Varying Paths and Motion Profiles in Multiple Robot Motion Planning

Carlo Ferrari(*), Enrico Pagello(*)(#), Matteo Voltolina(*), Jun Ota(§) and Tamio Arai(§)

(*) Department of Electronics and Informatics, The University of Padua, Via Gradenigo 6a, The University of Padua, I-35100 Padova, Italy

(#) Institute of System Sciences and Bioengineering - National Research Council (LADSEB-CNR) Corso Stati Uniti 4, I-35127 Padova, Italy

(§)Department of Precision Machinery Engineering, Graduate School of Engineering, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113, (Japan).

Abstract

We show how using anytime algorithms for solving the multiple robots motion planning problem allows to evaluate the opportunity of looking for alternative solution paths by generating small variations of robot motions in space and in time. By using the concept of plan robustness, we generate several alternative paths that are evaluated through various Performance Indices and Impact Factors, that allow to know how much a variation affects a given plan. Finally, we outline some recent experiments.

1. Introduction

The problem of *planning motion for multiple robots* seems to be quite complex; but can this problem be solved by making time-dependent the search for an alternative path for each single robot? If the answer is positive (and we show in the paper how to do this) we believe it would lead to a great flexibility.

Indeed, Boddy and Dean, in [2], investigated the use of *flexible computation* for robot-tour problem. They showed how to minimize the total amount of time consumed by a planner both in letting the robot sitting and deliberating about what to do next, and in actually making the robot moving around its environment. The statistics on travel time reduction and on tour length reduction were conveniently summarised in some performance profiles, that show how the expected savings in travel time increases as a function of time spent in path planning, and the expected length of the tour decreases as a fraction of the shortest tour for a given amount of time spent in tour improvement.

Once assumed that a robot starts out with some initially randomly selected tour, the 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 [3].

We would like to be able to take into account various kind of *motion variations* while planning, to choose a plan which is robust to those variations. In such a way, we could apply the deliberative scheduling approach to generating alternative path with respect to plan robustness. Therefore, in this paper, we investigate how to obtain better solutions from the original good one generated by using our Multiple Robot Motion Planner presented in [6], by *looking for alternative paths* with respect to some *quality parameters* that give heuristic rules for evaluating plan robustness, according to the definition of robustness given in [7]. 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.

Solving the multiple robots motion planning problem means essentially finding for each robot (or, in general, for each autonomous vehicle) a path and a proper velocity profile that meets all geometrical and task-related constraints.

The research work on motion planning for multiple mobile robots finds its foundation on a few papers that appeared in the early 80's, among which, one of the most important, is [9], 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 [5], where a configuration space-time was used to represent the time-varying constraints imposed by the other moving and stationary objects, and in [14], where an algorithm, based on a global cell decomposition approach was presented. Another important contribution was the use of algorithms with priority [2], [5]. In [8], the case when the environment contains obstacles whose existing periods are dependent on time, was considered, allowing in such a way 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 [11]).

A common view, however, in evaluating a multiple robots motion planning system, is to classify the planner

0-8186-8138-1/97 $10.00

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, we have to rely on some effective heuristic in order to develop a practical solution to the problem. Therefore, in our research in multiple robot motion planning, we tried to use simple, even if reliable, heuristics. To this aim, in [6], we introduced *performances indices:* the planned path, with associated highest index value, may preclude the motion of a robot along another path, with lower index value. The penalty given, in such a way, by the fact that we find out non-optimal solutions is highly compensated by the possibility of applying our heuristic in an effective way.

Later, in [7], we introduced the concept of *motion plan robustness.* In fact, we said that a motion plan is robust if it can be used in spite of small variations in the motion context.

Motion plan robustness has not yet been investigated enough. It is particularly useful when examining environments filled up with many robots. In this situation, indeed, a small variation in the execution plan of one of the robots may reflect in large variations to the other robots' execution plans, bringing, for example, to the necessity of re-building the complete plan for all robots, with a great waste of computation.

