COMPLETE AND SCALABLE MULTI-ROBOT PLANNING IN TUNNEL ENVIRONMENTS

Mike Peasgood [*] **John McPhee** [**]
Christopher Clark [*]

[*] *Lab for Intelligent and Autonomous Robotics,*
Department of Mechanical Engineering,
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1
{mike, chris}@lair.uwaterloo.ca
[**] *Motion Research Group,*
Department of Systems Design Engineering,
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1
mcphee@real.uwaterloo.ca

Abstract: This paper addresses the challenging problem of finding collision-free trajectories for many robots moving to individual goals within a common environment. Most popular algorithms for multi-robot planning manage the complexity of the problem by planning trajectories for robots sequentially; such decoupled methods may fail to find a solution even if one exists. In contrast, this paper describes a multi-phase approach to the planning problem that guarantees a solution by creating and maintaining obstacle-free paths through the environment as required for each robot to reach its goal. Using a topological graph and spanning tree representation of a tunnel or corridor environment, the multi-phase planner is capable of planning trajectories for up to $r = L-1$ robots, where the spanning tree includes L leaves. Monte Carlo simulations in a large environment with varying number of robots demonstrate that the algorithm can solve planning problems requiring complex coordination of many robots that cannot be solved with a decoupled approach, and is scalable with complexity linear in the number of robots.

Keywords: mobile robots, efficient algorithms, path planning, trajectory planning

1. INTRODUCTION

The use of multiple mobile robots in a common environment is required for the automation of many operations, such as underground mining and warehouse management. In such applications, multiple vehicles are required to drive autonomously between different locations, preferably taking the shortest possible route while avoiding collisions with static objects and other vehicles. This paper presents an algorithm for efficiently determining collision-free paths for many vehicles in environments composed of tunnels or corridors, as may be found in these applications. The problem addressed by this research is demonstrated by the multi-robot planning task pictured in Figure 1(a).

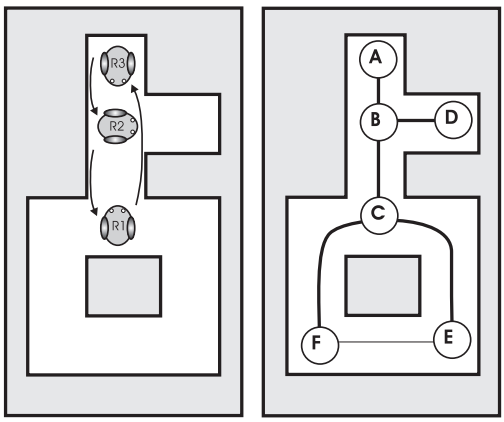
In this scenario, the environment is constructed of corridors or tunnels that are wide enough for only a single robot to travel, and we assume differential drive robots that can rotate in place. The objective in this example is to shift the

positions of each robot, such that robot R_1 moves to the initial position of R_3, R_3 to the position of R_2, and R_2 to the position of R_1. Our goal is to find an algorithm that can solve the simple problem shown in Figure 1(a), yet is scalable to a large number of robots (> 100) densely situated in a large environment.

Many methods have been proposed for planning the motion of one or more robots; refer to (Latombe, 1991) and (LaValle, 2006) for detailed reviews. Planning algorithms can be evaluated in terms of completeness (whether they are guaranteed to find a solution if one exists), complexity, and optimality.

Most multi-robot planning algorithms fall into one of two categories, *coupled* and *decoupled*. *Coupled* algorithms, such as (Svestka and Overmars, 1998), plan the trajectories of all robots in the environment concurrently. By combining the states (poses) of the individual robots together into a system state representation, a sequence of state transitions can be found that will move all robots to their respective goals. Using complete search methods, such as A*, coupled algorithms can achieve completeness and optimality, and can solve the problem shown in Figure 1(a). Their limitation is in searching the large configuration space that grows in dimension as each additional robot is added to the environment. One approach to reducing the size of the search space is to create probabilistic roadmaps (PRMs) through the environment; this method was shown in (Svestka and Overmars, 1998) to be probabilistically complete and demonstrated in simulation for up to 5 robots.

