ELSEVIER

Robotics and Autonomous Systems 25 (1998) 219–229

Robotics and Autonomous Systems

Multirobot motion coordination in space and time

Carlo Ferrari [a,*], Enrico Pagello [a,b,1], Jun Ota [c,2], Tamio Arai [c,3]

[a] *Department of Electronics and Informatics, The University of Padua, Via Gradenigo 6a, I-35131 Padova, Italy*

[b] *Institute of System Sciences and Bioengineering – LADSEB-CNR, Corso Stati Uniti 4, I-35127 Padova, Italy*

[c] *Department of Precision Machinery Engineering, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan*

Abstract

This paper describes a solution to the multirobot motion planning problem based on a decoupled analysis in the space domain and in the time domain. It investigates the practical use of the notion of motion plan quality and of the motion plan robustness measures for computing safe motions. The use of anytime algorithms allows one to evaluate the opportunity of looking for alternative solution paths by generating small variations of robot motions affecting both its geometrical path and its scheduled velocity. By using the concept of plan robustness, several alternative paths are generated and evaluated through various *performance indices and impact factors*, using heuristic rules. These indices allow one to know how much a variation affects a given plan. Finally, some recent experiments are outlined.

Keywords: Multirobot motion planning; Multirobot coordination; Practical planning method

1. Introduction

The problem of coordinating the motion of several robots moving in the same cluttered environment is becoming more and more interesting as long as it is reasonably possible to forecast a large use of multirobot systems both in industrial and civil environment. Unfortunately well-established planning method dealing with a single robot cannot be used when there are many different autonomous vehicles that share a common environment, because of the inherent intractability characteristic of the problem. Moreover, it is clear that planning cannot be avoided using pure reactive schemata, that rely only on on-line computation of sensor data. The reactive approach does not guarantee convergence. In fact, the local nature of sensing can easily trap robots in deadlock situations or it can delay to infinity their goal satisfaction. Hence, multirobot systems still require to study off-line planning methods, that can be used both for generating a first basic plan, and for updating that plan when some major exceptions occur at run-time. However, practical real-world situations call for a practical planning method that can give a reasonable solution in a reasonably small amount of time. As pointed out in [2], the time for reaching an admissible solution is a variable that should play a role in the planning algorithms. The general layout of a practical planning algorithm for the multirobot motion planning problem can be devised as an iterative refinement process that starts from an easy-to-find suboptimal solution. A solution consists of a path for each robot and a collision-free velocity

* Corresponding author. Tel.: +39-049-827-7729; fax: +39-049-827-7699; e-mail: carlo@dei.unipd.it

[1] E-mail: pagello@ladseb.pd.cnr.it

[2] E-mail: ota@prince.pe.u-tokyo.ac.jp

[3] E-mail: arai@prince.pe.u-tokyo.ac.jp

0921-8890/98/$ – see front matter
PII: S0921-8890(98)00051-7

profile for the robot running along its path. This very first solution can be refined by introducing some local and possibly small variations, in order to improve the overall quality of the solution itself. The refinement process can continue until it is reasonable to guess an improvement that can pay the time spent in finding the improvement itself. When there are no significative improvements over time, the algorithm stops, and the last solution with the best quality is picked up. This approach needs to investigate the notion of *solution quality*, i.e., the notion of a method to weigh how far a solution is from the optimal, but maybe unknown, solution.

This paper is devoted to present the main results and implementation related to a multirobot motion planner. The planner can take into account various kind of *motion variations* in order to choose a plan which is robust to those variations. We will present how to obtain better solutions from an original good one by *looking for alternative paths* with respect to some *quality parameters*. Heuristic rules for evaluating plan robustness, according to a proper definition of robustness [6], will form the core structure of the planner. In our view, this should lead to a better general performance for the robot motion planner when several tens, or even hundreds of robot are involved.

The research work on motion planning for multiple mobile robots (i.e. multirobots) finds its foundation on a few papers that appeared in the early 1980s, among which, one of the most important, is [10], where it is showed that even the 2D problem of moving arbitrarily many rectangles, in a rectangular region, is PSPACE-hard. Significant improvements in this field were later done in [4], where a configuration space–time was used to represent the time-varying constraints imposed by the other moving and stationary objects, and in [15], where an algorithm, based on a global cell decomposition approach was presented. Another important contribution was the use of algorithms with priority [3,4]. In [9], the case when the environment contains obstacles whose existing periods are dependent on time, was considered, allowing one to model a variety of time-varying situations that can arise in application domains. The previous list, of course, is not exhaustive, since many other interesting approaches would deserve to be quoted (see for instance [12]).

