Constraint-based Optimization of Priority Schemes for Decoupled Path Planning Techniques

Maren Bennewitz[1], Wolfram Burgard[1], and Sebastian Thrun[2]

[1] Department of Computer Science, University of Freiburg, Freiburg, Germany,
[2] School of Computer Science, Carnegie Mellon University, Pittsburgh PA, USA

Abstract Coordinating the motion of multiple mobile robots is one of the fundamental problems in robotics. The predominant algorithms for coordinating teams of robots are decoupled and prioritized, thereby avoiding combinatorially hard planning problems typically faced by centralized approaches. While these methods are very efficient, they have two major drawbacks. First, they are incomplete, i.e. they sometimes fail to find a solution even if one exists, and second, the resulting solutions are often not optimal. In this paper we present a method for finding and optimizing priority schemes for such prioritized and decoupled planning techniques. Existing approaches apply a single priority scheme which makes them overly prone to failure in cases where valid solutions exist. By searching in the space of priorization schemes, our approach overcomes this limitation. It performs a randomized search with hill-climbing to find solutions and to minimize the overall path length. To focus the search, our algorithm is guided by constraints generated from the task specification. To illustrate the appropriateness of this approach, this paper discusses experimental results obtained with real robots and through systematic robot simulation. The experimental results illustrate the superior performance of our approach, both in terms of efficiency of robot motion and in the ability to find valid plans.

1 Introduction

Path planning is one of the fundamental problems in mobile robotics. As mentioned by Latombe [10], the capability of effectively planning its motions is "eminently necessary since, by definition, a robot accomplishes tasks by moving in the real world."

In this paper we consider the problem of motion planning for multiple mobile robots. In particular, we are interested in planning paths for multiple robots operating in a single, shared environment, where physical limitations impose restrictions among the paths of the various robots. In such multi-robot problems, undesirable situations include congestions or deadlocks, which may prevent robots from reaching their goal locations. Since the size of the joint state space of the robots grows exponentially in the number of robots, planning paths for teams of mobile robots is significantly harder than the path planning problem for single robot systems. Therefore, existing approaches for single robot systems cannot directly be transferred to multi-robot systems.

The approaches for multi-robot path planning can roughly be divided into two major categories [10]: centralized and decoupled. In the *centralized* approach [3,19] the configuration spaces of the individual robots are combined into one composite configuration space which is then searched for a path for the whole composite system. Because

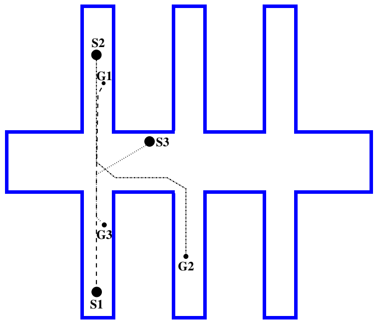


Figure 1. Situation in which no solution can be found if robot 3 has higher priority than robot 1.

the size of the joint configuration space grows exponentially in the number of robots, this approach suffers intrinsic scaling limitations. The major alternative are *decoupled* approaches [7,17,13,5,21,1,8]. Decoupled approaches compute separate paths for the individual robots. Subsequently, they apply heuristics for resolving conflicts between different robots (e.g., two robots attempt to occupy the same location at the same time). To deal with the still enormous search space, it is common practice to assign *priorities* to the individual robots [7,5,21,1,8]. Planning and re-planning is performed in accordance with these priorities. Priority schemes provide an effective mechanism for resolving conflicts that is computationally extremely efficient.

However, the priority scheme has a strong influence on whether a solution can be found and on how long the resulting paths are. To illustrate this, let us consider two examples. Figure 1 shows a situation in which no solution can be found *if* robot 3 has a higher priority than robot 1. Since the path of robot 3 is planned without considering robot 1, it enters the corridor containing its target location (marked G3) before robot 1 has left this corridor. Since the corridors are too narrow to allow two robots to pass by, robot 3 blocks the way of robot 1 so that it cannot reach its target point G1. However, if we change the priorities and plan the trajectory of robot 1 before that of robot 3, then robot 3 considers the trajectory of robot 1 during path planning and thus will wait in the hallway until robot 1 has left the corridor. Another example is shown in Figure 2 (left). If we start with robot 1 then we have to choose a large detour for robot 2 (see Figure 2, center). This is because robot 1 blocks the corridor. However, if the path of robot 2 is planned first, then we can obtain a much more efficient solution (see Figure 2, right). These two examples illustrate that the priority scheme has a serious influence on whether a solution can be found and on how long the resulting paths are. Moreover, it suggests that no single prioritization scheme will be sufficient for all possible multi-robot motion problems.