Decoupled methods plan for the motion of individual robots, rather than planning the motion of all robots simultaneously. Such methods may use a decentralized architecture, allowing independent planning based methods such as maze-searching (Lumelsky and Harinarayan, 1997) or potential fields (Ge and Cui, 1997), or they may use a centralized architecture planning for all robots with a single processor, allowing for coordination of collision-free plans for all robots. Centralized decoupled planners typically determine individual trajectories sequentially and combine the plans of all robots to avoid collisions, as in (Erdmann and Lozano-Perez, 1987), (Bennewitz *et al.*, 2001) and (Guo and Parker, 2002). By planning the motion of robots sequentially, decoupled methods have lower complexity and greater scalability than a coupled planner; however, this comes at the cost of completeness and optimality. The problem in Figure 1(a) for example cannot be solved by a sequential planner. By selecting the optimal plan for any robot independently, an obstacle is created in the space-time map that cannot be avoided by the other two robots.



(a) A planning problem (b) Graph-based map

Fig. 1. A multi-robot planning problem requiring coordination of 3 robots, and a graph-based representation of the environment.

This paper presents an alternative *multi-phase* planning method that can solve these coordinated planning problems, and is scalable to a large number of robots in a large environment. A graph representation of the environment is first created, and a spanning tree through the graph is selected. For the tunnel and corridor environments considered here the segments are only one lane wide, reducing the complexity of a suitable topological map generation process compared to the general case. A multi-phase planning approach then takes advantage of the properties of the graph and spanning tree to create and maintain obstacle-free paths while robots move to their respective goals.

2. MAP REPRESENTATION

Occupancy grids are a common map representation for robot navigation, and are easily derived from range sensor measurements. However, for motion planning problems, graph representations such as topological maps and roadmaps are often more efficient.

For the simple example of Figure 1(a), a topological graph G can be constructed as shown in Figure 1(b), consisting of $N = 6$ nodes and $E = 6$ edges. We assume that the initial and goal positions of all robots lie on the nodes of the graph; in this representation, the goal positions of robots R_1, R_2, and R_3 are nodes A, C, and B respectively.

Given the graph representation, we can also select a spanning tree T^* in the graph, that is, a subset of edges connecting all nodes without forming any loops. A given spanning tree has L *leaf nodes*, and $N - L$ *interior nodes*. A suitable spanning tree for the example is shown in Figure 2 where node C, closest to the geographic center of the map, is selected as the root. Selecting all edges except for $E - F$ into the spanning tree as shown gives

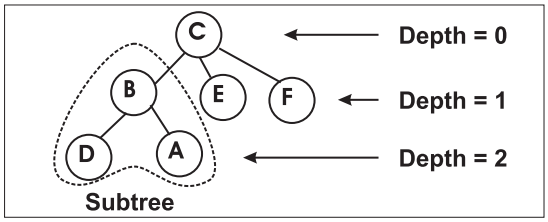


Fig. 2. A spanning tree T^* for the graph representation of the environment rooted at node C, and a subtree T_B rooted at node B.

$L = 4$ leaf nodes, A, D, E, and F, and two interior nodes, B and C.

In general the spanning tree is not unique, and a heuristic approach for tree selection is required that tends to maximizes the number of leaves and minimize the distance between leaves. We have found an effective approach is to iteratively add edges to the tree that lead to the nodes with the maximum number of branches.

3. MULTI-PHASE PLANNING ALGORITHM

The multi-phase algorithm finds a feasible solution to the trajectory planning problem by breaking the problem into a sequence of four subproblems, each of which can be solved in time proportional to the number of robots by taking advantage of the graph and spanning tree structure developed above.