A common view in evaluating a multirobot motion planning system is to classify the planner as a centralised planning system, or a decentralised one [1]. In *centralised planning*, all decisions are taken by a single decision maker. We definitively took this approach, by relying on the application of some suitable performance indices.

In engineering applications, it is often useful to introduce some effective heuristic in order to develop a practical solution to the problem. Therefore, we have designed simple, but reliable, heuristics based on *performance indices* [5]. Performance indices measure the quality of some path attribute or the quality of the robot behaviour while moving. They can be combined and used in the decision process figuring out some global property measure.

In [6], the concept of *motion plan robustness* has been introduced. The basic idea is that a motion plan is robust if it can be used in spite of small variations in the motion context. Motion plan robustness is particularly useful when examining environments filled up with many robots, because a small variation in the execution plan of one of the robots may reflect in large variations to the other robots' execution plans, bringing, to the necessity of re-building the complete plan for all robots.

Studying motion plan robustness has several advantages. First, it becomes possible to set a proper library of prototype plans that can be slightly modified to cope with the classes of similar applications. Then, some general heuristic rules, dealing with the most common problems, can be extracted and used to improve the goodness of proposed plans. Finally, it is possible to approach the problem of merging off-line and on-line methods by locally replanning portions of the solution paths. Any learning method can make a full use of the notion of plan quality and plan robustness both at the planning level and at the execution and monitoring level. In fact the availability of a library of plans, can help in the process of planning, while performance indices can be used and updated while the robots are moving and acquiring more information about the environment and the object in it.

In [2], the idea of *flexible computation* was used for the robot-tour problem. Once assumed that a robot starts out with some initially selected tour, a planner can figure out how much time to devote to tour improvement in order to minimise the expected time spent in stationary deliberation and in combined deliberation and path traversal. In [8], we proposed to

use anytime algorithms in the context of multirobot motion planning. Solutions are labelled with a quality measure that takes into account both the solution performance indices and the solution robustness with respect to small variations.

The paper follows this track. Section 2 presents how to approach the multirobot motion planning problem by decoupling the space analysis and the time analysis. Moreover, it is devoted to summarise the performance indices and robustness measures used by the planner. Section 3 details how to use the deliberative approach for selecting a good plan, while Section 4 describes some experimental results. Final comments and future work are in the conclusions.

2. Multirobot motion planning problem solved in space and time

The generic solution of the multirobot motion planning problem is a proper set of motion commands that determine both the position and the motion direction, for all the robots in the environment under exam. The motion command for each robot can be generated as a result of a proper choice of a geometrical path and of a velocity profile along that path. Synchronisation among all the moving objects can be obtained by a proper tuning of their velocity. Hence a velocity schedule must be determined for each robot, in order to avoid collisions with both the other robots and the moving obstacles. It should be clear that solving the multirobot motion planning problem involves some sort of reasoning in the space domain as well as in the time domain.

The approach we followed keeps the two domains apart. More specifically, the computation of the velocity profiles will follow the choice of safe geometrical paths. The advantage of this approach is that it avoids to consider space and time in a unique extended iper-space. This space cannot be considered as a 4-Euclidean space because the axis representing time has particular characteristics, that model the idea that it is not possible to go back to the past. On the other hand, decoupling the analysis in the space domain and in the time domain make less evident that there exists a correlation between the choice of a path and the computation of a velocity schedule; i.e., a simple and short path can have more intersection points with other paths, than a longer one. Increasing the number of intersection points can increase the request of synchronisations between robots, and it can result in a less efficient plan. We approached this problem by introducing some quality measures applied both to each path and to the overall plan and using these quality measures in all the planning phases. Moreover, in order to reduce the backtracking request between the analysis in the space domain and the analysis in the time domain, we did not compute a single geometrical path for each robot. Instead each robot was assigned to some different path. We do not consider all the infinitely many feasible geometrical paths because it is possible to group many of them with respect to some invariant measures with respect to the velocity schedule characteristic. Hence many different alternative paths, that form a family of paths, were assigned to a robot to have several significative choices when searching for a good solution of the motion problem.

Given r robots and k different paths toward the goal for each robot, there are k^r possible solutions to the *path scheduling problem*, because for each robot you can choose one of its k different paths. What is the best choice among all the paths? What does differentiate all the possible solutions? In order to answer these questions it is necessary to evaluate how good a solution is, i.e., it is necessary to introduce a method for evaluating the *quality* of the solution. A good solution is a good set of paths and of velocity schedules according to some quality parameters. Note that the solution quality must be a property of the whole solution, and it depends on all the paths and on all the velocity schedules in the motion plan.