In this paper, we present a technique that searches in the space of all priority schemes while solving hard multi-robot planning problems. Our approach performs a randomized hill-climbing search in the space of possible priority schemes. Since each change of a scheme requires the computation of the paths for many of the robots, it is important to focus the search. Our method achieves this by exploiting constraints between the different robots which are derived from the task specification. This has two serious

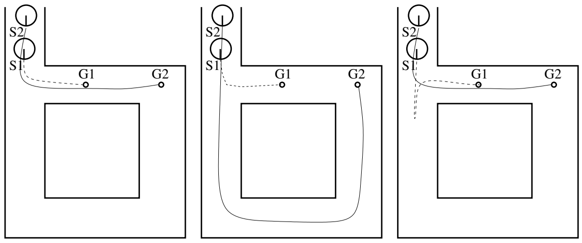


Figure 2. Independently planned optimal paths for two robots (left), sub-optimal solution if robot 1 has higher priority (center), and solution resulting if the path for robot 2 is planned first (right) .

advantages. First, it reduces the time required to find a solution, and second, it increases the number of problems for which a solution can be found in a given amount of time. Additionally, our algorithm is able to reduce the overall path length. It has anytime characteristics [22], which means that the quality of the solution depends on the available computation time; however, a solution may be available at any point in time.

Our approach has been successfully applied to physical mobile robots. These results are complemented by extensive simulations, to characterize the relation between the planning performance and various problem parameters. In all experiments, we found that our approach produces highly efficient motion plans even for very large teams of robots, for different environments, and using two different decoupled path planning techniques.

The paper is organized as follows. After discussing related work in the following section, we introduce two decoupled path planning techniques that will be used throughout this paper. Section 4 describes our algorithm for searching for priority schemes during planning. Finally, in Section 5, we present systematic experimental results illustrating the capabilities of our approach. The paper is concluded in Section 6.

2 Related Work

The problem of coordinating multiple mobile robots has received considerable attention in the robotics literature. As already mentioned above, the techniques for multi-robot path planning can roughly be divided into the *centralized* and the *decoupled* approaches [10].

Centralized methods consider the composite configuration space of all robots and search for a solution in the whole composite system. While these approaches (at least theoretically) are able to find the optimal solution to any planning problem for which a solution exists, their time complexity is exponential in the dimension of the composite configuration space. In practice, one is therefore forced to use heuristics for the exploration of the huge joint state space. Many methods use potential fields [2,3,20] to guide the search. These techniques apply different approaches to deal with the problem of local minima in the potential function. Other methods restrict the motions of the robots to reduce the size of the search space. For example, [9,19,11] only consider trajectories that lie on independent road-maps. The coordination is achieved by searching the

Cartesian product of the separate road-maps. Nevertheless, centralized approaches scale poorly to large numbers of robots.

Decoupled planners, in contrast, determine the paths of the individual robots independently and then employ different strategies to resolve possible conflicts. According to that, decoupled techniques are incomplete, i.e., they may fail to find a solution even if there is one. A popular decoupled approach is planning in the configuration time-space [7], which can be constructed for each robot given the positions and orientations of all other robots at every point in time. Techniques of this type assign priorities to the individual robots and compute the paths of the robots based on the order implied by these priorities. The method presented in [21] uses a fixed order and applies potential field techniques in the configuration time-space to avoid collisions. The approach described in [8] also uses a fixed priority scheme and chooses random detours for the robots with lower priority.

Another approach to decoupled planning is the path coordination method which was first introduced in [17]. The key idea of this method is to keep the robots on their individual paths and let the robots stop, move forward, or even move backward on their trajectories in order to avoid collisions (see also [6,4]). To reduce the complexity in the case of huge teams of robots, Leroy and colleagues [13] recently presented a technique to separate the overall coordination problem into sub-problems. Their approach, however, assumes that the overall problem can be divided into very small sub-problems. As various examples described below demonstrate, this assumption may not be justified in certain situations.

Unfortunately the problem of finding the optimal schedule is NP-hard for most of the decoupled approaches. To see, we notice that the NP-hard Job-Shop Scheduling problem with the goal to minimize maximum completion time [14,12] can be regarded as a special instance of the path coordination method. The decoupled and prioritized methods described above leave open how to assign the priorities to the individual robots. In the past, different techniques for selecting priorities have been used. For example, in [5] heuristic techniques are described that assign higher priority to robots which can move on a straight line from the starting point to their target location. In [1] all possible priority assignments are considered. Due to its (exponential) complexity this approach has only been applied to groups of up to three robots.

In this paper we present an approach to optimize priority schemes for arbitrary decoupled path planning methods. We perform a randomized hill-climbing search in the space of priority schemes. Thereby, we interleave the search for an optimal priority scheme with the planning of the paths of the robots. To guide the search, our algorithm exploits constraints between the robots that are extracted from the task description. As a result, our approach seriously reduces the time needed to find a solution to the path planning problem. Once a solution has been found, our algorithm is able to optimize the priority scheme in order to minimize the overall path length.

