were reassigned) 5.2 times during execution.

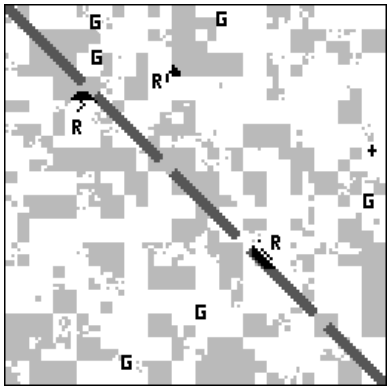

Figure 6: Advanced Scenario

6. Conclusion and Future Work

This system, along with prior work in dynamic planning for outdoor navigators, has demonstrated that complex missions can be performed using a reasonable computational capacity via a straightforward and feasible expansion on existing planning and navigational frameworks. It is expected that this design will permit the execution and optimization of a large class of previously unattempted missions for mobile robots.

The dynamic environments shown in these examples only include those where obstacles are added and a path always exists to all goals. This system could easily be extended to handle obstacles which are time-dependent by having the map cleared of all obstacles when the current map implies a goal cannot be reached. This would cause the robots to re-explore their environment in its entirety.

Given the ability of this type of planner to effectively select between millions of alternative possibilities in a manner which permits real time optimization of mission plans, it is the intent of future work to implement a mission planner capable of understanding the full grammar discussed in Section 2. The system will have the ability to distribute computation, use local inter-robot collision avoidance schemes, and gracefully degrade optimality when presented with computationally intractable plans. The system will be demonstrated on two autonomous off-road mobile robots as well as being further validated in simulation.

7. Acknowledgments

This research was sponsored by ARPA, under contracts “Perception for Outdoor Navigation” (contract number DACA76-89-C-0014, monitored by the US Army Topographic Engineering Center) and “Unmanned Ground Vehicle System” (contract number DAAE07-90-C-R059, monitored by TACOM.)

8. References

[1]Allen, J., et al. *Readings in Planning*, Morgan Kauffman Publishers, San Mateo, California, 1990

[2]Cameron, J.M., MacKenzie, D.C. “MissionLab: User Manual for Missionlab preliminary version 0.5,” Personal Communication. December, 1994.

[3]Feldman, J.A., Sproull, R.F. “Decision Theory and Artificial Intelligence II: The Hungry Monkey,” *Cognitive Science 1*, pp.158-192, 1977.

[4]Fikes, R.E., Nilsson, N.J. “STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving,” *Artificial Intelligence*, Vol. 2., pp.199-208, 1971.

[5]Kelly, A. “An Intelligent Predictive Control Approach to the High Speed Cross Country Autonomous Navigation Problem,” Carnegie Mellon Robotics Institute Technical Report, CMU-RI-TR-95-33.

[6]Lacroix, S., et al. “Autonomous Navigation in Outdoor Environment: Adaptive Approach and Experiment,” Proc. ICRA 1995, pp. 426-432.

[7]Langer, D., et al. “An Integrated System for Off-Road Navigation,” Proc. ICRA, 1994, pp. 414-419.

[8]Latombe, J.C. *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[9]Lawler, E.L., et al. *The Travelling Salesman Problem*, John Wiley & Sons, 1985.

[10]Le Pape, C. “A Combination of Centralized and Distributed Methods for Multi-Agent Planning and Scheduling,” Proc. ICRA 1990, pp. 488-493.

[11]Mataric, M.J. “Minimizing Complexity in Controlling a Mobile Robot Population,” Proc. ICRA, 1992, pp. 830-835.

[12]Parker, L. *Heterogeneous Multi-Robot Cooperation*, Ph.D. Thesis, MIT, Feb. 1994.

[13]Parsons, D., Canny, J. “A Motion Planner for Multiple Mobile Robots,” Proc. ICRA, 1990, pp. 8-13.

[14]Rugg-Gunn, N., Cameron, S. “A Formal Semantics for Multiple Vehicle Task and Motion Planning,” Proc. ICRA, 1994, pp. 2464-2469.

[15]Stentz, A. “Optimal and Efficient Path Planning for Partially-Known Environments,” Proc. ICRA, 1994.

[16]Stentz, A. “The Focussed D* Algorithm for Real-Time Replanning,” Proc. IJCAI, August 1995.