It is straightforward that the algorithm for finding the optimal solution by enumerating all the possible alternatives, has an exponential complexity. We were interested in a search for a nonoptimal solution that could be done in polynomial time. The approach we followed to obtain a suboptimal solution can be sketched as follows: a new problem can be set, where r' new robots, one for each possible path, are taken into account ($r' = kr$); i.e., any robot of the original problem is substituted by a family of k robots each running along one of the different paths connecting the starting position and the goal position. In this way, a *suboptimal solution* can be achieved by solving the synchronisation problem for all r' robots at the same time (ignoring intra-family

collisions) and then choosing a single robot from each family.

The search of a good solution is based on an anytime path planner that build paths bringing to better solutions according to some heuristic rule based on the quality indices. Then a path for each robot is chosen to improve solution's robustness. The process can be continued until a deliberative scheduler says to stop.

A particular aspect of the solution quality is its *robustness*. We studied the concept of robustness applied to motion plans and we defined a set of robustness indices to evaluate the robustness of a plan.

In finding a solution there could be some degree of freedom in choosing or modifying some local characteristics. These small changes could make an improvement locally, without affecting the quality of the solution as a whole. Hence robustness indices can be used to analyse the effect of local changes. We used these robustness indices not only to evaluate the goodness of a solution, but even to guide the search for new and better solutions, following the deliberative scheduling approach.

2.1. The path planning and the velocity planning problems

Even in a crowded multirobot environment, each single robot has to face at least two basic problems, namely, avoiding collisions with fixed obstacles and with moving obstacles. In the literature there are quite well-established techniques for modelling and solving these two problems. We found particularly interesting the approach proposed originally in [11] for a single point robot. That method was based on the separation of the space analysis from the time analysis. Hence we firstly designed a planner for a single robot [14] where the trajectory planning problem was divided into the two subproblems of planning a path to avoid collision with static obstacles, (the path planning problem or PPP), and of planning the velocity along the path to avoid collisions with moving obstacles, (the velocity planning problem or VPP). The pure geometric nature of PPP allowed us to design and use fast and original algorithms for collision detection resulting in a very efficient planner [13]. In the same spirit we generalised to the multirobot case that motion planner, approaching the problem of multirobot motion planning by separating the space analysis from the time analysis [5].

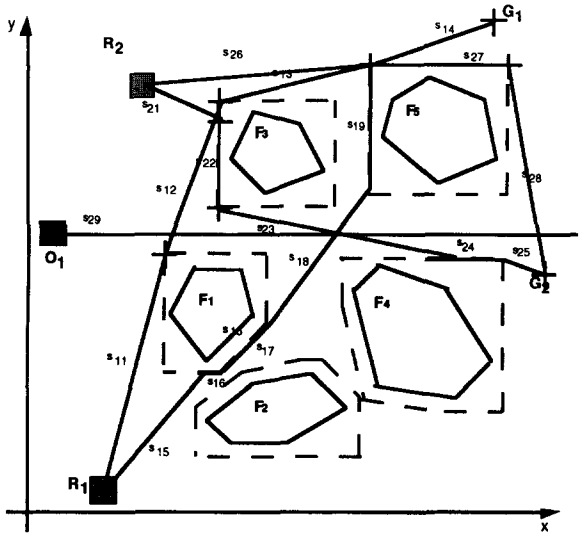


Fig. 1. The multirobot motion planning problem.

Let us look at PPP first (see Fig. 1). We solved PPP by using a rough representation of fixed obstacles in the C-space, based on the idea of enclosing the C-obstacles in boxes, that are the closest rectangular approximation of the C-obstacles. These boxes were locally refined (when needed) using our fast collision checking algorithms.

The PPP is solved firstly, i.e., some free paths for each robot, connecting its start position with its goal position among the fixed obstacles, are computed. The output of the PPP phase is a family of paths related to each robot in the environment. Those paths are collision-free with respect to the fixed obstacles.

Two robots will collide if some segments of their paths intersect and they are running on them in the same time period. Hence in the multirobot motion planning algorithm we need to consider a collision checking step to determine if two robots running on intersecting segments will collide. Furthermore, we also need a policy for solving collisions, i.e., there should be a delaying step to decide which is the robot to stop (or delay) and to recompute the basic time scheduling for each robot.

As the PPP output is a set of paths for each robot the VPP is approached by considering all those paths simultaneously, and computing the schedule for each

robot running on all of its path. At the end of the VPP phase the plan is built by picking up the best path for each robot.

