An experimental study of distributed robot coordination

Stefano Carpin[1] Enrico Pagello[2]

[1] *School of Engineering*
University of California, Merced – USA
[2] *Department of Information Engineering*
University of Padova – Italy

Abstract

Coordinating the path of multiple robots along assigned paths is a computationally hard problem with great potential for applications. We here provide a detailed experimental study of a randomized algorithm for scheduling priorities we have developed, also comparing it with exact and approximated solutions. It turns out that for problems of reasonable size our algorithm exhibits an appealing compromise between speed and quality.

Key words: Multi-robot systems, intelligent mobility, randomized algorithms, motion planning

1 Introduction

Planning the motion of multiple robot systems has been a task investigated since the early days of mobile robotics. While the problem is interesting in itself, because of the inherent computational complexity it exhibits, it has to be acknowledged that few applications have been presented up to now in the context of autonomous mobile robots. This is partly due to the fact that systems with a remarkable number of robots have not been deployed yet. Moreover, when multiple robots are to operate in a shared unstructured environment, one of the holy grails of multi-robot research, their motion is commonly governed by reactive navigation modules rather than by precisely planned paths. That said, it is not implied that the problem in itself lost interest from the applicative point of view. The demand for systems capable of governing the motion of multiple objects in shared environments is instead ever increasing. Applications include for example luggage handling systems

at airports, storage systems in factories, moving containers in harbors, and more. The theme of *intelligent mobility* is envisioned to play a major role in the foreseeable future. Possibly, one of the major differences that will be seen will be a decrease of individual robots' motion freedom. Sticking to the storage systems in factories example, mobile carts are not free to wander wherever they want, but are rather constrained to proceed along predefined paths, usually hard wired in hardware. Given a set of predefined paths, coordinating the motion of vehicles along these routes will be asked more and more often. And, of course, it will be asked to find solutions that optimize certain performance indices, like for example, consumed energy, time needed to complete the task, and the alike. From a computational point of view these problems have been studied since quite some time, and their inherent complexity has soon been detected. Approximated and heuristic based solutions are therefore a must when one is required to deal with multiple moving objects. In this paper we offer an experimental assessment of a simple randomized approach to solve the coordination problem we have proposed in the past. Section 2 discusses related literature, while the problem is formalized in section 3. The solving algorithm is illustrated in section 4, and the experimental framework and results are shown in section 5. Conclusions are finally provided in section 6.

2 Related work

The problem of multi-robot motion planning has been continuously studied in the past. The first major distinction concerns *centralized* versus *decentralized* approaches. When a centralized motion planner is used, one process has to plan the motion for the whole set of robots. The obvious drawback is in the high dimensionality of the composite configuration space to be searched. In a decentralized approach every robot plans its own motions, and then has to undergo a stage of negotiations to solve possible collisions with other robots. Decentralized approaches are inherently incomplete, but much faster. Sánchez and Latombe [1] speculated that decentralized approaches are likely to show their incompleteness often when used in industrial production plants. In the case of mobile robot systems, however, the environment is likely to be less cluttered and hence these problems are less likely to occur. Efficient methods to solve the single robot motion planning are available [2][3][4] and will not be further discussed here (recent books on motion planning like [5] and [6] provide extensive and up to date coverage of the topic). A common approach to solve the multi-robot motion planning problem consists in assigning priorities to robots and planning their motion according to them [7]. Paths for robots are computed one after the other, according to their priority. When planning the motion of a robot with priority p_j, it is necessary to take into consideration the already planned motions for robots with priority p_i, where

$p_i < p_j$. Finding a good priority schema is a hard problem in itself [8]. A related problem that we address in this paper consists in coordinating the path of a set of robots along given specified paths. As the paths may intersect with each other, it might be necessary to stop certain robots when approaching an intersection point in order to give way to other robots and avoid collisions. In this context one is generally interested to minimize certain parameters, like, for example, the time needed by all robots to reach their final destination. This rules out certain trivial coordination schemas, like for example the one where just one robot moves and all the others remain stationary, since the overall time would be too high. LaValle and Hutchinson solved the problem using a game-theoretic approach based on multi-objective optimization [9][10]. The approach allows to tune the algorithm behavior between centralized planning and complete decentralized planning. The authors show that optimal results can be found, but a significant amount of time is needed. Simèon et al. [11] solved the path coordination problem using a resolution complete algorithm. They show how it is possible to split a given path in segments where the robot will not collide with any other robot, and segments where paths intersect. The authors illustrate results involving up to 150 robots, but where no more than 10 robots are intersecting each other's path. Computation time is in the order of minutes. Akella and Hutchinson [12] solve the problem of robot coordination along predefined routes by simply varying the start time of each robot. Once a robot starts to move, it never stops until it reaches its target position. Peng and Akella recently extended these ideas addressing the case of robots with kinodynamic constraints [13].