[17]Stentz, A., Hebert, M. “A Complete Navigation System for Goal Acquisition in Unknown Environments,” *Autonomous Robots*, 2(2), 1995.

observe these unknown obstacles with their sensors, these areas are colored black, as was the ravine initially, indicating that they are now known. Each robot's sensors can view approximately 4 pixels away from the robot itself.

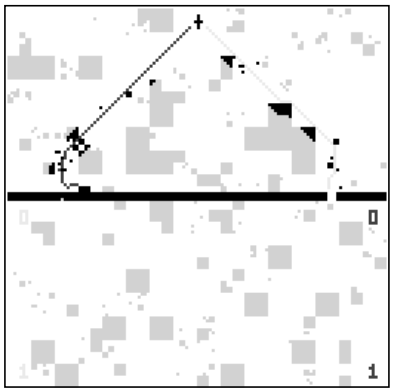

Figure 3: Simulation Step 2

In Figure 3, the dark robot has reached the left bridge. However, the left doorway is now black, indicating that the robots sensors have observed the bridge and thus, that the planners are aware that the robot cannot move through this region. Here, the Mission Planner realizes the difficulty and redefines each robot's goal selection to minimize the time to mission completion. The light robot will now go to the left goals, since it is closer to them than the dark robot, and the dark robot will go to the right goals.

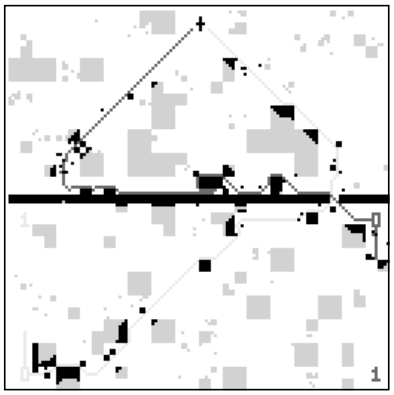

Figure 4: Simulation Step 3

Figure 4 shows that the light robot swapped the order of its goals since it went slightly out of the way while travelling to the upper left goal. This occurred because the swap reduced the total time for mission completion. In addition, each robot has reached the first of its two goals and has started heading towards the second. Finally, in Figure 5, both robots returned to base, and the light robot picked a quicker route around the right side of the indicated obstacle because of received obstacle information from the dark robot during its less efficient traverse to the left side.

By continuously comparing the cost of all alternative missions, the Mission Planner ensured that each robot was always moving to the correct goal with respect to optimizing the total time to complete the mission. The Dynamic Planner ensured the path followed to each goal was optimal. Map information was shared between robots to minimize the time for repeated traverses of familiar areas. Each robot encountered many unknown objects in the environment, and navigated in an optimal manner to its current goal given its current knowledge of the world. The worst case cycle time of the mission planner was under one second on a Sun Microsystems Sparc10. All processing was performed on one computer. Typically, in a real scenario, one computer per robot (at a minimum) would be available.

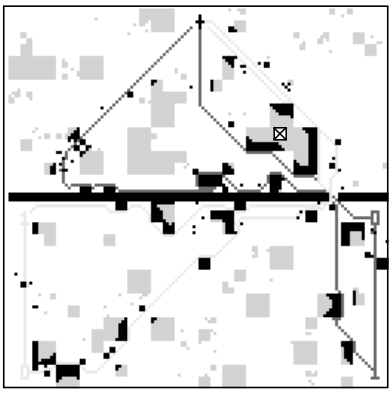

Figure 5: Simulation Step 4

If dynamic mission planning had not been performed, light would have reached the goals on the right and returned to the base quickly, leaving dark far behind, having to take much extra time to reach its goals (given that bridge was actually blocked.) Instead, the robots effectively cooperated through the Mission Planner, optimizing time to mission completion via reassignment of goals as permitted by the mission statement. Furthermore, since dynamic planning was also used when the robot was navigating toward its single goal, it minimized computation performed by not requiring complete replanning of the paths to its goal each time an unexpected obstacle was encountered.