Then, we analysed VPP, looking first for those path segments that are in use in the same time interval, i.e., making a temporal collision analysis, and checking later for spatial intersections between the pair of segments in use, i.e., making a spatial collision analysis. The basic point here is that the time analysis is performed first. Only if two path segments temporally overlap, the system checks for spatial path segments intersections. If a collision occurs, one of the two robots is stopped until the other has cleared the path, or it can be applied a shape changing algorithm [8] to locally modify the geometric structure of the intersection point. We should note that the intra-family robot collisions are ignored.

With this method we can avoid a combinatorial explosion of the solution space, but we were not assured to find out the optimal solution.

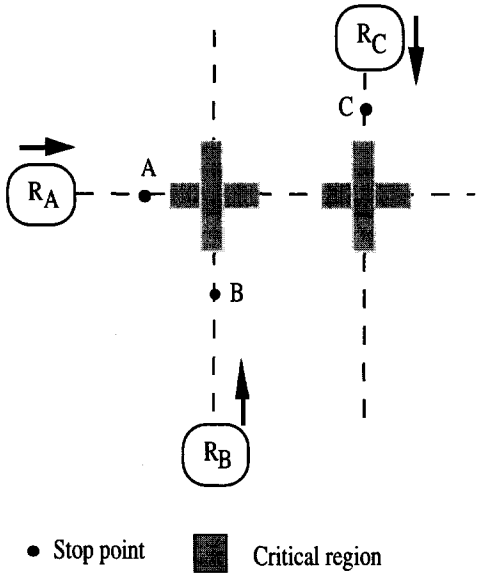


Fig. 2. VE evaluation for robot path.

2.2. Performance indices

We associate to a path three performance indices. The first one is the *running time* RT, i.e., the minimum time a robot should need for reaching the goal using that path. The second one is the *motion error* ME, that measures how a robot can move away from its path without colliding with obstacles or other robots. The third performance index is the *velocity error* VE, that measures how much a robot can vary its velocity schedule without worsening the global plan.

Performance indices can be used to choose which robot has to be delayed and which is the path to be considered for each robot. Any time some robot is stopped or some path is modified they are recomputed.

RT is the minimum time a robot needs for reaching the goal using its assigned path. This index is proportional to the maximum speed of the robot. The RT index can be used to choose which robot has to be delayed and which is the path to be considered for each robot. Each time a robot is stopped this index is recomputed. Note that RT is not a property of a path, but a property of the solution: it depends not only on the single path but on the set of paths that compose the chosen solution, and on the velocity schedule of all robots in the plan. In fact, each robot may interact with all the others, so that a variation in the motion of a robot can affect all the other robots in the system.

ME measures how a robot can move away from its path without any risk of collision, i.e., without colliding with fixed obstacles. The ME index is computed during the PPP phase and it is proportional to the distance of the path from the obstacles. A locally optimal path as regards ME can be constructed using Voronoi graphs (whose edges have the property of being equally distant from each obstacle). However, this local optimality does not solve the problem of ME optimisation. In fact, ME can be computed as the minimum of path's distance from obstacle or as a mean distance. In both cases it is possible to find different paths, for each robot, with different ME. So, even if locally ME is optimal for a single stretch of path computed by Voronoi graphs, the value of ME for the whole path depends on the particular path chosen.

VE measures how a robot can vary its scheduled velocity without causing any collision with other robots. See for example Fig. 2: if robot R_A is scheduled to stop at point A for a delaying time interval Δ (waiting for robot R_B to pass), robot R_B cannot arrive later than scheduled, without any risk to collide with R_A. Hence robot R_B has a value of VE equal to zero. Instead, robot R_A can arrive later than scheduled, without introducing any collision with R_B. Moreover, if R_C is

scheduled to stop at point C for a delaying time interval Δ (waiting for robot R_A to pass), robot R_A should not arrive at its stop point A later than the time scheduled for its re-start, otherwise this delay could affect the following synchronisation with R_C. The value of VE for R_A has a finite value different from zero. The value of VE for R_C, in this example, is positive infinite, because R_C is not involved in any further intersection.

Note that once again VE is not a property of the single path, but a property of the whole solution. A way to search a solution with a good VE is to look for different paths for each robot with different RT. As we have already seen, the idea is to take into account different choices for each robot, building a path family for each one, planning the motion for all the possible paths and then choosing the best for each robot.

2.3. Robustness with respect to environmental model

In our geometric representation, C-space obstacles are approximated by bounding boxes, that may have, or not, a large tolerance. It is possible to refine boxes around some robot path to estimate segment path safety. If the path is not safe enough, then it will be assigned a low ME index value. Then, there exists a *path substitution problem*: if a path p is not any more feasible, then we need to look for a new path q.