Studying robustness has several advantages. Firstly it becomes possible to set a proper library of prototype plans that can be slightly modified to cope with classes of applications. Secondly, 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, bringing to a better use of computation time

The paper is organized as follows. In the *second section,* we recall the main results of our previous research on the motion plan generation for multiple robots systems, we clarify better the role of collision avoidance in our planner, introducing the shape changing method, and we add to the performance index set, illustrated in our past papers, a new performance index, the Velocity Error parameter. In the *third section,* we how to apply anytime algorithms to the Multiple Robot Motion Planning Problem. In the *fourth section,* we illustrate some experimental results obtained using our new planner AnyRob. Finally, we give some *conclusions.*

2. The Multiple Robot Motion Planning problem solved in Space and Time.

As we will see with more details, in the next sub-section, the motion planning problem can be divided into two distinct phases: *Path Planning Problem* (PPP) and *Velocity Planning Problem* (VPP). In the first phase (i.e. in PPP) a geometrical path is assigned to each robot to avoid collisions with fixed obstacles, while in the second phase (i.e. in VPP) a velocity schedule is determined, to avoid collisions among robots and with moving obstacles.

For multiple robots you can build many different paths for each robot, to have more choices when searching for a good solution, i.e. a good set of paths and of velocity schedules (according with some quality parameter that we will see in the next sections)

Suppose we have r robots and suppose that we can build k different paths toward the goal for each robot. Then there are k^r possible solutions to the "path scheduling problem", because for each robot you can choose one of the k different paths built by the path planner. To find the optimal solution has an exponential complexity, but a search for non-optimal solution can be done with polynomial complexity.

A way to obtain a sub optimal solution is the following: a new problem can be set and solved, where r' = kr new robots, one for each possible path, is taken into account. That is, any robot of the original problem can be 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 *sub optimal solution* can be achieved solving the VPP problem for all r' robots at the same time (ignoring intra-family collisions) and then choosing a single robot from each family.

When we find a sub-optimal solution, a new problem arise of evaluating how good this sub-optimal solution is, i.e. of evaluating the *quality* of the solution. Note that the solution quality is a property of the whole solution, i.e., for almost all the quality indices, it depends on all the paths and on all the velocity schedules in the solution, so that it is often very difficult to give a quality parameter for each single path. A particular aspect of solution quality is *robustness.* We proposed in [7] a set of robustness indices to evaluate the robustness of a plan; in this paper we propose some new index. Moreover, we can now use these robustness indices not only to evaluate the goodness of a solution, but even to search for new and better solutions following the deliberative scheduling approach.

The idea is to use an anytime path planner to build paths which should bring to better solutions according with some heuristic rule based on robustness indices. Then a path for each robot is chosen to improve solution's robustness indices. The process continues until a deliberative scheduler says to stop.

2.1 The Path Planning and the Velocity Planning Problems.

In our previous paper [6], we generalized to the multiple robot case our motion planner in space-time, firstly designed for a *single robot* [13]. Therefore, we approached the problem of *multiple robot motion planning* by separating the space analysis from the time analysis, in the same spirit of what was done in [10] for a point robot. Performance indices were associated to the various candidate robot paths. Collisions were solved using a STOP-and-GO strategy.

To analyse the motion of the moving objects, we followed the approach, originally introduced in [10], of decomposing the trajectory planning problem into the two sub problems of planning a path to avoid collision with static obstacles, the PPP, and of planning the velocity along the path to avoid collisions with moving obstacles, the VPP.

In order to complete the space and time analysis, we first solved PPP, that is, we first computed some free paths (for each robot) among the fixed obstacles, connecting the start position with the goal position. Then, we analysed VPP, looking first for those path segments that are in use in the same time interval, that is making a Temporal Collision Analysis, and checking later for spatial intersections between the pair of segments in use, that is making a Spatial Collision Analysis.

We solved PPP by using a 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 [12]. The output of the PPP phase is a family of paths related to each robot in the environment. The VPP phase considers all those paths simultaneously, and computes the schedule for each robot running on all of its path. At the end of the VPP phase the plan is build by picking up the best path for each robot .

In multiple robot motion planning, the following particular cases must be considered:

- A collision checking step determines if two robots running on intersecting segments will collide
- A delaying step decides which is the robot to stop (or delay) and recomputes the basic time scheduling for each robot

Note that two robots will collide if some segments of their paths intersect and they are running on them in the same time slot (intra-family robot collisions are ignored).