3 Prioritized A^*-based Path Planning and Path Coordination

The basic algorithm to compute optimal paths for single robots, which will be used throughout this paper, is a variant of the popular A^* search procedure [16]. To represent the environment of the robots we apply occupancy grids [15] which separate the environment into a grid of equally spaced cells and store in each cell $\langle x, y\rangle$ the probability $P(occ_{x,y})$ that it is occupied by a static object. In this section we also present the key ideas of decoupled prioritized path planning and describe how the A^* procedure can be utilized to plan the motions of teams of robots.

3.1 A^*-based Path Planning

Our system applies the A^* procedure to compute the cost-optimal paths for the individual robots, in the remainder denoted as the independently planned optimal paths for the individual robots. A^* addresses the problem of finding a shortest path from an initial state to a goal state in a graph. To search efficiently, the A^* procedure takes into account the accumulated cost of reaching a certain location $\langle x, y\rangle$ from the starting position, and an estimate of the cost of reaching the target location $\langle x^*, y^*\rangle$ from $\langle x, y\rangle$. By doing so, A^* tends to focus its search in parts of the state space most relevant to the problem of finding a shortest path. This property, which makes A^* an efficient search algorithm, has given A^* an enormous popularity in the robotics community. However, A^* also requires a discrete search graph, whereas robot configuration spaces are continuous. In our case we assume that the environment is readily represented by a discrete occupancy grid map—which is common in the mobile robotics literature.

The cost for traversing a cell $\langle x, y\rangle$ is proportional to its occupancy probability $P(occ_{x,y})$. Furthermore, the estimated cost for reaching the target location is approximated by $c \cdot ||\langle x, y\rangle - \langle x^*, y^*\rangle||$ where $c > 0$ is chosen as the minimum occupancy probability $P(occ_{x,y})$ in the map and $||\langle x, y\rangle - \langle x^*, y^*\rangle||$ is the straight-line distance between $\langle x, y\rangle$ and $\langle x^*, y^*\rangle$. Since this heuristic is admissible (see [16]), A^* determines the cost-optimal path from the starting position to the target location.

3.2 Decoupled Path Planning for Teams of Robots

A^* can easily be extended to the problem of decoupled and prioritized path planning. Recall that in the multi-robot path planning problem, many robots simultaneously seek to traverse an environment. If the robots could move freely regardless of other robot's positions, the problem could easily be decoupled into many local path planning problem, in which each robot applied A^* to determine its optimal path. However, the impossibility for robots to occupy the same location at the same point in time introduces non-trivial restrictions that have to be incorporated into the individual robot paths.

A common approach is the following. In a first path planning step, each robot computes its optimal path using A^*, without any consideration of the paths of the other robots. Clearly, the resulting paths might not be admissible since they lead to collisions, if executed. Thus, in a second planning step, each robot checks for possible conflicts with all other robots. Conflicts between robots are then resolved by introducing a priority scheme. A priority scheme determines the order in which the paths for the robots are re-planned. The path of a robot is then planned in its configuration time-space computed based on the map of the environment and the paths of the robots with higher priority.

Please note, the A^* search can also be used to plan the motions of the robots in the configuration time-space. As in the standard approach described above, the cost of traversing a location $\langle x, y \rangle$ at time t is determined by the occupancy probability $P(occ_{x,y})$. To incorporate the restrictions imposed by the paths of the other robots, however, we do not allow a robot to enter a cell that is occupied by a robot with higher priority at time t. In addition to the general A^*-based planning in the configuration time-space we consider a second and restricted version of this approach denoted as the path coordination technique [13]. It differs from the general approach in that it only explores a subset of the configuration time-space given by those states which lie on the initially optimal paths for the individual robots. The path coordination technique thus forces the robots to stay on their initial trajectories. The overall complexity of both approaches is $O(n \cdot m \cdot \log(m))$ where n is the number of robots and m is the maximum number of states expanded by A^* during planning in the configuration time-space (i.e. the maximum length of the OPEN-list). Due to the restriction during the search, the path coordination method is more efficient than the general A^* search. Its major disadvantage, however, lies in the fact that it fails more often.

As already discussed above, the introduction of a priority scheme for the decoupled path planning leads to serious reduction of the overall complexity. Whereas there are schemes leading to a viable solution with collision-free paths, it is easy to see that there are schemes for which no solution can be found. In addition to the fact, that the order in which the robots may plan their paths has a profound impact on the ability of finding a solution, even the quality of the solution depends heavily on the priority scheme. Examples of such situations were already discussed in the introduction to this paper. Unfortunately, the problem of finding the optimal priority scheme, is a non-trivial matter. More specifically, the NP-hard Job-Shop Scheduling problem with the goal to minimize maximum completion time [14,12] can be regarded as an instance of the path coordination method. Therefore, we have to be content with possibly sub-optimal planning orders.

4 Finding and Optimizing Solvable Priority Schemes

This section describes our approach to searching in the space of priority schemes during decoupled path planning. As the examples given in Figures 1 and 2 illustrate, the order in which the paths are planned has a significant influence on whether a solution can be found and on how long the resulting paths are. This raises the question of how to find a priority scheme for which the decoupled approach does not fail and how to find the order of the robots leading to the shortest paths.