Consider an alternate path q for a robot r already running on a path p. Call parent plan the plan that has been already computed without considering q. It is very time expensive to evaluate RT for the new solution obtained by substituting path p with path q, because the whole solution should be recomputed. Thus, we evaluate a *collision impact factor* (CIF) for q, where CIF is the ratio between the number of robots whose paths cross q and the total robot number. A CIF almost equal to 1 suggests to discard the new path, because there are too many candidate collisions.

In our experimentation, we found that a CIF so defined is not a good parameter for deciding to discard a path or not, because the kind of geometrical intersection is much more interesting than the bare number of intersections. For this reason, it is better to estimate the length of the piece of path inside the critical collision area. So, it is possible to define the CAF quality measure as the ratio between the length of path inside critical collision area and the total path length. CAF has demonstrated to be a good parameter to decide if to discard a path or not both when we try to optimise RT and when we try to optimise VE.

Each proposed plan has a quality measure associated to it that is computed using the performance indices of the paths related to that plan. If the emphasis is on the "time to reach the goal", then the quality of the plan is given by the highest value of the RT index associated to the plan. If the quality must take into account security issues, then the quality measures can be computed by a weighted average of the RT, ME and VE indices of the paths. The new plan's quality index is also computed using the quality indices of the new path(s). Note that collision impact factors take part in the choice of paths that will constitute the solution to the motion plan problem. Such parameters are very useful because, as we said, RT and VE depends on the whole solution, and not on the single path, and their computation is not always simple, so we need a simple coefficient to help us choose a path or another. CIF and CAF give an evaluation criterion for choosing which paths will be used in the new motion plan.

3. Applying deliberative scheduling to families of varying paths

The quality criteria described previously play a major role in the procedure for finding a suboptimal solution to the multirobot motion planning problem. It is useful to point out that a complete search algorithm in the number of robots is not interesting from a practical point of view due to its exponential complexity. The planner uses the quality indices in anytime algorithms by applying the deliberative scheduling approach [17]. Within this approach the time for building a good solution together with the solution quality measures are the control variables of the planning algorithms. Moreover, this approach calls for iterative refinement of the candidate solutions, that adds more and possibly better solution to the initial ones. The process of growing path families is controlled by a *monitor*, which continuously evaluates the robustness indices for the solution paths found by the path planner and measures the time passing by. The monitor decides continuing or stopping the elaboration of the path planner, depending on the values evaluated for the solution paths.

The space of the possible paths for each robot with the possible velocity schedules along each path is the *search space*, that can be increasingly built starting from an initial and easy-to-find solution, and then adding to it several alternative choices, i.e., many different paths or velocity schedules. It is necessary to point out that we need some criterion to generate those alternative choices, and that their generation should be incremental, in order to provide the search algorithm with a growing space to be investigated, so that it may ideally continue its search gradually coming nearer to the optimal solution.

The first step requires to provide the planner with an initial set of paths. Suppose we use any roadmap path planning method (visibility graphs, voronoi diagrams, silhouettes, etc.) to find a path in the free-space for each robot in the system. Then, it is possible to find some different path for each robot, by forcing the robots to go through some particular points in the free-space. In this way, it is possible to generate a family of paths for each robot, each family with a growing number of paths in it, as time goes by.

These families are the growing space to be investigated: the search algorithm chooses a single path for each robot selecting it from the respective family, according to some criteria, as for example "choose the set of paths which gives the best robustness index".

The second and third steps are the core of the planner and deal with the ways of *generating the growing family* and of *selecting one path from each* family. Criterion to build alternative paths starting from the first given solution and to design the search algorithm able to choose one path for each robot need to be defined. One of the simplest choices for building alternative paths is to choose randomly a point in the free space at a growing distance from the mid-point of the first solution path. We have experimented this strategy for testing our approach, as it is illustrated in Section 4. This choice showed good performance, even if it may not give any particular interesting information about the goodness of initial solution. Experimentation showed that, as the growing distance grows too much, the paths added to the family are too far away to be useful.

An important point is to decide how the search algorithm operates. Given a number r of robots, each one with a family of k different paths, the algorithm chooses one path from the first family evaluating which one of the paths in the family is the best; then it is the turn of the second family, and so on. Naturally the given solution is suboptimal, but we found that almost always it is very near to the optimal one. In particular, each time the search algorithm tries a different path from a family, it must recalculate or reestimate the values for the robustness index of all paths in the solution, unless the robustness criterion chosen is ME because, only this one is a property of the single path, while the others are a property of the whole solution.