A time analysis is performed first. When two path segments temporally overlap, the system checks for spatial path segments intersections. If a collision occurs, one of the two robot is stopped until the other has cleared the path, or a shape changing algorithm (explained in the next section) is applied.

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

2.2 Collision Avoidance among Robots

Stop & Go method

Solutions to collisions are obtained by Stop & Go Strategy, which consists in modulating the robot velocity to avoid collisions with other robots and with mobile obstacles by introducing some stops in robot motions. Note that the shape of paths, in Stop & Go method, is to be considered fixed.

In particular, robots are stopped before they enters the "critical area" to allow the other robot or moving obstacle to go on without collisions (figure 1).

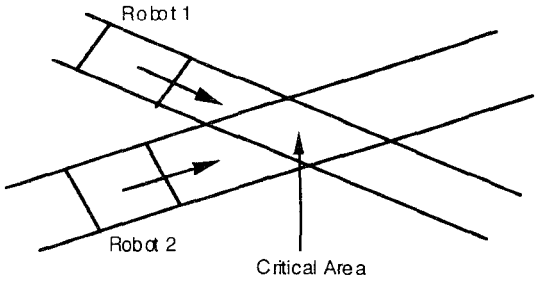

Figure 1

The Stop & Go algorithm might bring to a situation of dead-lock. There's an analogy between robots and parallel processes: critical collision areas are just like resources that can not be shared among robots (i.e. processes).

If it happens that more robots try to access the same resources (i.e. areas) at the same time, because the areas are superimposed, a dead-lock may generate (figure 2).

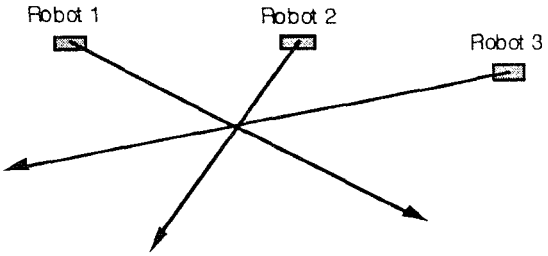

Figure 2

An obvious solution to the dead-lock problem is to give a priority to the robots.

Shape Changing Method

We improved this approach by introducing an algorithm called Shape Changing, which allows to solve collision introducing local variations on the shape of paths (figure 3).

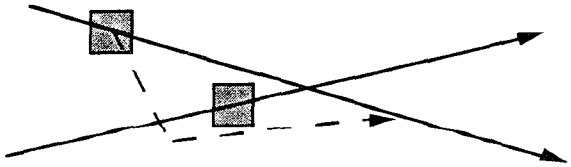

Figure 3

In this way, good solutions for avoiding collision can be achieved with regard to robots' running time, but the disadvantage is that the shape changing algorithm has an elevated computational complexity. Anyway, as we will see later, this algorithm has also interesting implication in

paths robustness.

In figures 4 and 5, we see an example of a situation where shape changing offers a much better solution to the problem solving collisions than Stop & Go algorithm.

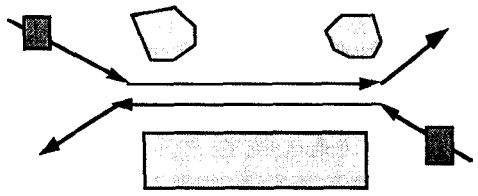

Figure 4

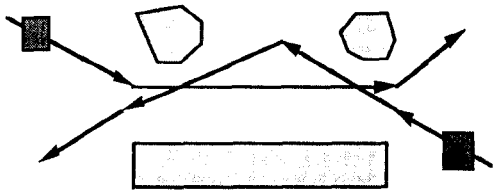

Figure 5

Summing up, collisions can be solved both with a Stop & Go algorithm and with a shape changing algorithm. A way to decide which of the two algorithm to apply has been proposed, based on heuristic considerations.

Performance Indices

As performance indexes, we associate to a path the *running time RT,* that is the minimum time a robot should need for reaching the goal using that path, and the *motion error ME,* that measures how a robot can move away from its path without colliding with obstacles or other robots.