3 Problem formulation

The problem we aim to study is the following: given n robots, assume that n paths, one for each robot, are provided. We suppose that each path has been subdivided into *free* and *occupied* segments. Given a path, an occupied segment is a part of the path such that the robot can collide with other robots while it is moving along that part of the path. Occupied segments arise when different paths intersect each other, or are very close (see figure 1 for an example). Any segment that is not occupied is declared free. In light of results presented in [11] and [13], free and occupied segments can be efficiently determined. Hence, a path p_i can be seen as a sequence $p_i^1 \dots p_i^{s(i)}$, where each p_i^j is either a free or an occupied segment, and $s(i)$ is the number of segments composing path p_i. The task is to find a coordination schema, i.e. a mapping:

$$C : [0, T] \rightarrow \{1 \dots s(1)\} \times \{1 \dots s(2)\} \times \dots \times \{1 \dots s(n)\}. \quad (1)$$

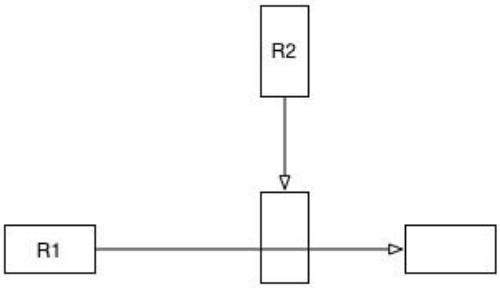

Fig. 1. The simplest case of robot coordination, involving two robots only. R1's path can be divided into three segments, free, occupied and free respectively. R2's segment can be divided into two segments. If R2 is given a priority higher than R1, and both robots travel at the same speed, R1 will not be able to reach its final destination, because of R2 stopping on its path.

such that for each time $0 \leq t \leq T$ no two or more robots are moving along path segments that collide with each other. In the beginning, robot i is positioned at segment p_i^1, and in the end it has to reach $p_i^{s(i)}$. While moving through the different segments, a certain amount of time will be spent to traverse each of them. Let $t(p_i^j)$ be the time spent by robot i to traverse segment p_i^j $(1 \leq j \leq s(i))$. Throughout the paper we assume that robots only move forward along their path, though they can stop at certain points to give way to other robots. However, they never backtrack along their route. The goal is to find a coordination schema that minimizes the time needed by all robots to complete their motion. Formally we aim to minimize the following quantity

$$z = \max_{1 \leq i \leq n} \sum_{j=1}^{s(i)} t(p_i^j). \tag{2}$$

It can be shown that this problem is equivalent to the Job Shop Scheduling (JSS) Problem, which is known to be NP-hard [14]. The JSS problem asks how to schedule n jobs that have to be processed through m machines in a way that the overall required time is minimized. The constraints are that no machine can process more than one job at the same time, and that each job has to be processed by the machines in a given order. In the path coordination problem, each robot is a job, and each free or occupied segment is a machine. The reader should note that while reducing the robot motion planning coordination problem to an instance of the JSS, not every job needs to be processed by every machine (i.e. not every robot has to travel through all the possible segments). Under the assumption that $P \neq NP$, the search for a coordination schema that minimizes the time needed to complete the motion task is doomed to take exponential time. This motivates the great number of approximated and heuristic approaches that have been proposed along the years.

4 Random rearrangements

In our former work [15] we proposed a simple distributed schema to solve the multi-robot motion planning problem. Similar ideas were later used in [16]. The idea is slightly modified here to describe how the various robots can operate to find a valid coordination schema, and is depicted in algorithm 1. The algorithm assumes that a data structure *SpaceTime* is available. *SpaceTime* records which part of the space is occupied or free at a given time. The SpaceTime data structure can be accessed by providing two indices, one for the space and one for the time. The algorithm picks a random priority schema (line 1), and then schedules the robot motions according to the selected priority schema. The first considered robot will be scheduled to move straight along its path with no stops, and SpaceTime will be accordingly updated. When scheduling successive robot motions, it is necessary to check whether the robot can move to its next path segment or if it is already occupied (line 6). If this is possible, the robot moves to its next segment (line 7), or a delay is inserted (line 11). In both cases SpaceTime is updated to record robots' position (line 12) while time evolves (lines 8 and 11).

