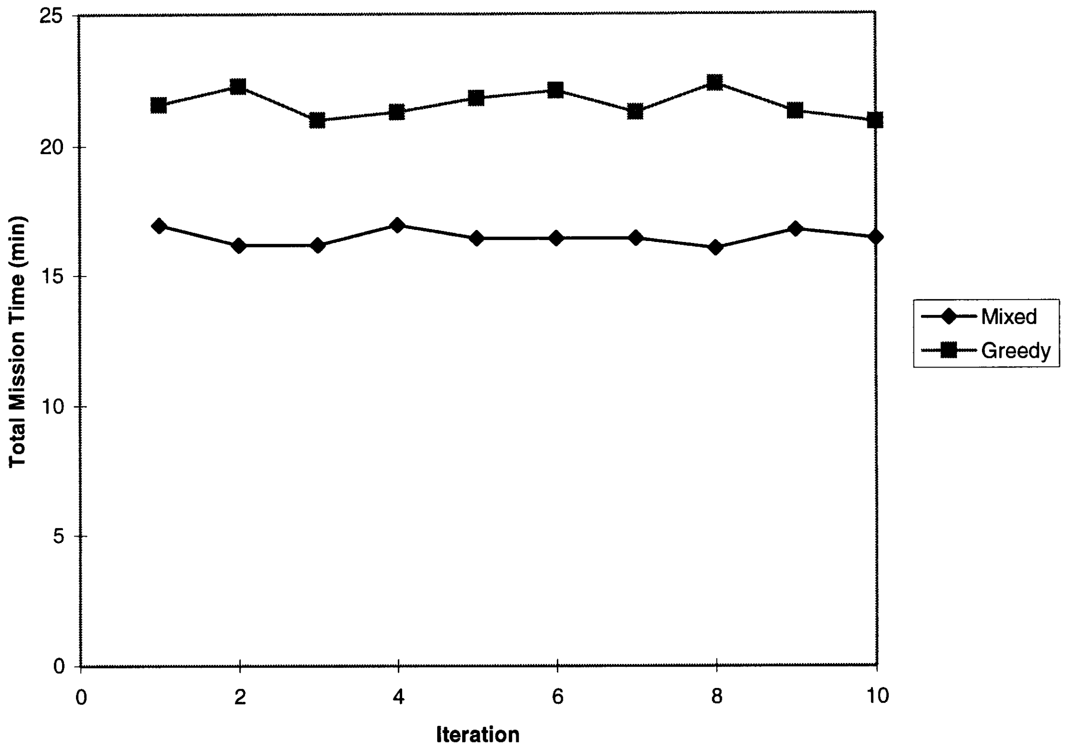


Figure 6-14: Multiple Iterations on One Road-Building Mission

From the curve in the Figure 6-14, both of the planning algorithms can be considered to perform consistently. The mixed combined algorithm completes the mission slightly faster in all of the runs. This result was expected since the mixed algorithm is assumed to generate much more optimal routing solutions than the greedy algorithm. More results that compare the two planning algorithms are presented later.

The reason why the curve is not quite straight between runs is that some of the unknown obstacles may or may not get avoided during the plan verification phase. Each simulated unknown obstacle is assigned with a probability of obstacle-avoidance success. Thus, depending on the probability they may get avoided in some runs and may not get avoided in some other runs. The total average time for this mission is approximately 30 to 35 minutes. Thus, three vehicles take 17 to 25 minutes to build three separate obstacle-free roadways that are connected at the home node.

6.2.2 Guideline for the Number of Vehicles

To derive a guideline to determine the most efficient use of the number of vehicles in performing the road-building mission, a randomly generated mission is simulated with a varying number of vehicles. Table 6-2 lists the key simulation parameters used for the tests to determine the most efficient use of the number of vehicle in a given mission. Figure 6-15 shows the simulated test results.

Table 6-2: Simulation Parameters for the Mission with the Varying Number of Vehicles

Parameter	Parameter Value
# of UXO Node	50
# of Known Obstacle Node	10
# of Unknown Obstacle Node	40
# of Home Node	1
# of ODA Node	0
Workspace	200 x 200 ft^2
Separation	10 ft
# of Vehicle	1-10

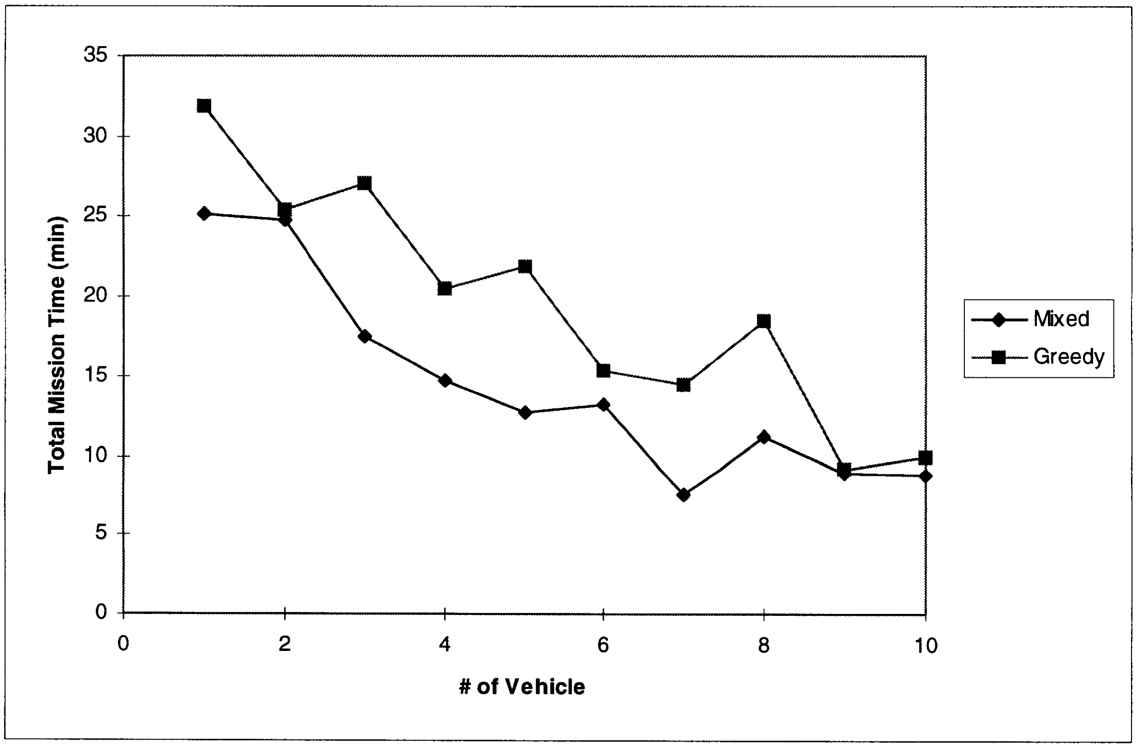


Figure 6-15: Effectiveness of Multiple Vehicles with Increasing Number of Vehicles

From the curve in the Figure 6-15, the effectiveness of using multiple vehicles for the road-building mission is shown. The simulated test results above show that the most optimal way to carry out this mission is to use four or five vehicles using the mixed combined algorithm. The results using the greedy combined algorithm don't really show a clean-cut number of the vehicles to use. The initial planner will plan for either four or five paths depending on the road planning algorithm used and the verification manager will coordinate four or five vehicles to complete the mission. The total mission time of this road-building mission using one vehicle is much greater than using four or five vehicles. Note that the total mission time using the mixed algorithm is slightly less than using the greedy algorithm.

6.2.3 Comparison Tests of the Road Planning Algorithms

Random Mission Cases

Several test conditions have been run to compare the efficiency between the two planning algorithms. The first test condition is that ten random missions with same workspace have been run using both planning algorithms. The simulation parameters for these tests are the same as listed in Table 6-2 except that three vehicles have been used for all the mission cases. Figure 6-16 shows the results of ten different missions.

From the Figure 6-16, the mixed combined algorithm results in slightly shorter mission time in six cases, while the greedy combined algorithm gives slightly shorter mission time in four cases. Thus, the greedy combined algorithm performs better than the mixed combined algorithm in some mission cases. The mixed routing algorithm produces more optimal routes than the greedy routing algorithm. However, the SFC and "winner" distributing algorithms are inconsistent in grouping the nodes. Thus, neither of the distributing algorithms is superior to the other in terms of the total mission time. Although the workspace is the same for all ten test cases, the total mission time varied greatly between any two test cases (i.e. mission).

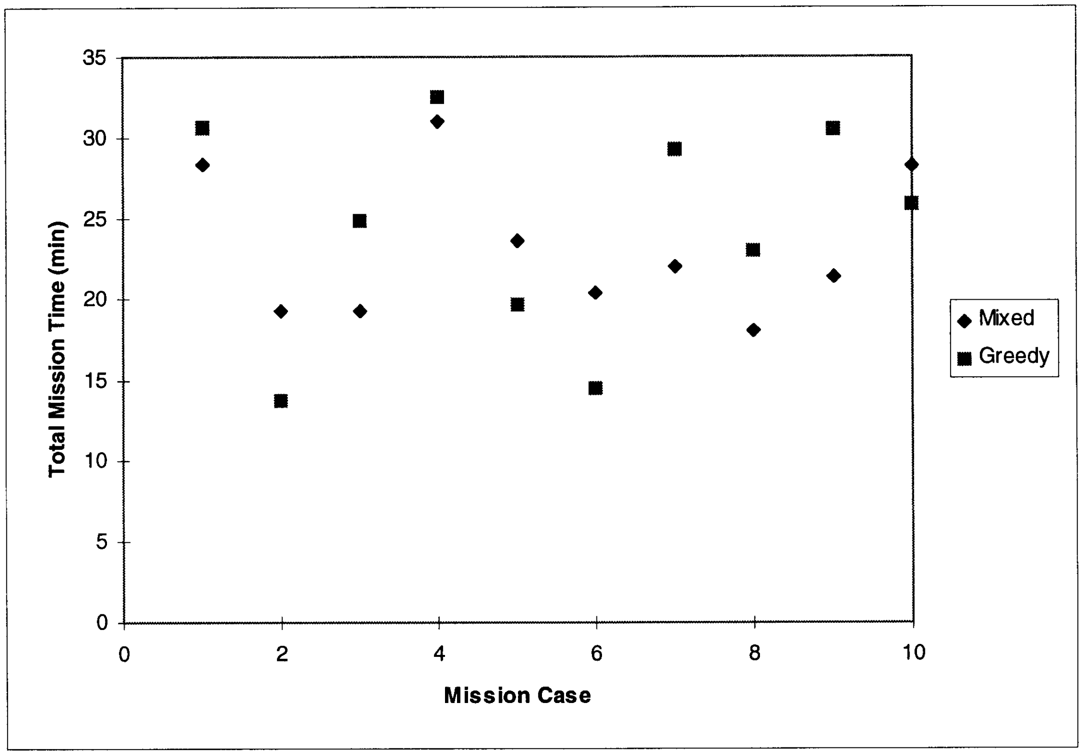


Figure 6-16: Simulated Test Results for the Ten Mission Cases

No-Obstacle Mission Cases

The third set of simulation parameters is listed in the Table 6-3. The specific test condition is that neither known nor unknown obstacles are present in the mission. Thus, this mission is strictly optimizing the order of the nodes that must be visited by the vehicles. Of course, no re-planning is done since no obstacles are present in the mission. Ten different missions with same workspace have been run to test the two routing algorithms using one vehicle. Figure 6-17 shows the tested results for the ten different missions with no obstacles. Figure 6-18 shows the total planning (i.e. initial planning) time for the given missions.

Table 6-3: Simulation Parameter for the No Obstacle Condition Tests

Parameter	Parameter Value
# of UXO Node	30
# of Known Obstacle Node	0
# of Unknown Obstacle Node	0
# of Home Node	1
# of ODA Node	0
Workspace	80 x 80 ft^2
Separation	8 ft
# of Vehicle	1

From Figure 6-17, the mixed combined algorithm yields slightly better solutions than the greedy combined algorithm. This is consistent with all the other results of the two routing algorithms. In some mission cases, the mixed algorithm is not quite optimizing the routing of a vehicle since it's intended to create a tour but not a single path. On the other hand, the greedy algorithm creates a single path that connects the nearest nodes. In addition, the mixed algorithm must be improved in some steps to yield the optimal solution more consistently. Furthermore, the planning time for the mixed algorithm is slightly more than the greedy algorithm (Figure 6-18). This is an expected result since the mixed algorithm considers many more choices than the greedy algorithm. However, the total planning times are on the order of milli seconds so that they can be ignored.

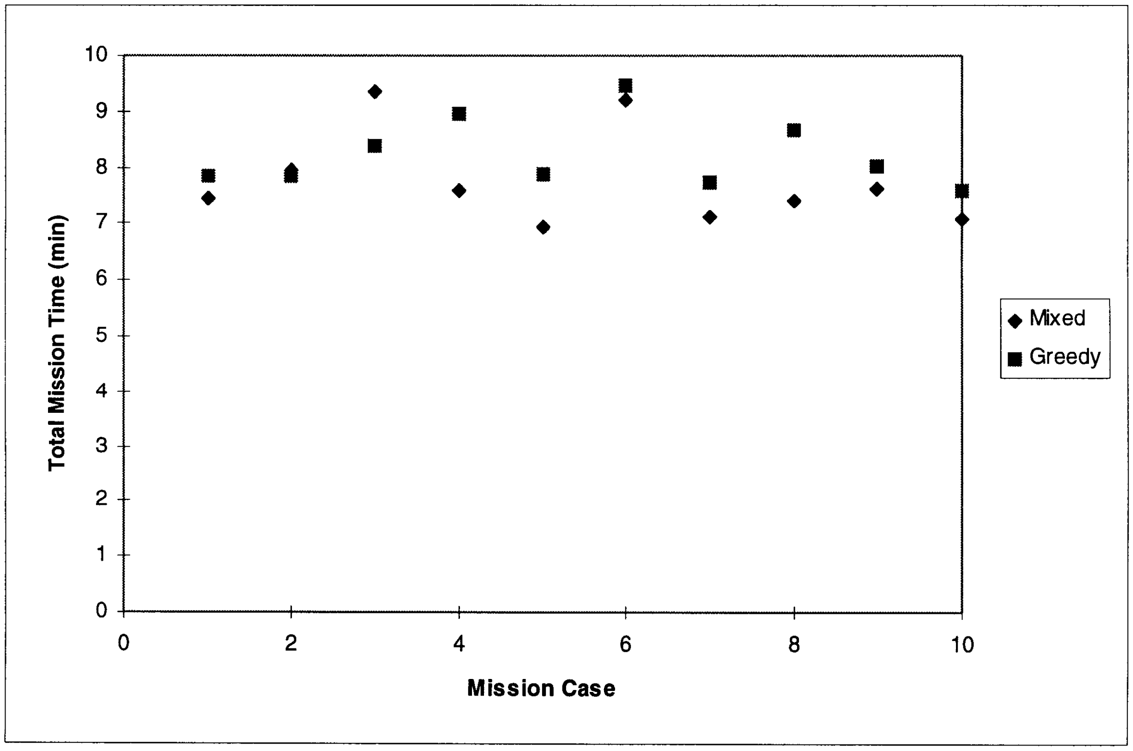


Figure 6-17: Effectiveness of the Planner with No Obstacles

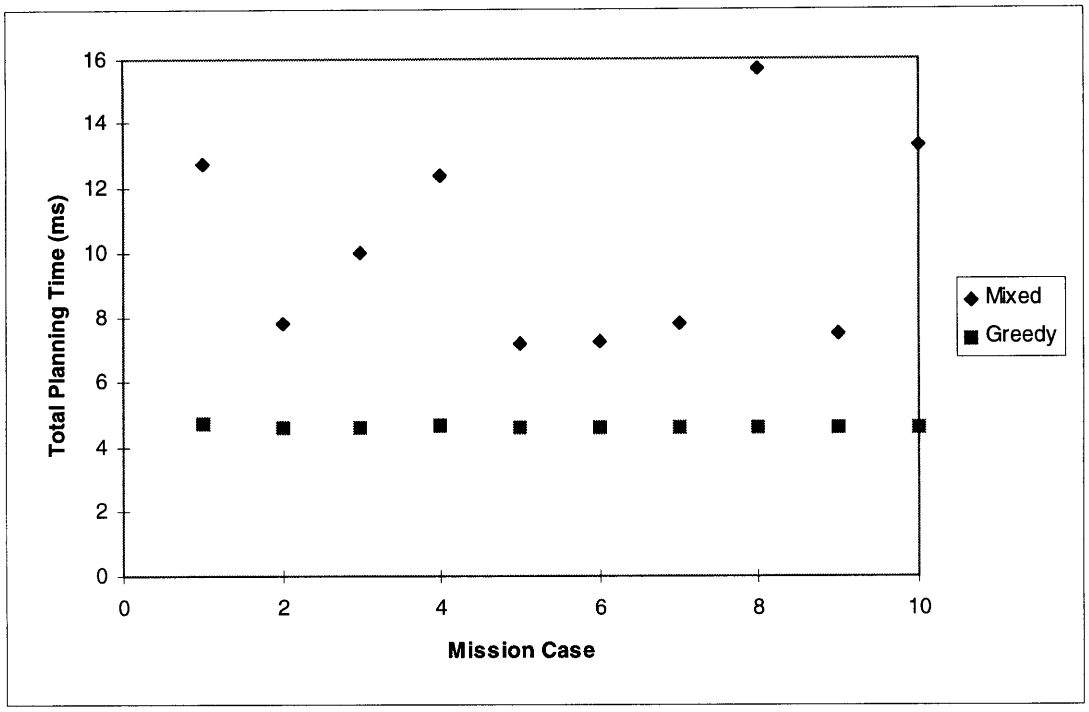


Figure 6-18: Planning Time for the Missions with No Obstacles

No Unknown Obstacle Mission Cases

The fourth set of simulation parameters is listed in the Table 6-4. This specific test condition is that the unknown obstacles are not present in the mission but some known obstacles are present in the mission. Thus, this mission is optimizing the order of the nodes that must be visited by the vehicles taking into account the known obstacles. Of course, no re-planning is done using the mixed combined algorithm since no unknown obstacles are present in the mission. However, some re-planning may be done using the greedy combined algorithm depending on the complexity of the known obstacle map. Ten different missions with the same workspace have been run to test the two routing algorithms using one vehicle. Figure 6-19 shows the tested results for the ten different missions with no unknown obstacles. Figure 6-20 shows the total planning (i.e. initial planning) time for the given missions.

Table 6-4: Simulation Parameter for the No Unknown Obstacle Condition Tests

Parameter	Parameter Value
# of UXO Node	50
# of Known Obstacle Node	28
# of Unknown Obstacle Node	0
# of Home Node	1
# of ODA Node	0
Workspace	150 x 150 ft^2
Separation	5 ft
# of Vehicle	1

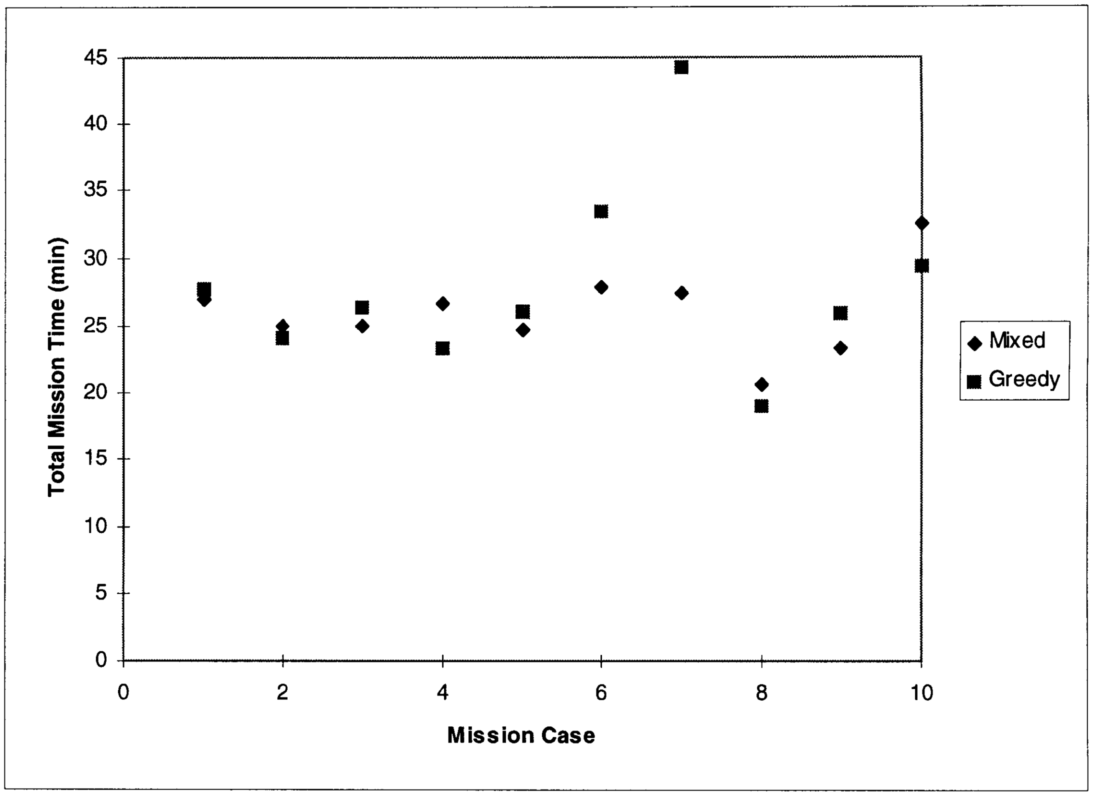


Figure 6-19: Effectiveness of the Planner with No Unknown Obstacles

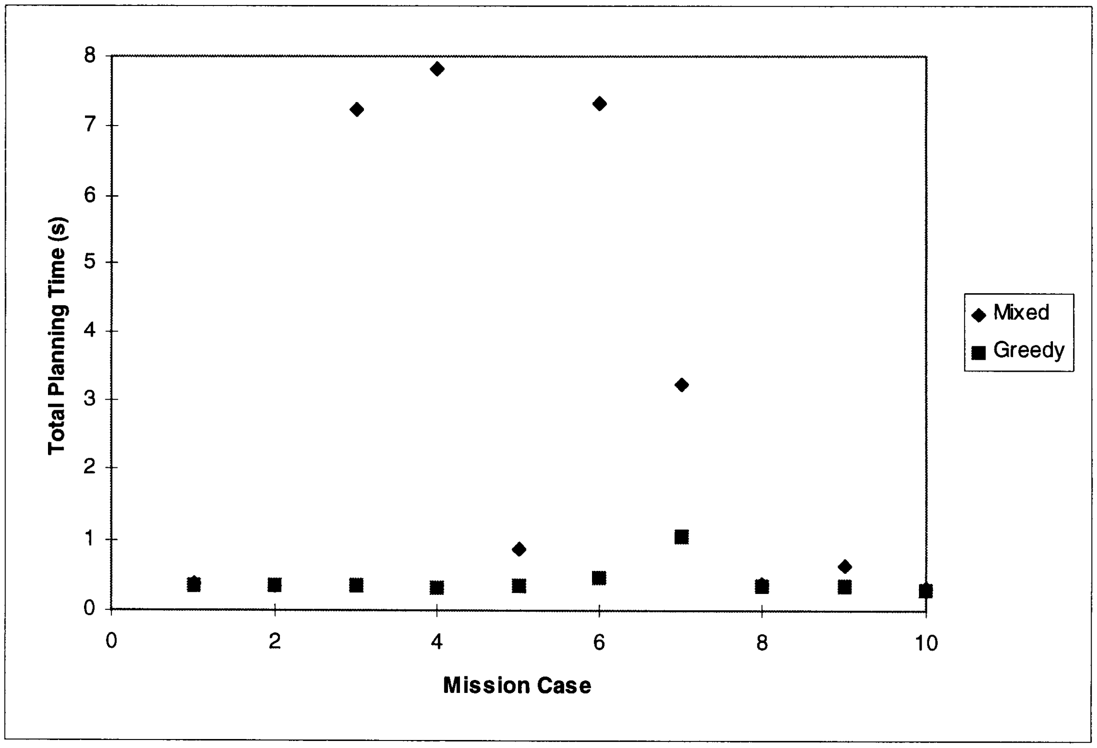


Figure 6-20: Planning Time for the Missions with No Unknown Obstacles

From Figure 6-19, the mixed combined algorithm yields slightly better solutions than the greedy combined algorithm in most of the mission cases. This is consistent with the results for the mission cases with no obstacles. Again, the planning time for the mixed algorithm is higher than for the greedy algorithm. Furthermore, the planning time with no known obstacle is indeed faster than the planning time with some known obstacles. This is an expected result since the planner takes longer to generate an initial road plan that avoids the known obstacles.