A more complex scenario involving 3 robots and 6 goals is in shown in Figure 6. Randomly generated scenarios of this sort[3] were simulated 1000 times, both with the type of dynamic planner used in the first example, and with a static planer which never changed the ordering of the goals from the initial plan. Dynamic mission planning was found to result in missions which were approximately 25% more efficient on average than those statically planned. On average, missions were changed (i.e. goals

3. The obstacles, number of open/closed bridges along the ravine, and location of robots and goals are all randomized.

sible to move to a simulated environment which mimics the real environment without loss of applicability, as the identical software could be used to control the robot. This permitted testing multiple robot scenarios without having to build the communications infrastructure necessary to support a live test. The only changes from an existing system[17] which are necessary to make a live run with a single robot would be to permit the Mission Planner to switch which Dynamic Planner (D*) advises the Local Navigator.

Second, the perception processing which converted range images into obstacle (cost) maps was eliminated by using a world which contained a binary obstacle map and a perception system which accessed this obstacle map directly.

Also, the processing was done on a single machine as a single process. The difficulties of using asynchronous processes had been solved for much of this application[17], therefore, it was possible to perform useful tests on an improved system without the additional complexity introduced by asynchronisity.

The prototype system functions as follows. A world of random obstacles is generated, with M goals and N robots located randomly in this world. M dynamic planners are created, each tasked with maintaining the optimal path for all N robots to the planner's assigned goal. On each cycle of the main loop of the prototype system, each dynamic planner performs a fixed amount of computation updating the current $N \times M$ costs of all robots to all goals based on the new perception information provided. Periodically, the Mission Planner exhaustively searches the alternative solutions to the mission using the latest computed costs. Based on the current best solution to the MTSP problem (which may have been computed on a earlier cycle if the Mission Planner was not active during the current cycle), each robot is then moved a single step in the direction of its goal as determined by the Dynamic Planner assigned to this goal. In this way, the planners cycle more slowly than the local navigator.

D* (Dynamic A*) is used as the dynamic planner to maintain the costs from each robot to each goal. The D* algorithm was developed to replan paths in real-time. D* produces an initial plan based on known and assumed information, and then incrementally repairs the plan as new information is discovered about the world. The repaired plan is guaranteed to be optimal and is functionally equivalent to replanning from scratch. The robot moves in such a way that at every point P along its traverse, the robot is following an optimal path from P to the goal, given all information known in aggregate at P. For large applications (i.e., environments with a million states or more), D* is more than 200 times faster than brute-force replanning. D* is fast for many motion planning problems because new information is generally discovered by sensors carried on-board the robot and thus impacts the only portion of the plan "local" to the robot's current state; therefore, most of the time only a small portion of the existing plan is affected.

The Mission Planner uses a straightforward exhaustive search of all possible alternatives for solving the MTSP which comprises the mission statement. Segments of the mission can be limited to the paths planned independently by D* because the triangle inequality holds; D* always maintains the shortest path between two points, and there is always a path AC which costs no more than sum of the costs of AB and BC for any point B. Since most reasonable workspaces are relatively open, explicit methods for avoiding robot-robot collisions are not used. It is assumed local methods will be effective at avoiding such conflicts. As goals are reached they are removed from the Mission Planner's search, thus speeding response time as the mission proceeds. Whenever the mission plan is changed (a robot needs to drive towards a different goal), a different D* instantiation provides the driving direction for that robot. The only change in functionality necessary to enable this prototype system to solve more complex plans is an improved mission planning component.

5. Results

A straightforward scenario begins in Figure 2. In this

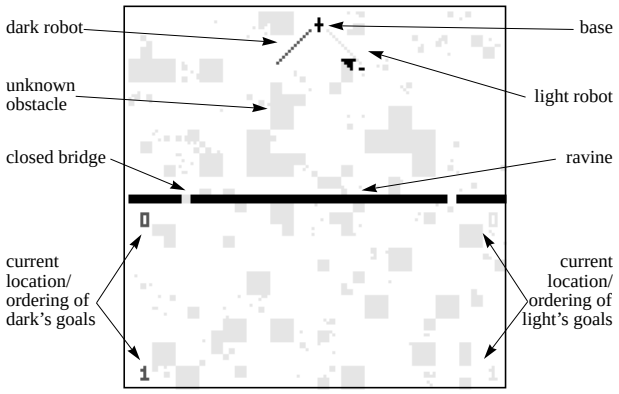

Figure 2: Simulation Step 1

example, N=2 and M=4. The cross in black at the top center is the base, and the left and right lines indicate the path of the "dark" and "light" robots. The Mission Planner has already planned that the dark robot should go to the goals on the left (numbered 0 and 1 in dark), and light should go to the goals on the right (numbered 0 and 1 in light). The black portion of the image corresponds to a ravine with two bridges that divide the environment in half. The grey regions, including in particular the blocked bridge on the left, are unknown obstacles in the environment. Thus, the dark robot believes it can go over the left bridge even though it is actually blocked. As the robots move and