Algorithm 1 Coordination of n robots

```
 1: pick a random permutation of the set {1 ... n}
 2: for i ← 1 to n do
 3:    Time ← 0
 4:    j ← 1
 5:    while j ≤ s(i) do
 6:       if SpaceTime(Time,p_i^j) is free then
 7:          Advance robot i through the segment p_i^j
 8:          Time ← Time + t(p_i^j)
 9:          j ← j + 1
10:       else
11:          Time ← Time + Delay
12:       update SpaceTime
```

We here stick to the hypothesis formulated in [15], i.e. that each robot will apply this procedure to compute a coordination schema, and that in the end the one leading the best value for the variable z formerly defined will be used to execute the real motion. So, when n robots are involved, n independently chosen random priority schemas are tried.

The coordination schema does not compute any path and does not alter any of the paths it starts with. Hence there are cases when certain priority schemas do not lead to any solution. For example, if robot i's goal position is on the path p_j assigned to robot j, and i has a higher priority than j, robot i may reach its goal position before j passes through the intersection. At that point robot j will not be able to reach its target position. This could be avoided if robots would be allowed to move backwards along their path, but

this extension is not considered here. From a practical point of view, it has to be mentioned that even a naive implementation of the above algorithm can solve a coordination task in a fraction of a second. The reader should not be misled by this statement. The algorithm just computes a random permutation and tries to schedule robots one after the other according to this schema. No paths are computed and no optimization is tried, hence the small time required to find a solution. The twist of our study is indeed to show that with such a simplified approach one still gets good results in practice.

5 Experimental results

The conceptual equivalence between the path coordination problem and the JSS problem has been outlined since long time, but to the best of our knowledge there have been no attempts to set up an experimental framework to compare the exact approach and heuristic solutions proposed along the years. While one can easily guess that solving the JSS problem will take long time, it is interesting to assess by how much the best solution found by an heuristic algorithm differs from the optimal one. With respect to the algorithm presented in section 4, it is also interesting to measure how likely it is to select a random priority schema that leads to a deadlock situation. We have then developed three experimental scenarios. The first one is based on the formerly described algorithm, while the second is based on a freely available implementation of Mixed Integer Linear Programming algorithms [17] and is used to get exact solutions to the associated JSS problem instances. In addition, we have also implemented the approximated algorithm described in [18]. The algorithm described by Shmoys et al. therein is the first approximation algorithm that gives a guaranteed approximation ratio with high probability. It is instructive to compare three algorithms that span the whole range of possibilities, namely an exact solution, an approximated solution that has a guaranteed bound with high probability, and a heuristic algorithm with no guaranteed performance.
The operating scenario has been simplified to a square grid, with robots starting and ending at random positions and moving along randomly generated paths. Generated paths are not straight, but rather present various turns and are of variable length. Figure 2 shows an extremely simple scenario with two robots moving on a 6×6 grid. The first two panels illustrate the path of the two robots. On the right, the two paths are superimposed and the grey cells show possible collisions to be resolved. In this example, however, the algorithm determines that if the robots move at full speed no collision occurs in space time and produces a schedule with no stops at all. It is worth observing that these hypothesis do not oversimplify the problem. We anticipated in section 2 that there exist efficient algorithms to both compute paths and decompose them into free and occupied segments. Here we are only interested in the suc-

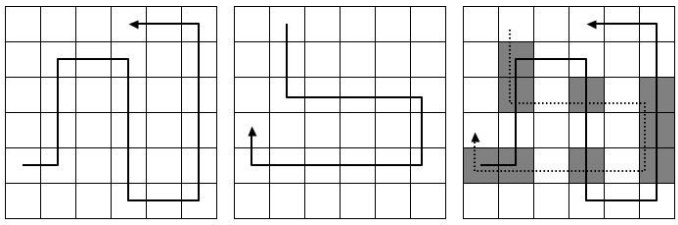

Fig. 2. A simple case of coordination: two robots moving in the same grid.