6.2.4 Number of Nodes

Varying the Number of UXO Nodes

To derive the limitation of the mission manager (i.e. initial planner and re-planner), a series of tests using multiple vehicles with an increasing number of UXO nodes have been run. The number of known and unknown obstacles have been increased accordingly to make the mission more challenging. Table 6-5 lists the simulation parameters for these tests with varying number of UXO nodes. Figure 6-21 displays the tested results.

Table 6-5: Simulation Parameter for the Mission Cases with Varying Number of UXO Nodes

Parameter	Parameter Value
# of UXO Node	10 - 100
# of Known Obstacle Node	2 - 20
# of Unknown Obstacle Node	8 - 80
# of Home Node	1
# of ODA Node	0
Workspace	100-200 x 100-200 ft^2
Separation	10 ft
# of Vehicle	4

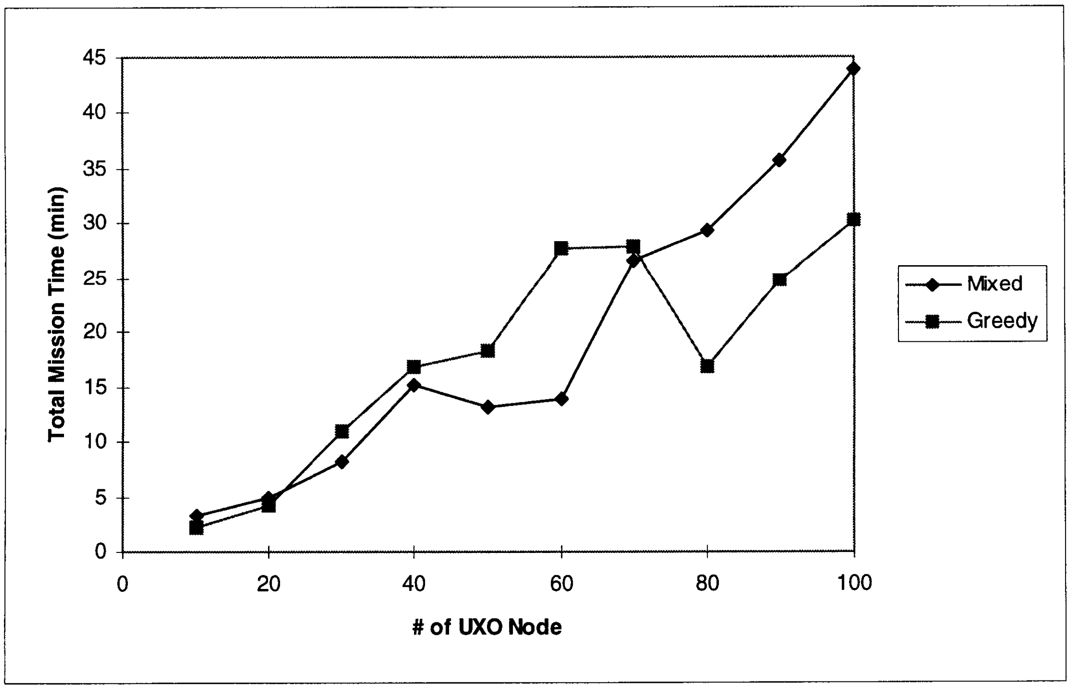


Figure 6-21: Mission Performance with Increasing UXO Node

From the curve in the Figure 6-21, the total mission time is increasing somewhat linearly as the number of UXO nodes increases. The mission time decreases by 10 minutes from 70 to 80 UXO nodes using the greedy combined algorithm. This decrease is due to the efficiency of the "winner" distributing algorithm. The workload for the four vehicles are evenly distributed by the "winner" algorithm in this case. However, the time increases rapidly after about 50 UXO nodes. The total mission time will definitely increase even faster as more UXO nodes are added to the problem. This can be considered to be one of the limitations on the mission manager.

Varying the Number of Known Obstacles

The next test condition is to vary the number of known obstacles and keep other parameters constant. Table 6-6 lists the simulation parameter for this test condition. Figure 6-22 displays the tested results with the increasing number of known obstacles. Figure 6-23 shows the total planning time for this series of tests.

Table 6-6: Simulation Parameter for the Mission Cases with Varying Number of Known Obstacles

Parameter	Parameter Value
# of UXO Node	30
# of Known Obstacle Node	5-50
# of Unknown Obstacle Node	15
# of Home Node	1
# of ODA Node	0
Workspace	300 x 300 ft^2
Separation	10 ft
# of Vehicle	3

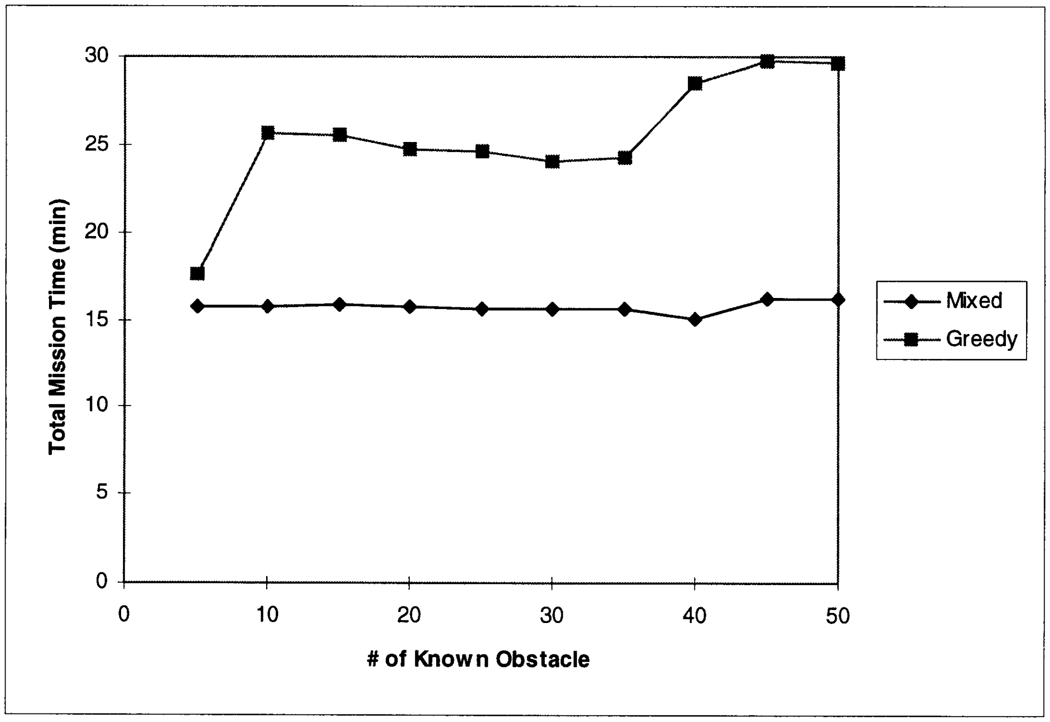


Figure 6-22: Mission Performance with Increasing Known Obstacles

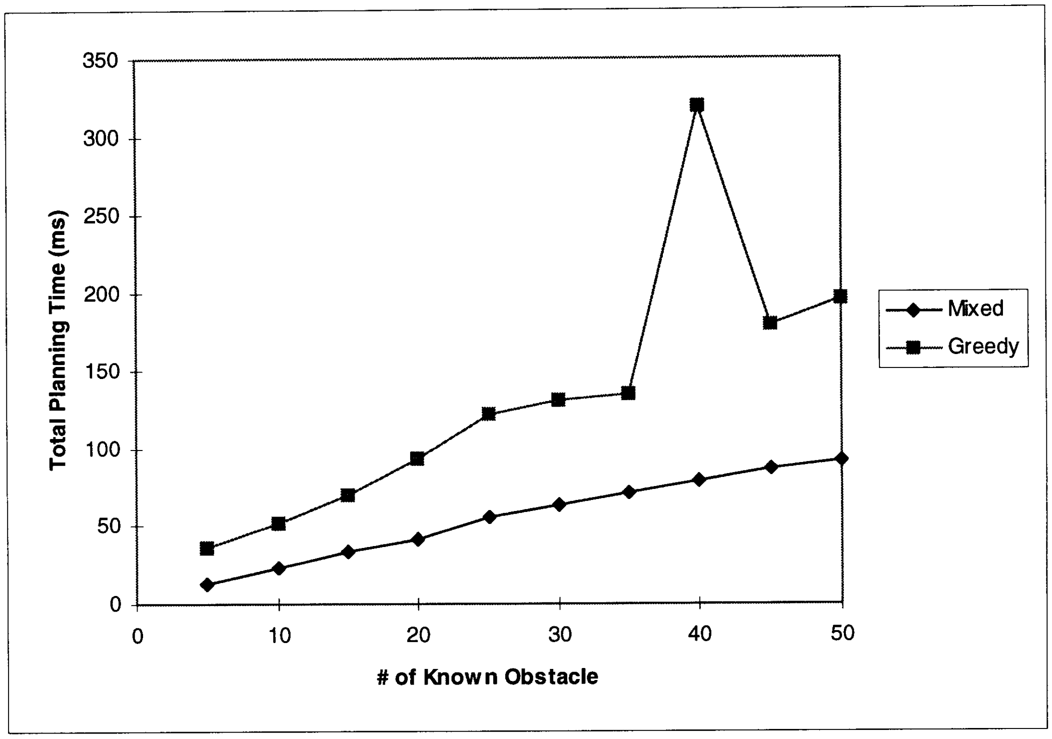
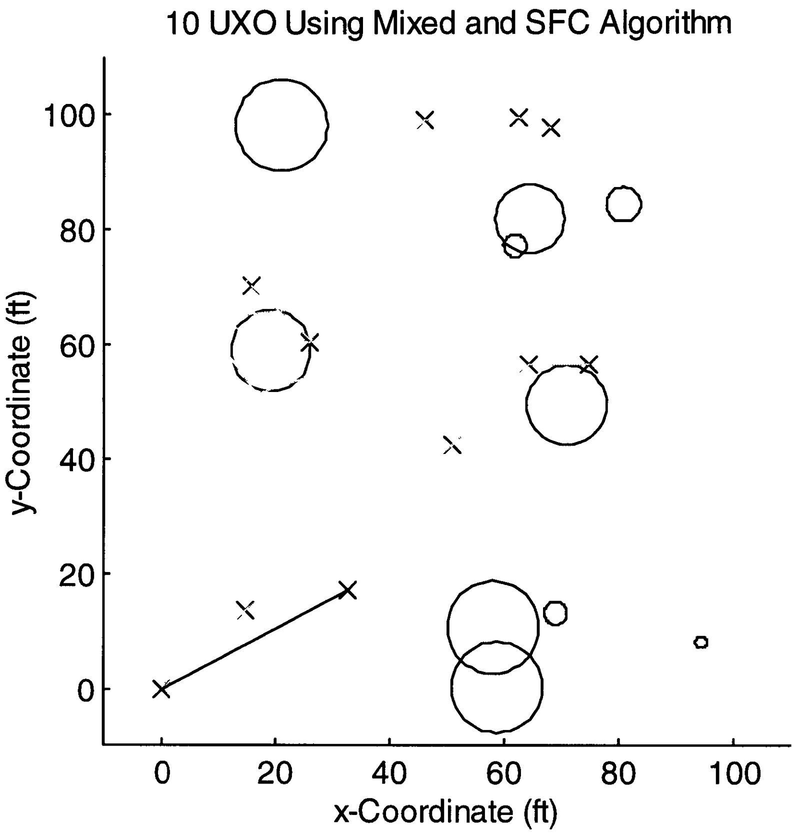


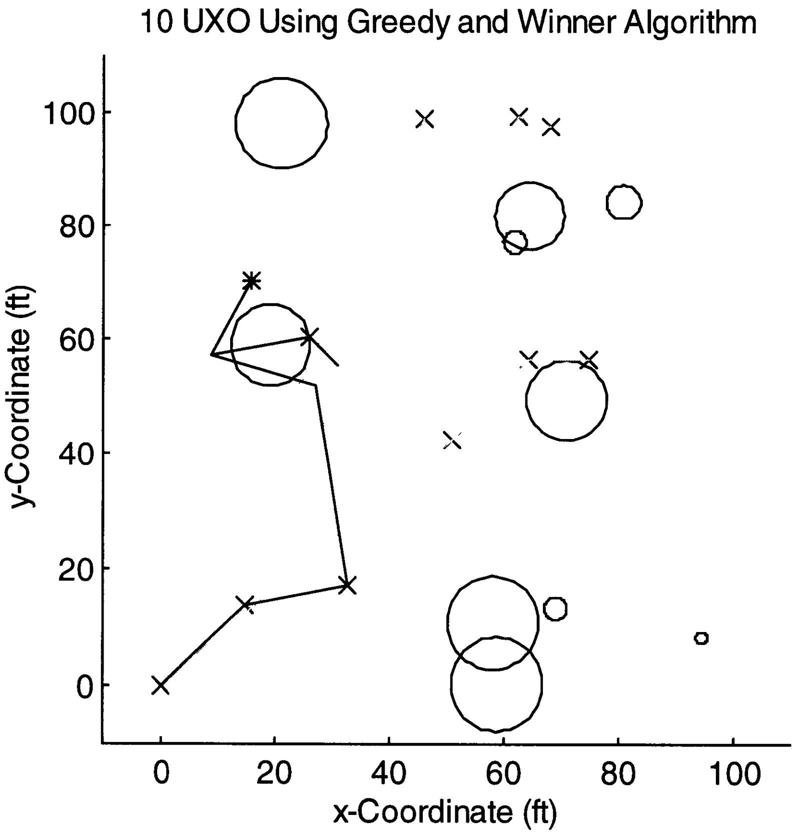
Figure 6-23: Planning Time with Increasing the Number Known Obstacles

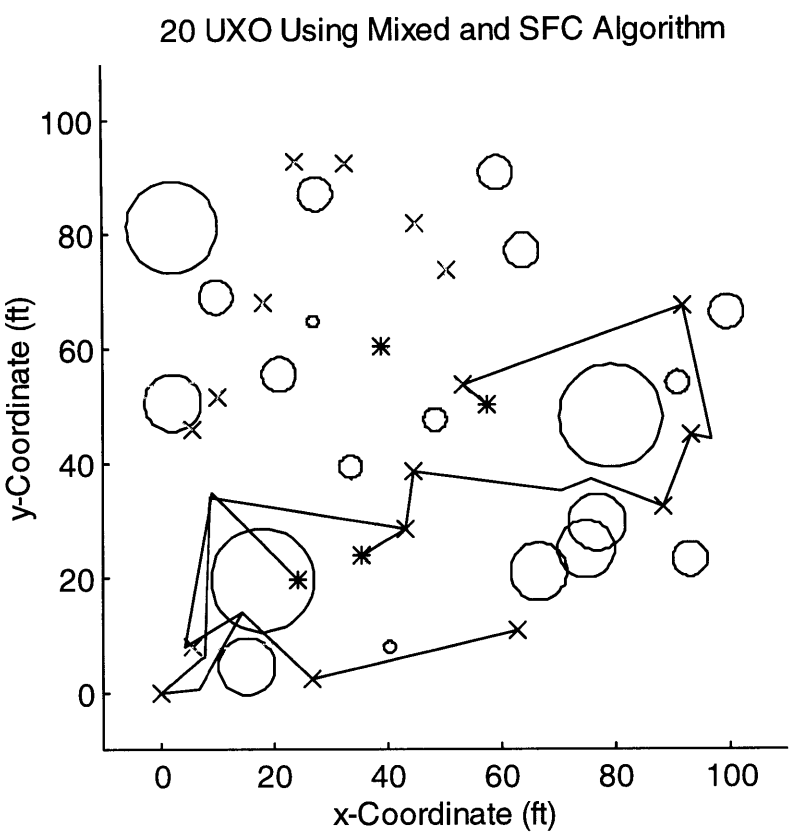
The mixed combined algorithm generally yields better solution than the greedy combined algorithm. The missions with more than 35 known obstacles become very time consuming using the greedy algorithm. The time increases rapidly as the number of obstacles exceed the number of UXO nodes. This is due to the fact that the greedy algorithm excludes many UXO nodes due to known obstacles during the initial road plan. Thus, the initially excluded nodes have to be re-planned at the end of the plan verification process. If many excluded nodes exist, then many re-planning and re-plan verification steps will have to be accomplished to complete the mission. Therefore, more planning/re-planning and re-plan verification time is required using the greedy algorithm. The mixed algorithm includes all the UXO nodes in the initial plan. Thus, less initially excluded nodes may exist using the mixed algorithm. The mixed algorithm is more useful in the road-building missions when the mission contains a complicated map of obstacles.

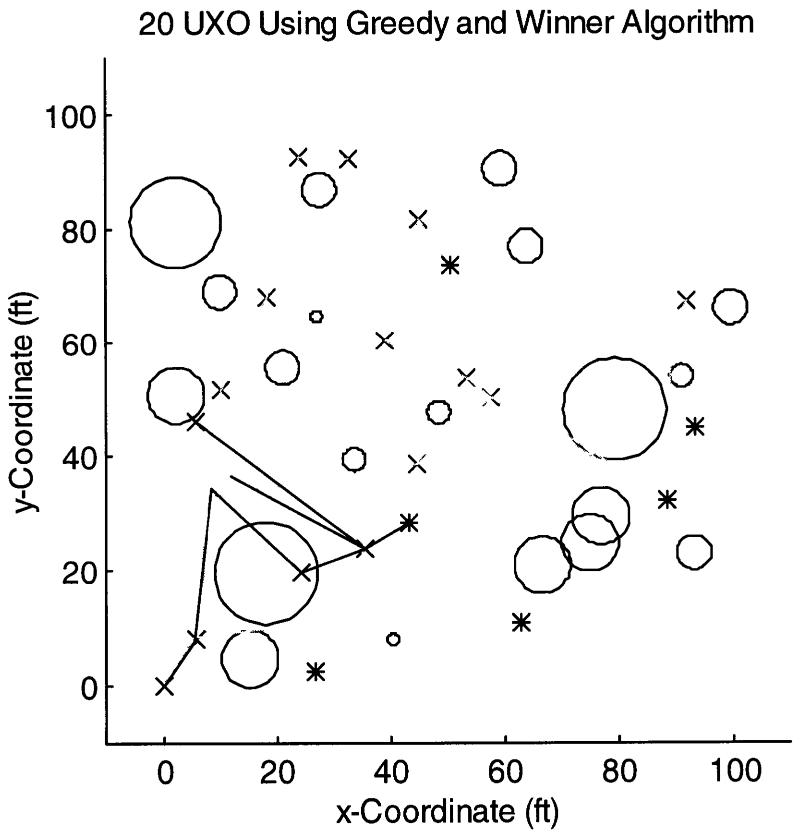
Graphical Representations of the Mission Cases with Varying Number of UXO and Obstacle Nodes

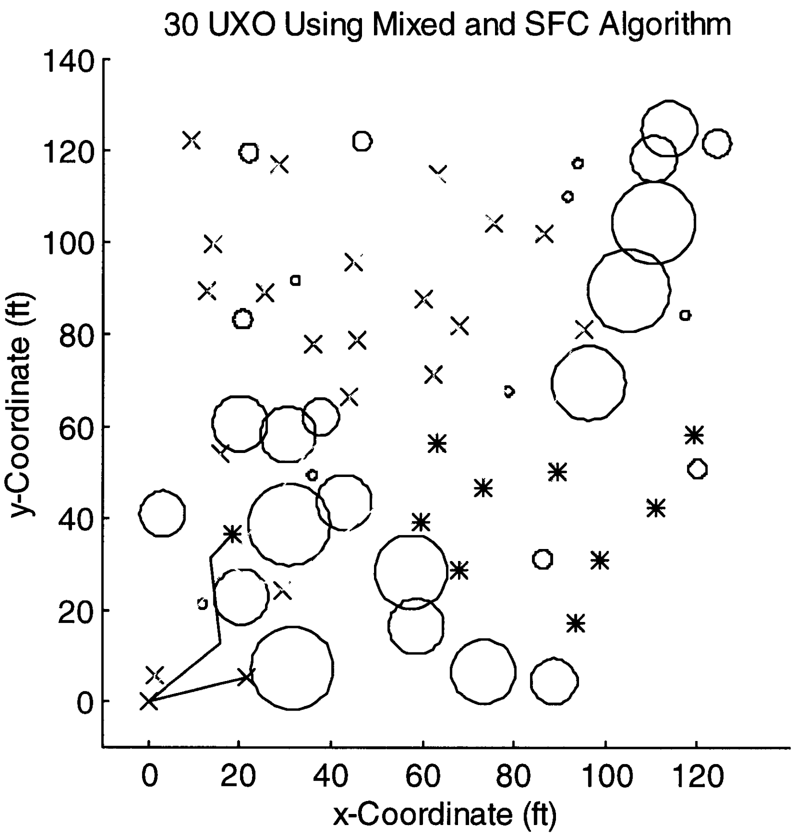
The following are graphical representations of the results of ten mission cases using four vehicles with increasing number of UXO nodes, and known and unknown obstacle nodes. The asterisks represent excluded nodes at the end of the plan verification process. From the following results, the planner makes reasonable plans of paths around the known and unknown obstacles for most of the mission. The following results show some faults in the simulations. Several simulation problems cause the vehicles to appear to travel along "infeasible" roads. These problems mostly have to do with the effectiveness of the simulated obstacle avoidance maneuvers. Obstacle-avoidance maneuvers and the simulation thereof are beyond the scope of this thesis. In addition, some of the re-planned paths aren't shown correctly in the simulation. The simulator seems to loose track of some segments of the actual paths that the vehicles use to visit the excluded nodes. The planner produces correct road plans with the given road map to cover all the UXO nodes. However, the simulator isn't quite simulating the environmental and verified roadways correctly. Despite these difficulties, the figures below provide an indication of the basic behavior of the planning and management system and serve to verify that reasonable decisions are generated.