4.1 The Randomized Search Technique

Our algorithm for finding eligible priority schemes is a randomized search technique, similar to those reported in [18]. More specifically, our approach performs a randomized and hill-climbing search in order to optimize the planning order for decoupled and prioritized path planning techniques. Our approach starts with an arbitrary initial priority scheme Π and randomly exchanges the priorities of two robots in this scheme. If the new order Π' results in a solution with shorter paths than the best one found so far, we continue with this new order. Since hill-climbing approaches like this frequently get stuck in local minima, we perform random restarts with different initial orders of the robots. Thus, our approach interleaves the search for collision-free paths with the search for a solvable priority scheme.

4.2 Exploiting Constraints to Focus the Search

Whereas the plain randomized search technique produces good results, it has the major disadvantage that often a lot of iterations are necessary to come up with a solution. For example, we found that for ten robots in the environment shown in Figure 1 more than 20 iterations on average were necessary to find a solvable priority scheme. In this section we therefore present a technique to focus the search that tends to reduce the search time significantly. Our approach can be motivated through the situation depicted in Figure 1. In this situation, it impossible to find a path for robot 1 if the path of robot 3 is planned first, because the goal location of robot 3 lies on the optimal path for robot 1. The key idea of our approach is to introduce a constraint $p_i > p_j$ between the priorities of two robots i and j, whenever the goal position of robot j lies on the optimal path of robot i. In our example we thus obtain the constraint $p_1 > p_3$ between the robots 1 and 3. Additionally, we get the constraint $p_2 > p_1$, since the goal location of robot 1 lies too close to the trajectory of robot 2.

Although the satisfaction of the constraints by a certain priority scheme does not guarantee that valid paths can be found, orders satisfying the constraints more often have a solution than priority schemes violating constraints. Unfortunately, depending on the environment and the number of the robots, it is possible that there is no order satisfying all constraints. In such a case the constraints produce a cyclic dependency. The key idea of our approach is to initially reorder only those robots that are involved in such a cycle in the constraint graph. Thus, we separate all robots into two sets. The first group R_1 contains all robots that, according to the constraints, do not lie on a cycle and have a higher priority than the robot with highest priority which lies on a cycle. This set of robots is ordered according to the constraints and this order is not changed during the search. The second set, denoted as R_2 contains all other robots.

As an example, Figure 3 (left) shows a simulated situation with ten robots. Whereas the starting positions are marked by $S_0, \ldots, S_9$ the corresponding goal positions are marked by $G_0, \ldots, G_9$. The independently planned optimal trajectories are indicated by solid lines. Given these paths we obtain the constraints depicted in Figure 3 (right). According to the constraints, six robots belong to the group of robots whose order (at least in the beginning) remains unchanged during the search process. The robots in their order of priorities are 3, 6, 7, 2, 4, 9.

Initially, our algorithm only changes the order of the robots in the second group. After k iterations, we include all robots in the search for a priority scheme. In extensive experimental results we figured out that this approach leads to better results with respect to the overall path length, especially for large numbers of iterations. The complete algorithm is listed in Table 1.

If we apply this algorithm to the example shown in Figure 3 (left) under the constraints shown in Figure 4, the system quickly finds a solution. One typical result is the the order 0, 1, 5, and 8, for those robots that generate a cycle in the constraint graph. The corresponding collision-free paths for all robots are shown in Figure 3. This

Table 1. The algorithm to optimize priority schemes.

```
count := 0
FOR tries := 1 TO maxTries BEGIN
    IF count > k   // extensive search after k iterations
        select random order Π
    ELSE
        select order Π given fixed order for R1
        and random order for R2
    IF (tries = 1)
        Π* := Π
    FOR flips := 1 TO maxFlips BEGIN
        IF count > k // extensive search after k iterations
            choose random i, j with i<j
        ELSE
            choose random i, j with i<j and i,j∈ R2
        Π' := swap(i, j, Π)
        count := count+1
        IF moveCosts(Π') < moveCosts(Π)
            Π := Π'
    END FOR
    IF moveCosts(Π) < moveCosts(Π*)
        Π* := Π
END FOR
RETURN Π*
```

demonstrates, that the constraints drastically reduce the search space and still allow the system to quickly find solvable priority schemes.

5 Experimental Results

Our approach has been tested thoroughly on real robots and in extensive simulation runs. The two key questions addressed in our experiments were: (1) Solvability: Does our approach succeed more frequently in finding valid multi-robot paths than approaches with fixed prioritization? (2) Optimality: If our approach succeeds, does it generate more efficient plans? All experiments were carried out using different environments. To evaluate the general applicability, we applied our method to the two decoupled and prioritized path planning techniques described above. The current implementation is highly efficient. It requires less than 0.1 seconds on a 1000 MHz Pentium III to plan a collision-free path for one robot in all environments described below. The whole optimization for 10 robots with 10 restarts and 10 iterations per restart requires approximately one minute.