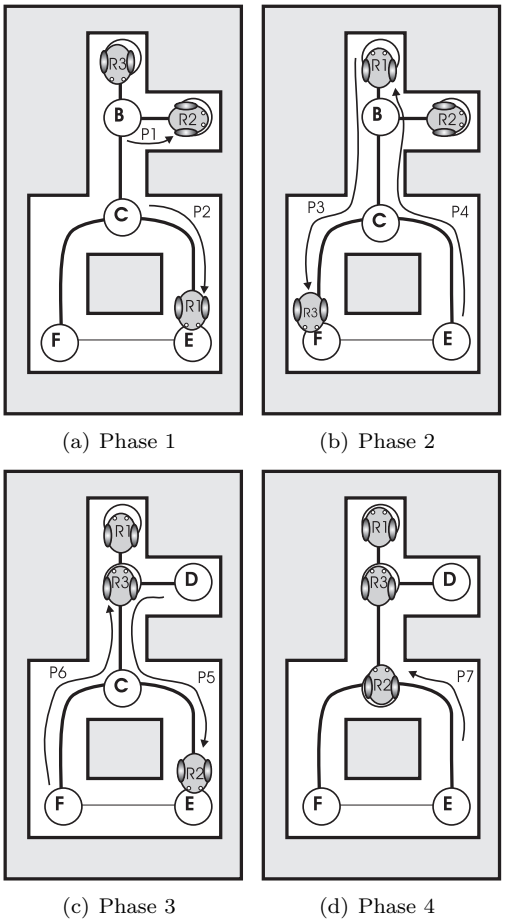
First, the robots move to the leaves of the spanning tree. We then use the following observations to plan a sequence of paths to drive each robot to its goal. For a system with $r < L$ robots:

Lemma 1: When all robots occupy leaf nodes, any robot can move to any interior node in the graph G.

Lemma 2: When all robots occupy leaf nodes, any two robots can swap positions.

Lemma 1 is clear since an obstacle-free path can be found between any two nodes through the spanning tree T^*, and no robots remain as obstacles on the interior nodes of the tree. Lemma 2 follows, since with $r < L$ robots, there is always one unoccupied leaf N_{tmp} in the spanning tree. Robots R_i and R_j at nodes N_i and N_j can swap positions by moving R_i to N_{tmp}, R_j to N_i, and R_i to N_j.

Note that these lemmas guarantee that one path can be found using the spanning tree through the graph under certain conditions. However, a shorter path may exist using graph edges that are not in the tree. Where an A* search is used in the following steps, the entire graph is searched, and the shortest paths will be selected.



(a) Phase 1 (b) Phase 2

(c) Phase 3 (d) Phase 4

Fig. 3. A multi-phase solution to the planning problem of Figure 1(a). Refer to text for details of each step.

3.1 Phase 1: Reaching Leaf Nodes

In Phase 1, we develop a plan that will move all robots to leaf nodes of the spanning tree. This is accomplished by repeatedly selecting a robot R_i that is not currently on a leaf node, and selecting an unoccupied leaf node L_i. This is guaranteed to succeed, since there are L leaf nodes, and $r < L$ robots to occupy them. In the example in Figure 1(a), node D may be selected as the leaf node for robot R_1.

An A* search is then used to find a path (sequence of nodes) P_i, from the initial position of robot R_i to the target leaf node L_i, *ignoring all other robots in the system.* If any robots are found on a node of the path P_i, let R_j be the robot on a node of P_i that is closest to L_i. In this case, we plan for R_j to move to L_i instead, using the obstacle-free subpath of P_i that connects R_j to L_i.

In Figure 3(a), since robot R_2 is an obstacle between the selected robot R_1 and leaf node D, a path P_1 moving R_2 from node C to D is added instead. Continuing the process, R_1 remains to be moved to a leaf node, and either node E or F may be selected, indicated by path P_2.

3.2 Phase 2: Filling Leaf Node Goals

In Phase 2, we require that all robots with goal positions located on leaf nodes fill those goals. Since Phase 1 has moved all robots to leaf nodes, referring to lemma 2, we can move any robot to any leaf goal node and maintain an obstacle-free path through the spanning tree.