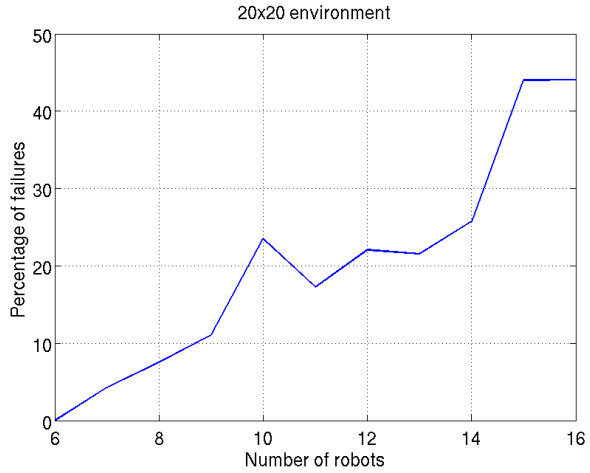

Fig. 3. Percentage of failures (y axis) versus number of robots (x axis) on a 20×20 grid.

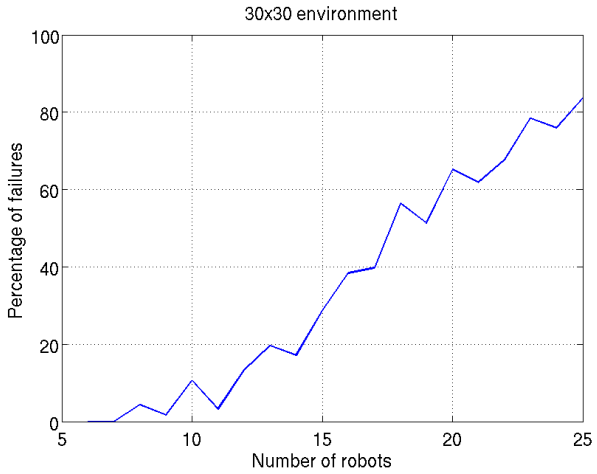

Fig. 4. Percentage of failures (y axis) versus number of robots (x axis) on a 30×30 grid

cessive step, i.e. the coordination along these provided paths. The first set of tests analyzes the possibility of not finding a valid priority schema. Figures 3, 4 and 5 illustrate the results. Fixing three different environment sizes, we have solved 20 randomly generated problems involving a varying number of robots, ranging from 6 to 35. The figures illustrate the percentage of generated priority schemas that are not valid. A sort of *saturation* effect can be observed, i.e.

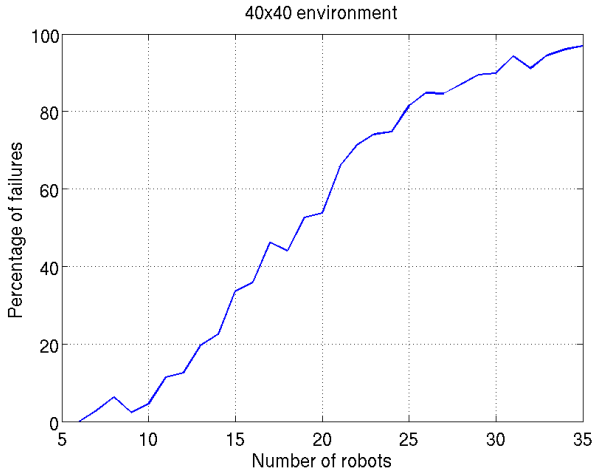

Fig. 5. Percentage of failures (y axis) versus number of robots (x axis) on a 40×40 grid

when the number of robots increases, so does the fraction of invalid priority schemas. The reader should however observe that this effect manifests itself when the number of robots is high and the environment is big. The bigger the environment, the longer the paths travelled by each robot and the number of collisions. Moreover, also the chance that a robot will terminate its run on some other path is higher. This was decided on purpose, to test the algorithm under extreme conditions. It is also worth noting that when trying to coordinate n robots the algorithm always tries only n of the possible $n!$ priority schemas. In the most extreme case with 35 robots, only 35 out of the 35! possible priority schemas were tried. This is clearly a negligible fraction that could be easily increased in order to give the algorithm more chances to find valid schedules. Next, we have compared the performance of our algorithm with the algorithm illustrated in [18]. In particular, we have compared the length of the schedules produced by the two algorithms. Results are provided in figures 6, 7 and 8. The figures show the ratio between the schedule length produced by ours and Shmoys' algorithms. For a fixed number of robots multiple comparisons are displayed because multiple randomly generated test cases are solved and compared. In all considered cases the ratio is below 500%, i.e. solutions produced by our algorithm are at most 5 times longer. Interestingly, the trend shows that this gap reduces when the problem becomes more challenging, i.e. when the number of robots increases. For the most hard problems the solution found by our algorithm is usually twice as long.