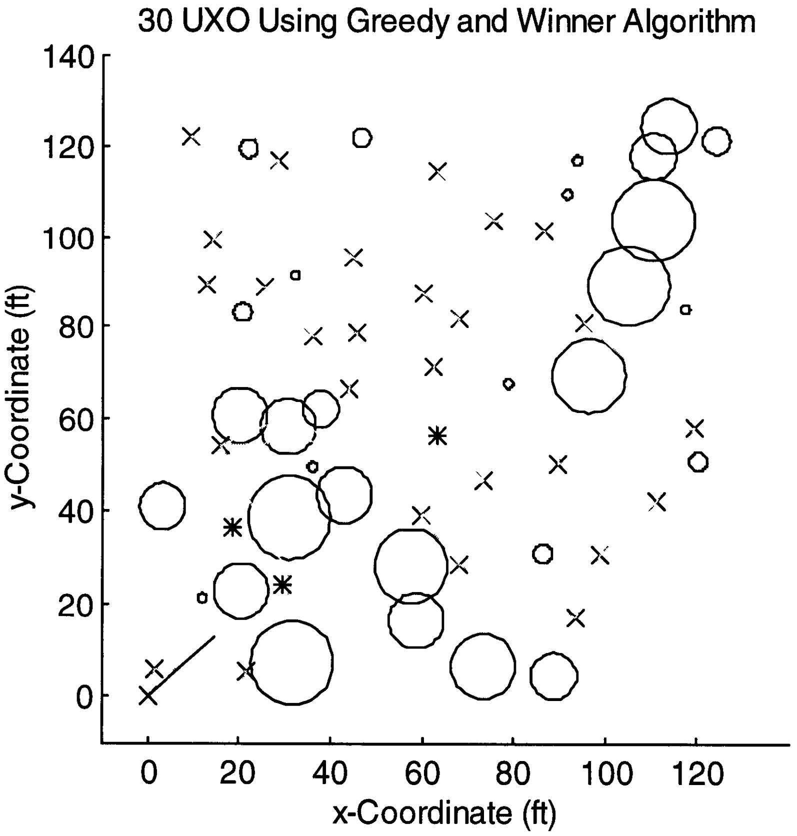

10 UXO Using Mixed and SFC Algorithm
y-Coordinate (ft)
x-Coordinate (ft)
0
20
40
60
80
100

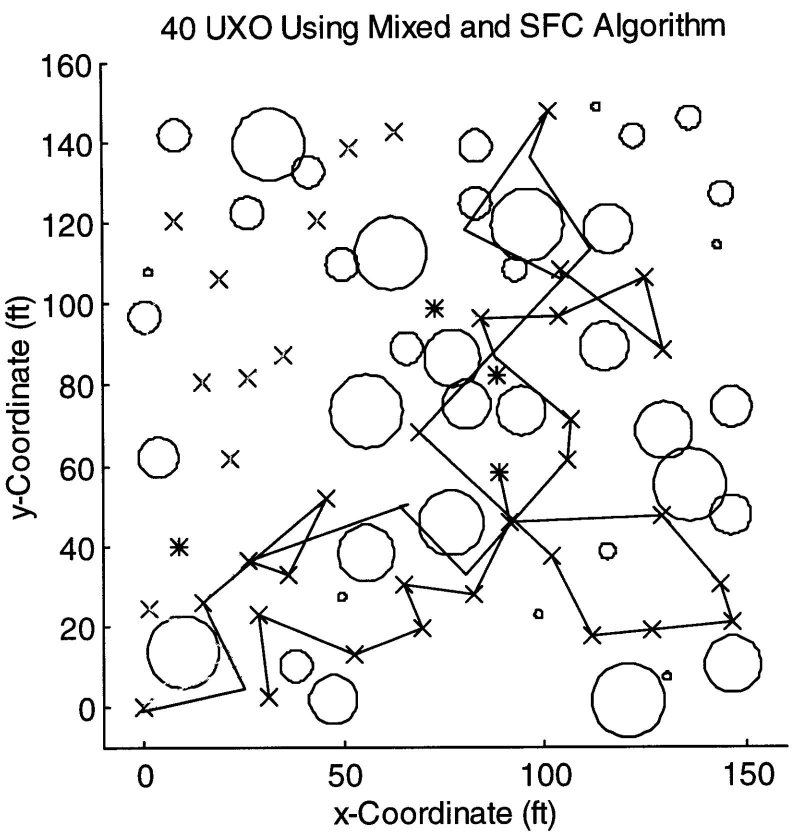

10 UXO Using Greedy and Winner Algorithm
y-Coordinate (ft)
x-Coordinate (ft)
0
20
40
60
80
100

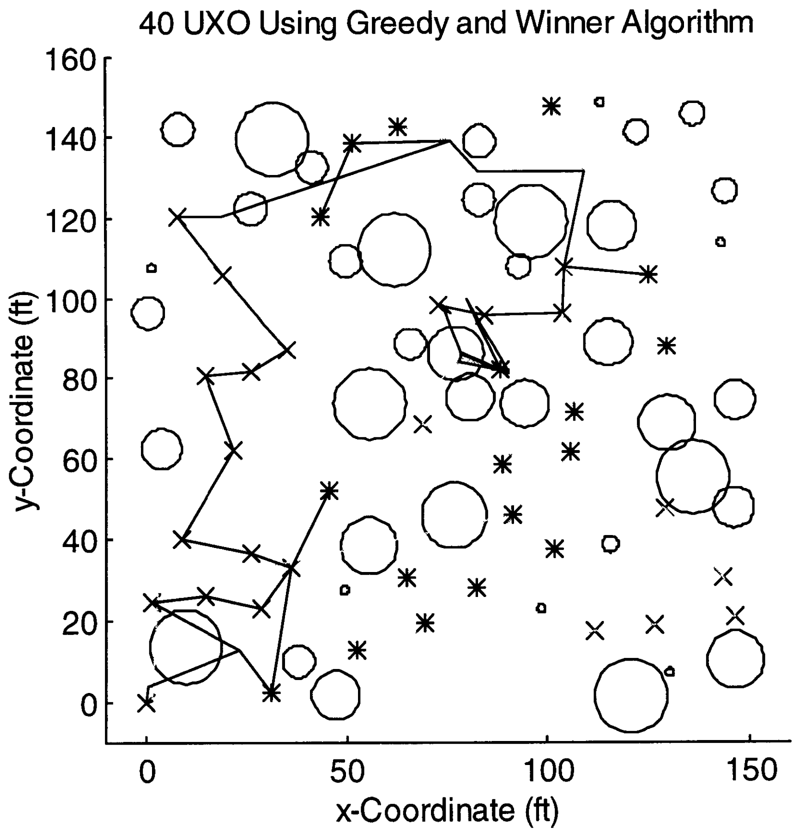

20 UXO Using Mixed and SFC Algorithm
y-Coordinate (ft)
x-Coordinate (ft)
0
20
40
60
80
100


20 UXO Using Greedy and Winner Algorithm
y-Coordinate (ft)
x-Coordinate (ft)
0
20
40
60
80
100

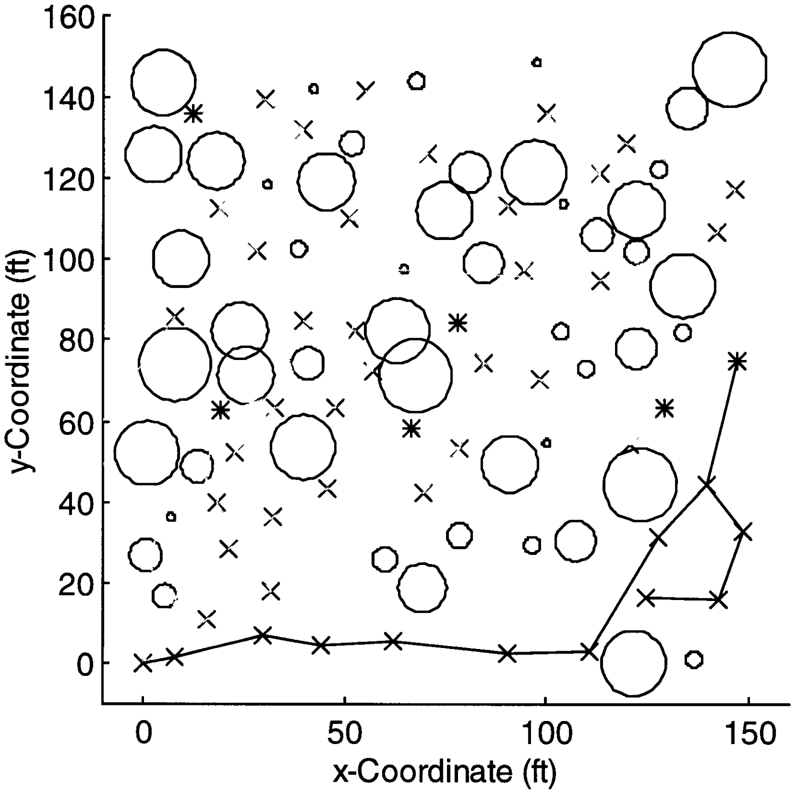

30 UXO Using Mixed and SFC Algorithm
140
120
100
80
60
40
20
0
y-Coordinate (ft)
0
20
40
60
80
100
120
x-Coordinate (ft)


30 UXO Using Greedy and Winner Algorithm
140
120
100
80
60
40
20
0
y-Coordinate (ft)
0
20
40
60
80
100
120
x-Coordinate (ft)

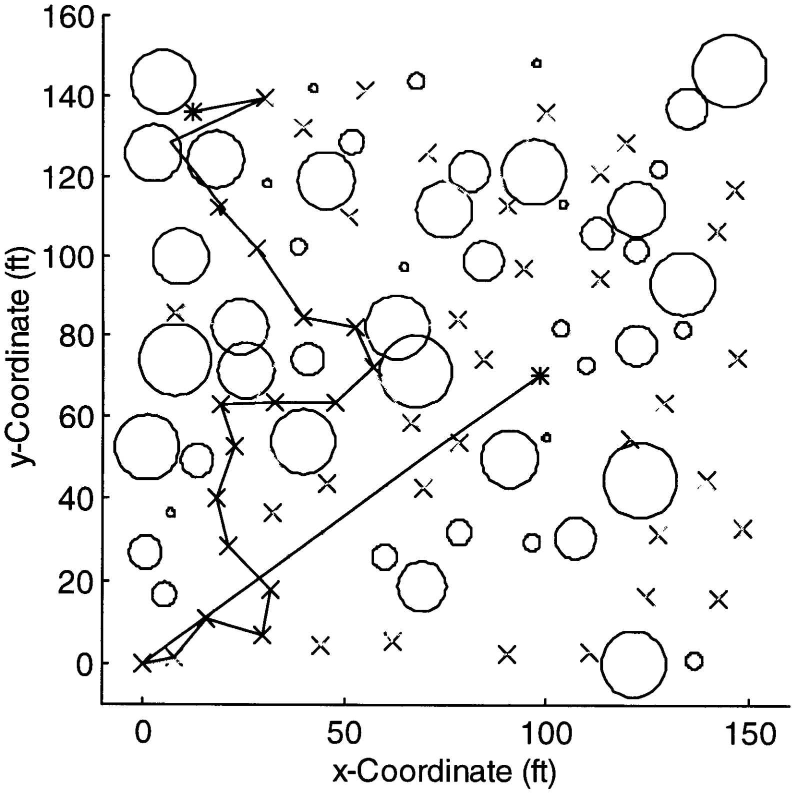

40 UXO Using Mixed and SFC Algorithm
160
140
120
100
80
60
40
20
0
y-Coordinate (ft)
0
50
100
150
x-Coordinate (ft)

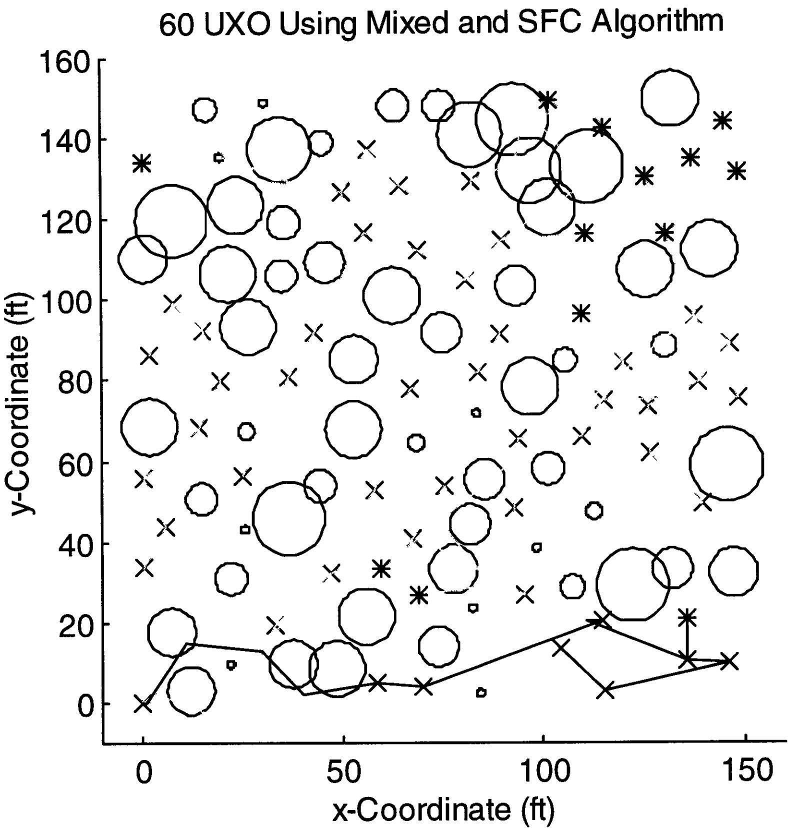

40 UXO Using Greedy and Winner Algorithm
160
140
120
100
80
60
40
20
0
y-Coordinate (ft)
0
50
100
150
x-Coordinate (ft)

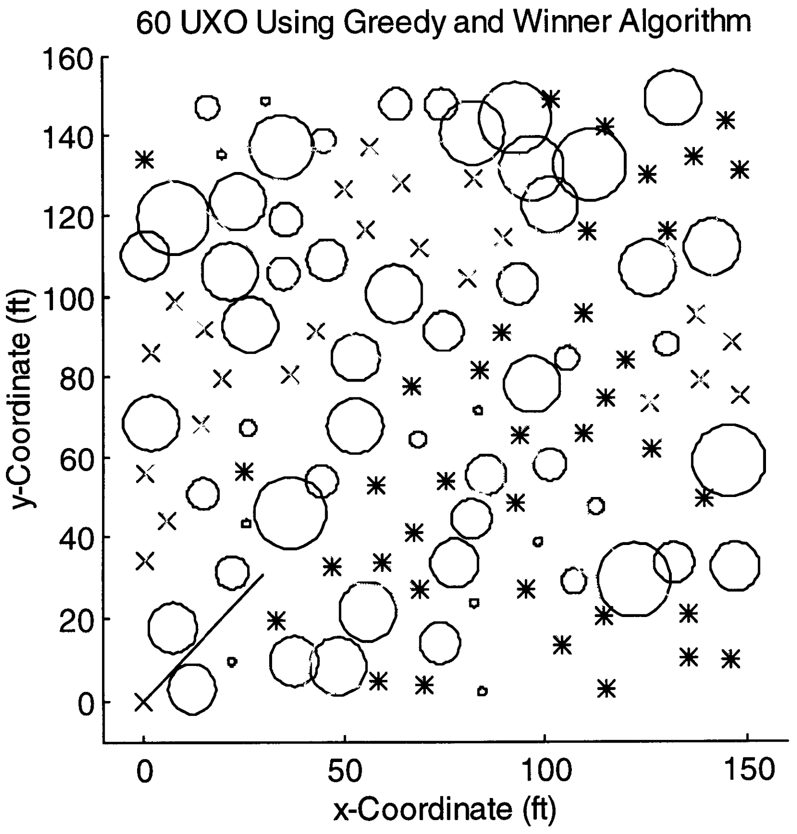
50 UXO Using Mixed and SFC Algorithm

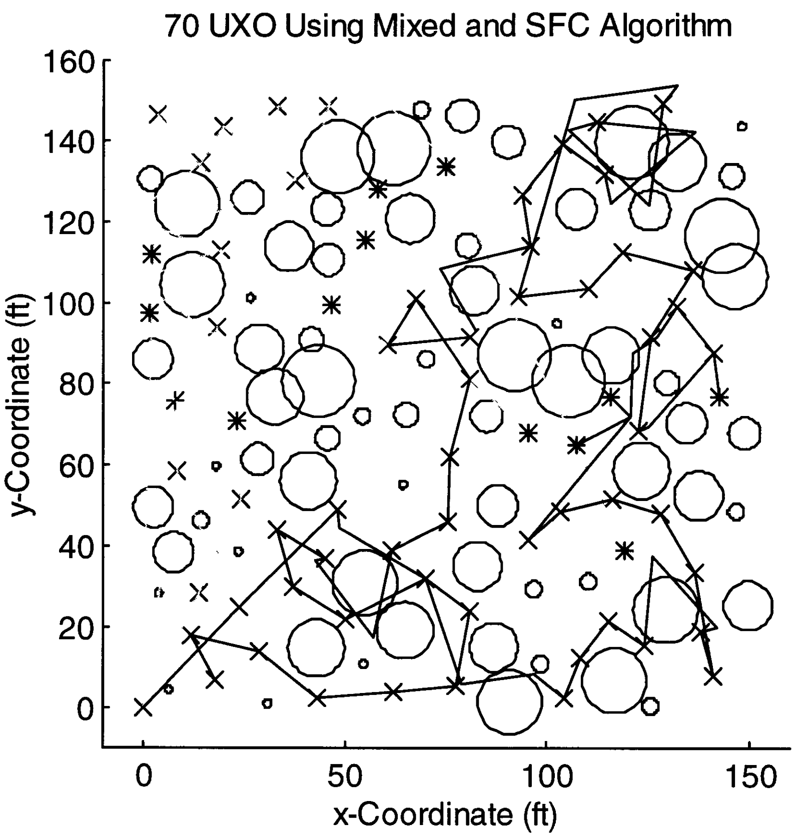


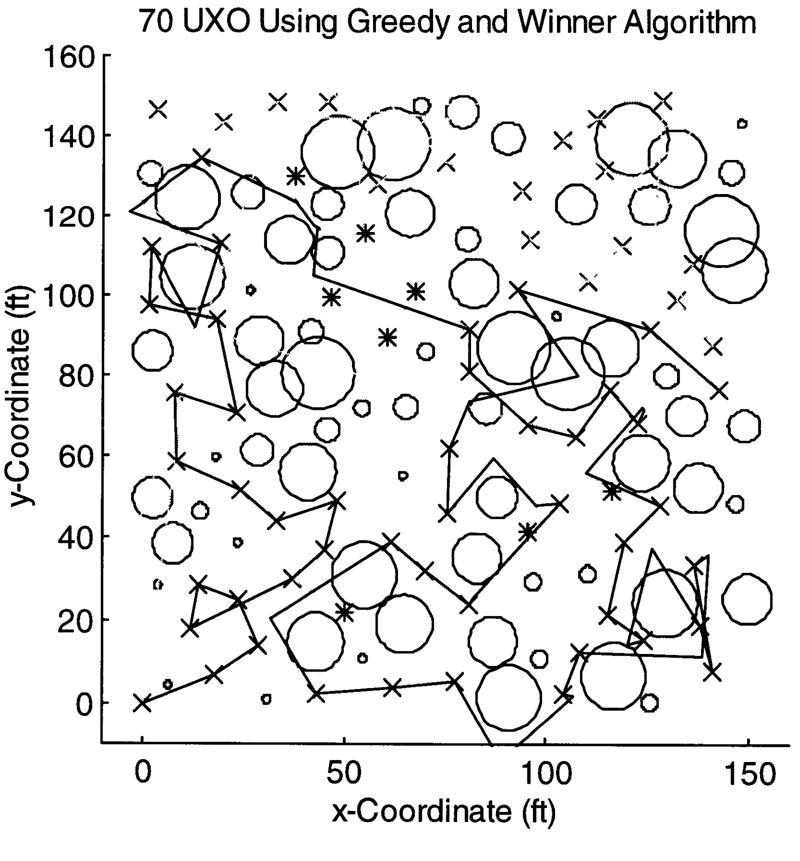
50 UXO Using Greedy and Winner Algorithm