A 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 index 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 modified they are recomputed

RT is the minimum time a robot need 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 paths, but a property of solutions: 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). But this local optimality doesn't solve the problem of ME optimization. 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 (see figure 6).

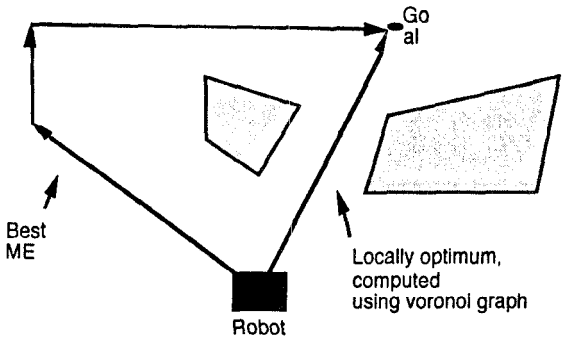

Figure 6

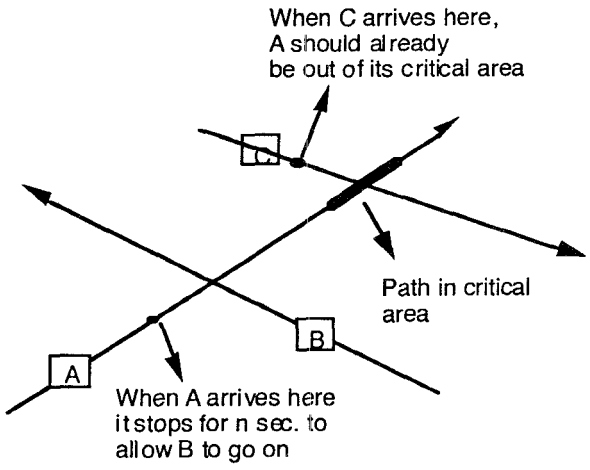

Figure 7

VE measures how a robot can vary its scheduled velocity without causing any collision with other robots. See for example figure 7 : if robot A is stopped waiting for robot B to pass, robot B has a VE=0 because if B arrives later than scheduled, B will collide with A. Note that only delays affect VE, because if a robot arrives earlier it can simply stop and wait for a while. Otherwise, if A arrives later than scheduled, no collision will happen, so that the value of VE for A, in this example, seems to be positive infinite. But, if A should arrive to its stopping point later than the time scheduled for its re-start, this delay could

affect the following collision with robot C; that is the reason why VE has a value different from positive infinite, even for robot A. Please, note that here we defined VE for each single path. Thus, we can define a "global" VE as the worst VE among all robots, or as the medium value for all VE.

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 paths family for each one, planning the motion for all the possible paths and then choosing the best for each robot. We will return later to this point.

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 substituting path p with path q, because the whole solution should be re-computed. Thus, we evaluate a *Collision Impact Factor* (CIF) for q, where

CIF = (number of robots whose paths cross q) / (total robot number)

A CIF almost equal to one suggest 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 if discard or not a path, because the kind of geor- -trical intersection is much more interesting than tl_ bare number of intersections. In fact, a collision of the kind illustrated in figure 4 cause a delay much greater than a big number of orthogonal intersections. For this reason, it is better to estimate the length of the piece of path inside the critical collision area. So, a better definition than CIF is the following:

CAF = (length of path inside critical collision area) / (total path length)

CAF has demonstrated to be a good parameter to decide if discard or not a path both when we try to optimize RT and when we try to optimize 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 the RT, ME and VE indexes 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 approach to families of varying paths

As we said before, we are able to find a sub-optimal solution, according with some quality criterion, to the problem of multiple robot motion planning, in a satisfactory time, while a complete search algorithm in the number of robots it not interesting from a practical point of view.

Looking for sub optimal algorithms, we presented in [6] a planner which calculates a good solution with a polynomial complexity. However, in that paper we did not further use quality indices of the solution paths, and in particular those particular measures of quality given by the robustness indices introduced in [7]. We can use these indices in anytime algorithms [15]. Therefore, we take now those quality indices into account by applying the deliberative scheduling approach [2]. In particular, we control the process of growing path families 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 if continuing or stopping the elaboration of the planner, depending on the values evaluated for the solution paths.