The last point would be comparing the solution found with the proposed algorithm with the one found by the exact JSS solution. Doing this comparison is however computationally hard. Solving a single instance of the coordination problem with environments bigger than 20×20 or more than 10 robots takes a remarkable number of hours of intensive computation. For the scenario involving a 20×20 grid and a number of robots ranging from 6 to 9, no significant

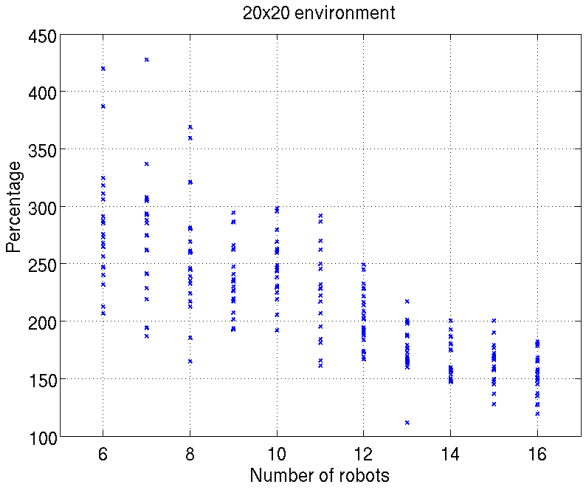

Fig. 6. Relative performance on a 20×20 grid

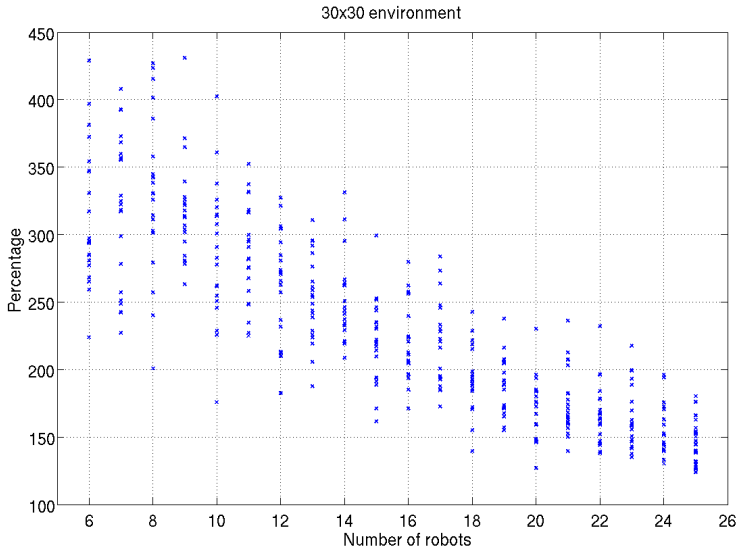

Fig. 7. Relative performance on a 30×30 grid

differences were observed. In 97% of the cases the produced schedules had the same length. When differences were noticed, the schedules differed by 1 unit of time (versus a schedule length always higher than 50 time units, i.e. the observed mismatch was less than 2%).

6 Conclusions

In this paper we have experimentally analyzed a randomized algorithm for the coordination of multiple robots along predetermined paths. The algorithm selects some random priority schemas and then tries to schedule the robots' motions according to them, inserting suitable delays when collisions in space