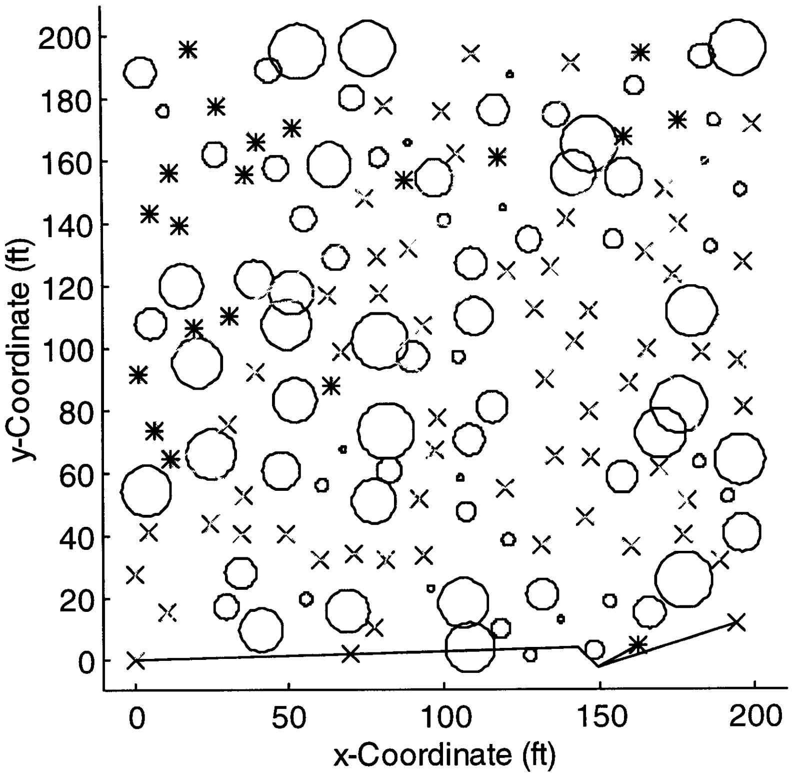

60 UXO Using Mixed and SFC Algorithm
160
140
120
100
80
60
40
20
0
y-Coordinate (ft)
0
50
100
150
x-Coordinate (ft)


60 UXO Using Greedy and Winner Algorithm
160
140
120
100
80
60
40
20
0
y-Coordinate (ft)
0
50
100
150
x-Coordinate (ft)

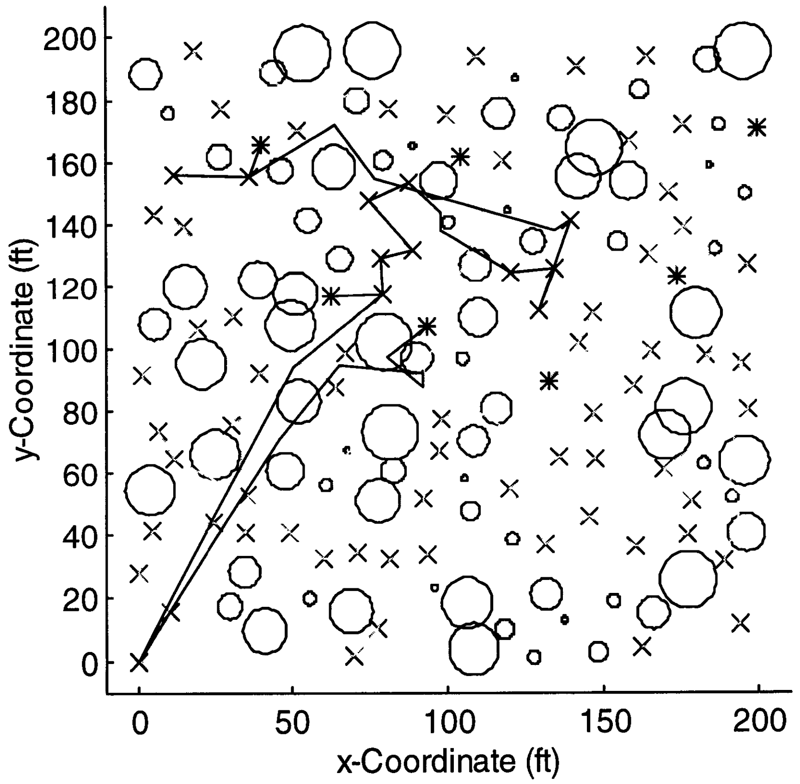

70 UXO Using Mixed and SFC Algorithm
160
140
120
100
80
60
40
20
0
y-Coordinate (ft)
0
50
100
150
x-Coordinate (ft)


70 UXO Using Greedy and Winner Algorithm
160
140
120
100
80
60
40
20
0
y-Coordinate (ft)
0
50
100
150
x-Coordinate (ft)

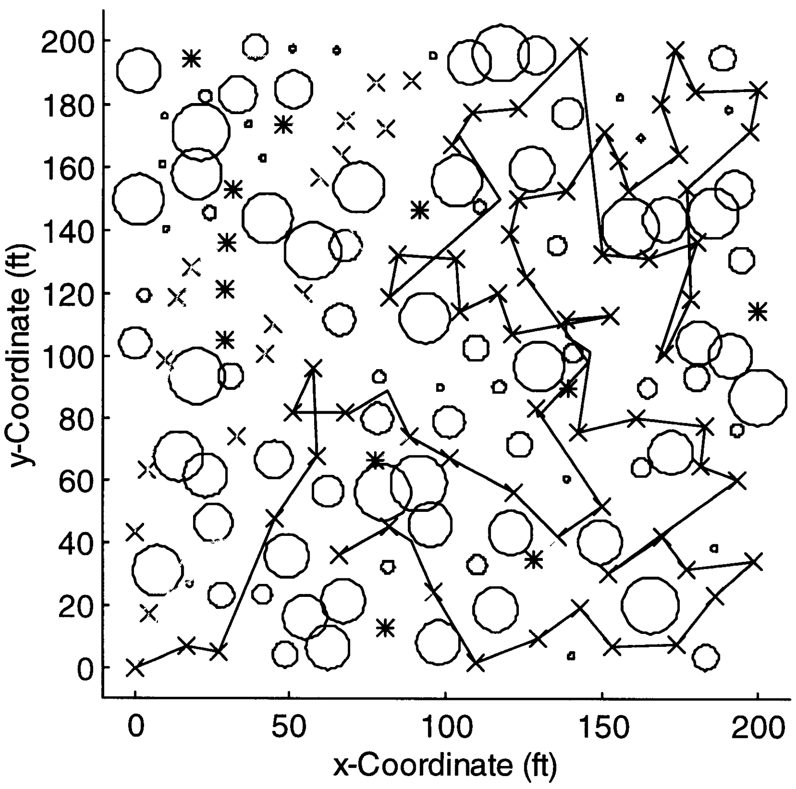
80 UXO Using Mixed and SFC Algorithm



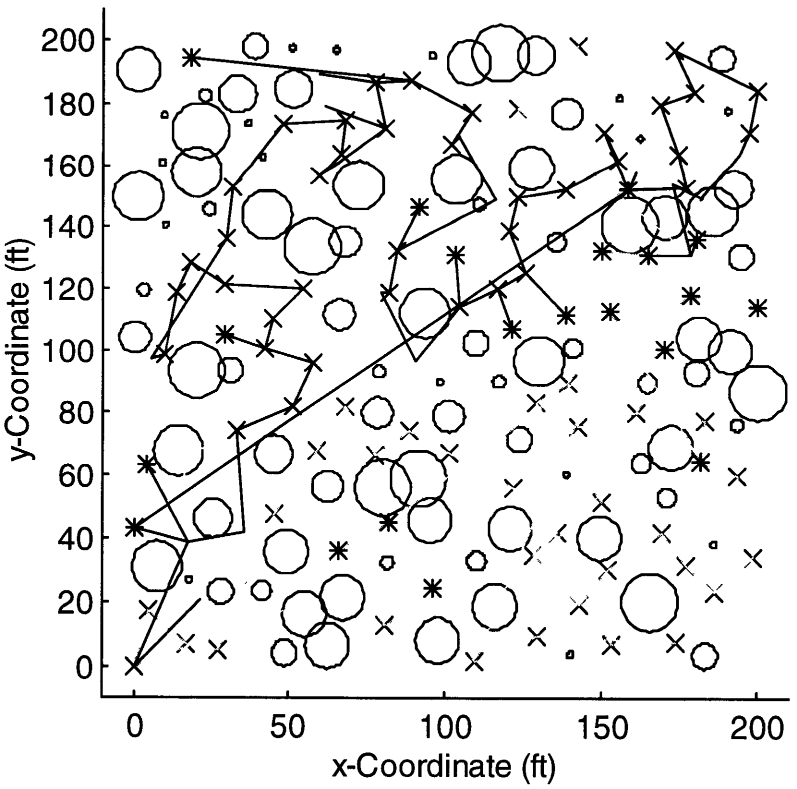
80 UXO Using Greedy and Winner Algorithm



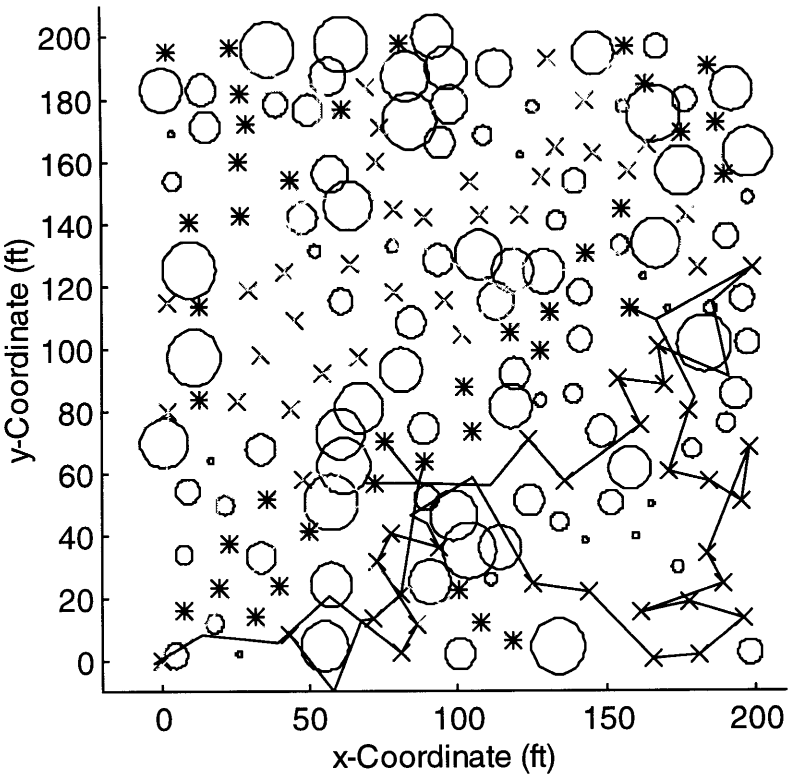
90 UXO Using Mixed and SFC Algorithm



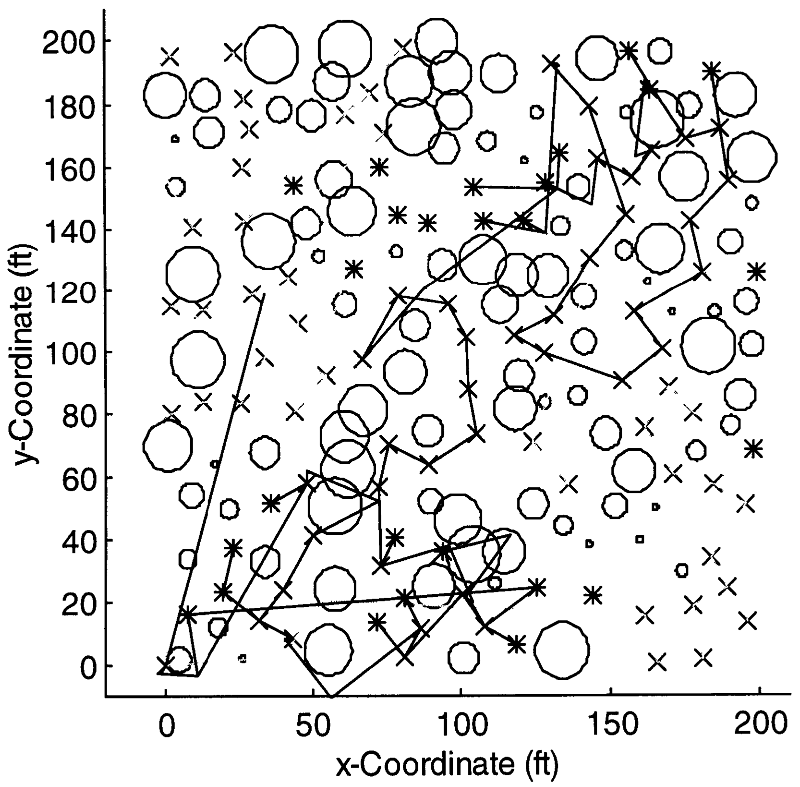
90 UXO Using Greedy and Winner Algorithm



100 UXO Using Mixed and SFC Algorithm



100 UXO Using Greedy and Winner Algorithm



Chapter Seven : Conclusion

7.1 Summary

The annotated example test that has been presented in the Chapter 6 has shown that the selected planning/re-planning and managing concepts are indeed reasonable and feasible to accomplish a typical road-building mission. Furthermore, the random test results in Chapter 6 give guidelines for using the proposed mission manager to build obstacle-free roadways for the UXO clearing mission. One of the important guidelines is that there is a limit on the number of vehicles used in accomplishing a given road-building mission. In other words, an optimal number of vehicles does exist depending on the mission and the road planning algorithm. Another important guideline is that many road planning algorithms must be run to find the most optimal solution to a given MVRP. No particular planning algorithm works best for every given mission. For instance, the two proposed routing algorithms (i.e. mixed and greedy) seem to have both advantages and disadvantages. In the missions with many obstacles, the mixed algorithm usually presented much more optimal routing solutions. In the missions with the simplest obstacle map (i.e. almost no obstacles), the greedy usually yields better solutions than the mixed algorithm. However, the user must note that the two algorithms have been originated from the solution methods that solve different problems. Likewise, the distributing algorithms (i.e. SFC and "winner") have both advantages and disadvantages. The "winner" distributing algorithm resulted in much more evenly distributed road plans for the multiple vehicles. Both SFC and "winner" distributing algorithms can be improved by adding simple modifications in the procedure.

7.2 Merits of the Approach

Although many improvements to the mission manager must be developed, the currently developed road-building mission manager satisfies the baseline work of the road-building mission requirement. One of the important issues that has been drawn from the testing the planning algorithms is that the total mission time is essentially driven by the efficiency in distributing the workload among the vehicles. From the simulated test results, the mission time for a typical road-building mission is on the order of 30 minutes to visit 100 UXO nodes using 3 vehicles. This is considered to be a reasonable rate of exploring for the obstacle-free roads.

The planning/re-planning time using both routing algorithms is significantly smaller than was expected. Both of the algorithms have been designed to use the procedures that will lead to the least computational time and memory storage. Although the currently developed software for the mission manager still requires further development, the tested results show that it has potential to be a "good" autonomous manager for coordinating the multiple vehicles system. The integrated manager software for the road-building mission and the planner software for the UXO-clearing mission can definitely improve the current slow and expensive UXO-clearing process.

7.3 Suggestions for Further Development

Although four combinations of the routing and distributing algorithms have been proposed, only two combined algorithms have been fully implemented to test the efficiency. The mixed routing with the SFC distributing algorithm and the greedy routing with the "winner" distributing algorithm have been fully implemented and tested. Thus, the other two combined algorithms (the mixed with the "winner" and the greedy with the SFC algorithms) should be implemented to provide options for an initial plan. Furthermore, some other heuristic methods that will generate more optimal solutions can be derived from the various existing algorithms to provide more options for an initial road plan. One of the serious challenges in the planner or re-planner is to optimize the distribution of the workload among the available vehicles. Thus, the distributing algorithm in the planner or re-planner should definitely be improved to distribute the workload evenly among the vehicles. This will allow the vehicles to build obstacle-free roads in the least time.

The mission problem with too many UXO nodes and for obstacle nodes requires the usage of some partitioning method. This method will partition the large problem into many smaller problems that can be managed by the currently implemented mission manager. A simple partitioning concept has been proposed but has not been implemented (i.e. coded) to test its functionality. Thus, this function should be implemented fully to solve a large mission problem. Likewise, the mission problem with multiple home nodes needs to be developed further. A heuristic method has been proposed to deal with the multiple home nodes but has not been implemented. Therefore, this function should also be coded to test its functionality.

The selected mission manager satisfies the minimal requirements of the road-building mission. The mission manager should definitely be improved to manage the real-time operations of the multiple EOD vehicles system. The selected verification manager deals only with the primary events that may occur during the verification process (i.e. avoiding obstacles). Some other real-time managing issues (i.e. detonated or idle vehicle) have been given with some proposed conceptual solutions; however, they haven't been implemented to test functionality. Furthermore, many other issues must be solved for the mission manager to truly manage the real-time operations of the EOD mission. For instance, EOD vehicles have some constraints such as limited battery lifetime, limited turning capability, and others so that the mission manager must account for their constraints.

The most critical constraint is associated with the limited battery lifetime. The battery packs that are chosen for EOD vehicles only last about 30 minutes at most. The batteries take several hours to recharge. Thus, each of the vehicles must have several battery packs reserved to them to operate a normal road-building mission. Since the charged batteries will be available for the vehicles at a station that is at a specific location, the mission manager must make sure that none of the vehicles run out of power before they reach the service station during the mission. Therefore, the distances that the vehicles travel and the distance that the vehicles will have to travel back to the station for new batteries have to be updated periodically. To simplify the problem, an initial plan can still be created using the method discussed in Chapter 2 and just keep track of the battery usage for each of the vehicles. Furthermore, the service station can be assumed to be located at the initial location of the vehicles (i.e. home node). Thus, the mission manager keeps track of the vehicle's traveled distance and the distance to the service station before the battery runs out. Then, the manager decides when a vehicle must travel back to the station for a new battery pack. The vehicle will be traveling back to the station by following the route that has already been verified to be obstacle-free so that it can travel at the fast speed. Since the vehicle can travel up to 6 fps in the fast-speed mode, the manager should divide the distance to travel back to the station by a factor of 6. In other words, if a vehicle has traveled 60 feet then the distance to travel back to the station is 10 feet due to the fast speed. Once the battery pack is replaced, the vehicle resumes the mission and travels back to the place where it had left off at the fast speed.

The turning constraint on the vehicles is another important issue that must be solved for real-time operations. EOD vehicles are theoretically able to turn 180 degrees without any problem (i.e. tank-turn). However, the tank turns require much more time than a smaller angle turn. Therefore, the mission manager must make sure that the vehicles are given enough time to carry out the large angle turns to prevent the vehicle losing track of the heading.

The mission manager software for the road-building phase needs to be integrated into the mission manager for UXO-clearing phase. As mentioned in the Chapter 1, the mission manager software has been implemented by another MIT student. Therefore, the two high-level software tools can be interfaced to provide the complete mission planner for the EOD mission. Furthermore, the combined verification manager and verification manager software needs to be integrated into the real-time simulation. This will allow the real-time interactions between the mission manager and the actual vehicles.

Appendix A: IUVC (Intelligent Unmanned Vehicle Center)

The IUVC was first established in 1992 as the Planetary Rover Baseline Experiment (PROBE) Laboratory at the Charles Stark Draper Laboratory. This laboratory started to research, design, develop, and manufacture a small autonomous vehicle for planetary exploration. These vehicles were originally designed to meet specifications set by NASA for the MESUR Pathfinder Mission[19] [11]. The small, lightweight, and autonomous vehicles are suitable for the space exploration missions since they reduce the cost of transportation to the planet surface. The first prototype MITy-1, which will become the basis for future vehicle designs in IUVC, is a proof-of-concept platform. This vehicle is equipped with an earthbound sensor suite including three acoustic range finders and a compass. After the analysis of the test data from the first prototype, a second vehicle MITy-2 (Figure A-1) was developed. This vehicle was designed to be much more intelligent and autonomous. For extensive research of the mechanical platform, a third vehicle was designed (Figure A-2). This vehicle doesn't have intelligence but it allowed IUVC to research into various steering methods in a variety of terrains.

The MITy-series of micro-rovers led the PROBE Laboratory to evolve into the Unmanned Vehicle Laboratory (UVL) in 1994. This laboratory was responsible for investigating possible applications of autonomous and semi-autonomous robots in the real life scenarios. The laboratory was also responsible to provide possible design concepts and developed prototypes for a particular application. Then, the IUVC replaced the UVL in April, 1996 [11]. This center is still responsible for the research of unmanned, intelligent, and autonomous vehicles system for a variety of applications. Thus, the primary function of the IUVC is to design and develop platforms for small and autonomous vehicles. The IUVC has designed and developed both autonomous and semi-autonomous vehicles for air, land, water, and space terrain.

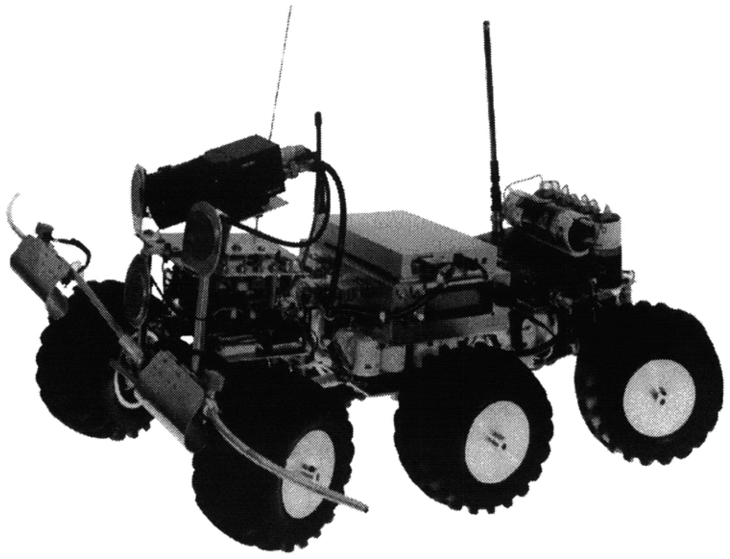


Figure A-1: MITy-1 Proof-of-Concept Vehicle

[19] The Mars Environmental Survey (MESUR) Pathfinder Mission proposed the landing of a probe on Mars to observe the planet's surface and atmosphere. Mars Pathfinder will land on Mars, open up and allow a six-wheeled robotic rover to drive out and begin exploring the Martian terrain. Pathfinder was launched on December 4, 1996 aboard a Delta rocket.

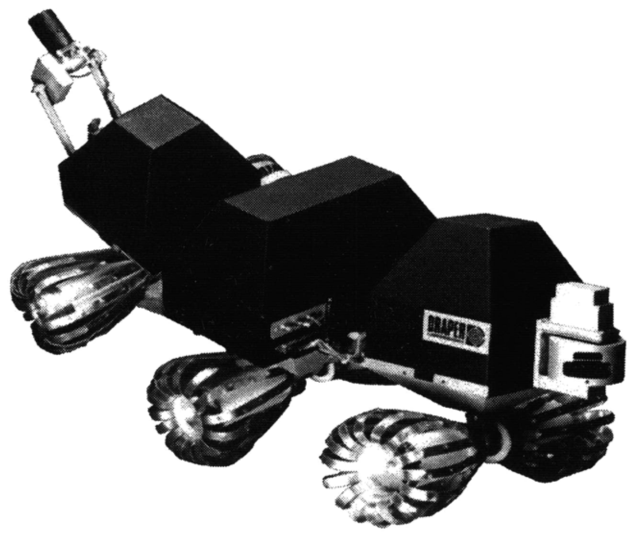


Figure A-2: MITy-3 Prototype Vehicle with Conical Wheels and an Instrumental Arm

A.1 Micro-Rover

Most of the research projects in the IUVC has been the development of small robotic vehicles, referred to as micro-rovers. These vehicles are generally small, lightweight, and affordable. The laboratory developed three different proof-of-concept rovers. The first vehicle called MITy-1 demonstrated the autonomy and mobility concepts and became the baseline of the future designs. This vehicle was developed for earth-based missions and was equipped with basic computing power and off-the-shelf sensors. The onboard processor is an M68HC11 micro-controller with motor drivers, analog-to-digital conversion (A/D), and digital I/O lines [13]. Navigation is done by using a magnetic compass and a drag wheel to determine heading and position respectively. A sonar range finding module, switch-actuating bumpers are used to detect hazard and avoid obstacles. MITy-1, which uses only four feet wide path, is able to navigate through terrain filled with random obstacles with full autonomy. This vehicle is also able to climb rocks, curbs, and ledges that are taller than half its height. The capabilities of obstacle avoidance and hazard sensing are the elements, which led the vehicles to be completely autonomous. These capabilities also increased system's reliability and effectiveness. Some of the sensors on MITy-1 are 3 acoustic range finders for obstacle avoidance, bumpers for collision detection, electronic compass for heading, and odometer for distance. The key pieces of hardware on this vehicle are video camera and transmitter, wireless command and control link, and microprocessor [11].

The second vehicle in the MITy series showed major improvement in the overall performance such as mission-worthy sensors and advanced algorithms. A ground-based control station for MITy-2 has been developed to monitor and control the vehicle by a radio-modem link. MITy-2 sensor suite includes a scanning laser range finder for obstacle avoidance, gyroscope and a drag wheel with optical encoder for navigation, bumpers for collision detection, two forward-looking proximity and tilt sensors for cliff detection and terrain slope measurement, a customized sun sensor for heading, and odometer and tachometers for distance and speed control. Unlike MITy-1, which is limited to earth-based operation, MITy-2 is able to perform autonomous missions on the moon or Mars using the various sensors listed above. Furthermore, MITy-2 has a modular hardware and software design. In other words, other types of platform can replace each platform on the vehicle. This design concept is appropriate for changing sensor, processor, and power systems to either replace defected module or carry out different mission. This vehicle also has path-planning capability to accomplish a mission autonomously. The key pieces of hardware on this vehicle are same as MITy-1 except for sonar array [11].

A.2 Companion

The design of a sensor fusion package led to the test platform called Companion (Figure A-3) which is built on an electric wheelchair. This vehicle is a larger system with onboard hierarchical planning and mapping capabilities. The Companion includes two 486 laptop computers, one runs the path planning [3] and map building [18] software and the other is used to interface the software to the hardware.

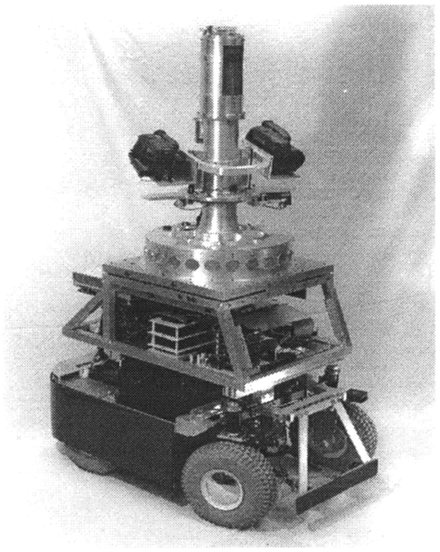


Figure A-3: "Companion" Prototype Vehicle - 5 Feet Tall

A.3 Aerial Vehicle

An autonomous helicopter was developed to enter the Sixth Annual International Aerial Robotics Competition in 1996. This helicopter (Figure A-4) can take-off, fly, and land completely autonomous. One of the mission requirements in the competition was to identify hazardous waste symbols on barrels in a field by using a complex vision system. Some of the instrumentation on the vehicle include Novatel RT-20 differential GPS, six-DOF inertial measurement unit, downward-looking video camera, compass, and ultrasonic altimeter sonar [19].