We can denote the plan execution time as T_{exe}. From the definition of the quality indices we have

$$T_{\text{exe}} = RT.$$

Let T_{elab} be the time actually elapsed and used for the plan elaboration. The total time can be given as follows:

$$T_{\text{tot}} = T_{\text{elab}} + T_{\text{exe}}.$$

The monitor can estimate T_{tot} by measuring on-line T_{elab} and by estimating T_{exe} on the base of the actual best solution provided by the search algorithm. If T_{tot} is going to result better and better (i.e. smaller and smaller) the monitor continues to allocate time for new elaboration cycles to the search algorithm, but when T_{tot} begins to grow, the monitor stops the computation and starts the execution of the plan, because probably any further time dedicated to computation would delay the end of the execution. In this way, the planner tries to optimise the total time elapsed since the start of the computation till the end of the execution of the plan.

4. Experimental system

Our experimental system (called AnyRob) was built in C++, on an Indy Silicon Graphics workstation [7]. We present now an example of how AnyRob works.

In Fig. 3 we show an environment with some fixed obstacles. The points marked with a plus sign (+) are the robot start positions while the points marked with a cross sign (×) are the robot goal positions. The obstacles have been enclosed within their bounding boxes, and the robots shrink to a point. In Fig. 3, some

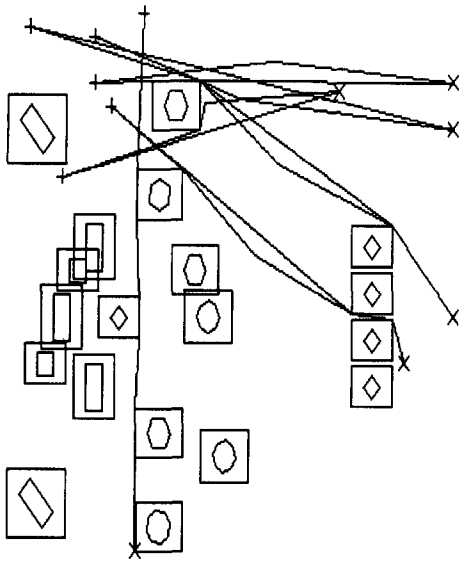


Fig. 3. An initial solution for motion planning.

tentative solution paths are also shown. These paths form the initial solution.

The planner starts constructing incrementally the path families (see Fig. 4). The alternative paths are generated using the mid-point method which consists of forcing the paths to go through some points whose distance from the generator path gradually increases.

The control monitor watches the elaboration of new solutions by estimating T_{tot} each time new paths become available. The monitor can interrupt the elaboration at any time. One path from each family is selected by the search algorithm and the total time of elaboration and of plan execution, T_{tot}, is estimated. Note that the time of execution is just RT. If T_{tot} is getting worse, the monitor stops the path family growing phase and it activates the collision avoider, otherwise it will ask for new paths.

The collision avoidance phase solves chronologically all the possible collisions using the *Stop & Go* method or the *Shape Changing* method. The Stop & Go method consists in modulating the robot velocity to avoid collisions with the other robots and with mobile obstacles by introducing some stops in the robot motions. The Shape Changing method solves collision by introducing local variations on the shape of paths. Shape Changing method demonstrated to be very powerful, in all the cases in which the Stop & Go method introduces very long delays.

The result of the experiment is shown in Fig. 5.

We give in the following some more experimental results. With 12 robots working in a simple environment, AnyRob elaborates for 19 s to find a good solution to the PPP problems (trying four different paths

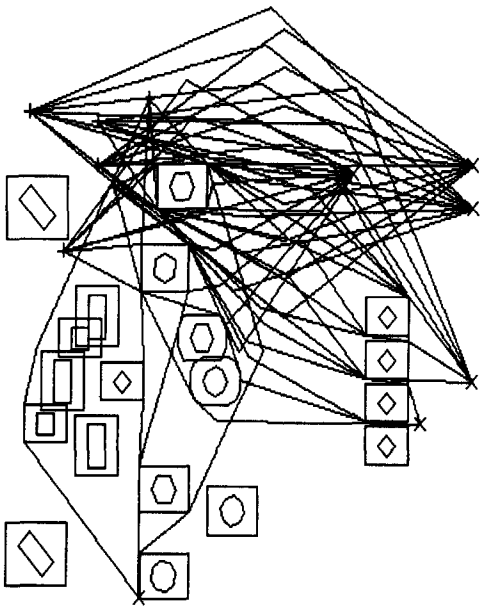


Fig. 4. The path family growing phase.