Navigators to bias choice of steering direction) and updating costs of the these paths (used by the Mission Planner to reassign robots and goals if a more efficient solution is found.) The Mission Planner can cycle slowest of all, as changes it makes typically have a large effect, and thus need not happen instantly. This is possible because the delay from the time when the relevant data arrives until the time when the change is computed is much smaller than the savings in mission completion time produced by the change. For example, if during a drive to work it is discovered that the current route has heavy traffic, it is profitable to continue driving while checking a map (before deciding to change the route) because the change will typically save significantly more time than the 1-2 minutes of "wasted" driving done while the decision was being made.

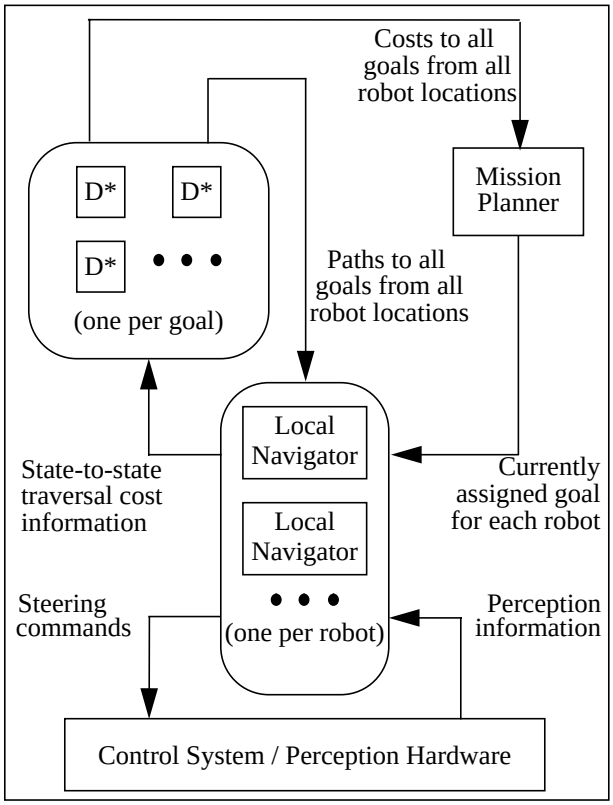

Figure 1: System Data Flow

This system can be generalized easily because of the abstracted data passed between layers. Any local navigator which:

- permits the description of the world as an undirected graph of locations (a 4/8-connected grid is acceptable),
- possesses a method for translating perception data into state-to-state transition cost information,[2] and
- accepts advice concerning a path to a goal,

can be used with this system. This permits easy implementation of the planning system on a variety of navigators and their corresponding perception systems.

4. Prototype System

4.1. Example Problem

Given the above system design, the only limitation in the complexity of feasible missions is the ability of the Mission Planner to understand them and find optimal solutions for them in a timely fashion. The example problem used in this prototype is a variety of the Multiple Travelling Salesman Problem (MTSP). The scenario which the Mission Planner optimizes is one where there are N robots each starting at different locations, M goals which must be visited by some robot, and 1 Base to which all robots must eventually return. The Mission is optimized with respect to getting all robots back to the Base as quickly as possible. All robots are homogenous in all respects.

This problem exhibits the most challenging aspect of designing a general Mission Planner - the problem it must optimize is NP-hard[9]. Virtually any mission involving multiple robots is going to require that the planner evaluate thousands (if not millions) of alternatives as quickly as possible. By picking a scenario with this order of difficulty, the feasibility and benefit of a such a solution can be tested. Any optimal solution to this problem is going to run in time exponential to the number of robots and/or goals. Thus, by picking a problem where the cycle time of the Mission Planner is much greater than the cycle time of the Local Navigator, it's possible to determine if a benefit from such a process can be gained, and thus, if a more general Mission Planner can be successfully implemented which optimizes more complex missions.

4.2. Prototype Operation

A number of simplifications were made to the design of the prototype system to permit its rapid implementation without loss of applicability.