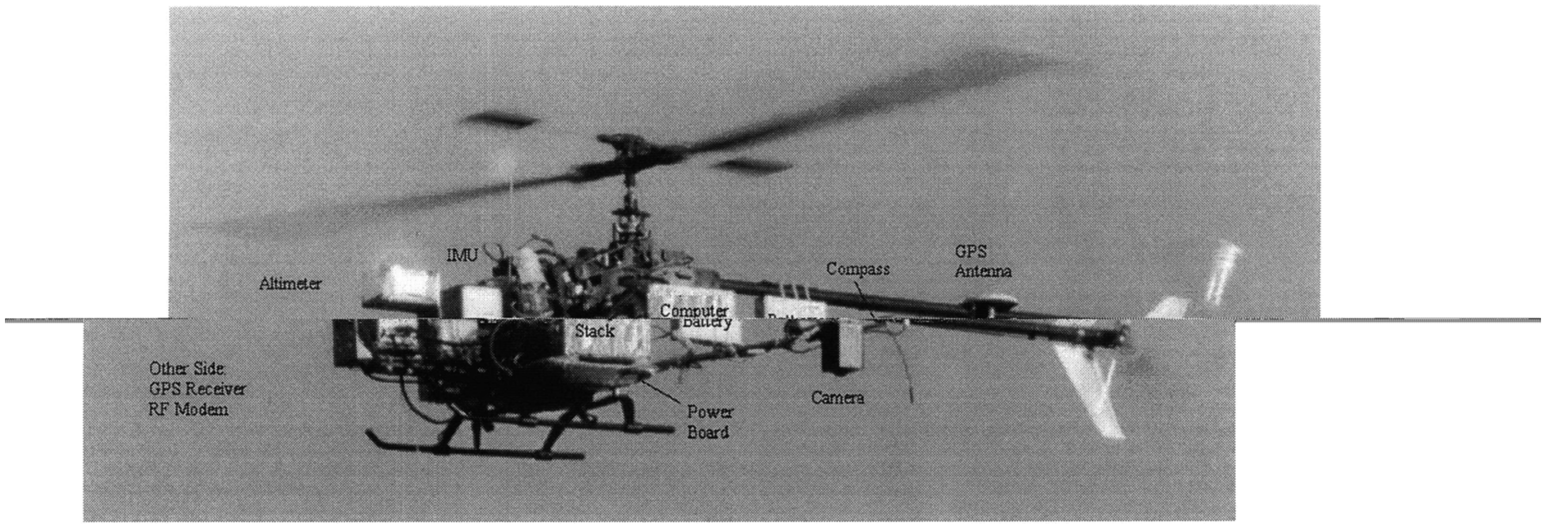


Figure A-4: Draper Small Autonomous Aerial Vehicle (DSAAV)

A.4 Undersea Vehicle

A highly maneuverable autonomous undersea vehicle (Figure A-5) is being developed based on the swimming motion of a fish. The flexible-hull undersea vehicle is driven by a propulsion system based on the manipulation of the wake vorticity created by foil flaps positioned at the side of the vehicle. The objective of this research is to combine a submarine, which has the long range and high endurance, and a remotely operated undersea vehicle [11].

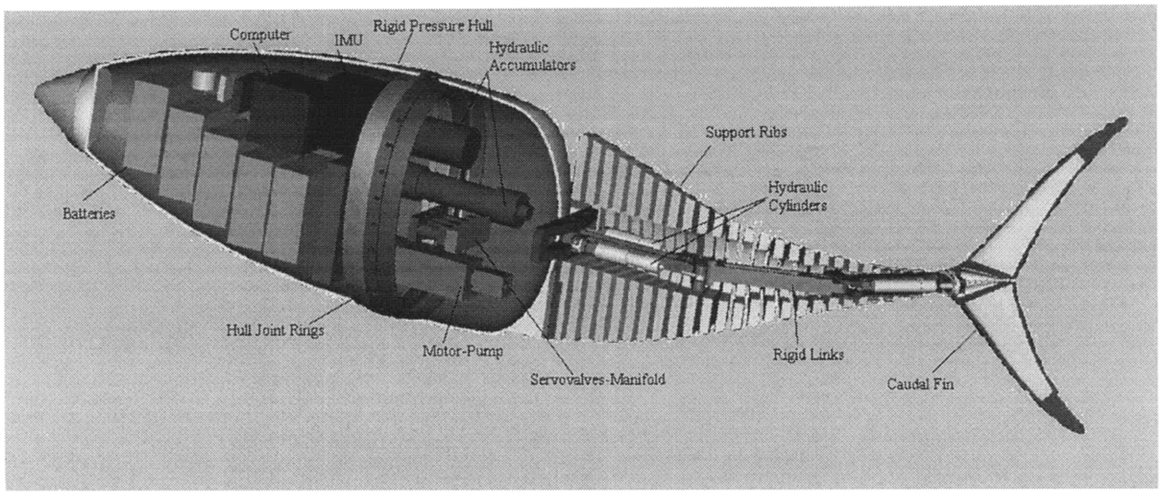


Figure A-5: Vorticity Control Unmanned Underwater Vehicle (VCUUV) System Design

Appendix B: Formulation of TSP and VRP

B.1 The Traveling Salesman Problem (TSP)

Problem Definition: A traveling salesman has to visit *n* cities or customers from a depot. He must visit each of the other *n*-1 cities only once and return to the depot that is the *n*th city or node. The travel cost between any two cities, say from city *i* to *j*, is represented as c_{ij} in a cost matrix denoted by *C*. This travel cost may be in terms of distance, time, money, or others. The problem is to create a tour through all the *n* cities with the minimum total cost of the tour. This is a classical operations research problem of unconstrained TSP. An Euclidean TSP is when the cities that need to be visited all lie on the same plane and the travel cost between any two cities is the Euclidean distance between them. This unconstrained TSP can become constrained by adding time window. In other words, if t_i is the time that the salesman visits *i*th city, then t_i must satisfy $l_i <= t_i <= u_i$ where l_i and u_i are the lower and upper bounds of a time window. The location of depot, a series of *x* and *y* co-ordinates for all *n* cities and a set of time windows are defined initially for the problem [4].

Mathematical Formulation of TSP: A tour is a chain that goes through all the *n* cities or nodes and that the first and the last nodes coincides. This type of tour is referred to as a Hamiltonian cycle. Let a tour be defined by $t = (i_1, i_2, ..., i_l)$ and the cost of this tour be defined by the Equation (B.1).

$$C(t) = \sum_{j=1}^{n-1} c_{i_j i_{j+1}} + c_{i_n i_1} \tag{B.1}$$

A graph G = {*N*, *A*} consists of *N*, a set of *vertices* or the number of nodes, A, a set of *edges* or the number of arcs that connect the nodes, and a cost of distance c_{ij} for each arc (*i*, *j*). The TSP is the problem of obtaining the minimum cost tour (Equation (B.2)) that connects all the nodes of *N*.

$$MIN \sum_{i,j} c_{ij} x_{ij} \tag{B.2}$$

Equations (B.3) and (B.4) are the constraints that must not be violated. These two constraints guarantee that each node will be visited exactly once.

$$\sum_i x_{ij} = 1 \qquad j = 1, ..., n \tag{B.3}$$

$$\sum_j x_{ij} = 1 \qquad i = 1, ..., n \tag{B.4}$$

Equations (B.5), (B.6), and (B.7) represent constraints that ensures the final solution being a single tour that starts and ends at the depot node [1]. In other words, the sub-tours are eliminated from the tour.

$$Y_i - Y_j + m x_{ij} \leq n - 1 \qquad i \neq j = 2,3,...,n \tag{B.5}$$

$$Y_i = \text{arbitrary real number} \qquad i = 1, 2, ..., n \tag{B.6}$$

$$x_{ij} = \{1, \text{if the salesman goes directly from city i to city j; 0, otherwise}\} \tag{B.7}$$

B.2 Multiple Traveling Salesmen Problem (MTSP)

The multiple traveling salesman problem (MTSP) is easily formulated into a single TSP. *M* salesman requires *M* copies of the origin are in the cost matrix $C = [c_{ij}]$. Each of the *M* copies of the origin indicates a unique stop but has same costs with respect to the other nodes. These copies are also assigned with extremely large cost to avoid inclusion in a solution [1].

B.3 The Vehicle Routing Problem (VRP)

Problem Definition: The vehicle routing problem (VRP) is a subset of TSP. In other words, the VRP is just a constrained TSP. VRP takes into account the vehicle's limits on capacity and operation. Due to the computation limit on computers, the small-scaled problems that involve around 30 nodes are considered only for the exact optimal solutions [4].

Mathematical Formulation of VRP: The formulation of VRP is same as the TSP except the number of vehicles denoted *m* is introduced to define the vehicle's constraint denoted by Q_k. Each customer has an order denoted by r_i and an integer programming technique is used to represent the VRP mathematically. The VRP seeks to minimize the Equation (B.8).

$$\sum_i \sum_j \sum_k c_{ij} x_{ij}^k \qquad \text{(B.8)}$$

Equations (B.9) thorough (B.16) are the constraints that must not be violated. Constraint Equation (B.9) ensures that the kth vehicle visits each node exactly once.

$$\sum_i \sum_k x_{ij}^k = 1 \qquad j = 1, 2, \ldots, n \qquad \text{(B.9)}$$

Constraint Equation (B.10) requires that if a vehicle visits a node, it must leave from the same node.

$$\sum_i x_{ip}^k - \sum_j x_{pj}^k = 0 \qquad \begin{matrix} k = 1, 2, \ldots, K \\ P = 1, 2, \ldots, n \end{matrix} \qquad \text{(B.10)}$$

Constraint Equations (B.11) and (B.12) represent the capacity and operational limits of the kth vehicle.

$$\sum_i \sum_j r_i x_{ij}^k \le Q_k \qquad k = 1, 2, \ldots, K \qquad \text{(B.11)}$$

$$\sum_i \sum_j c_{ij} x_{ij}^k \le D_k \qquad \text{(B.12)}$$

Constraint Equation (B.13) guarantees that a vehicle is used once but only once.

$$\sum_i x_{ij}^k = 1 \qquad k = 1, 2, \ldots, K \qquad \text{(B.13)}$$

Constraint Equations (B.14), (B.15), and (B.16) ensure that the final solution will be a single tour that starts and ends at the depot node. In other words, the sub-tours are eliminated from the tour [1].

$$Y_i - Y_j + n\sum_k x_{ij}^k \le n-1 \qquad i \ne j = 2,3,\dots,n \qquad \text{(B.14)}$$

Y_i = arbitrary real number i = 1, 2, ..., n (B.15)

x^k_{ij} = {1, if the k th vehicle goes directly from city i to city j; 0, otherwise} (B.16)

B.4 Multiple Vehicles Routing Problem

Mathematical Formulation of Multiple VRP: The multiple vehicle routing problem can also be formulated by extending the single vehicle routing case. Assume *V* number of vehicles are available to visit *N* number of nods with the capacity of vehicle *k* be Q_k. Two more variables are assumed to have value of 0 or 1 depending on the conditional Equations (B.17) and (B.18) [8].

Y_{rk} = {1, if task *r* is processed by vehicle *k*; 0, otherwise} (B.17)

x^k_{ijk} = {1, if vehicle *k* visits from node *i* to *j* at step *t*; 0, otherwise} (B.18)

The multiple VRP seeks to minimize the cost Equation (B.19).

$$MIN\sum_k \sum_t \sum_i \sum_j c_{ij} x_{ijk}^t \qquad \text{(B.19)}$$

Equation (B.20) assigns each task to one available vehicle.

$$\sum_{k=1}^{V} Y_{rk} = 1 \qquad r = 1, \dots, N \qquad \text{(B.20)}$$

Expression (B.21) and (B.22) ensure that each node is visited only by the vehicle to which the node is assigned.

$$\sum_{t=1}^{M}\sum_{j=1}^{M} x_{a(r)jk}^t = Y_{rk} \qquad r = 2, \dots, N \qquad \text{(B.21)}$$

$$\sum_{t=1}^{M}\sum_{j=1}^{M} x_{b(r)jk}^t = Y_{rk} \qquad k = 2, \dots, V \qquad \text{(B.22)}$$

Constraint Equation (B.23) requires that if a vehicle visits a node, it must leave from the same node.

$$\sum_i x_{ip}^k - \sum_j x_{pj}^k = 0 \qquad k = 1, 2, \dots, K \quad P = 1, 2, \dots, n \qquad \text{(B.23)}$$

Constraint Equations (B.24) and (B.25) represent the capacity and operational limits of the k th vehicle.

$$\sum_i \sum_j r_i x_{ij}^k \le Q_k \qquad k = 1, 2, \dots, K \qquad \text{(B.24)}$$

$$\sum_{i}\sum_{j} c_{ij} x_{ij}^{k} \le D_k \qquad \text{(B.25)}$$

Constraint Equation (B.26) guarantees that a vehicle is used once but only once.

$$\sum_{i} x_{ij}^{k} = 1 \qquad k = 1, 2, ..., K \qquad \text{(B.26)}$$

Constraint Equations (B.27), (B.28), and (B.29) ensure that the final solution will be a single tour that starts and ends at the depot node. In other words, the sub-tours are eliminated from the tour [1].

$$Y_i - Y_j + n\sum_{k} x_{ij}^{k} \le n - 1 \qquad i \ne j = 2,3,\ldots,n \qquad \text{(B.27)}$$

Y_i = arbitrary real number   i = 1, 2, ..., n   (B.28)

x^{k}_{ij} = {1, if the kth vehicle goes directly from city i to city j; 0, otherwise}   (B.29)

Therefore, the multiple VRP requires *V* times as many variables and constraints as the single VRP case. However, neither the single nor the multiple VRP case will not be solved by direct optimization techniques. Either heuristic or mixed algorithms can be applied to solve the single as well as the multiple VRP cases for reasonable problem size.

Appendix C: Details of the Optimal Algorithms

An optimal solution approach involves considering every possible solution to the TSP and VRP. This exhaustive search method guarantees the best solution to a problem. The optimal solution method will be efficient to solve a problem that has a reasonable number of possible solutions. However, this method will be inefficient and often impossible to solve a problem that has an "infinite" number of possible solutions. TSP and VRP often have a great number of possible solutions.

Although many optimal solution methods exist for TSP or VRP, the following algorithms have been selected to illustrate few of the many existing optimal algorithms [1] [8] [12] [15]. The first three algorithms are the most popular exact algorithms (branch-and-bound, integer programming, and dynamic programming methods) to solve a general TSP.

C.1 Branch-and-Bound

Branch-and-bound method simply creates partial tours or sub-tours, calculates the lower bounds for each of the sub-tours, and compares them to the upper bound (i.e. maximum distance of the optimal tour). If any of the lower bounds for the sub-tours exceeds the upper bound then that particular sub-tour is eliminated. The following are the expressions that define the assignment problem of TSP where the objective is the most efficient assignment of *n* vehicles to *n* nodes. The expression (C.1) represents the objective function that is to minimize the cost Equation.

$$MINz\sum_{i,j=1}^{n} c_{ij}x_{ij} \tag{C.1}$$

The two Equations (C.2) and (C.3) represent the constraints that ensures each node will get visited once but only once.

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad i = 1, 2, ..., n \tag{C.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i \neq j = 2,3,\ldots,n \tag{C.3}$$

x_{ij} = {1, if the salesman goes directly from city i to city j; 0, otherwise (C.4)

The assignment problem of the TSP may be solved easily in polynomial time by relaxing the constraints. The optimal solution to the assignment problem that is the relaxed TSP is not a feasible solution to the basic TSP. In other words, the solution using the two constraint equations above is not guaranteed to reduce to a single tour. Thus, the two sets of the equality constraint equations are necessary but not sufficient for the standard TSP. However, this solution contains *Z* value that is a lower bound on the cost of the TSP. Furthermore, any feasible TSP solution marks an upper bound on the problem. The solution to the assignment problem may not be feasible to the TSP if sub-tours exist. Then, one should branch into *k* (the number of arcs in one of the sub-tours) sub-problems. Assigning an infinite or large cost to one of the k sub-tour arcs in each of the sub-problems eliminates the sub-tours. These *k* assignment problems are solved to compute the bounds. The process runs again if either any new sub-tours exist or the calculated lower bound is less than the best feasible tour found previously. The iterations of the process guarantee the optimal solution. However, the size of the branching tree can be large for complicated problems. But if tight bounds are provided then the size of the branching tree will be reduced greatly. Therefore, tight bounds are necessary conditions for a good branch-and-bound algorithm. This algorithm should not only eliminate sub-tours but also achieve feasibility of the solution [1].

The following are the steps of the solution approach to the branch-and-bound algorithm in general terms. The set *activeset* holds the "live" nodes[20] at any point and *U* that is an upper bound on the optimal cost holds the cost of the best complete solution at any point in time [15]:

STEP 1: INITIALIZATION
activeset = {0} where 0 refers to the original problem
U = a large number
currentbest = any number

STEP 2: CANDIDATE TOUR SELECTION
Select a branching node *k* from the *activeset* and remove this node from the *activeset.* Then go to *STEP 3*. Otherwise, *stop*, there are no live nodes in the *activeset.*

STEP 3: BRANCHING
Generate the children nodes $i = 1, \ldots, N_k$ for the branching node *k* and the corresponding lower bounds Z_i. For each child node *i*, repeat *STEPS 3.1* through *3.2* and at the end go to *STEP 4.*

STEP 3.1: If $Z_i \geq U$ then eliminate child node *i* and go to next child node. Otherwise, go to *STEP 3.2.*

STEP 3.2: If child node *i* is a complete solution then $U = Z_i$ and *currentbest* = child node *i.* Then go to next child node. Otherwise, add child node *i* to *activeset* and go to the next child node.

STEP 4: Go to the next branching node in the *activeset* and go to *STEP 3.* Otherwise, *stop*, there are no more live nodes in the *activeset.* The *activeset* contains the best tour solution with *U* as the optimal cost solution.

The procedure listed above shows that the sub-tour with the least lower bound among the live sub-tours will be considered for further branching. The branching idea creates complications for large branching trees; however, it minimizes computer storage requirements. The upper bound *U* is updated in the *STEP 3.2* when a complete tour that has smaller value than the required solution is obtained [15].

C.2 Integer Programming

A node-covering problem such as TSP or VRP can be formulated using integer-programming concept.

Some notations are defined as the following to be used in the formulation [15].

Let Nodes → 1, 2, …., N.
C_{jk} → the distance from node j to node k.
C_{jj} → make this distance a very large number such that it will not be considered.
$X_{jk} = \{$ 1 if the vehicle goes from node j to node k; 0 otherwise.
U_j → order in which nodes are traversed.

The expression (C.5) represents the objective function that is to minimize the cost equation.

$$MINz \sum_{\substack{i,j=0 \\ i \neq j}}^{n} c_{ij} x_{ij}$$

[20] These nodes are the remaining nodes from which branching is still possibly fruitful.

$0 \le x_{ij} \le 1$ all i, j

x_{ij} integer all i, j (C.5)

The Equations (C.6), (C.7), and (C.8) are the necessary constraining linear system of equations for TSP.

$$\sum_{j=1}^{N} x_{jk} = 1 \qquad \text{for all k} \leftarrow \text{arrive once} \qquad (C.6)$$

$$\sum_{k=1}^{N} x_{jk} = 1 \qquad \text{for all j} \leftarrow \text{arrive once} \qquad (C.7)$$

$$u_j - u_k + N x_{jk} <= N - 1 \qquad \text{where j is not equal to k, } j = 2, \ldots, N, \; k = 2, \ldots, N. \qquad (C.8)$$

So if X_{jk} is 1 then the Equation (C.8) yields $U_j = U_k - 1$ which simply indicates that j^{th} node must be visited before k^{th} node is visited. Thus, this eliminates any sub-tour for all X_{jk}.

However, for EOD vehicle routing problem the equalities in the two constraining Equations (C.6) and (C.7) can be relaxed and set to inequalities. Thus, TSP is the worst case for the EOD vehicle routing problem. EOD VRP can still be solved by equating the Equations (C.6) and (C.7) to 1. Therefore, this integer programming formulates three linear system of equations. The number of equations equals the number of unknowns. Even though the system of equations is linear, they have to be solved through an exhaustive iteration. This exhaustive approach to solve for the optimal solution for the vehicle routing problem becomes inefficient as well as not feasible if too many nodes are considered.

The following is the overall algorithm of the solution approach to the integer programming technique. This overall algorithm is usually referred to as a *fractional dual algorithm*[21] [15]:

```
Procedure fractional dual
begin
        solve the relaxation of ILP to obtain the optimal solution denoted by $x^*$;
        feasible = "yes";
        while ($x^*$ is not an integer and feasible = "yes") do
        begin
        select a source row denoted by i;
        add the generated Gomory cut and a corresponding basic variable denoted by s;
        apply the dual simplex algorithm;
        if (dual is unbounded) then feasible = "no";
        set $x^*$ = the new optimal value;
        end
end
```

C.3 Dynamic Programming

An example of a shortest partial TSP path from a source node or the origin node 1 to another node denoted by *j* that passes through nodes 2, 3, ..., *k*-1 is illustrated using the dynamic programming technique. Let S = {2, 3, ..., *n*} and C(S, *k*) = the optimal cost of starting from city 1 to city *k* including the visits to all of the nodes in S. The method starts by finding Equation (C.9) for |*S*| = 1.

[21] Fractional dual algorithm have fractional entries and dual feasibility is maintained.

$$C(\{k\}, k) = d_{1k} \qquad \text{for all } k = 2, \ldots, n \tag{C.9}$$

For $|S| > 1$, C(S, k) can be calculated from the Equation (C.10). This equation assumes that node m is visited immediately before node k for all m. Also, C(S – $\{k\}$, m) is accessible from the preceding table.

$$C(S,k) = \min_{m \in S-\{k\}} [C(S-\{k\},m)+d_{mk}] \tag{C.10}$$

The recursion Equation (C.10) must be calculated for all sets of S and for each considered city m in S. The city m with a minimum also needs to be retained so that the optimal tour can be constructed by backtracking.

Assuming C(S, k) takes one storage location, the complexity of the problem using dynamic programming method is approximately $O(n2^n)$. This is an exponential functions of the problem size n nodes. Although branch-and-bound method has been proven to be more efficient for TSP than dynamic programming technique, this method is better than any other algorithms discovered for TSP [15].

Example C.1 (Back-Solving Problem)

Consider a simple problem where we calculate the cost-to-go in order to determine the best time-optimal path between two points or nodes. The numbers on the grid in the Figure C-1 are the times needed to travel the legs of the grid. Find the time-optimal path from A to B, traveling either up-right or down-right at each corner.