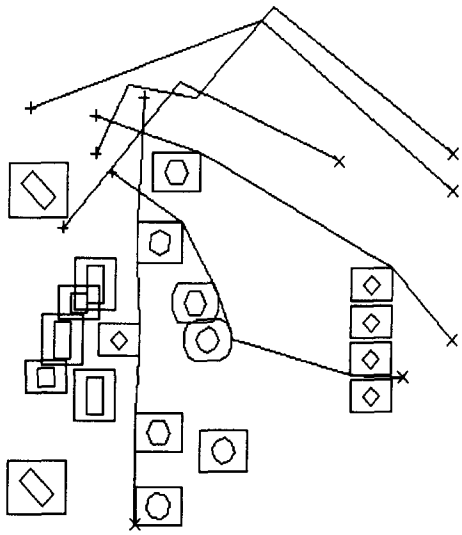


Fig. 5. The final solution.

for each robot). Then, the control passes to the collision avoider which finds a solution using Stop & Go and Shape Changing algorithms in 6 s. The RT value was 284 s while the RT for the first simple solution found was of more than 400 s with 121 robots, AnyRob elaborates for 463 s in the PPP phase, and for 500 s in the collision avoiding phase, while RT was 142 s. In this situation, controlling the algorithm with our approach is not too efficient, because there are too many robots and the first solution found is still the best, because the time to find a second solution is too high.

5. Conclusions

In this paper, we presented a solution to the multirobots motion planning problem based on the definition of plan robustness and on the use of flexible computation. We applied the idea of time-dependent planning to the problem of searching for alternative robot paths to a given initial solution. Variations of initial solutions for collision-free robot paths are obtained with respect to quality parameters that give heuristic rules for evaluating plan robustness. As quality parameters, we used both collision impact factors (CIF and CAF), for evaluating the quality of a single path, and performance indices (RT, ME and VE), for evaluating the overall quality of a plan. A prototype of the multirobot motion planner, called AnyRob, has been implemented in C++ and tested on an Indy Silicon Graphics workstation.

A particular problem which emerged from experimenting AnyRob is the existence of critical paths: there are some robots which arrive at their destination (goal) later than the other, because they are much slower or have paths much longer than the other robots. Robots corresponding to critical paths can be identified since the beginning of computation, in order to optimise the plan for these robots and for all those which are involved in collisions with them. If only the family corresponding to critical robots is allowed to grow, there will be less new paths, and the computation step will be quicker.

Moreover, the criterion for creating new paths can be improved. For example, the new paths can be chosen in a way that allows to avoid collisions (improving RT and ME), i.e., the algorithm of *shape changing* can be used in the family-growing step rather than in the collision-avoiding step to obtain better families.

Another interesting problem is the one of premature convergence. It might happen that the system finds a solution which seems to be optimal because all efforts have been concentrated to refine a first good solution, while much better solutions exist, but are not found because they are too distant from the first good solution. A way to avoid the problem is always to try some solution different from the actual best one, just like it is done using genetic algorithms [16]. Indeed an interesting analogy may be done between our approach and GA. We plan to further investigate the relation between these two approaches.

Acknowledgements

This research is being pursued in co-operation between the Department of Electronics and Informatics (DEI) of the University of Padua, from the Italian side, and the Department of Precision Machinery Engineering of the University of Tokyo, from the Japanese side. Financial support has been provided by the Italian Ministry of University, and Scientific and Technological Research. We like to acknowledge L. Marangoni, M. Martello, E. Modolo and M.Voltolina, master students at University of Padua, for their contribution in implementing some software modules of the motion planner.

References

[1] T. Arai, J. Ota, Motion planning of multiple mobile robots, in: Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems, Raleigh, NC, 1992, pp. 1761–1768.

[2] M. Boddy, T.L. Dean, Deliberation scheduling for problem solving in time-constrained environments, Artificial Intelligence 67 (1994) 245–285.

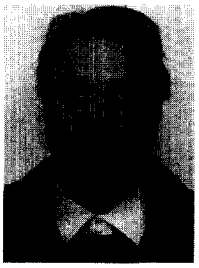
[3] S.J. Buckley, Fast motion planning for multiple moving robots, in: IEEE Proceedings of the International Conference Robotics and Automation, Phoenix, AZ, 1989, pp. 322–326.

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

[5] C. Ferrari, E. Pagello, J. Ota, T. Arai, Planning multiple autonomous robots motion in space and time, in: Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, Pittsburgh, PA, 1995, pp. 253–259.