The space of the possible paths for each robot and of the possible velocity schedules along each path becomes a *search space* that can be increasingly built starting for an initial, and easy, solution, and then adding to it several alternative choices, i.e. many different paths or velocity schedules. Please, note that we need some criterion to generate those alternative choices, and that their generation should be incremental, to provide to a search algorithm a growing space to be investigated, so that it may ideally continue its search forever.

Suppose to 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, for example by forcing the robots to go through some particular points in the free-space. In this way, we can generate a family of paths for each robot, each family with a growing number of paths in it, as time goes by. Those family are the growing space to be investigated: we must define a search algorithm that chooses a single path

for each robot selecting it from the respective family, according with some criteria, as for example "choose the set of paths which gives the best robustness index I".

What is now to be decided are the ways of *generating the growing family* and of *selecting one path from each family,* i.e. we need to define the 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. One of the simplest choice 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 of the paper. This choice showed good performance, even if it may not give any particular interesting information about the goodness of initial solution, and it has some problem. In particular, 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 form 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 sub-optimal, 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 re-calculate or re-estimates the values for the robustness index of all paths in the solution, unless the robustness criterion chosen is ME because, as we have already seen, only this one is a property of the single path, while the others are a property of the whole solution.

Our experimental system has been called *AnyRob.* It is built in C++, and runs on a Silicon Graphics workstation. It operates in such a way to obtain a good solution with respect to RT. We plan to experiment also the use of ME and VE. We have chosen, as first, RT because it is easy to understand how much time the monitor (which controls the duration of the elaboration) must grant to the anytime algorithm with creates the solutions spaces and search for a good solution. In fact, if we say t_{exe} as the plan execution time, then:

$$t_{exe} = RT$$

Now we can define as t_{elab} the time actually elapsed (used for the elaboration). Then, the monitor can estimate:

$$T_{tot} = t_{elab} + t_{exe}$$

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, then it is better to stop the computation and start the execution of the plan, because probably any further time dedicated to computation would delay the end of the execution. In such a way, the planner tries to optimize the total time elapsed since the start of the computation till the end of the execution of the plan.

In AnyRob the possible collisions among robots are avoided using two methods: the first is the classical Stop & Go method, while we the second "Shape Changing" method. It consists of introducing local variations in the shape of the paths to avoid collisions. Shape changing method demonstrated to be very powerful, especially in situations like those in figure 4 and 5, where the Stop & Go method introduces very long delays.

4. Using AnyRob in a simple experiment.

When a solution to the multiple robot motion planning problem is available, it is possible to evaluate each of the robustness indices RT, VE and ME, simply measuring it. Then, starting from a first simple solution, we can build new different paths for each robot, using some heuristic technique which have an interpretation in terms of robustness, i.e. that are supposed to improve some robustness index.

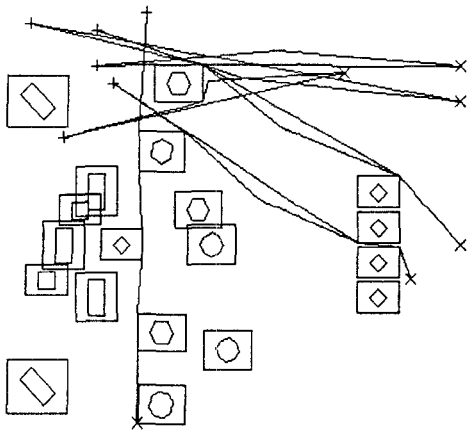

Figure 8

For example, if we want to improve ME, we can select those path segments with the worst ME, i.e. those segments whose distance to the some fixed obstacle is the smallest, and try to find a path segment with a better ME. If we want to improve VE, we can use a shape changing algorithm which avoids collisions introducing a local variation on the shape of the paths (figure 3). In such a way collisions can be avoided, and VE is expected to improve. Finally, if we want to improve RT, we can use once more a shape changing algorithm, because almost always avoiding collisions means improving the running time (see figure 4 and 5).

Once we have some alternative paths, we can try to

build a new solution using some of these new path. To do this it is necessary to choose a single path from each family, by a search algorithm, and then start a "collision avoiding phase" which outputs the final solution.