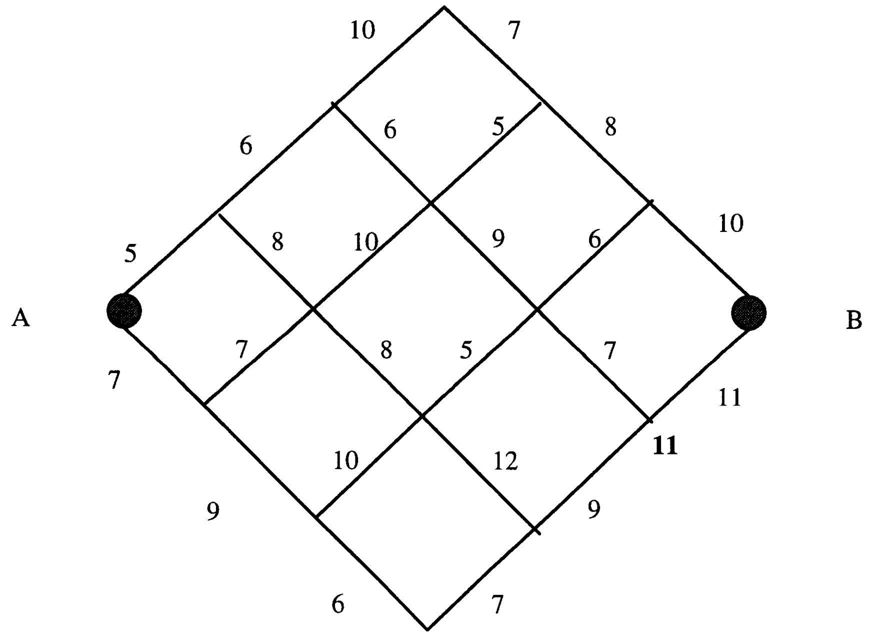


Figure C-1: Travel Grid with Times to Travel Each Leg

The time-optimal path from A to B and the optimal cost-to-go functions from any point to B are shown in the Figure C-2. The dashed line indicates the optimal path from A to B. The bolded numbers represent the optimal cost-to-go functions from any point to B. Basically, the optimal cost-to-go in the Figure C-2 is calculated by working backwards from B to A. The principle of the optimality ensures that if the optimal path from a given point, say C, to the goal is known, then any longer path which passes through C must follow the same path from C to the goal B. Thus, working backwards and comparing cost-to-go at each point, the optimal path is found.

The relative complexity or difficulty of computing the optimal path using a brute force method or exhaustive search method and using dynamic programming are compared. As a measure for this comparison, the number of additions which have to be computed in order to find the solution are considered. However, a different measure such as the number of comparisons needed to find the optimal path can be considered in computing the complexity. For a grid size n, the total number of possible paths from A to B is given by the Equation (C.11):

$$N_{paths} = \binom{2n}{n} = \frac{(2n)!}{n!n!} \tag{C.11}$$

This can be seen by noting that the total number of paths is actually given by the center coefficient in the binomial expansion of $(a+b)^{2n}$. In other words, the Equation (C.11) indicates 2n choose n. The number of arcs from A to B contains 2n times the total number of arcs. This brute force method or exhaustive search method yields the Equation (C.12).

$$N_{brute} = 2n(N_{paths}) \tag{C.12}$$

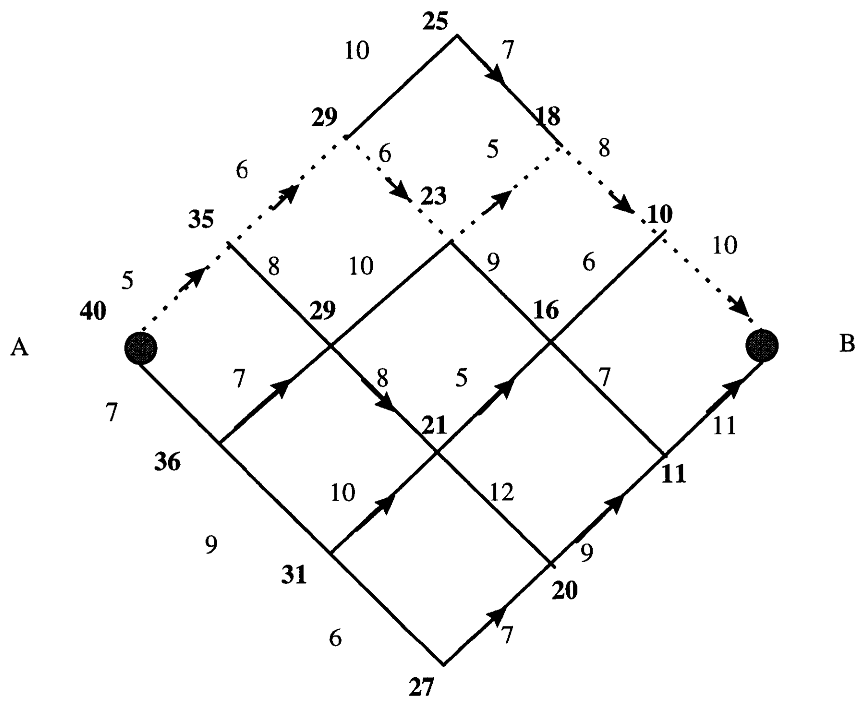


Figure C-2: Travel Grid with Times to Travel and the Optimal Cost-to-Go

The number of sub-routes to try in the dynamic programming method is the number of arcs on the grid. This is also the number of additions that are required to compute the cost-to-go for the full grid since each leg is added only once. This can be written in terms of n as the Equation (C.13):

$$N_{dynprog} = 2n(N+1) \tag{C.13}$$

Therefore, comparison of the Equations (C.12) and (C.13) shows that the number of paths increases much faster with growing grid size than the number of additional legs. Thus, applying the optimality principal and finding the optimal path by using the dynamic programming technique brings a significant advantage for the problems with large grids. Table C-1 shows how the number of possible paths grow according to the grid size for both methods.

Table C-1: Complexity of the Brute Force Method and the Dynamic Programming Method

Grid Size (n)	1	2	3	4	5	6	7	8
Possible Paths (N_{paths})	2	6	20	70	252	924	3,432	12,870
Brute Force (N_{brute})	2	24	120	560	2,520	11,088	48,048	205,920
Dynamic Programming ($N_{dynprog}$)	4	12	24	40	60	84	112	144

C.4 Lagrangian Relaxation

The Equation (C.14) represents the objective function with relaxation of the constraint equations and duplication of them.

$$MaxF(w) = Min\sum_{t=1}^{M}\sum_{i=1}^{M}\sum_{j=1}^{M}(d_{ij}+w_i)x_{ij}^{t} - \sum_{i=1}^{M}w_i \tag{C.14}$$

For any value of w that is the Lagrangian multiplier, the Equation (C.14) creates a tour that will visit each node exactly once without violating the vehicle constraints [8].

Two algorithms exist that reduce the value of the Lagrangian multiplier, which will eventually maximize the value of the objective function in the Equation (C.14). The first algorithm is Lagrangian ascent procedure that produces 90% of optimal lower bounds initially. Thus, this procedure finds either the optimal solution or solution that is extremely close to the optimal solution. If the optimal solution is not discovered by Lagrangian ascent procedure alone, then the branch-and-bound method is applied on the top of it to find the optimal solution. Since the Lagrangian technique finds a 90% optimal solution initially, the branch-and-bound procedure can quickly find the optimal solution. However, one drawback in the Lagrangian ascent procedure is an additional computational cost. After the initial iterations, the procedure either has no improvement or minimal improvement in the result [8].

The second algorithm is Lagrangian 1-trees, which solves the symmetric TSP. The symmetric TSP means that $c_{ij} = c_{ji}$ for every i and j. Also, a l-tree is a tree that connects all the vertices, say 2, 3, ..., n, to the vertex 1 using two edges. Therefore, this tree has vertex with degree 2 (Degree 2 means that only two edges are either coming out or going into the node). The 1-tree is obtained by finding a minimum spanning tree with vertex 2 to n and connecting the two edges that cost the least to vertex 1. The 1-tree approach can only be used for the symmetric TSP. However, a majority of the TSP is the symmetric TSP [1].

Example C.2 (Lagrangian 1-tree for Symmetric TSP)

Table C-2: Distance Matrix for Depot and Nine Nodes

From\To	1	2	3	4	5	6	7	8	9	10
1	0	25	43	57	43	61	29	41	48	71
2	25	0	29	34	43	68	49	66	72	91
3	43	29	0	52	72	96	72	81	89	114
4	57	34	52	0	45	71	71	95	99	108
5	43	43	72	45	0	27	36	65	65	65
6	61	68	96	71	27	0	40	66	62	46
7	29	49	72	71	36	40	0	31	31	43
8	41	66	81	95	65	66	31	0	11	46
9	48	72	89	99	65	62	31	11	0	36
10	71	91	114	108	65	46	43	46	36	0

Consider the following nine cities (referred to as the nodes) that the salesman has to visit starting from the node 1 or the depot node and ending at the same node. The salesman wishes to find the shortest route to visit all the nine cities. Figure C-3 shows the location of the depot or the node 1 and the nine nodes numbered arbitrarily as 2 through 10. Also, Table C-2 lists the Euclidean distances for all pairs of nodes [12].

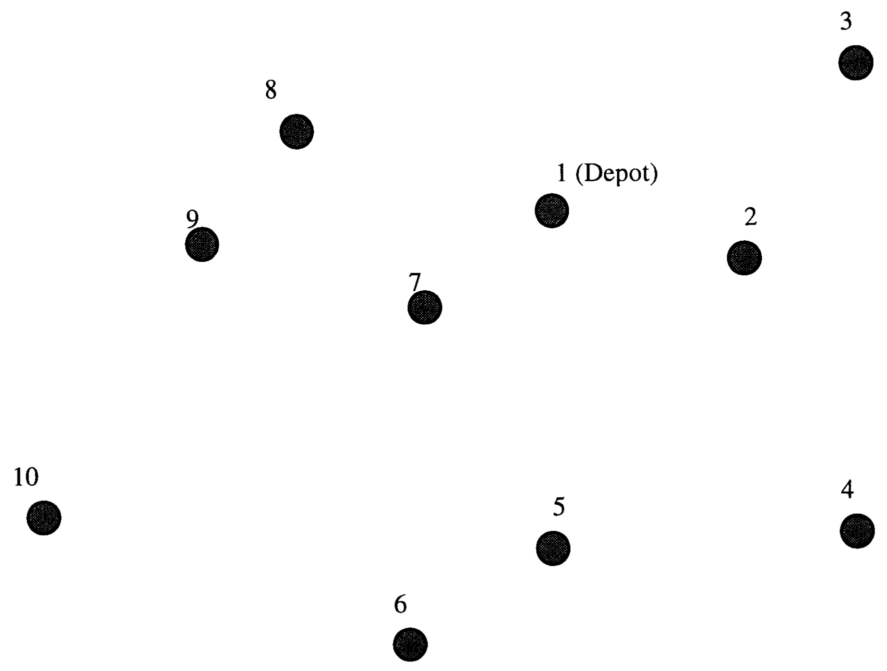


Figure C-3: Depot and Nine Nodes to be Visited

Now applying the Lagrangian 1-tree algorithm for the TSP. The first step is to find the minimum spanning tree using nodes 2 through 10 (Figure C-4).

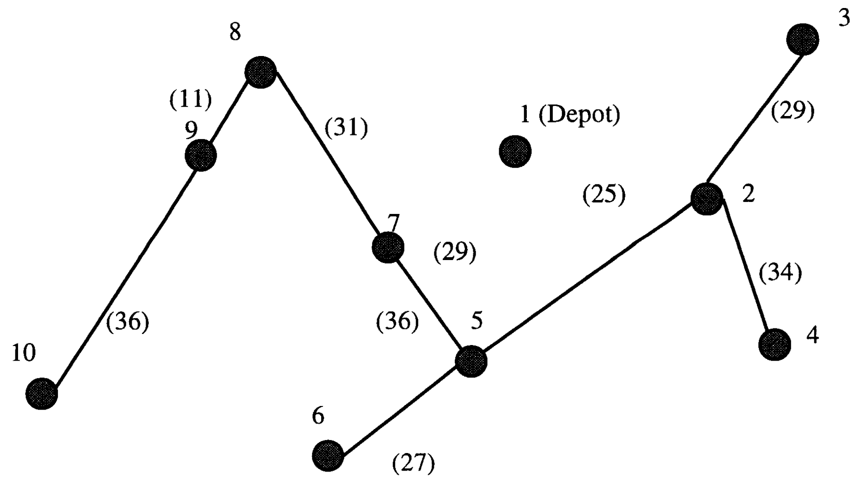


Figure C-4: Minimum Spanning Tree for Lagrangian

Then, connect the two edges with the least distance to the vertex 1 or the node 1. Figure C-5 shows the minimum spanning tree with two cheapest edges connected to the node 1. This provides a lower bound on the length of the optimum traveling salesman tour. The lower bound for this example is 258 units. The true optimal solution happens to be the tour {1, 3, 2, 4, 5, 6, 10, 9, 8, 7, 1} with the total length of 331 units [12].

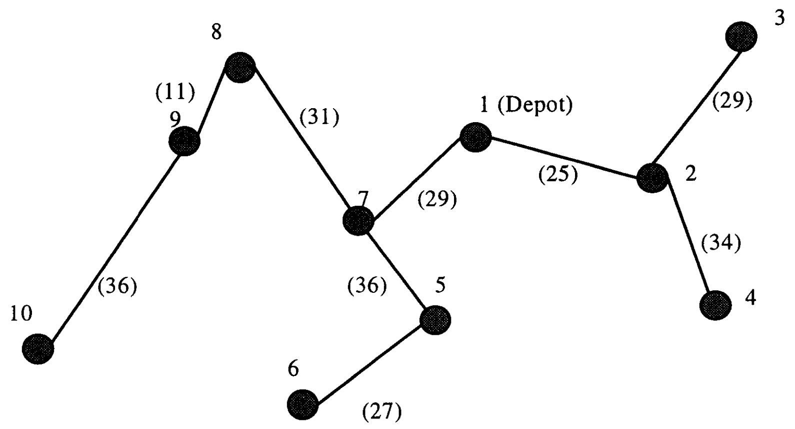


Figure C-5: Minimum Weight Lagrangian 1-tree

Appendix D: Details of Heuristic Algorithms

As mentioned in the Appendix C, the optimal solution approach to TSP and VRP requires an exhaustive search for the best solution. An exhaustive search of the best solution is limited to small-scaled problems (i.e. the problem that has only several possible solutions). However, heuristic algorithms consider only few of the possible solutions to any sized problems using "rule of thumb". Thus, heuristic solution method can be used to solve a large-scaled problem in a reasonable time. This solution may not be the best solution, but a feasible solution is found in a reasonable computing time. Thus, the heuristic solution can be considered to be sub-optimal solution.

Heuristic algorithms [1] [4] [12] [15] are generally categorized into three broad classes: tour construction heuristics, tour improvement heuristics, and composite heuristics. Tour construction approaches always begin the process with a single node and add nodes to build a complete tour. Tour improvement approaches start with the initial tour obtained in the tour construction procedure and tries to improve the tour. Lastly, composite approaches are a combination of the first two approaches. These procedures construct an initial tour by using the tour construction approaches and improve the initial tour by using the tour improvement approaches [4].

D.1 Tour Construction Heuristics

Tour construction procedure creates a complete tour that visits one node at a time by using the known distance or cost matrix. This procedure adds a node to the partially constructed tour in the VRP only if it will be included in the final solution. Experimental results show that tour construction heuristics generally don't produce good results. Nevertheless, many tour construction procedures are available to construct an initial tour for any given problem with or without constraints. Some of the tour construction procedures are nearest neighbor or greedy heuristic, Clarke-Wright savings algorithm that usually solves the VRP, and various insertion procedures [1].

D.1.1 Nearest Neighbor

The followings are the steps to the nearest neighbor algorithm in the general terms [4] and an example that uses the greedy algorithm follows the algorithm [15]:

STEP 1: Select an arbitrary node as the starting point of a sub-tour.

STEP 2: Find the closest node that is not yet included to the last node included in the sub-tour. If this node is found then add it to the current sub-tour. Then repeat the *STEP 2.* Otherwise, *stop* and go to *STEP 3.*

STEP 3: Connect the first and the last node in the completed sub-tour in *STEP 2.*

Example D.1 (Minimum Spanning Tree Using Greedy Algorithm)

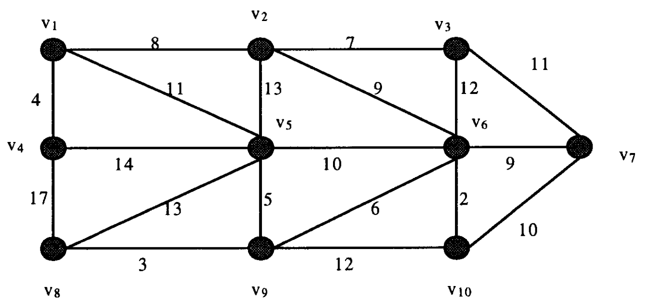


Figure D-1: Minimum Spanning Tree Problem

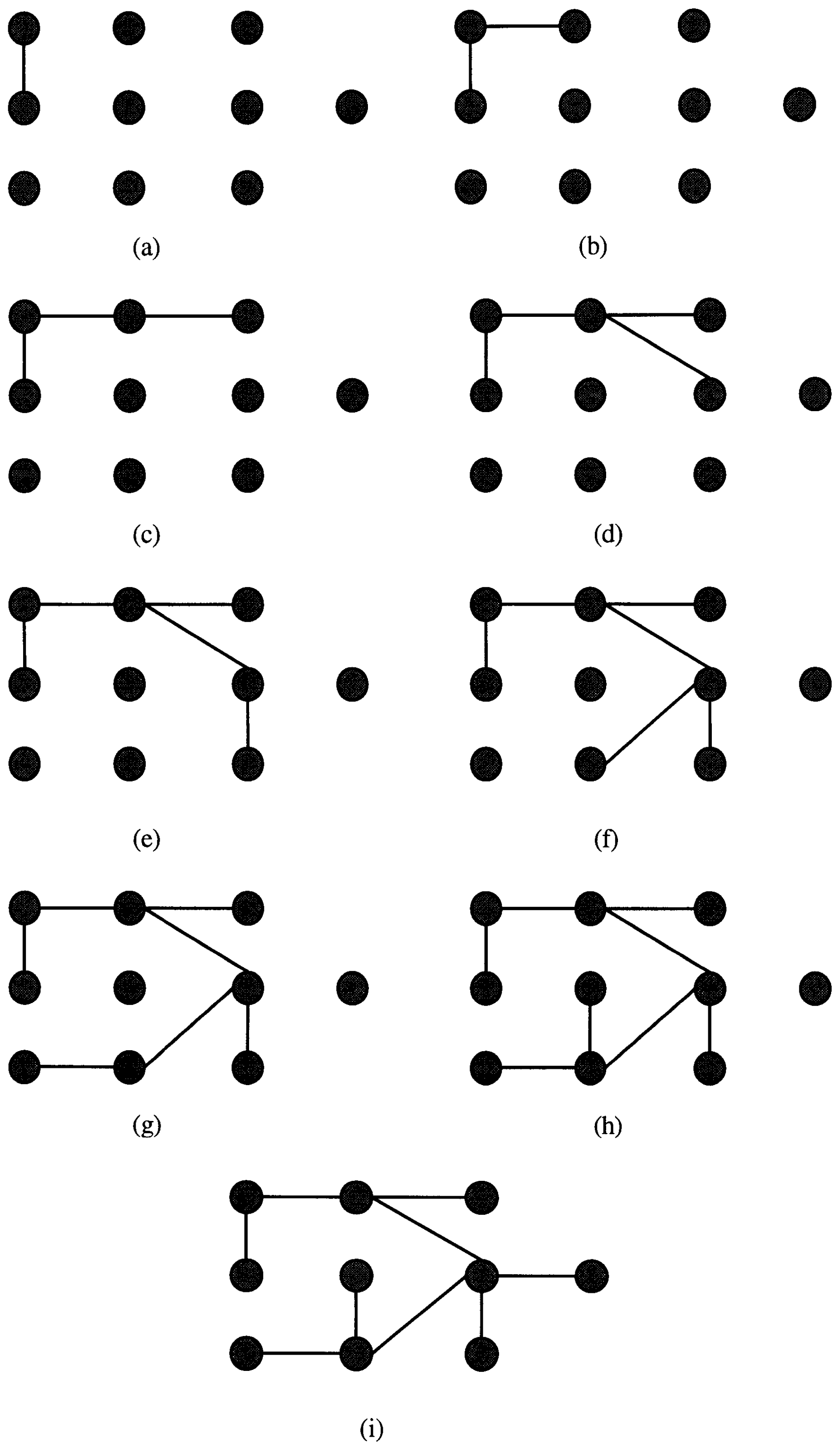


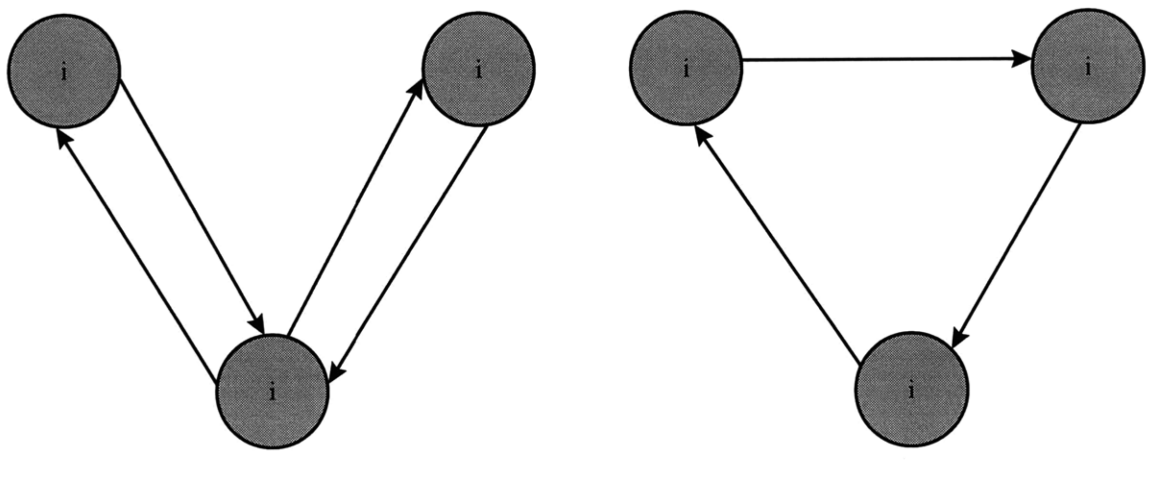
Figure D-2: Nine Stages of the Minimum Spanning Tree by Using Greedy Algorithm

D.1.2 Clarke-Wright Savings

To illustrate the concept of Clarke-Wright savings algorithm, consider nodes *i* and *j* being linked into one node. The saving in the cost (denoted by S_{ij}) is calculated by the Equation (D.1). The complete tour is constructed in the order of the decreasing savings that are computed without violating any constraint.

