Finding and Optimizing Solvable Priority Schemes for Decoupled Path Planning Techniques for Teams of Mobile Robots

Maren Bennewitz[†] Wolfram Burgard[†] Sebastian Thrun[‡]

[†]Department of Computer Science, University of Freiburg, 79110 Freiburg, Germany
[‡]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. The goal is to compute trajectories for the individual robots such that collisions between the robots are avoided. Especially in the context of multi-robot systems different undesirable situations can occur like congestions or even deadlocks. 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, the existing approaches for single robot systems cannot directly be transferred to multi-robot systems.

The existing methods for solving the problem of motion planning for multiple robots can be divided into two categories [10]. In the *centralized* approach [3, 15, 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. In contrast, the *decoupled* approach [6, 8, 13, 18] first computes separate paths for the individual robots and then resolves possible conflicts of the generated paths.

While centralized approaches (at least theoretically) are able to find the optimal solution to any planning problem for which a solution exists, their time complexity is ex-

ponential 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 field techniques [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, the approaches presented in [9, 11, 19] restrict the trajectories of the robots to lie on independent road maps. The coordination is achieved by searching the Cartesian product of the separate road maps.

Decoupled planners 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 [6] 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 [7] 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 [18]. The key idea of this technique 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 [4]). To reduce the complexity in the case of huge teams of robots [13] recently presented a technique to separate the overall coordination problem into sub-problems. This approach, however, assumes that the overall problem can be divided into very small subproblems, a serious assumption which, as various examples described below demonstrate, is often not justified. In general, therefore, a prioritized variant has to be applied.

The 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. [5] applied a heuristic which assigns higher priority to robots which can move on a straight line from the starting point to their target location. In [1] all possible assignments are considered. Due to its complexity this approach has only been applied to groups of up to three robots.

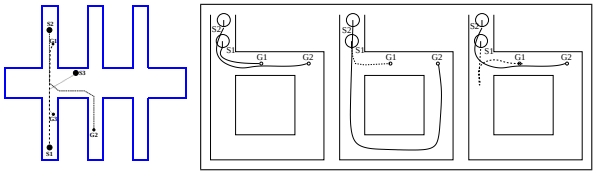

Figure 1: Situation in which no solution can be found if robot 3 has higher priority than robot 1 (leftmost image) and independently planned optimal paths for two robots (second image), sub-optimal solution if robot 1 has higher priority (third image), and solution resulting if the path for robot 2 is planned first (right).

For decoupled and prioritized methods the order in which the paths are planned has a serious influence on whether at all a solution can be found and how long the resulting paths are. To illustrate this, let us consider two examples. Figure 1 (leftmost image) 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 G_3) 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 G_1. 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 1 (three images on the right). If we start with robot 1 then we have to choose a large detour for robot 2 (see Figure 1, third image). 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 1, 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 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 once a solution has been found. It has anytime characteristics [22], which means that the quality of the solution depends on the available computation 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. In the following section, we introduce two decoupled path planning techniques that will be used throughout this paper. Section 3 describes our algorithm for searching for priority schemes during planning. Finally, in Section 4, we present systematic experimental results illustrating the capabilities of our approach.

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

2.1 A^*-based Path Planning

Our system applies the A^* procedure [17] to compute the cost-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.

Occupancy grids [16] 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. 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 [17]), A^* determines the cost-optimal path from the starting position to the target location.

2.2 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 problems, 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. In the remainder we denote the paths generated in this step as the independently planned optimal paths for the individual robots. Clearly, these 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.

As already mentioned, A^* search can also be used to plan the motions of the robots in the configuration time-space. To incorporate the restrictions imposed by the other robots we do not allow a robot to enter a cell that is occupied by a robot with higher priority at that time. 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 independently planned 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 a 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.

3 Finding and Optimizing Solvable Priority Schemes

This section describes our approach to searching in the space of priority schemes during decoupled path planning. The examples given above illustrate that 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 for which the length of the resulting paths is minimized.