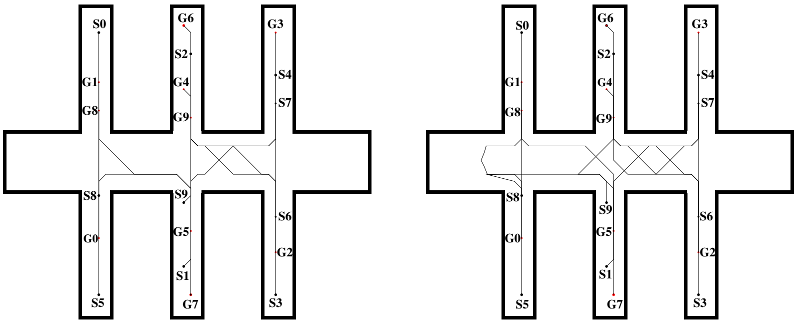


Figure 3. Independently planned paths for ten robots (left) and the paths resulting after priority optimization (right).

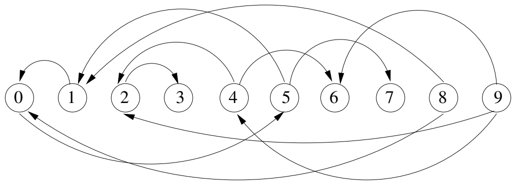


Figure 4. Constraintgraph generated according to the paths shown in Figure 3 (left).

5.1 Simulation Experiments

To elucidate the scaling properties of our approach to larger number of robots, we performed extensive simulation experiments. In particular, we were interested in characterizing the dependence between the performance of our system on various components of our approach. In our experiments, we analyzed the number of planning problems that can be solved using our strategy, the speed-up obtained by exploiting the constraints, and the reduction of the overall path length. In all experiments, we found that our approach produces highly efficient motion plans even for very large teams of robots, for different environments, and regardless of the specific baseline path planning technique (e.g., A^*).

Solved Planning Problems This first set of experiments was designed to characterize the effect of our search scheme on the overall number of failures. For each number of robots considered, we performed 100 experiments. In each experiment we randomly chose the starting and target locations of the robots. We applied four different strategies to find solvable priority schemes:

1. A single randomly chosen order for the robots.
2. A single order which satisfies the constraints for the robots in R_1 and consists of a randomly chosen order for the robots in R_2.

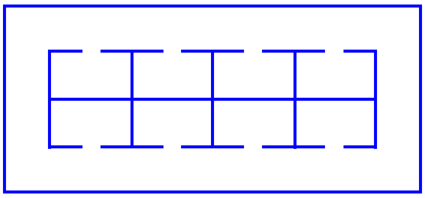


Figure 5. Cyclic corridor environment used for the simulation runs .

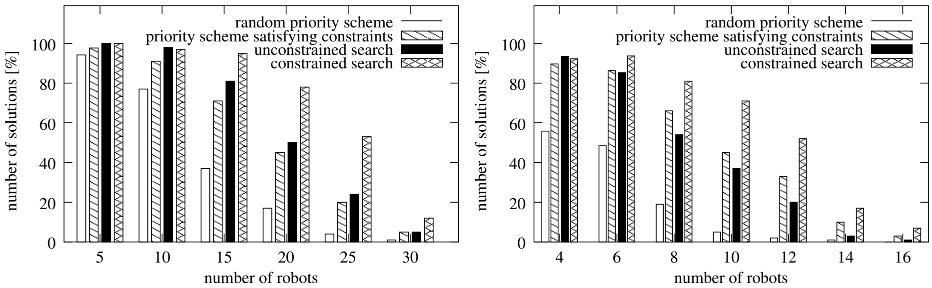


Figure 6. Solved planning problems for different strategies using A^*-based planning in the configuration time-space in the cyclic corridor environment depicted in Figure 5 (left) and the corresponding results obtained in the noncyclic corridor environment shown in Figure 3 (right).

3. Unconstrained randomized search starting with a random order and without considering the constraints.
4. Constrained randomized search starting with an order computed in the same way as strategy 2).

All four strategies can be cast as special cases of our algorithm. In the first two strategies the corresponding values for `maxTries` and `maxFlips` are 1. For the first strategy the value of the threshold k is 0. The strategies 3 and 4 only differ in the value of the threshold k. Whereas the unconstrained search is obtained by setting $k = 0$, the constrained search corresponds to a value of $k = \infty$.

Please note that in this experiment we chose a small number of iterations for the last two strategies in order to assess the advantages of the constrained search under serious time constraints. Particularly, we chose a value of 3 for the parameters `maxFlips` and `maxTries`. Obviously, the larger the number of iterations, the higher is the probability that a solution can be found by an arbitrary randomized search. However, larger numbers of iterations drastically increase the computation time. For each technique, we performed A^*-based planning in the configuration time-space and counted the number of solved planning problems.