First, work by Stentz and Hebert[17] has demonstrated the feasibility of using D* coupled with a local navigator to drive a single real robot to a single goal. Thus, it was pos-

1. The notion of "optimal" in dynamic planning is different from that in static planning. In static planning, the environment is fully known and the optimal path, once planned, remains constant during execution. When the state of the world is not fully known, a mission plan is optimal if each robot R, at each point P_R along its traverse given the aggregate information available at P_R, follows the lowest cost path to its goal such that the mission is completed with the cost of the longest path any robot traverses is minimized

2. Formally, the perception system must convert workspace information into a C-space representation for planning. The assumption of relatively open operational workspaces implies that there should be sufficient space between obstacles to allow planning in 2-D C-spaces via region growing.

Stentz[15][16].

Several multi-robot systems which operate in a largely reactive fashion exist. Examples of these include Mataric's work on flocking behaviors[11], Parker's ALLIANCE[12], and Georgia Tech's MissionLab[2]. While each of these systems is capable of moving robots safely through their environment, little attention is paid to optimizing robot motion, building world models which may be used by other robots, or solving missions which involve explicit choices between alternatives. Others, such as Rugg-Gunn[14] and Le Pape[10], use more traditional world models and a centralized planning framework to plan and execute complex missions. While these systems do optimize motion and permit complex plans, they are not designed for the efficient replanning necessary for operation in dynamic environments.

In summary, though existing AI Planners are effective at sequencing actions and Motion Planners are effective at keeping the robot safe while achieving a single goal, neither alone addresses optimizing the execution of tasks which may involve multiple goals and multiple robots. Current multi-robot systems either are not designed to reason about complex mission-oriented plans or are not effective in dynamic environments. This paper discusses a mission planning system designed to satisfy the demands of dynamic planning for multiple goals in unstructured environments as needed for a variety of robot applications.

2. Problem Statement

The problem addressed can be stated as follows:

To support:

- motion planning and
- complex missions,

for:

- multiple robots,
- multiple concurrent goals, and
- dynamic environments,

using mobile robots which have:

- relatively open operational workspaces, and
- effective positioning, communication and perception,

permit:

- mission cost optimization,
- successful, optimized task completion, and
- dynamic replanning as world knowledge increases.

To address a general class of mission planning problems applicable to a variety of applications, we propose the following grammar in which missions may be expressed:

$$m \rightarrow M(r, g) \mid (m \wedge m) \mid (m \vee m) \mid (m \Rightarrow m)$$
$$r \rightarrow R_i \mid (r \wedge r) \mid (r \vee r)$$
$$g \rightarrow G_j \mid (g \wedge g) \mid (g \vee g) \mid (g \Rightarrow g)$$

where M, R, and G, conceptually represent Move, Robot, and Goal, respectively. M denotes the "Move" operator and subscripted capital letters represent constants, so $M(R_1,G_1)$ means that Robot number 1 is instructed to move to Goal number 1. $A \wedge B, A \vee B, A \Rightarrow B$ are interpreted as A "and", "or", "then" B, respectively.

To demonstrate how a robotic mission might be expressed in this grammar, consider the following expression:

$$M(R_1 \vee R_2, G_1) \rightarrow M(R_2, G_2 \rightarrow G_3)$$

This states that either Robot 1 or Robot 2 should move to Goal 1, then Robot 2 should visit Goal 2 and Goal 3 in that order. This could correspond to the data gathering scenario where either robot must visit one location (Goal 1), and then Robot 2 (which is perhaps equipped with a sensor that Robot 1 lacks) should go to Goals 2 and 3.

This paper studies mission expressions which are a subset of this grammar to determine the feasibility of a system which can address a large class of mission planning problems defined by the grammar above and constrained by the demands of dynamic planning. The problem consists of N robots initially located at different locations, M goals which must be reached (by any robot), and 1 base to which all robots must return. The performance metric is the longest distance travelled by any robot while visiting goals and then returning to base. In the grammar, this mission is expressed as:

$$M(R_1 \vee \ldots \vee R_N, G_1 \wedge \ldots \wedge G_M) \rightarrow M(R_1 \wedge \ldots \wedge R_N, G_{base})$$

The rest of this paper will discuss the technical approach of this prototype, its applicability to real tasks, the results obtained with this system, and directions for future work.