For the example scenario, R_1 is the only robot with a leaf node goal at node A, and that node is occupied by robot R_3. The obstacle R_3 is therefore first moved to the unoccupied leaf node F, after which a plan for R_1 to its goal node is added, as indicated by paths P_3 and P_4 in Figure 3(b). The final step of the swap is not required in this phase; R_3 may remain at node F leaving the interior nodes unoccupied.

3.3 Phase 3: Robots with Interior Node Goals

In Phase 3, we move all robots with goals on interior nodes into positions where they can reach those goals without creating an obstruction for another robot. The need for this arrangement step can be seen in 3(b): robots R_2 and R_3 have goals on the interior nodes C and B respectively, and if either moves directly to its goal, it will create an obstacle for the other.

For a general algorithm to resolve this potential deadlock, we consider the problem in terms of robot positions relative to their goals within the spanning tree structure. Let T_{G_i} be a subtree of the spanning tree with root at the goal node G_i of robot R_i. The deadlock condition occurs only if another robot R_j is in the subtree T_{G_i}, and is blocked from reaching its goal outside of the subtree when R_i occupies its goal node G_i. We can prevent this condition by:

(1) moving robots to nodes within the subtree of their goal nodes, and
(2) ordering the depth of the robots within the subtree based on the depth of their goals.

To accomplish this task, we process robots in the order of the *depth* of their goals, that is, the distance from the goal node to the root of the spanning tree. For each robot R_i, we determine whether it is already in T_{G_i}. If not, we test whether filling the goal G_i will create an obstacle for any robots in the subtree T_{G_i}. In so, we select the deepest positioned such robot R_j within T_{G_i}, and swap the positions of R_i and R_j, as described in lemma 2. If no robots will be blocked into the subtree, R_i can be moved G_i directly. This method achieves the two conditions required above to avoid deadlock conditions when filling interior node goals.

The total path length can be reduced by only partially completing the swap in some cases:

- If the temporary unoccupied leaf used for swapping is not in T_{G_i}, robot R_j may remain at that leaf rather than completing the swap to the previous position of R_i.
- If R_j is the only robot that would be blocked into the subtree, robot R_i can fill its goal node immediately after robot R_j has been moved.

In the example, the goals of robots R_2 and R_3 are interior nodes C and B, with C being the root of the spanning tree T^*. R_3 has the deepest goal node B, so is processed first. Node B is the root of the subtree containing nodes A and D, as shown in Figure 2, so we must check for robots that would be blocked into the subtree. Referring to Figure 3(b), R_2 at node D is such a robot. We therefore move R_2 to an unoccupied leaf node, then plan robot R_3 to its goal node, indicated by paths P_5 and P_6. This leaves R_2 and R_3 in subtrees of their goal nodes, and in the same depth order as their goals, as required.

3.4 Phase 4: Filling Interior Node Goals

In Phase 4, we fill the remaining goals on interior nodes. If we plan for robots with goals closest to the top of the tree first, an obstacle-free path for each robot is guaranteed by the arrangement determined in Phase 3, where the robots are sorted in order of the depth of their goals. For the example scenario, this requires planning robot R_2 to its goal at node C, resulting in the desired goal configuration shown in Figure 3(d).

3.5 Complexity Analysis

The plan completed at the end of Phase 4 will move all robots to their respective goals, as required for a *complete* planner. In each of the 4 phases, we iterate once over the set of r robots, and required at most 3 (in the case of swapping) A* plans for each. Each A* search has a fixed complexity C that depends on the size of the graph and the heuristic used, but remains independent of the number of robots in the environment. The total complexity of the 4-phase planning method is therefore $O(r \cdot C)$ for r robots.