Figure 6 (left) summarizes the results we obtained for the cyclic corridor environment depicted in Figure 5. The horizontal axis represents the number of robots, and the vertical axis depicts the percentage of solved path planning problems. As this result

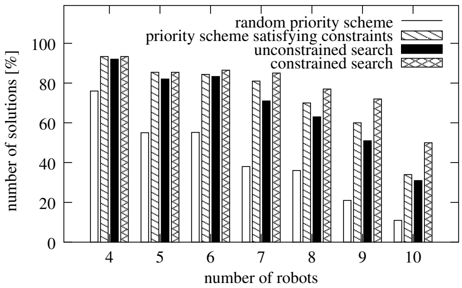


Figure 7. Solved planning problems for all four strategies using the path coordination method in the noncyclic environment depicted in Figure 3 (left).

illustrates, our constrained search technique succeeds more often than any of the alternative strategies. It is interesting to note that the second strategy, which exploits the constraints but considers only one scheme in each experiment, shows a similar performance than the unconstrained randomized search. To complement these results, we performed a similar series of experiments for the noncyclic corridor environment depicted in Figure 3. The results are shown in Figure 6 (right). Again, our constrained-based search outperforms all other strategies. All these and the following results are significant on the 95% confidence level.

To investigate the performance using a different baseline path planning algorithm, we applied all four strategies using the path coordination method instead of plain A^*. We used a variant of the environment depicted in Figure 3 with five corridors on both sides. Since the path coordination method restricts the robots to stay on their independently planned optimal trajectories, the number of unsolvable problems is much higher compared to the A^*-based planning in the configuration time-space. As can be seen from Figure 7, our constrained search leads to a much higher success rate that actually increases with the number of robots involved.

Speed-up Obtained by Exploiting the Constraints In this section, we are interested in one particular aspect of our approach, namely the ability to guide the search in the space of all priority schemes. More precisely, we pose the question how much the computation time necessary to find a solution can be reduced by constraining the search.

For the next set of experiments we increased the values of `maxFlips` and `maxTries` to 10 and evaluated in which iteration the first solution was found if the planning problem could be solved. Figure 8 (left) plots the results obtained for different number of robots in the cyclic corridor environment and Figure 8 (right) shows the same evaluation for the noncyclic environment. As can be seen, for both environments the unconstrained search needs significantly more iterations to generate a solution.

As these experiments suggest, the advantages of our constrained search is two-fold. On one hand, it requires fewer iterations than the unconstrained counter-part. On the other hand, it requires less computation, since the search is restricted to a subset of the robots, which reduces the number of paths that are generated in the search.

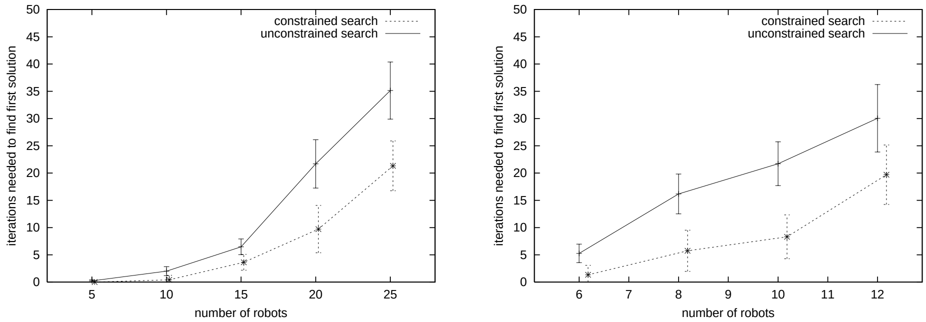


Figure 8. Iteration in which the first solution was found if the planning problem could be solved for the cyclic corridor environment (left) and the corresponding results obtained int the noncyclic corridor environment (right).

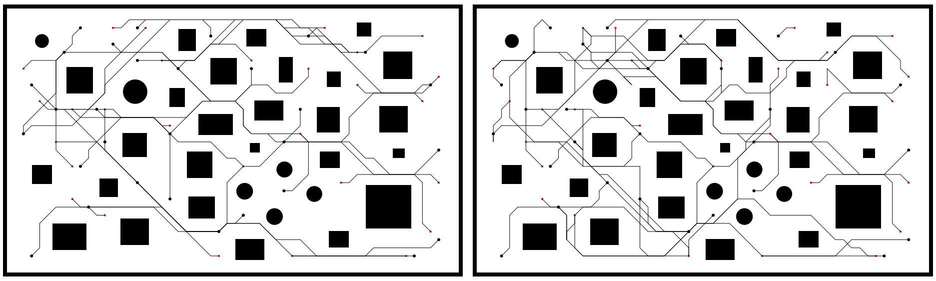
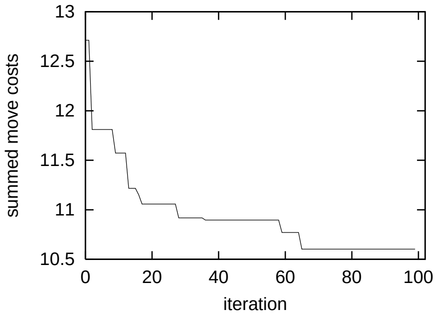


Figure 9. Independently planned optimal paths for 30 robots (left) and the resulting paths after priority optimization (right).