3.1 The Randomized Search Technique

Our algorithm for finding eligible priority schemes for decoupled and prioritized path planning techniques interleaves the search for collision-free paths with the search for a solvable priority scheme. It performs a randomized search combined with hill-climbing. It starts with an arbitrary initial priority scheme and randomly exchanges the priorities of two robots in this scheme. If the new order results in a solution with shorter paths than the best one found so far, it continues with this new order. Since hill-climbing approaches like this frequently get stuck in local minima, it performs random restarts with different initial orders of the robots. The number of restarts and priority exchanges are controlled by the two parameters `maxTries` and `maxFlips`.

3.2 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 a situation with ten robots in the environment shown in Figure 2 (leftmost image) 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 the leftmost image of Figure 1. In this situation, it is 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. Initially, our algorithm only changes the order of the robots in the second group. After a certain number of iterations, we include all robots in the search for a priority scheme. In our experiments we figured out that this leads to better results with respect to the overall path length, especially for large numbers of iterations. The number of iterations carried out using the robots in R_2 only is controlled by a parameter denoted k in the remainder of this paper.

To illustrate our approach, again consider the situation with ten robots shown in the leftmost image of Figure 2. Whereas the starting locations 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 the center image of Figure 2. According to the constraints, in the beginning the order of the six robots 3, 6, 7, 2, 4, and 9 remains unchanged during the search process. Given the restricted search space, our system quickly finds a solution. In this example, we obtained the order 0, 1, 5, and 8, for the robots lying on a cycle in the constraint graph. The resulting corresponding collision-free paths for all robots are shown in the right image of Figure 2. This demonstrates, that the constraints can drastically reduce the search space and still allow the system to quickly find solvable priority schemes.

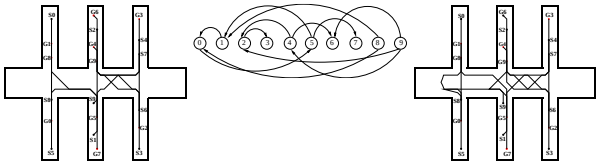

Figure 2: Independently planned paths for ten robots (left), resulting constraints (center), and the paths resulting after priority optimization (right).

4 Experimental Results

Our approach has been tested thoroughly on real robots and in extensive simulation runs. The key questions addressed in our experiments were: (1) Practicability: is our approach relevant and applicable to real robot systems? (2) Solvability: Does our approach succeed more frequently in finding valid multi-robot paths than approaches with fixed prioritization? (3) 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.

4.1 An Example with Real Robots

The goal of the first experiment is to demonstrate the applicability of our approach to real robot systems. This experiment has been carried out using the pioneer I robots of the CS-Freiburg RoboCup team. The task of the robots is to get into their initial formation which has to be done at

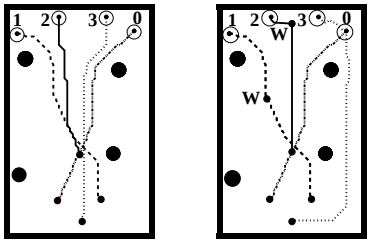

Figure 3: An application example with the Robots of the CS-Freiburg RoboCup team. The left image shows the independently planned optimal paths for the four robots and the resulting collision-free paths computed by our algorithm are depicted on the right.

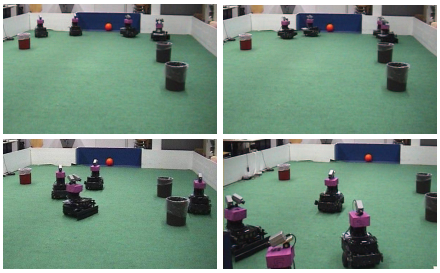

Figure 4: These four pictures show the robots carrying out the navigation plans. The top left images depicts the initial situation. In the top right image robot 2 makes space for robot 1 while robot 3 takes a detour. The lower left images shows robot 1 waiting to let robot 0 pass by. The lower right image shows the robots at their final locations.

the beginning of each match. Thereby they have to avoid colliding with other robots that are on the field already. In the particular example described here, the robots are deployed on one side of the field and have to move to their home positions on the other side. The left image of Figure 3 shows this initial configuration along with the independently planned optimal paths. As can be seen from this figure, these paths cross each other close to the center of the field leading to several several conflicts. Here we applied our approach to A^* search in the configuration time-space, and the paths of the robots were planned in the order 0, 1, 2, and 3. The resulting paths are depicted in the right image of Figure 3. As can be seen, the paths of the robots are changed in order to avoid collisions. Whereas the robots 1 and 2 shortly wait to let robot 0 pass by (the corresponding positions are marked with a "W" in the right image of Figure 3), robot 3 has to take a detour.