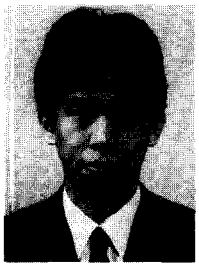
[6] C. Ferrari, E. Pagello, J. Ota, T. Arai, A framework for robust multiple robots motion planning, in: 1996 IEEE/RSJ Proceedings of the International Conference on Intelligent Robots and Systems, Osaka, Japan, 1997, pp. 1684–1690.
[7] C. Ferrari, E. Pagello, M. Voltolina, J. Ota, T. Arai, Varying paths and motion profiles in multiple robot motion planning, in: Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA, 1997, pp. 186–194.
[8] C. Ferrari, E. Pagello, M. Voltolina, J. Ota, T. Arai, Multirobot motion coordination using a deliberative approach, in: Proceedings of the Second Euromicro Workshop on Advanced Mobile Robots, Brescia, Italy, 1997, pp. 96–103.
[9] K. Fujimura, Motion planning amid transient obstacles, The International Journal of Robotics Research 13 (5) (1994) 395–407.
[10] J.E. Hopcroft, J.T. Schwartz, M. Sharir, On the complexity of motion planning for multiple independent objects; PSPACE hardness of 'Warehouseman's Problem', The International Journal of Robotics Research 3 (4) (1984) 76–88.
[11] K. Kant, S.W. Zucker, Towards efficient planning: The path–velocity decomposition, International Journal of Robotics Research 5 (2) (1986) 72–89.
[12] J.C. Latombe, Multiple moving objects, in: Robot Motion Planning, Kluwer Academic Publishers, Dordrecht, 1991 Chapter 8.
[13] C. Mirolo, E. Pagello, Local geometric issues for spatial reasoning, in: Robot Motion Planning, 1991 IEEE/RSJ Proceedings of the International Workshop on Intelligent Robots and Systems, Osaka, Japan, 1991, pp. 569–574.
[14] E. Modolo, E. Pagello, Collision avoidance detection in space and time planning for autonomous robots, in: IAS-3 Proceedings of the International Conference on Autonomous System, Pittsburgh, PA, 1993, pp. 216–225.
[15] D. Parsons, J. Canny, A motion planner for multiple mobile robots, in: Proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, OH, 1990, pp. 8–13.
[16] J. Xiao, Z. Michalewicz, L. Zhang, K. Trojanowski, Adaptative evolutionary planner/navigator for mobile robots, IEEE Transactions on Evolutionary Computation 1 (1997) 18–28.
[17] S. Zilberstein, S.J. Russell, Anytime sensing, planning and action: A practical model for robot control, in: Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93), Chambery, France, 1993, pp. 1402–1407.

Carlo Ferrari was born in 1960 in Genua, Italy. He received the Laurea degree from the University of Genua in 1985, and the doctoral degree from the University of Padua in 1992. He was visiting the University of California at Berkeley from 1990 to 1991. Since 1992 he is Assistant Professor in Computer Science at the School of Engineering of the University of Padua, within the Department of Electronics and Informatics (DEI). His research interests are motion planning for multiple mobile robot system, assembly planning, RISC robotics, dynamic simulation.



Enrico Pagello is an Associate Professor of Computer Science, since 1983, at the School of Engineering of the University of Padua, where he belongs to the Department of Electronics and Informatics (DEI). He has also a joint appointment with the Institute Ladseb of CNR as a part-time Senior Scientist. He is the Chairman of the Working Group on Robotics (GLR) of the Italian Association for Artificial Intelligence (AIIA). He is the Chairman of the Italian RoboCup Scientific Committee. He is a member of the Council Board of the Italian Association of Robotics (SIRI). He will be the General Chairman of next Intelligent Autonomous System 6, to be held in Venice, in July 2000. He is an Associate Editor of IEEE/Trans. on Robotics and Automation, since 1997.

Tamio Arai was born in 1947 in Tokyo, Japan. He received the M.S. degree and D.S. degree in Engineering from the University of Tokyo in 1972 and 1977, respectively. He was a visiting researcher in the Department of Artificial Intelligence, Edinburgh University in 1979–1981, He has been a professor in the Department of Precision Machinery Engineering, the University of Tokyo since 1987. He has mainly worked on robotics and manufacturing engineering. Currently his research interests include (1) automatic assembly, (2) planning and control of plural mobile robots, and (3) robot language and protocols in manufacturing.



Jun Ota was born in 1965 in Saitama, Japan. He received M.S. and D.S. degrees from the Faculty of Engineering, the University of Tokyo in 1989 and in 1994, respectively. From 1989 to 1991, he joined Nippon Steel Cooperation. In 1991, he was a Research Associate of the University of Tokyo. In 1996, he became an Associate Professor at Graduate School of Engineering, the University of Tokyo. From 1996 to 1997, he was a Visiting Scholar at Stanford University. His research interests are multiple mobile robot system, environmental design for robot systems, human–robot interface, and cooperative control of multiple robots.