The search algorithm can be run many times, as newer and newer paths are added to the first set, and the process can keep on running until a monitor decides that it will bring no improvement in the solution (for example, estimating T_{tot} as we have already seen, if RT is the performance index used in growing path families).

To help understanding our method, we present a simple example of how AnyRob works, where the new paths are built though the mid-point method, which consist in forcing the paths to go through some points whose distance from the first path gradually increases:

1. First, given an initial solution, as shown in figure 8, let us construct incrementally the "Paths families" by any flexible algorithm, as shown in figure 9, where the alternative paths are generated by the mid-point method in existing paths discussed above.

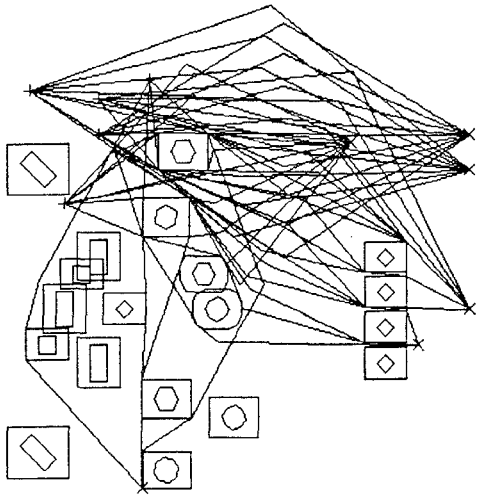

Figure 9

2. 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, Ttot, is estimated by the monitor each time a solution is available. Note that the time of execution is just RT.

3. If Ttot is getting worse, the monitor activates the collision avoider, otherwise it gives back the control to the flexible algorithm (step 1.)

By these three steps, Anyrob solves the problem of analyzing many different alternatives looking for a solution that has a good RT. Note that, with the same algorithm, AnyRob could provide a solution which has a good ME, while it's more difficult, at this level, to estimate VE.

4. then starts the Collision avoidance step. The collisions are solved chronologically by Stop & Go or by Shape changing (figure 10).

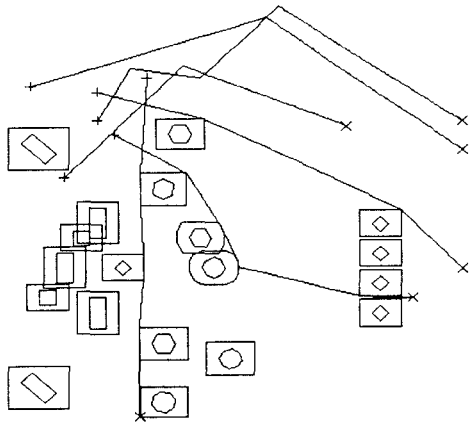

Figure 10

It is very interesting to note that in this example the algorithm of shape changing was used to obtain better solutions in terms of RT, but the same algorithm can be used even to find better solutions in terms of VE, in facts shape changing helps to avoid collisions, bringing a higher VE. Although we developed Anyrob to deal only with RT, the algorithm can be re-used to deal easily with ME, while dealing with VE it may be just a little bit more complex. We can say that it's even more complicated to deal with RT than with ME, because while ME can be computed during the PPP phase, RT has to be estimated using various parameters, among which the most important is CIF. CIF can be used to find paths that are supposed to bring a better VE. Then, if the plan effectively has associated to itself a better VE, the path is accepted. Else, the old plan can be re-used. To refine a good plan, shape changing algorithm can be used, which helps finding paths with less collision problems, and therefore best VE.

We give in the following some experimental results. With 12 robots working in a simple environment, AnyRob elaborates for 19 sec. to find a good solution to the PPP problems (trying 4 different paths for each robot). Then, the control passes to the collision avoider which finds a solution using Stop & Go and Shape Changing algorithms in 6 sec. The RT value was 284 sec. while the RT for the first simple solution found was of more than 400 sec.
With 121 robots, Anyrob elaborates for 463 sec. in the PPP phase, and for 500 sec in the collision avoiding phase, while RT was 142 sec. In this situation, controlling the algorithm with our approach was not 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.