Influence on the Overall Path Length The previous experiments investigated the number of cases in which a solution can be found, as a function of the algorithm used for path planning. In this section, we will be interested in plan efficiency, that is, the overall plan execution time.

To show that our optimization technique is not limited to typical corridor environments, Figure 9 (left) shows the independently planned optimal paths for a team of 30 robots in an unstructured environment. By optimizing these paths over 100 iterations, we obtain the solution illustrated in Figure 9 (right). Figure 10 (left) plots the evolution of the summed move costs of the best solution found so far over time. As can be seen from the figure, after 100 iterations the overall move costs are reduced by 15%.

The final experiment in this section is designed to analyze the performance of our algorithm with respect to the overall path length. Since our algorithm in the beginning only considers a restricted set of priority schemes, and after k iterations explores the whole set of priority schemes, we are especially interested in how long the resulting paths are compared to the unconstrained search. We performed over 100 experiments in the cyclic corridor-environment and determined the average overall move costs at every iteration. The corresponding graphs are shown in Figure 10 (right). This plot con-



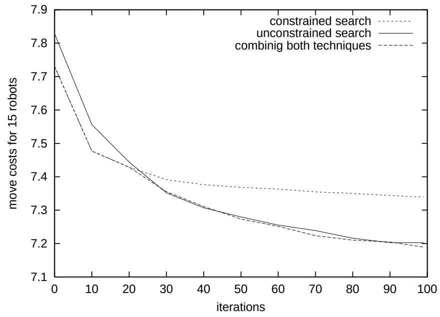


Figure 10. Summed move costs plotted over time for the planning problem with 30 robots shown in Figure 9 (left) and summed move costs plotted over time averaged over 100 planning problems for 15 robots in the cyclic environment (right).

tains the average move costs for three different strategies at each iteration. The first data set was obtained for the constrained search which corresponds to $k = \infty$. Using this strategy we reorder only those robots which lie on a cycle in the constraint graph. The data for the unconstrained search was obtained using $k = 0$. In this case our algorithm chooses arbitrary priority schemes regardless of the constraints which were extracted given the task specification. Finally, the third function labeled "combining both techniques" corresponds to the results obtained with our algorithm given $k = 20$.

Since the constrained search, which is guided by our heuristics, focuses the search on the robots that pose the most serious restrictions to the other robots, it more quickly finds a solution and accordingly has more time to optimize it. Thus, in the beginning, the constrained search outperforms the unconstrained search. After 20 iterations, however, the situation completely changes. Because the unconstrained search can explore many more priority schemes, it more often finds better solutions than the constrained search. Thus, after 20 iterations, the unconstrained search leads to better results than the constrained search. As can be seen from the figure, our approach combines the advantages of both methods. In the beginning, it applies the constraints to focus the search and to quickly find a first solution which is optimized subsequently. After 20 iterations it considers arbitrary priority schemes so that the resulting path length is reduced as in the unconstrained search.

Accordingly, our randomized search that initially uses the constraints to focus the search for a viable solution and afterwards uses the unconstrained search to optimize this solution inherits the advantages of both techniques with respect to efficiency and the overall resulting path length.

5.2 An Example with Two Real Robots

Figure 11 (center) illustrates a typical application example carried out in our office environment with our robots Albert and Ludwig. The robots are shown in Figure 11 (left and right). In this example, we used the general A^* procedure in the configuration time-space for local path planning. While Albert starts at the right end of the corridor of

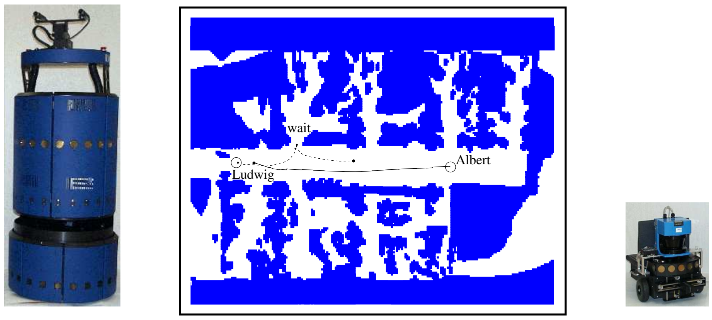


Figure 11. The mobile robots Albert (left) and Ludwig (right) and a real world application of A^*-based planning in the configuration time-space where Ludwig moves away in order to let Albert pass by (center) .

our lab and has to move to left end, Ludwig has to traverse the corridor in the opposite direction. Notice that no path for Albert can be found if the path of Ludwig is planned first, since Albert cannot reach its target point if Ludwig stays on its optimal trajectory. Because of that, the system alters the order of the two robots. Given the optimal path for Albert, our system plans a path for Ludwig which first leads it into a doorway in order to let Albert pass by. The resulting trajectories are shown in Figure 11 (center). Notice that at some point, the robot Ludwig waits to let the robot Albert pass by. In comparison, no solution can be found in this situation if the path coordination [13] technique is used.