4.2 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., general A^* or the path coordination method).

4.2.1 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.
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).

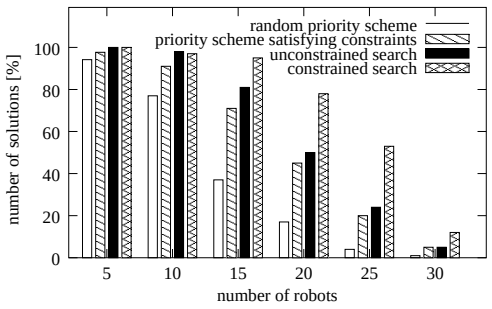

Figure 5: Solved planning problems for different strategies using A^*-based planning in the configuration time-space in the cyclic corridor environment depicted in Figure 6.

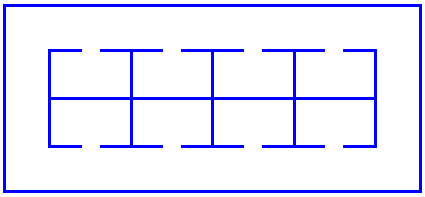

Figure 6: Cyclic corridor environment used for simulation experiments.

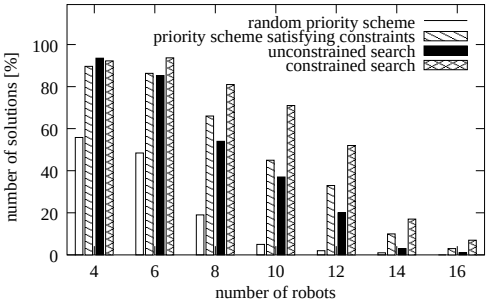

Figure 7: Solved planning problems for all four strategies using A^*-based planning in the configuration time-space in the noncyclic environment depicted in Fig. 2

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 $\texttt{k} = 0$, the constrained search corresponds to a value of $\texttt{k} = \infty$.

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 5 summarizes the results we obtained for the cyclic corridor environment depicted in Figure 6. The horizontal axis represents the number of robots, and the vertical axis depicts the percentage of solved path planning problems. As this result 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 2. The results are shown in Figure 7. 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 2 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 general A^*-based planning in the configuration time-space. As can be seen from Figure 8, our constrained search leads to a much higher success rate.

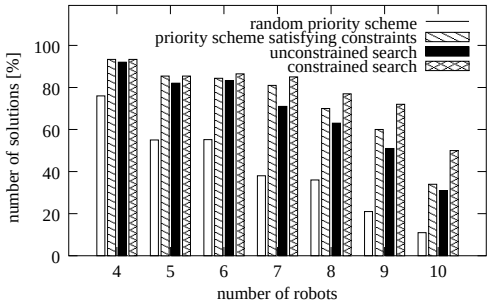

Figure 8: Solved planning problems for the four strategies using the path coordination method in the noncyclic environment.

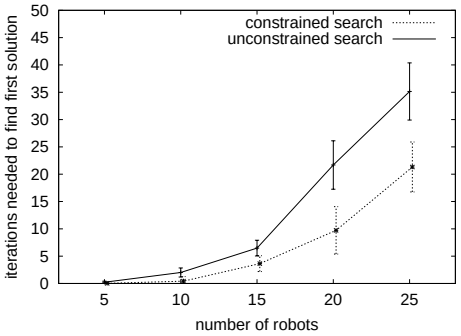

Figure 9: Iteration in which the first solution was found if the planning problem could be solved for the cyclic corridor environment.