A particular problem which emerged from experimenting AnyRob is the existence of critical Paths: the RT for the global system, defined as the maximum RT among all robots, depends on some of the robots, i.e. there are some robots which arrives at 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, so that the greatest efforts can be concentrated in optimizing the plan for these robots and for all those which are involved in collisions with them.

If the phase of family growth is more accurate, that is if only the family corresponding to critical robots is allowed to grow, correspondingly there will be less new paths, and the following 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. With these few modifications the algorithm should improve its performances.

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. This problem happens especially if all efforts are concentrated on critical paths. A way to avoid the problem is always to try some solution different from the actual best one, just like it is done in Genetic Algorithms. Indeed an interesting analogy may be done between our approach and GA. In another research project, we experimented GA for Multiple Robot Motion Planning Problem, and we are now comparing the two approaches. We will report in a future paper, or possibly, in a revised version of this one, these comparisons.

5. Conclusions

In this paper, we presented a solution to the multiple robots 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 some 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. All the proposed procedures are of practical use, in the sense that they give fast alternative solution paths. So far, a first prototype of a new multiple robots motion planner, called Anyrob, has been implemented in C++ and is under testing on a Indy Silicon Graphics workstation.

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 and M. Martello, 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". Proc. of the 1992 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS92). Raleigh, NC (USA), July 7-10, 1992, pp. 1761-1768

[2] M. Boddy & T.L. Dean: "Solving time-dependent planning problems", Proc. of IJCAI-89, Detroit, Mi, 1989

[3] M. Boddy & T.L. Dean: "Deliberation scheduling for problem solving in time-constrained environments". Atificial Intelligence, Vol. 67, 1994, pp. 245-285

[4] S.J. Buckley: "Fast Motion Planning for Multiple Moving Robots". Proc of 1989 IEEE Int. Conf. on Robotics and Automation (ICRA89), Phoenix May 1989, pp. 322-326 (ICRA89)

[5] M. Erdman & T. Lozano-Pérez: "On Multiple Moving Objects". Algorithmica, Vol. 2, 1987, pp. 477-521.

[6] C. Ferrari, E. Pagello, J. Ota, T. Arai: "Planning Multiple Autonomous Robots Motion in Space and Time", Proc of the 1995 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS95), Pittsburgh, Pennsylvania (USA), Aug 1995, vol 2 pp. 253-259.

[7] C. Ferrari, E. Pagello, J. Ota, T. Arai: "A Framework for Robust Multiple Robots Motion Planning", Proc of the 1996 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS96), Osaka, Japan November 1997, pp. 1684-1690

[8] K. Fujimura: "Motion Planning Amid Transient Obstacles". The Int. J. of Robotics Research, Vol. 13, No. 5, Oct. 1994, pp. 395-407.

[9] 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 Int. J. of Robotics Research, Vol. 3, No. 4, Winter 1984, pp. 76-88.

[10] K. Kant & S.W. Zucker: "Towards Efficient Planning: the Path-Velocity decomposition". Int. Journal of Robotics Research, Vol. 5, No. 2, 1986, pp. 72-89.

[11] J.C. Latombe: "Robot Motion Planning". chp. 8 on "Multiple Moving Objects", Kluwer 1991, pp. 356-402.

[12] C. Mirolo & E. Pagello: "Local Geometric Issues for Spatial Reasoning in Robot Motion Planning". Proc. of the IEEE/RSJ Int. Workshop on Intelligent Robots and Systems '91 (IROS'91). Osaka 1991, pp. 569-574.

[13] E.Modolo & E.Pagello: "Collision Avoidance Detection In Space and Time Planning for Autonomous Robots". Proc. of the Int. Conf. on Autonomous Sistem (IAS-3). Pittsburgh 1993, pp. 216-225.

[14] D. Parsons & J. Canny: "A Motion Planner for Multiple Mobile Robots". Proc. of the 1990 IEEE Int. Conf. on Robotics and Automation (ICRA90), Cincinnati May 1990, pp. 8-13.

[15] S. Zilberstein & S.J. Russel: Anytime sensing, planning and action: A practical model for robot control". Proc. of 13th IJCAI, Chambery, France, 1993, pp. 1402-1407.