Figure D-3 illustrates the savings method for two nodes *i* and *j*. A pair of nodes that have been linked in the process must remain linked. The savings algorithm stops once all the nodes in the problem have been included in the final tour [1].

$$S_{ij} = c_{1i} + c_{1j} - c_{ij} \tag{D.1}$$



(a) Before Linking (b) After Linking

Figure D-3: Concept of the Clarke-Wright Savings Heuristic

Two nodes can only be linked if their time windows overlap. In other words, a given node that has time window of 4 through 7 needs to be visited on the 4th, 5th, 6th, or 7th stop after the vehicle has left the depot or the origin. Suppose

S_k = {stop numbers in k's time window} where k = 2, 3, …, N,

$Lij = S_i \cup S_j$

Nodes *i* and *j* can only be linked together if the set for both nodes contains all of the stop numbers. For instance, if $S_i = \{2, 3, 4\}$ and $S_j = \{6, 7, 8\}$ then $L_{ij} = \{2, 3, 4, 6, 7, 8\}$. Nodes *i* and *j* can't be linked together since the stop number 5 is not an element of the set L_{ij}. The latest possible stop number is 4 for the node *i* while the earliest possible stop number is 6 for the node *j*. Basically, a tour with nodes *i* and *j* linked together must have the node *i* in the 4th stop and the node *j* in the 5th stop so that neither time window is violated. Another case is $S_i = \{2, 3, 4, 5, 6, 7\}$ and $S_j = \{4, 5, 6,7,8, 9\}$ then $L_{ij} = \{2, 3, 4, 5, 6, 7, 8, 9\}$. If the node *i* precedes the node *j* then the node *i* contains the stop number 3, 4, 5, 6, 7, or 8 in the final tour. On the other hand, if the node *j* precedes the node *i* then the node *i* contains only the stop number 5, 6, or 7 in the final tour. That is, if all the elements of S_i are less than the elements of S_j then the node *i* should precede the node *j* in the initial partial tour [1].

The criterion for linking two nodes is necessary for feasibility of the tour, but it doesn't guarantee that a feasible tour can be constructed. For instance, consider $S_i = \{4, 5\}$, $S_i = \{6, 7\}$, and $S_b = \{5, 6\}$. Positioning *i* at the stop 5 and the *j* at the stop 6 allows a feasible linking between *i* and *j*. However, node *b* can't be feasibly sequenced in the tour if I and j are linked by the position 5 and 6 respectively. In this case, the sequence in the Equation (D.2) is feasible with respect to *i*, *j*, and *b* where x's represent other nodes. Thus, the first two nodes for linkage need to be selected such that a feasible final tour is guaranteed. Moreover, an infeasible tour is almost always the result from ignoring the feasibility during the tour construction steps [1].

$$1\text{-x-x-x-i-b-j-x-}\ldots\text{-}1 \tag{D.2}$$

The Clarke-Wright savings algorithm basically adds new nodes to a current partial tour. These new nodes must be selected from the set that includes the candidate nodes. The new nodes can be added either in front of or behind the node in the current partial tour such that a feasible final tour can be constructed. Therefore, two feasible candidates are generated at each iteration. The following are the possible ways that the new nodes could be front or rear end candidates [1]:

Front End Candidate

- The new node is an unassigned origin whose corresponding destination is already sequenced.
- The new node is an unassigned origin as is its corresponding destination, but the current partial tour is small enough that the destination can be sequenced later.
- The new node is an unassigned destination whose origin has not yet been sequenced.

Rear End Candidate

- The new node is an unassigned destination whose corresponding origin is already sequenced.
- The new node is an unassigned destination whose corresponding origin has not yet been sequenced, but the current partial tour is small enough that the origin could be sequenced before the first node.
- The new node is an unassigned origin whose destination has not yet been sequenced.

Thus, the two feasible candidates (if exist) have to be identified with their savings values. Then, the one with the greater savings is selected to be added to the current partial tour. If both candidates have the same savings and the front end candidate is an origin then it needs to be added. Otherwise, the rear end candidate needs to be added to the partial tour. This process guarantees that the final tour is feasible because a candidate is added only if a feasible final tour would result.

On average, the Clarke-Wright savings approach provides tours that are as good as if not better than tours created by other heuristic methods for the typical VRP. Most of the sub-optimal solutions produced by the heuristics have an average error of 3.2% of the optimal solutions for the VRP of 10 nodes [1].

The Clarke-Wright savings method can be further improved by using the Yellow's model. This model adds a route shape parameter θ to the Equation (D.3). The parameter θ changes the significance of the cost between nodes *i* and *j* with respect to their costs relative to the depot or the origin. Thus, Yellow's method becomes Clarke-Wright savings method with $\theta = 1$. Therefore, the improved method will always produce at least as good solution as the Clarke-Wright savings technique [1].

$$S_{ij} = c_{1i} + c_{1j} - \theta\, c_{ij} \qquad \text{(D.3)}$$

The followings are the steps to the typical Clarke-Wright savings algorithm in the general terms [4]:

STEP 1: Select any arbitrary node as the central depot or the origin denoted by node 1.

STEP 2: Calculate the savings for nodes *i* and *j* using the Equation (D.1) for $i = j = 2, 3, \ldots, n$ and *i* doesn't equal *j*. Savings need to be computed for each pair of nodes in the problem.

STEP 3: Rank the savings from the largest to the smallest.

STEP 4: Start with the largest savings cost from the list produced in *STEP 3*, create sub-tours such that the next node added to the sub-tour would have the largest remaining savings. If no constraint is violated then link nodes *i* and *j*. If all nodes in the problem have been included in the tour then *stop*, a sub-optimal solution has been obtained. Otherwise, go to *STEP 4* again.

Example D.2 (Clarke-Wright Savings Algorithm for TSP)

Consider the nine cities referred to as the nodes that the salesman has to visit starting from the node 1 or the depot node and ending at the same node. The salesman wishes to find the shortest route to visit all the nine cities. Figure C-3 in the Example C.2 shows the location of the depot or the node 1 and the nine nodes numbered arbitrarily as 2 through 10. Also, Table C-2 in the Example C.2 lists the Euclidean distances for all pairs of nodes. *STEP 1* is completed by selecting the central depot to be the origin and denoted by node 1 as shown in the Figure C-3 [12].

STEP 2 is carried out by calculating the savings for all the pair of nodes in the problem using Equation (D.1). Then, *STEP 3* is done by arranging the savings in decreasing order as shown in the Table D-1.

Table D-1: Clarke-Wright Savings Listed in Decreasing Order

$S_{6,10} = 86$	$S_{2,4} = 48$	$S_{5,9} = 26$	$S_{2,7} = 5$
$S_{9,10} = 83$	$S_{3,4} = 48$	$S_{2,5} = 25$	$S_{2,10} = 5$
$S_{8,9} = 78$	$S_{4,6} = 47$	$S_{4,10} = 20$	$S_{3,8} = 3$
$S_{5,6} = 77$	$S_{6,9} = 47$	$S_{5,8} = 19$	$S_{4,8} = 3$
$S_{8,10} = 66$	$S_{7,9} = 46$	$S_{2,6} = 18$	$S_{3,9} = 2$
$S_{7,10} = 57$	$S_{2,3} = 39$	$S_{4,7} = 15$	$S_{2,9} = 1$
$S_{4,5} = 55$	$S_{7,8} = 39$	$S_{3,5} = 14$	$S_{2,8} = 0$
$S_{6,7} = 50$	$S_{5,7} = 36$	$S_{3,6} = 8$	$S_{3,7} = 0$
$S_{5,10} = 49$	$S_{6,8} = 36$	$S_{4,9} = 6$	$S_{3,10} = 0$

STEP 4 is applied to find the final good tour for the given TSP. Thus, using the largest saving that is in the first entry of the Table D-1 the sub-tour is created. Figure D-4 shows the final tour created using the Clarke-Wright savings algorithm. The optimal tour is {1, 3, 2, 4, 5, 6, 10, 9, 8, 7, 1} (Figure D-5). The final tour has total length of 349 units where the true optimal tour has 331 units. This final tour is less than 6% longer than the optimal tour. Thus, Clarke-Wright savings heuristics create very good solutions to TSP and VRP in the minimal time by considering the more distant nodes to be included in the final tour.

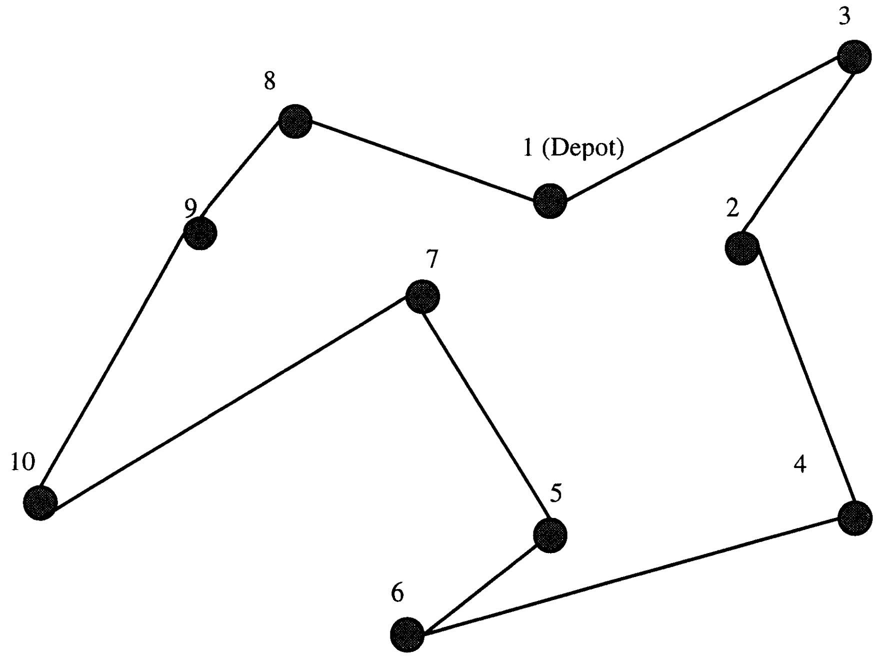


Figure D-4: Final Tour for Clarke-Wright Savings Algorithm

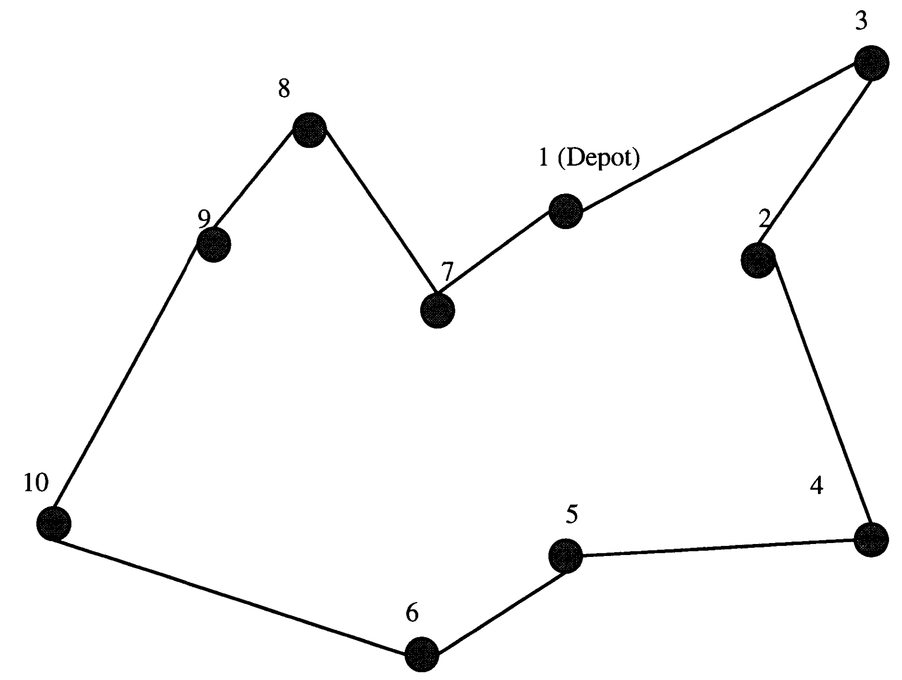


Figure D-5: The Optimum Tour Using Clarke-Wright Savings Algorithm

D.1.3 Insertion Procedures

The insertion is completed for node *k* by creating arc (*i*, *j*) in the tour that should minimize the Equation (D.4) without violating the constraints. The sub-optimal solutions discovered by the insertion procedure are within 3-5% of the optimal solution [1].

$$S_{ij} = c_{ik} + c_{kj} - c_{ij} \qquad \text{(D.4)}$$

The followings are an insertion algorithm in the general terms [4]:

STEP 1: Initial sub-tour. Create a TSP tour for a subset of the nodes N' ε N in G (the graph representing the network of the problem).

STEP 2: Selection step. Select a node *k* ε N-N' to be added to the sub-tour from *STEP 1.*

STEP 3: Insertion step. Provide an arc (*i*, *j*) within the existing sub-tour on N'. Insert node *k* between nodes *i* and *j* and add node *k* to N'.

STEP 4: If N = N', then *stop,* a Hamiltonian[22] cycle has been obtained. Otherwise, go to *STEP 2.*

The convex hull of the set of nodes N have been used to produce the initial sub-tour in the *STEP 1.* The optimal tour usually visits nodes on the boundary of the convex hull just as if the original boundary itself were followed. Furthermore, several comparisons of the insertion heuristics both with and without the use of the convex hull as the initial tour have been done. The comparisons favor of the insertion heuristic with the use of the convex hull as the starting point [4].

Furthermore, route insertion procedures combine the Clarke-Wright savings method with the insertion heuristics to consider early more distant nodes to be included in the final tour. Therefore, the expensive and

[22] A tour is a chain that goes through all the *n* cities or nodes and that the first and the last nodes coincides. This type of tour is referred to as a Hamiltonian cycle.

last minute changes to sequence the given nodes are prevented. In the route insertion algorithm, a pair of nodes is selected based on the selection criteria to be added into the partial tour. Pairs of nodes are selected for insertion in decreasing order of individual service cost. In other words, a pair of nodes is linked directly from the depot and the salesman or the vehicle returns to the depot after visiting the pair of nodes. For instance, for the pair of nodes *i* and *j* where *i* is the origin and *j* is the corresponding destination, the Equation (D.5) represents the individual service cost:

$$C_{ij} = c_{1i} + c_{ij} + c_{j1} \qquad \text{(D.5)}$$

The individual service cost needs to be computed for each pair of nodes in the problem. Then, the pair with the highest individual service cost is added to the partial tour just as in the Clarke-Wright savings method. Therefore, if the pair *(i, j)* has the highest service cost then the Equation (D.6) represents the initial partial tour:

$$1 - i - j - 1 \qquad \text{(D.6)}$$

Likewise, the pair with the next to the highest cost is inserted into the partial tour in the Equation (D.6). This process continues until the final tour is constructed [1]. Furthermore, this next pair is inserted into the partial tour in such a way to minimize the total service cost. The cost equations vary depending on the location of the new destination. The new destination node can be inserted directly after its corresponding origin or at a node later in the partial tour. Figure D-6 (a) displays a partial tour and Figure D.6 (b) and (c) show the pair *(i, j)* insertion into the partial tour. Figure D-6 (b) has *j* node inserted directly after the *i* node and Figure D-6 (b) has *j* node inserted at a later node than *i* node in the partial tour. Equations (D.7) and (D.8) represent the cost equations for the insertion in (b) and (c) respectively.

$$C = c_{ai} + c_{ij} + c_{jb} - c_{ab} \qquad \text{(D.7)}$$

$$C = c_{ai} + c_{ib} - c_{ab} + c_{cj} + c_{jd} - c_{cd} \qquad \text{(D.8)}$$

```
          1 - a - b - c - d - 1
```

(a) Partial Tour

```
              i - j
             /     \
          1 - a     b - c - d - 1
```

(b) j Inserted Directly After i

```
          i         j
        /   \     /   \
    1 - a    b - c     d - 1
```

(c) j Inserted at a Later NodeThan i

Figure D-6: Example Insertion Patterns for the Pair Insertion Heuristic

Example D.3 (Route Insertion Procedure for TSP)

Consider the nine cities referred to as the nodes that the salesman has to visit starting from the node 1 or the depot node and ending at the same node. The salesman wishes to find the shortest route to visit all the nine cities. Figure C-3 in the Example C.2 shows the location of the depot or the node 1 and the nine nodes numbered arbitrarily as 2 through 10. Also, Table C-2 in the Example C.2 lists the Euclidean distances for all pairs of nodes [15].

The route insertion procedure starts with computing the service costs for all the pair of nodes in the problem using Equation (D.5). Then, these service costs are arranged in the decreasing order as shown in the Table D-2.

Table D-2: Route Insertion Service Costs Listed in Decreasing Order

$C_{4,10} = 236$	$C_{6,10} = 178$	$C_{2,6} = 154$	$C_{6,7} = 130$
$C_{3,10} = 228$	$C_{6,9} = 171$	$C_{3,4} = 152$	$C_{2,4} = 116$
$C_{4,9} = 204$	$C_{6,8} = 168$	$C_{5,8} = 149$	$C_{2,5} = 111$
$C_{3,6} = 200$	$C_{3,8} = 165$	$C_{2,9} = 145$	$C_{5,7} = 108$
$C_{4,8} = 193$	$C_{3,5} = 158$	$C_{4,5} = 145$	$C_{7,9} = 108$
$C_{4,6} = 189$	$C_{8,10} = 158$	$C_{3,7} = 144$	$C_{2,7} = 103$
$C_{2,10} = 187$	$C_{4,7} = 157$	$C_{7,10} = 143$	$C_{7,8} = 101$
$C_{3,9} = 180$	$C_{5,9} = 156$	$C_{2,8} = 132$	$C_{8,9} = 100$
$C_{5,10} = 179$	$C_{9,10} = 155$	$C_{5,6} = 131$	$C_{2,3} = 97$

The largest service cost that is in the first entry of the Table D-2 is used to initiate the insertion procedure. Figure D-7 shows the final tour created using the route insertion algorithm. This is in fact the same tour for the given TSP in the Example D.2. The total length of the tour is likewise 349 units which is about 5.4% more than the optimal solution. Thus, route insertion heuristics create very good solutions to TSP and VRP in the minimal time by considering the more distant nodes to be included in the final tour.

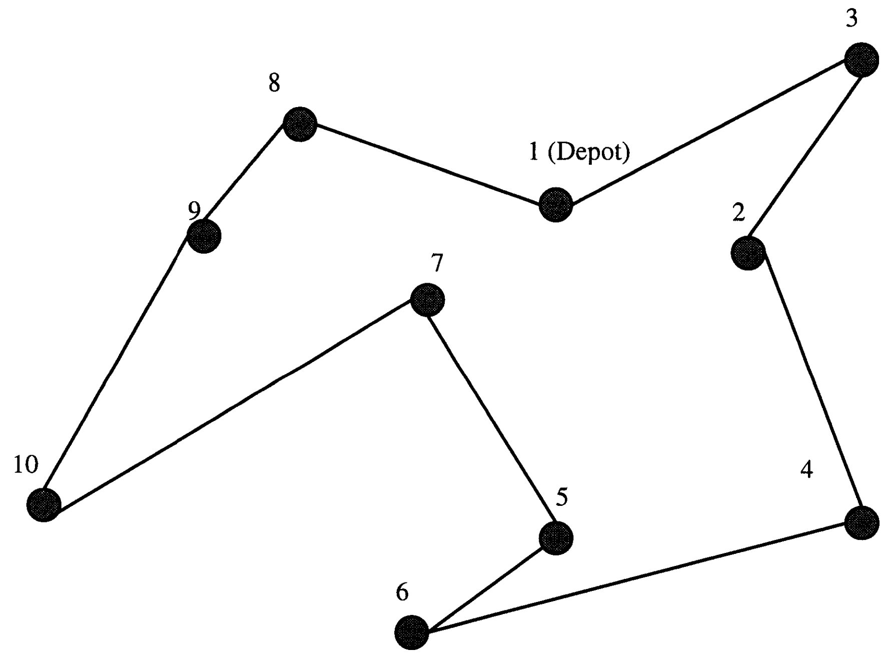


Figure D-7: The Optimum Tour Using Route Insertion Algorithm

D.2 Tour Improvement Heuristics

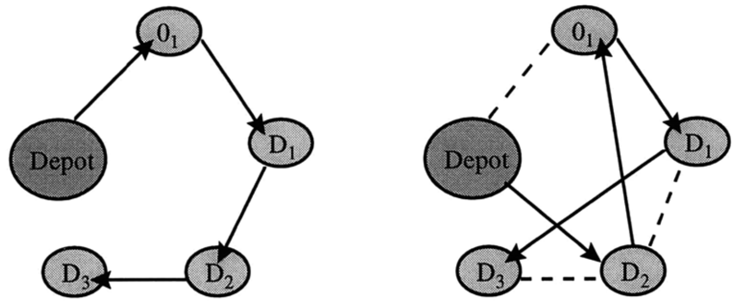
Two basic heuristics exist to improve a complete tour. They both use the arc interchange or branch exchange technique to improve the current tour [1].

3.3.2.1 r-optimality

The *r*-optimal procedure indicates that a feasible solution can't be further improved by eliminating any *r* links and replacing the links by *r* new links. In other words, this procedure stops when a local optimal solution has been obtained. TSP and VRP are usually solved by 2-optimal and 3-optimal interchanges. Higher values of *r* can be used to solve the problems; however, the computational cost increases significantly. Thus, a 2-optimal tour can be found first used as input to a 3-optimal algorithm for the best solution. The directed network may have infeasible patterns since an arc or a branch may not exist in the reverse direction [1].

3.3.2.2 λ-optimality

This procedure is similar to the r-optimal procedure except that λ is not fixed but starts at a value of 2 at any step of the iteration process. In other words, two arcs or branches are removed to find a 2-optimal solution at the start. Then a third arc or branch is removed to find 3-optimal solution and so forth. This iteration process continues until the tour cost can't be further improved. Therefore, if five arcs or branches are removed then the procedure seeks a better solution than the 4-optimal solution. The procedure guarantees that a feasible reconnection pattern exists if an arc or branch is removed. In other words, reconnection patterns are considered only if feasible solution exists by exchanging the arcs or the branches. However, this approach may never investigate an instance where 3-optimal reconnection yields a feasible tour because 2 removed arcs or branches can't be feasibly rejoined. For instance, consider the example in the Figure D-8 (a) where no possible way exists to remove two arcs or branches and reconnect them in a new pattern that satisfies the requirements. On the other hand, the Figure D-8 (b) shows three arcs or branches can be removed and reconnected in a new pattern. Therefore, problems with tight constraints would save costs from many reconnections [1].



(a) No 2 Arc Exchange Possible (b) A Feasible 3 Arc Exchange

Figure D-8: The Problem with Precedence Constraints Using λ-Optimality

D.3 Composite Heuristics

The composite heuristics are a combination of the tour construction and improvement heuristics. This procedure generates an initial tour by the tour construction method and improves the tour by the tour improvement method. Although many composite heuristics exist, two of them are of special interest to the VRP [1].

3.3.3.1 Sweep Algorithm

The sweep algorithm is characterized as solving the big problem. Thus, this algorithm divides the given big problem into *k* sub-problems where *k* is the number of available vehicles. The sweep algorithm basically ranges all the nodes to be visited by their polar coordinate angle. The tours are produced by sweeping through the polar angles and adding the nodes to the tour if no constraint is violated. The algorithm created by Lin-Kernighan can be applied to the sweep algorithm to improve the final tour. The sweep algorithm can generate feasible solutions to the problem with about 100 nodes. Moreover, the generated solutions are as good as the solutions produced by other heuristic approaches [1].

3.3.3.2 MTOUR (Multiple Tour)

The MTOUR method is developed to solve constrained multiple traveling salesmen problems or constrained VRPs. This technique solves the multiple salesmen or vehicles as a single salesman or vehicle problem on an extended network by using the approach created by Lin-Kernighan. Thus, an initial feasible solution is used as input to the MTOUR improvement heuristics to provide improved solution. The time window constraint is specifically considered in the multiple VRP and TSP. Although this constraint may not easily yield the initial solution or complicate the λ-optimality procedures, good results are provided by the MTOUR improvement heuristics [1]. Appendix E extends both the exact and heuristic algorithms for a single VRP to multiple VRP.