3. Technical Approach

Figure 1 shows the system data flow. There are three asynchronous processes operating. First, each Local Navigator (1 per robot) evaluates its current field of view and chooses a steering direction based on that field of view and its assigned goal. Second, the Dynamic Planners (D*'s) continually update the paths for all robots to all goals as necessitated by incoming map information from the Navigators. The Dynamic Planners provide this path information to the Local Navigators, and also provide each path's respective cost to the Mission Planner. Finally, the Mission Planner continually updates the assignment of robots to goals by selecting assignments which minimize the cost of the mission and then informs each Local Navigator of its currently assigned goal.

The benefit from the asynchronous operation of these processes is that each may cycle at a time scale appropriate to the assistance it provides towards optimizing[1] the mission performance. The Local Navigators cycle rapidly to ensure the robot never collides with anything in the environment. The Dynamic Planners cycle as quickly as possible, updating paths to the goals (used by the Local

Dynamic Mission Planning for Multiple Mobile Robots

Barry L. Brumitt, Anthony Stentz

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Planning for multiple mobile robots in dynamic environments involves determining the optimal path each robot should follow to accomplish the goals of the mission, given the current knowledge available about the world. As knowledge increases or improves, the planning system should dynamically reassign robots to goals in order to continually minimize the time to complete the mission. In this paper, an example problem in this domain is explored and performance results of such a dynamic planning system are presented. The system was able to dynamically optimize the motion of 3 robots toward 6 goals in real time, improving the average overall mission performance compared to a static planner by 25%. A preliminary design for a practical solution to a wider class of problems is also discussed.

1. Introduction

Work in outdoor autonomous mobile robotics to date has been primarily concerned with maintaining robot safety while moving in the environment. Mobile robotic tasks, particularly in the outdoor realm, have been largely limited to path tracking with obstacle avoidance[5][6][7], or to single-robot / single-goal scenarios[17]. For mobile robotics to be effective in real-world applications, more than one robot must be able to share a potentially unknown workspace, and complicated missions with interdependencies between these robots must be feasible.

In many such applications, there exists a need to coordinate the actions of robots to achieve a goal. Example applications include construction, military reconnaissance, warehouse automation, post-office automation, and planetary exploration. A mission in these applications could consist of a series of locations to be visited in any order by any robot, or perhaps a series of locations each robot should visit before returning to its recharging area. Existing systems for autonomous robots fail to permit the execution of complex missions for multiple robots because they do not adequately address the dynamic planning and mission planning aspects of these problems.

Dynamic planning is the process of efficiently updating the plans (i.e. changing each robot's plan) when further knowledge of the world is gained or when the control or local obstacle avoidance system causes actual execution to differ from planned. Every perception system can introduce unavoidable noise into the system, "surprise" the planner with unexpected terrain, limit the motion of the robot due to a narrow field of view, and/or "change its mind" about terrain when viewed from a different position. These problems imply that estimated costs of mission plans may change frequently and that the robot may not follow a previously recommended path to the goal.

Mission planning is the process of determining what each robot should do to achieve the goals of the mission. Since environments encountered in the above domains are unknown, the optimal mission plan may change dramatically due to new information received from any involved robot. Thus, a mission planner in an unknown environment must operate in a dynamic fashion.

Mission planning problems have been addressed in the fields of AI planning and motion planning. Complete multi-robot systems have also been developed which coordinate particular multi-robot scenarios.

AI planners tend to be domain independent and frequently use heuristic search methods to determine appropriate sequences of operators to reach a goal state. An excellent collection of papers on AI planning can be found in *Readings in Planning*[1]. A quintessential example of this type of symbolic system is STRIPS[4]. While such systems are very effective at finding sequences of actions to reach a goal state and (as extended by Feldman and Sproull[14]) minimizing the cost of accomplishing the task, there is a fundamental paradigm mismatch; physically situated mobile missions are not easily linked with logic-oriented symbolic planners.

Motion planning systems have ranged from simple control schemes for local obstacle avoidance to omniscient schemes which reason about the entirety of the robot workspace. Motion planners reason exclusively about movement, producing paths that avoid collision, maximize safety, and/or minimize distance travelled. Most traditional motion planning systems[8] (both for single and multiple[13] robot scenarios) are not immediately applicable to dynamic mission planning problems due to the constraints of a dynamic environment -- having to replan every time a map discrepancy is found or a robot fails is unacceptably slow. However, a dynamic planner capable of planning paths in real time for a single robot and a single goal set has been designed and implemented by