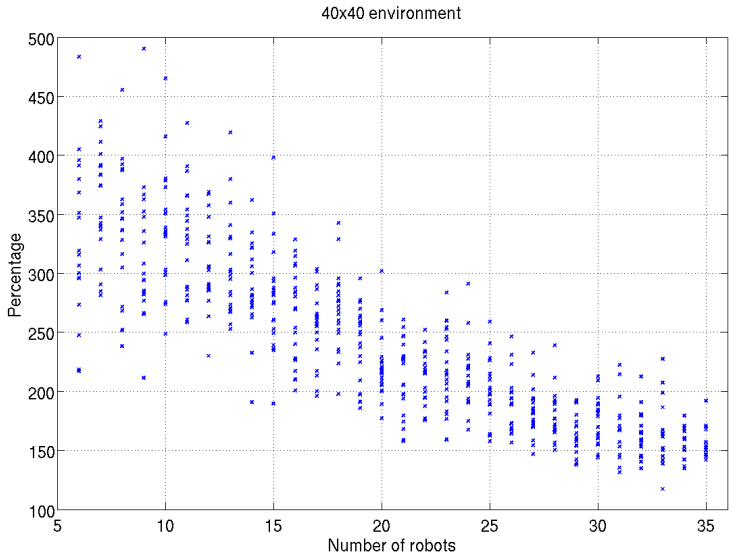

Fig. 8. Relative performance a 40×40 grid

and time are detected. The simple randomized procedure for finding priority schemas appears to be competitive under rather general conditions. Most of the benchmark problems can be solved using only a very limited number of priority rearrangements. This result is intriguing, if one considers the number of priority schemas tried versus the number of possibilities. We have also observed that the quality of the solution, intended as the time to bring all the robots to their final destination, is often not far from the optimal one. For the proposed algorithm we cannot provide analytical bounds specifying by how much the determined solution differs from the optimal one. However, experimental comparisons with an algorithm well characterized from the thoeretical point of view show that our algorithm compares favorably. The proposed approach shows that in many practical situations, suboptimal approaches can be quickly found and are still satisfactory, whereas the need of guaranteed optimal solutions offers very often an unbalanced ratio between quality increase and required time.

References

[1] G. Sánchez, J. Latombe, Using a prm planner to compare centralized and decoupled planning for multi-robot systems, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2002.

[2] S. Carpin, G. Pillonetto, Motion planning using adaptive random walks, IEEE Transactions on Robotics 21 (1) (2005) 129–136.

[3] L. Kavraki, P. Švestka, J. Latombe, M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Transactions on

Robotics and Automation 12 (4) (1996) 566–580.

[4] S. LaValle, J. Kufner, Randomized kinodynamic planning, International Journal of Robotics Research 20 (5) (2001) 378–400.

[5] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, Principles of robot motion, MIT Press, 2005.

[6] S. LaValle, Planning algorithms, Cambridge academic press, 2006.

[7] M. Erdman, T. Lozano-Pérez, On multiple moving objects, Algorithmica 2 (1) (1987) 477–521.

[8] M. Bennewitz, W. Burgard, S. Thrun, Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots, Robotics and Autonomous Systems 41 (2-3) (2002) 89–99.

[9] S. LaValle, Robot motion planning: A game-theoretic approach, Algorithmica 26 (3-4) (2000) 430–465.

[10] S. LaValle, S. Hutchinson, Optimal motion planning for multiple robots having independent goals, IEEE Transactions on Robotics and Automation 14 (6) (1998) 912–925.

[11] T. Siméon, S. Leroy, J. Laumond, Path coordination for multiple mobile robots: A resolution-complete algorithm, IEEE Transactions on Robotics and Automation 18 (1) (2002) 42–49.

[12] S. Akella, S. Hutchinson, Coordinating the motions of multiple robots with specified trajectories, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2002, pp. 624–632.

[13] J. Peng, S. Akella, Coordinating multple robots with kinodynamic constraints along specified paths, International journal of robotics research 24 (4) (2005) 295–310.

[14] M. Garey, D. Johnson, Computers and Intractability. A guide to the theory of NP-Completeness, W.H. Freeman and Company, 1979.

[15] S. Carpin, E. Pagello, A distributed algorithm for multi-robot motion planning, in: Proceedings of the fourth Eurpoean Conference on Advanced Mobile Robots, Lund (Sweden), 2001, pp. 207–214.

[16] Y. Guo, L. Parker, A distributed and optimal motion planning approach for multiple mobile robots, in: Proceedings of the IEEE Conference on Robotics and Automation, 2002, pp. 2612–2619.

[17] G. project, The gnu linear programming toolkit, www.gnu.org/software/glpk/glpk.html (2005).

[18] D. Shmoys, C. Stein, J. Wein, Improved approximation algorithns for shop scheduling problems, SIAM Journal of Computing 23 (3) (1994) 617–632.