3.3.3.3 General Composite Procedure

As stated previously, the basic composite algorithm simply appends a branch exchange procedure to a tour construction algorithm. This procedure is usually fast in terms of the computation time and yields relatively good results. The following steps are the general steps to a typical composite procedure [4]:

STEP 1: Obtain an initial tour using a tour construction procedure.

STEP 2: Apply a branch or an arc exchange algorithm to the tour created by the *STEP 1*. If no further improvement can be made to the current tour then *stop.* Otherwise, iterate the branch or arc exchange algorithm.

D.4 Mixed Solution

Any of the exact solution algorithms can be formulated as a heuristic method by terminating the technique early to result in a good solution but not the optimal or exact solution. In other words, calculate the total cost of the partial tour at each iteration of any exact algorithms to compare to each other. Therefore, if the partial tour has not improved more than some defined error value then the process terminates with the current tour as the final solution. The error value needs to be reasonably defined so that the process runs only few iterations.

Mixed routing algorithm exists in many different forms. A particular mixed heuristic algorithm uses the minimum spanning tree and some heuristics to improve the tour. The details and example of the mixed routing algorithm is included in Chapter 2, section 2.3.1.

Appendix E: Details of Extended Algorithms

The routing algorithms that have been discussed in the Appendix C and D considers a routing solution involves a complete tour (i.e. a path that connects all the nodes in a loop). This solution can be used using one or two vehicles. However, if the number of vehicles increases to more than two, then the single-tour solution is useless for routing multiple vehicles. Some of the optimal solution and sub-optimal solution methods can be modified using a heuristic algorithm to solve the MVRP. These extended algorithms [1] can be modified to solve various routing problems.

E.1 Extended Dynamic Programming

From the various choices of exact solution methods, only dynamic programming method is extended to solve multiple vehicle routing problem (MVRP). Greedy heuristic and insertion selection methods are chosen from the tour construction heuristics and extended to solve MVRP. The mixed routing algorithm is also extended to solve MVRP. The 3-optimistic heuristic (tour improvement heuristic) extension to cover MVRP is briefly discussed.

Example E.1 (Back-Solving Problem)

Consider Example C.1 with two available vehicles that need to cover the tour that is found using the dynamic programming technique discussed in the Appendix C. Figure E-1 shows the routing problem with the optimal path from A to B found using dynamic programming technique and the depot or origin node. From the depot node, each of the two available vehicles can follow the path on either side of the depot node. The two paths are indicated on the Figure by the arrows that point in the direction of the vehicle traveling. Likewise, the optimal path is indicated by the dashed line in the Figure E-1. The two paths are not exactly equal in distance but include the same number of nodes. The two vehicles can follow the two paths to build the routes simultaneously[23]. However, if an additional vehicle is added to the problem then the optimal tour found using the previous technique is not feasible anymore. No free path is available for the third vehicle to build the route. Therefore, the dynamic programming technique needs to be extended to consider more than two vehicles in the routing problem.

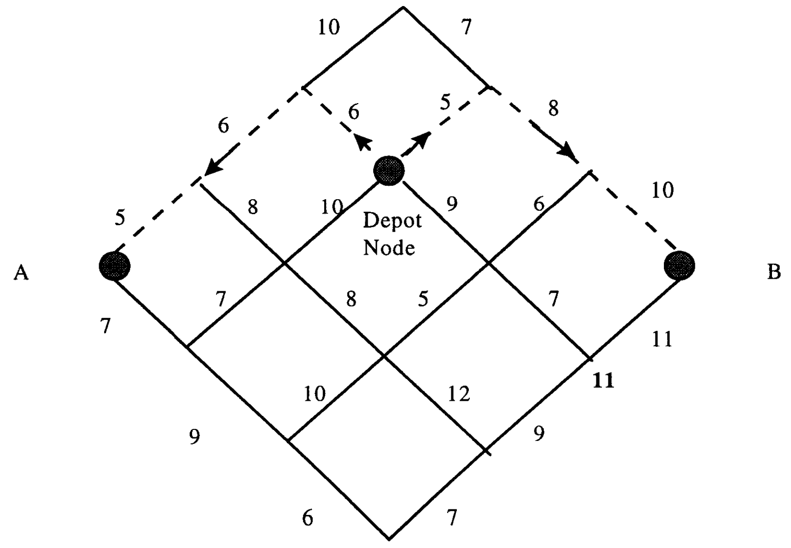


Figure E-1: Two Paths for the Two Vehicles Using Dynamic Programming

[23] Since the two vehicles are assumed to have same speed, one of the vehicles will complete the path earlier than the other vehicle.

The extended dynamic programming method is used to solve for more than two vehicles routing problem. The number of nodes that needs to be visited can be divided or assigned by basically two different ways. First consider the case of clustering or dividing the network first and routing each division second. Thus, the dynamic programming technique for a single vehicle can still be applied to the divided cluster of nodes. Consider the next example that is similar to the previous example except the objective is to visit all the nodes in the network not just the nodes that lie in the optimal path between A and B. Also, the vehicles can travel along any direction as required not only right-up or right-down direction. Therefore, the network is not a directed network.

Figure E-2 shows a way to divide the network given in the Figure E-1 into three smaller regions depicted by dashed lines for each of the three available vehicles. There may be other ways to divide the network and produce feasible solution to the problem. The area is divided into three regions geographically. The nodes in the different regions are distinguished by the patterns of the nodes. For instance, the nodes in one of the regions are plain white and the nodes in another region are filled with dots. The nodes in the last region are filled with waves. Two of the three created routes are approximately equal in the distance but the third one is slightly longer than the other two routes. Thus, these three routes are not necessary optimal or good solution to the three vehicles routing problem. The regions can be divided into different sections to yield a better solution that produces three routes that are approximately equal in the distance.

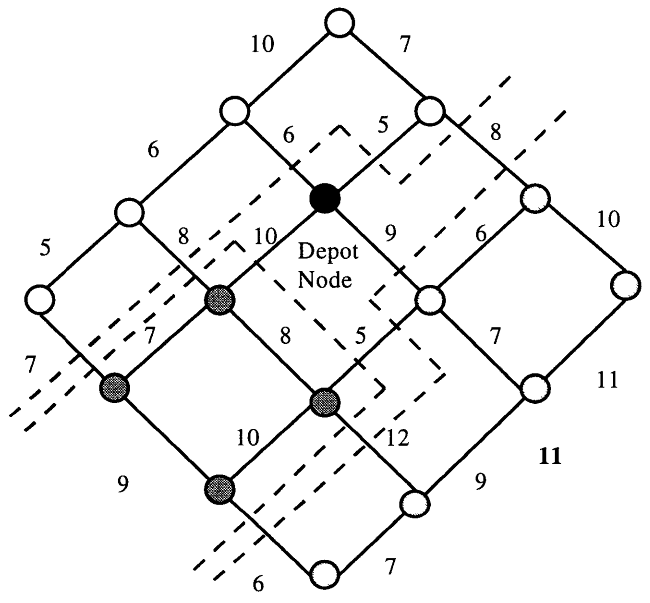


Figure E-2: Divided Regions for Three Vehicles Using Dynamic Programming

After the network has been divided or clustered then each of the smaller regions can be routed using the regular dynamic programming technique. Figure E-3 shows the three divided regions along with their routes for each of the three available vehicles. The arrows indicate the direction of each of the routes in the network.

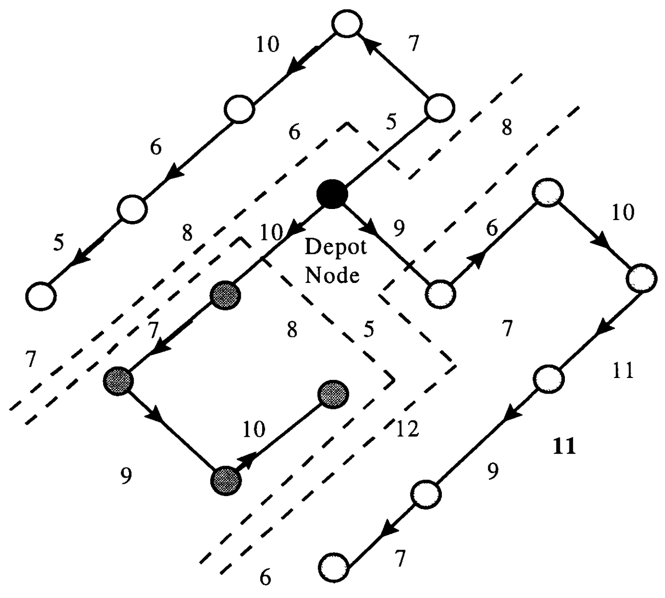


Figure E-3: Three Individual Routes for Three Vehicles Using Dynamic Programming

The second way to assign nodes to three available vehicles is to dispatch the vehicles simultaneously. In other words, apply the general dynamic programming technique to the network except for three different vehicles simultaneously. Thus, first three nodes are assigned to each of the three vehicles then the second nodes and so forth. This method usually prevents uneven routes that are unfavorable for multiple vehicles routing problem. Figure E-4 shows the simultaneous assignments of the nodes to three vehicles. The depot node is filled and labeled and the three different routes are marked by different patterns of the nodes. Again, two of the three created routes are approximately equal in the distance but the third one is longer than the other two routes.

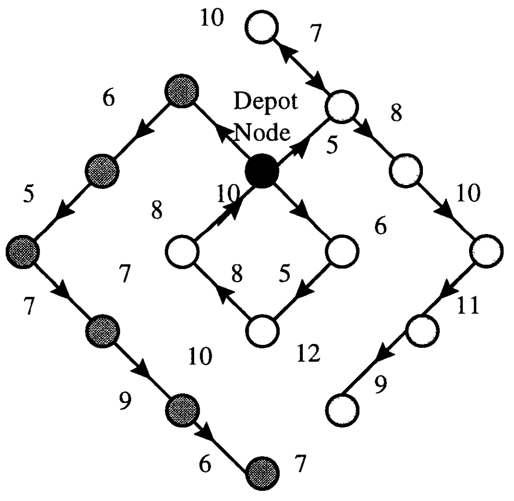


Figure E-4: Simultaneous Assignment of Nodes Using Dynamic Programming

A large storage is required for the approach stated above. For example, the least storage requirement for the stated problem with 25 nodes and 5 vehicles is approximately 265,650 locations. This storage requirement can be as large as 68 million locations for the similar problem depending on how the nodes are divided among the vehicles [1]. Therefore, any exact solution method is extremely inefficient and not feasible for the multiple vehicles routing problem.

E.2 Extended Greedy Heuristic

Greedy heuristic method exists in many different forms. A particular greedy heuristic algorithm uses the distribution of nodes and adding the closest node at a time to one or several routes. The algorithm details and example of the particular greedy method that has been considered is included in Chapter 2, section 2.3.2.

The greedy heuristic basically selects the nearest neighbor node to be included in the final tour without considering where the tour will go next. The distance over the next two nodes may be considered to produce a better solution. This is often referred to as the "greedy look ahead" algorithm. However, no appreciable difference occurs on the final results between the regular greedy heuristic and the greedy look ahead algorithms. Furthermore, the regular greedy algorithm is an order Z algorithm while the greedy look ahead algorithm is an order Z^2 algorithm. Therefore, the look ahead algorithm is not implemented further due to the heavier computational effort [1].

E.3 Extended Pair Selection

The pair selection heuristic assigns each pair of origin/destination pairs to the same vehicle. This method uses a savings value that is computed for each pair of nodes that are assigned to a vehicle. Once all of the nodes in the problem are assigned to a vehicle then the problem reduces down to a single VRP for each vehicle. The advantage of the pair selection method is to move from an origin to its corresponding destination. In other words, if two nodes can be visited as cheaply as one then it is advantageous to assign both nodes to the same vehicle. Figure E-5 illustrates two origin/destination pairs that can both be visited for the same cost as visiting the first pair alone. However, the EOD problem doesn't necessarily consider pairs of origin/destination nodes. Pick-up and delivery problems usually consider the origin/destination pairs of nodes. The pair selection heuristic can still be applied to EOD VRP if the destination node from the second pair is assumed to be the origin node in the second pair. Therefore, only three nodes are considered for the linkage in the network. This method reduces down to the pair insertion procedure that costs expensive computational effort for even the single VRP.

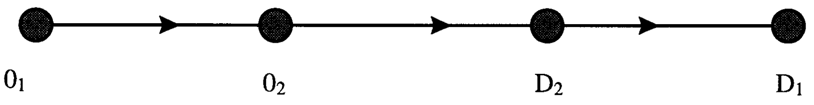


Figure E-5: Example of How Two Nodes Can Be Visited at the Cost of One

The savings value that is denoted by SAV are computed for all of the pairs of nodes in the network to determine how much can be saved by visiting two nodes on the same route. The SAV is the cost of linking the two origin/destination pairs minus the cost of linking the two pairs separately. This value may be either positive, negative, or blank space or zero for cancellation. Similar to the Clarke-Wright savings and pair insertion procedures, the most negative or least positive SAV value determines the first two origin/destination pairs to be assigned to the first vehicle. The next smallest SAV value determines the next two pairs to be assigned to the second vehicle and so forth. Therefore, if one pair has been assigned to a vehicle then the other pair is assigned to that same vehicle. Thus, this process is continued until all of the vehicles are assigned with a pair of nodes. Once all the available vehicles are assigned at least one pair of nodes then additional steps added to check the availability of the nodes. For instance, let *i* and *j* represent the next pair to be assigned based on the SAV being the minimum of the remaining values. If the node *i* is already assigned but the node *j* is not then *j* needs to be assigned to the same vehicle as *i*. On the other hand, if *i* has been assigned to a vehicle and *j* to another vehicle then the pair is already taken care of by the previous assignments. However, if neither node can be assigned to a vehicle then the pair is held until one of the nodes in the pair is assigned to one of the available vehicles. It is absolutely necessary to check that each assignment will lead to feasible individual routes for the vehicles. Thus, the pair selection method is similar to the pair insertion or Clarke-Wright savings algorithms.

The two vehicles routing problem can be solved by using the pair insertion that is a tour construction heuristic. Consider Example D.3 for two vehicles using the pair insertion procedure. The same technique can be applied to find the final tour in the Figure D-7 and assign two paths to the two available vehicles. Figure E-6 shows the final complete tour with two routes assigned to the two vehicles. The two routes are directed on either side of the depot node.

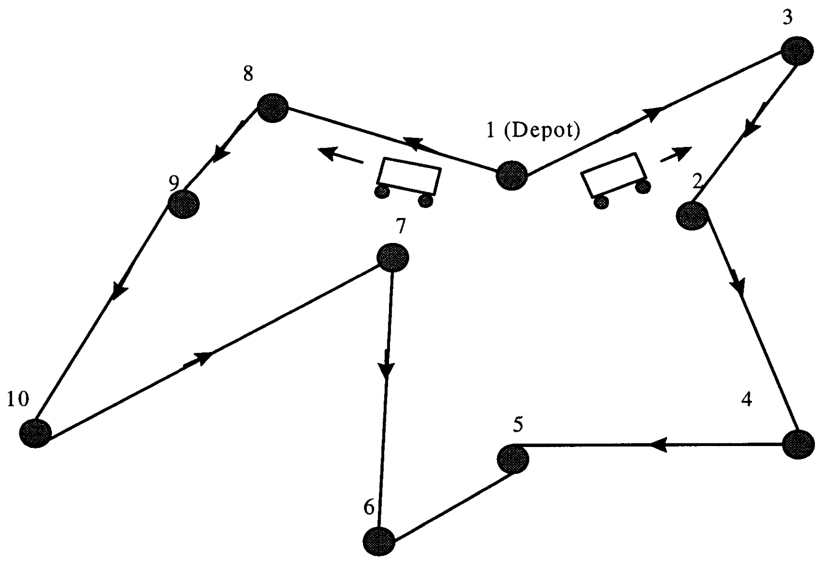


Figure E-6: Two Routes for Two Vehicles Using Pair Insertion

No additional vehicle can be added to the tour in the Figure E-6 without modifying the entire tour. Thus, pair selection or pair insertion algorithm can be applied to solve for more than two vehicles routing problem. The pair insertion or selection algorithm can be directly applied to the three vehicles problem. Therefore, the three vehicles are assigned to the nodes simultaneously. First, a node acting as an origin node is assigned to each of the three vehicles. Then, the pair of nodes that are associated with the highest saving value are assigned to the first vehicle, the second pair to the second vehicle and so forth. Thus, the simultaneous routes are built instead of just a single route being built. Consider Example D.3 that found a final tour using pair insertion procedure. Slightly different final tour is produced if three vehicles are considered for the same example. Figure E-7 shows the three individual routes that are produced using the pair insertion or selection algorithm. The nodes in the different routes are distinguished by the patterns on the nodes. By inspection, the routes are arranged slightly different order to improve the overall tour.

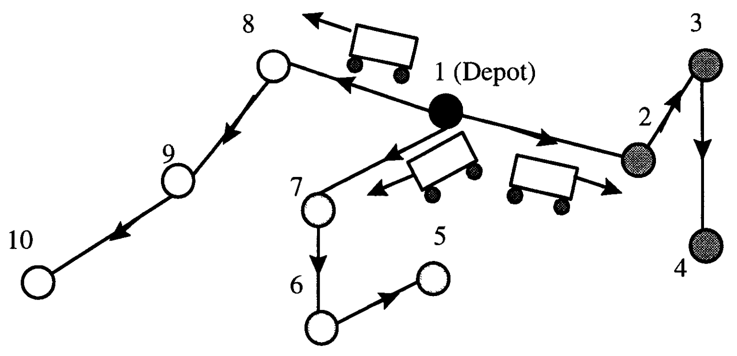


Figure E-7: Three Routes for Three Vehicles Using Pair Selection

The "cluster first, route second" concept can be applied to the same example above to produce three individual routes for the three available vehicles. First, the network in Example D.3 can be divided into three smaller regions just as in the Figure E-7. Then, the pair selection or insertion algorithm can be applied to each of the divided regions to produce three different routes for the three vehicles. The resulted three individual routes are exactly same as the ones shown in the Figure E-7. Thus, either approach of allocating the nodes to the three vehicles is feasible to produce the final routings.

Interchange heuristic can be applied to the pair selection method to improve majority of individual tours. This heuristic identifies two nodes on different routes that can be switched to reduce the combined cost of the two routes. The penalty cost of being on the tour that it is on and not on one of the other tours is computed for all of the candidate nodes to be switched. Then the penalty costs are compared to find the ones that appear to be good for a switch. All of the possible switches of two pair can't be considered for the practical reason. Thus, the pair of nodes that are associated with the highest four or five penalty costs can be attempted for a switch. Several experimental results show that the interchange heuristic rarely identified a profitable switch on the initial tour found using greedy heuristic or pair selection procedure. Therefore, the interchange heuristic is eliminated from further implementation.

E.4 Extended 3-Optimal Heuristic

For the single VRP, the 3-optimal heuristic generally resulted solutions that are within a few percentage of the optimal solution. However, the 3-optimal heuristic is limited by the precedence relationship in the multiple vehicle routing problem. In other words, the nodes that are assigned to a vehicle must be visited by the same vehicle. Otherwise, the nodes may get assigned to a vehicle and visited by another vehicle. Thus, arc interchanges are rarely profitable just as the interchange heuristic for the greedy algorithm. For instance, given the restriction that each of the three vehicles must visit the same number of nodes, there exists only one way to remove three arcs feasibly. If two arcs are removed from one route, then the arcs must come from each of the three individual routes, all at the same node. Some preliminary results show that the 3-optimal heuristic is not effective to solve MVRP. Thus, this heuristic is eliminated from further consideration in extending the algorithms to the MVRP.

E.5 Extended Mixed Solution

The mixed routing algorithm can be extended to solve multiple vehicle routing problem. The extended mixed routing algorithm uses the minimum spanning tree, some heuristics to improve the tour, and partitioning techniques. The details and example of the extended mixed routing algorithm is included in Chapter 2, section 2.3.1. Furthermore, the greedy heuristic and mixed routing algorithms for a single vehicle are combined to produce routes for multiple vehicles. Chapter 2, section 2.3.3 contains details of the algorithm and example of this algorithm.

References

[1] G. R. Armstrong, *The Single and Multiple Vehicle Pickup and Delivery Problem: Exact and Heuristic Algorithms,* Doctor of Philosophy's thesis, University of Tennessee, 1981.

[2] D. J. Bertsimas, G. V. Ryzin, *Stochastic and Dynamic Vehicle Routing Problem in the Euclidean Plane,* Master's thesis, Massachusetts Institute of Technology, 1990.

[3] T. Y. Chow, *Software Architecture, Path Planning, and Implementation for an Autonomous Robot,* Master's thesis, Massachusetts Institute of Technology, 1996.

[4] B. J. Chun, S. H. Lee, *Algorithms and Heuristics for Time-Window-Constrained Traveling Salesman Problems,* Master's thesis, Naval Postgraduate School, 1985.

[5] A. E. Goldfeld, H. E. Myers, "Ban the Land Mines," *Boston Globe*, Boston, MA, 1996.

[6] W. D. Hall, *Resource-Coordinated Hierarchical Planning for Real-Time Autonomous Systems*, Master's thesis, Massachusetts Institute of Technology, 1992.

[7] A. V. Hill, D. C. Whybark, "Comparing Exact Solution Procedures for the Multi-Vehicle Routing Problem," *The Logistics and Transportation review,* Berkeley, CA, 1976.

[8] J. J. Jarvis, O. Kirca, *Pick-Up and Delivery Problem: Models and Single Vehicle Exact Procedures,* Master's thesis, Georgia Institute of Technology, 1985,

[9] D. S. Kang, "Quarterly Technical Progress Report: SMART Program," Contract N00174-95-D-0075, CDRL A004, Cambridge, MA, 1996.

[10] B. Koontz, *A Multiple Vehicle Mission Planner to Clear Unexploded Ordnance from a Network of Roadways*, Master's thesis, Massachusetts Institute of Technology, 1997.

[11] B. Koontz, C. Tung, E. Wilson, D. Kang, "SMART: SMall Autonomous Robotic Technician," *Proceedings of the Symposium on Technology and the Mine Problem, Naval Postgraduate School,* Monterey, CA, 1996.

[12] R. C. Larson, A. R. Odoni, *Urban Operations Research,* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[13] E. J. Malafeew, *An Autonomous Control System for a Planetary Micro-Rover,* Master's Thesis, Massachusetts Institute of Technology, 1993.

[14] "Unexploded Ordnance (UXO) An Overview," prepared for the Federal Advisory Committee for the Development of Innovative Technologies in cooperation with: Department of Defense Explosives Safety Board; Naval Explosive Ordnance Disposal Technology Division; U.S. Army Environmental Center; Bureau of Land Management; PRC Environmental Management, Inc., Indian Head, MD, 1996.

[15] C. H. Papadimitriou, K. Steiglitz, *Combinatorial Optimization Algorithms and Complexity,* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[16] J. Pearl, *Heuristics: Intelligent Strategies for Computer Problem Solving,* Addison-Wesley, Reading, MA, 1984.

[17] J. F. Shapiro, *Column Generation Models for Vehicle Routing and Scheduling Problems,* Master's thesis, Massachusetts Institute of Technology, 1988.

[18] S. J. Steiner, *Mapping and Sensor Fusion for an Autonomous Vehicle,* Master's thesis, Massachusetts Institute of Technology, 1996.

[19] C. Trott, *Electronics Design for an Autonomous Helicopter,* Master's of Engineering thesis, Massachusetts Institute of Technology, 1997.

[illegible]