3.6 Building a concurrent plan

The plan segments P_i determined in phases 1-4 are collision-free with only one robot moving at any time. A plan of shorter duration can be created by overlapping the individual segments

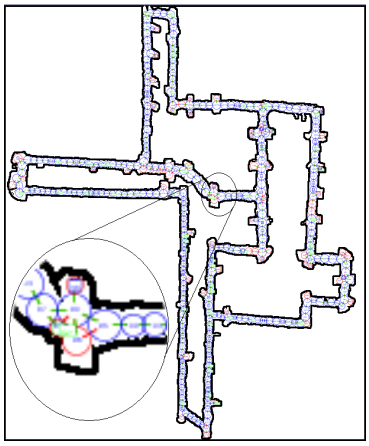


Fig. 4. Example simulation environment

in time as much as possible without introducing any collisions. Each successive segment of the original plan is added to a concurrent plan by first considering it appended to the end of the plan. The start position of the segment is then moved earlier in time until the motion in the new segment would create a collision between robots in the concurrent plan. The motion of the robot in the new segment is then incorporated into the concurrent plan.

4. SIMULATION RESULTS

The planner described above was implemented and evaluated in Monte-Carlo simulations in the underground mine map shown in Figure 4, using between 3 and 60 robots. A topological map was generated from an occupancy grid by finding adjacent circular regions of open space (nodes) and connecting all adjacent nodes by edges. The spanning tree selected for the graph contains 63 leaf nodes, allowing for motion planning of up to 62 robots in the environment. Random initial and goal positions are selected for each robot. As expected from the analysis above, the multi-phase planner finds a collision-free plan for every configuration.

For comparison, a *Decoupled Planner* using a sequential A* planning approach for each robot was also implemented, which randomly selects a priority sequence of robots. This sequential planner finds the shortest collision-free path for each robot through the space-time map, avoiding obstacles including the trajectories all previously planned robots. The results of such a planner are dependent on the priority sequence selected, so up to 250 randomly selected priority sequences were selected for each case in an attempt to find a sequence for which a plan could be found. The plots in the following sections show the results of applying the two different algorithms to the same randomly-generated problems in the mine map.

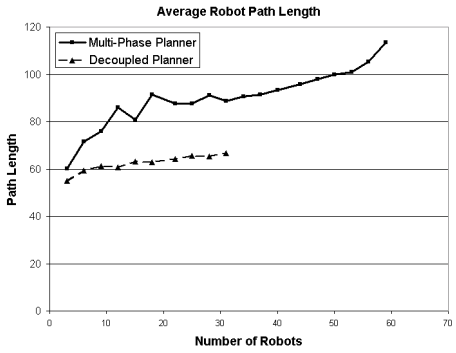


Fig. 5. Average robot path length generated by each planner

4.1 Planning Success Rate

The first measure of the algorithm performance is the success rate of finding a feasible solution. As expected for a complete algorithm, the success rate of the multi-phase planner is 100% for up to 62 robots given a spanning tree with 63 leaves. However, the sequential planner failed to find solutions for 10% of the randomly generated problems with 22 robots, and failed to find solutions for all problems with 34 or more robots.

4.2 Average Robot Path Length

The average distance required for each robot to travel to reach its goal is plotted in Figure 5. The results indicate that the multi-phase planner consistently generates longer paths for each robot, particularly as the number of robots increases. This is not unexpected, since the planner first directs robots to positions other than their goals in order to create an obstacle-free path for the final phases of the process. For 22 or more robots, where the sequential planner begins to fail for some problems, the average path length is computed only for those scenarios where a solution was found.

4.3 Search Cost

The search cost is a measure of the complexity of the planning algorithm, or the time required to complete the search for a feasible solution. Figure 6 shows the CPU time required by each algorithm; the processing time has been normalized by the number of robots in the plan, and plotted on a logarithmic scale to show the exponential growth in complexity of the decoupled planning method. The values indicate the time required to find a feasible solution given the graph representation, and not the (one-time) cost of generating the map.