In various other tests operating our two robots in our narrow hallways, we frequently observed the emergence of solutions where robots sensibly coordinated their behavior, e.g., by waiting for each other. However, we also notice that with only two robots, these experiments do not evaluate the utility of our search algorithm in priority scheme space, since there exist only two such schemes. Unfortunately, we currently have only two physical robots available in our lab, so that the experiment could not be carried out with larger groups of robots.

6 Conclusions

This paper presented an approach to optimize priority schemes for arbitrary decoupled and prioritized path planning methods for groups of mobile robots. Our approach performs a randomized hill-climbing search in the space of priority schemes in order to find a solution and to minimize the overall path length. To guide the search, our approach exploits constraints extracted from the current task specification.

The approach has been implemented and tested on real robots. In addition, extensive simulations were performed to complement the physical robot experiments, The experiments suggest that our technique significantly decreases the number of failures in which no solution can be found, compared to a range of alternative approaches. Additionally, our approach leads to a significant reduction of the overall path length. A further advantage of our method lies in its general applicability. Although we applied our optimization technique only to two different baseline path-planning techniques in this paper, it is not limited to these two techniques. Rather, it can be used to find and optimize paths generated with arbitrary prioritized path-planning techniques.

Apart from the promising results presented in this paper, there are different aspects for future research. First, in the experiments carried out here, we assumed equal constant velocities for all robots. In practice, teams often are inhomogeneous and contain different types of robots with different average velocities which has to be taken into account. Furthermore, the techniques considered here provide no means to react to possible deviations of the robots from their planned trajectories during the plan execution. For example, if one robot is delayed because unforeseen objects block its path, alternative plans for the robots might be more efficient. In such situations it would be important to have appropriate plan-revision techniques. Additionally, the delay of a single robot may result in a dead-lock during the plan execution. In this context, robot control systems require techniques to detect dead-locks while the robots are moving and to resolve such dead-locks appropriately.

References

1. K. Azarm and G. Schmidt. A decentralized approach for the conflict-free motion of multiple mobile robots. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1667–1674, 1996.
2. J. Barraquand, B. Langois, and J. C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Robotics and Automation, Man and Cybernetics*, 22(2):224–241, 1992.
3. J. Barraquand and J. C. Latombe. A monte-carlo algorithm for path planning with many degrees of freedom. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1990.
4. Z. Bien and J. Lee. A minimum-time trajectory planning method for two robots. *IEEE Transactions on Robotics and Automation*, 8(3):414–418, 1992.
5. S. J. Buckley. Fast motion planning for multiple moving robots. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1989.
6. C. Chang, M. J. Chung, and B. H. Lee. Collision avoidance of two robot manipulators by minimum delay time. *IEEE Transactions on Robotics and Automation, Man and Cybernetics*, 24(3):517–522, 1994.
7. M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica*, 2:477–521, 1987.
8. C. Ferrari, E. Pagello, J. Ota, and T. Arai. Multirobot motion coordination in space and time. *Robotics and Autonomous Systems*, 25:219–229, 1998.
9. L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars. Probabilistic road maps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, pages 566–580, 1996.
10. J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991. ISBN 0-7923-9206-X.
11. S. M. LaValle and S. A. Hutchinson. Optimal motion planning for multiple robots having independent goals. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1996.
12. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. Technical report, Centre for Mathematics and Computer Science, 1989.
13. S. Leroy, J. P. Laumond, and T. Simeon. Multiple path coordination for mobile robots: A geometric algorithm. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.
14. P. Martin and D.B. Shmoys. A new approach to computing optimal schedules for the job-shop scheduling problem. In *Proc. of the 5th International IPCO Conference*, pages 389–403, 1996.
15. H.P. Moravec and A.E. Elfes. High resolution maps from wide angle sonar. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 116–121, 1985.
16. N. J. Nilsson. *Principles of Artificial Intelligence*. Springer Publisher, Berlin, New York, 1982.
17. P. A. O'Donnell and T. Lozano-Perez. Deadlock-free and collision-free coordination of two robot manipulators. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1989.
18. B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard instances of satisfiability. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 1992.
19. P. Sveska and M. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1995.
20. P. Tournassoud. A strategy for obstacle avoidence and its application to multi-robot systems. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 1224–1229, 1986.
21. C. Warren. Multiple robot path coordination using artificial potential fields. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 500–505, 1990.
22. S. Zilberstein and S. Russell. Approximate reasoning using anytime algorithms. In S. Natarajan, editor, *Imprecise and Approximate Computation*. Kluwer Academic Publishers, Dordrecht, 1995.