4.2.2 Speed-up Obtained by Exploiting the Constraints

The second set of experiments was performed to investigate the ability or our approach to guide the search in the space of all priority schemes. We were especially interested in the question how much the computation time necessary to find a solution can be reduced by constraining the search. During these 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 9 plots the results obtained for different number of robots in the cyclic corridor environment and Figure 10 shows the same evaluation for the noncyclic environment. As can be seen, the unconstrained search needs significantly more iterations to generate a solution for both environments. Thus, 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 have to be generated in each iteration during the search.

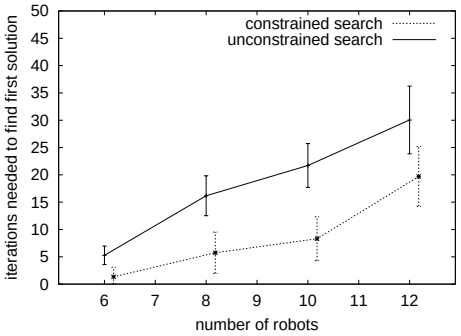

Figure 10: Iterations needed to find first solution in the noncyclic corridor environment.

4.2.3 Influence on the Overall Path Length

The next experiments were carried out 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 $\mathtt{k}$ 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 11. This plot contains the average move costs for three different strategies at each iteration. The first data set was obtained for the constrained search which corresponds to $\mathtt{k} = \infty$. 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 $\mathtt{k} = 0$. In this case our algorithm chooses arbitrary priority schemes

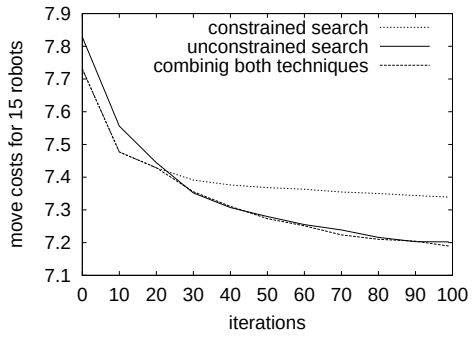

Figure 11: Summed move costs plotted over time averaged over 100 planning problems for 15 robots in the cyclic environment.

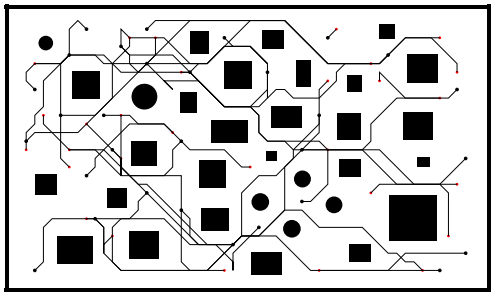

Figure 12: Resulting paths after priority optimization for a team of 30 robots.

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 $\mathtt{k} = 20$.

Since the constrained search 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.

4.2.4 Planning Paths for Large Teams of Robots

The final experiment in this paper is designed to illustrate that our system can be used to solve planning problems for even large numbers of robots. Figure 12 shows the paths planned for a team of 30 robots in an unstructured environment. In this particular example our system was able to generate a first solution in less than one second. The paths shown in the figure are the best solution found after 10 restarts with 10 iterations in each round. The paths are 20% shorter than the first solution found.

5 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 task specification.

The approach has been implemented and tested on real robots. One experiment carried out with the CS-Freiburg RoboCup team demonstrated that our approach can effectively be used for a team of mobile 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. 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 opti-

mization 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.

6 Acknowledgments

We would like to thank Thilo Weigel for his efforts in carrying out the experiments with the CS-Freiburg RoboCup team robots.

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] M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica*, 2:477–521, 1987.

[7] C. Ferrari, E. Pagello, J. Ota, and T. Arai. Multirobot motion coordination in space and time. *Robotics and Autonomous Systems*, 25:219–229, 1998.

[8] F. Gravot and R. Alami. An extension of the plan-merging paradigm for multi-robot coordination. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2001.

[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*, 12(4):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] M. McHenry. *Slice-Based Path Planning*. PhD thesis, University of Southern California, 1998.

[16] 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.

[17] N. J. Nilsson. *Principles of Artificial Intelligence*. Springer Publisher, Berlin, New York, 1982.

[18] 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.

[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.