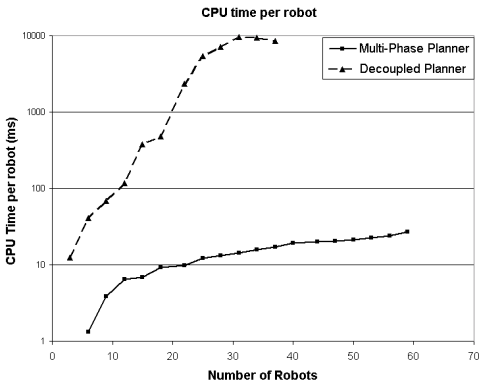


Fig. 6. Average CPU time used by each planner

These results demonstrate that while a decoupled approach can find shorter paths for simpler planning problems, the multi-phase planner involves much less computational cost. The cost of the sequential planner grows exponentially, since it requires many attempts with different random priority sequences to find a solution. The cost of the multi-phase planning algorithm, however, increases linearly with the increase in number of robots. For 60 robots, feasible plans were computed in less than 2 seconds (< 30 ms per robot) using a 1.5 GHz Pentium M processor.

5. DISCUSSION

In this paper, maps of tunnels and corridors were considered specifically, since they have a relatively simple topological representation and present a challenging environment for the coordination of a large number of robots. For more general cases, including arbitrary obstacles and non-holonomic motion constraints, the generation of a suitable roadmap or graph representation can be a challenging problem in itself. However, once a suitable graph is created, the multi-phase algorithm can be applied directly.

Considering the performance comparison between the sequential planner and the multi-phase planner, it may be advantageous to consider a hybrid approach, taking advantage of the features of both algorithms. By first generating a plan using the multi-phase planner, a feasible solution can be generated very efficiently. To search for a more optimal plan, a sequential planner could then be applied to the same problem, and permitted to run within the time bounds of the application.

The algorithm as presented here assumes a centralized planning architecture, where all information and resources are available at a single processing point. Incorporating this centralized planner into a distributed planning architecture, as proposed in (Clark *et al.*, 2003), will be another subject of future work.

6. CONCLUSIONS

This paper presented a multi-robot planning algorithm for tunnel and corridor environments that is based on a topological graph and spanning tree representation. By breaking the planning algorithm into several different phases, it was shown that the algorithm guarantees a solution to the planning problem, and is scalable with linear increase in complexity for up to $r < L$ robots given a spanning tree with L leaves. In comparison to a decoupled sequential planning algorithm, the multi-phase planner typically produces longer paths, but at a much reduced computational cost when planning for many robots.

7. ACKNOWLEDGEMENTS

We would like to thank Sebastian Thrun of the Stanford Artificial Intelligence Lab for the use of the robot-generated maps of underground mines.

This work is funded in part by NSERC.

REFERENCES

Bennewitz, M., W. Burgard and S. Thrun (2001). Optimizing schedules for prioritized path planning of multi-robot systems. In: *Proc. IEEE Int. Conf. on Robotics and Automation.* pp. 271–276.

Clark, C., S. Rock and J.C. Latombe (2003). Motion planning for multi-robot systems using dynamic robot networks. In: *Proc. IEEE Int. Conf. on Robotics and Automation.* pp. 4222 – 4227.

Erdmann, Michael and Tomas Lozano-Perez (1987). On multiple moving objects. *Algorithmica* **2**, 477–521.

Ge, S.S. and Y.J. Cui (1997). Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots* **13**(3), 207–222.

Guo, Y. and L. Parker (2002). A distributed and optimal motion planning approach for multiple mobile robots. In: *Proc. IEEE Int. Conf. on Robotics and Automation.* pp. 2612–2619.

Latombe, J.C. (1991). *Robot Motion Planning.* Kluwer Academic Publishers.

LaValle, S. M. (2006). *Planning Algorithms.* Cambridge University Press.

Lumelsky, V.J. and K.R. Harinarayan (1997). Decentralized motion planning for multiple mobile robots: The cocktail party model. *Autonomous Robots* **4**(1), 121–135.

Svestka, P. and M. Overmars (1998). Coordinated path planning for multiple robots. *Robotics and Autonomous Systems* **23**, 125–152.