In: International Conference on Robotics and Application (RA´99), Santa Barbara, USA, Oct. 28-30, 1999

MULTI-GOAL PATH PLANNING FOR INDUSTRIAL ROBOTS

CHRISTIAN Wurll, DOMINIK Henrich AND HEINZ Wörn
Institute for Process Control and Robotics (IPR),
Computer Science Department,
P.O. Box 69 80,
University of Karlsruhe (TH)
D-76128 Karlsruhe, Germany,
[Wurll, dHenrich, Woern]@ira.uka.de,
http://wwwipr.ira.uka.de/~paro/

Abstract

A new problem for the automated off-line programming of industrial robot application is investigated. The Multi-Goal Path Planning is to find the collision-free path connecting a set of goal poses and minimizing e.g. the total path length. Our solution is based on an earlier reported path planner for industrial robot arms with 6 degrees-of-freedom in an on-line given 3D environment. To control the path planner, four different goal selection methods are introduced and compared. While the Random and the Nearest Pair Selection methods can be used with any path planner, the Nearest Goal and the Adaptive Pair Selection method are favorable for our planner. With the latter two goal selection methods, the Multi-Goal Path Planning task can be significantly accelerated, because they are able to automatically solve the simplest path planning problems first. Summarizing, compared to Random or Nearest Pair Selection, this new Multi-Goal Path Planning approach results in a further cost reduction of the programming phase.

Keywords: motion planning, search algorithms, traveling salesman problem, shortest sequence

1 INTRODUCTION

The todays production lines usually consist of multiple robots, interacting with a wide range of equipment and fixtures. Programming these capital intensive installations can be made off-line in powerful robot simulation systems. The off-line programming is still a complex task and the resulting programs strongly depend on the programmer's capabilities. Let us for instance consider a spot welding task, in which a robot has to reach several spot welding points. In this scenario, the main goal of the programmer is to generate a collision-free robot program, which can be executed as quick as possible in order to achieve short cycle times, thus, increasing the total throughput. But even an experienced programmer often needs a lot of time to find at least an sub-optimal solution. Depending on the problem complexity, it is very difficult to choose the optimal sequence as well as to find a collision-free path between two spot welding points. Yet, neither in the current state of the art nor in the existing robot simulation tools like ROBCAD, IGRIP or CATIA, tools are available to solve the Multi-Goal Path Planning problem.

The main part of the Multi-Goal Path Planning is the finding of a collision-free path. The issue of robot path planning has been studied for a couple of decades and many important contributions to the problem have been made [14]. Point-to-Point (PTP) Path planning algorithms, which can find a collision-free path from one start configuration (point) to a goal configuration (point) are of great theoretical interest, but are rarely used in practice because of their computational complexity [16]. In the last two years a few new PTP path planning approaches have been published, which promise good results (see e.g. [2, 3, 10]).

The *Multi-Goal Path Planning* (MTP) problem which computes a collision-free path as well as the optimal sequence has not yet been considered. In our opinion, solving the MTP problem can improve the off-line generated programs and therefore reduce the total programming time.

In this paper, we introduce a first approach to solve the MTP problem. More precisely, we consider the following: Given an industrial robot (usually with six degrees-of-freedom) and a set of static obstacles. Both, the robot and the obstacles are provided as CAD models. Additionally, a

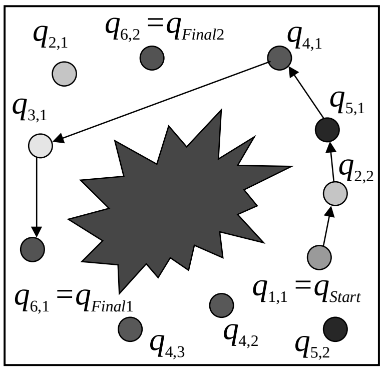

Figure 1: Illustration of a MTP problem in a two-dimensional C-space with one static obstacle (star) and several different goal configurations (dots). Dots with the same first index belong to the same pose, the arrows indicate the optimal sequence.

set of different goal poses representing the position and orientation of the robot's tool center point in the work space is given. According to ambiguous inverse kinematics of the robot, a pose can be reached by several different configurations in the configuration space (C-space). Now, the MTP problem is to compute a collision-free path between these poses and to find the optimal pose sequence, thus, reaching every pose at least once while minimizing the total path length (see Figure 1).

The remainder of the paper is organized as follows: Section 2 gives an overview of our MTP approach. The three basic modules "MTP Control", "MTP Path Planning" and " Shortest Sequence Planning" of this approach are explained in detail in Section 3, 4 and 5, respectively. Experimental results are presented in Section 6. The paper closes with the conclusion and an outlook to the future work in Section 7.

2 MULTI-GOAL PATH PLANNING APPROACH

In order to solve the MTP problem, the following concept was developed. It consists of three major modules, which are embedded in a main control loop (see Figure 2).

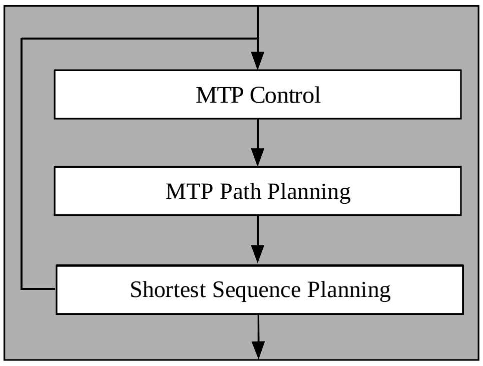

Figure 2: The MTP approach for solving MTP problems

At the beginning, every configuration represents a node in an initial graph (Figure 3a). In every iteration, the *MTP Path Planning* module computes one collision-free path between a set of start configurations and a set of goal configurations. The solution path is then inserted as the corresponding edge in the graph (see Figure 3b). In this iteratively growing graph the *Shortest Sequence Planning* module tries to find the shortest sequence in order to solve the given MTP problem (see Figure 3c).

In the worst case, the maximal number $R_{Max} = N*(N-1)/2$ of planning runs might be necessary for N configurations to find the optimal sequence. But depending on the chosen Path Planning algorithms, which are described in Section 4 and according to the regarded robot application, the total runs as well as the total solving-time can be reduced significantly.

3 MTP CONTROL

The MTP Control module plays an important role in the MTP approach. Its main task consists of selecting the suitable start and goal configuration pair for the MTP Path Planning in every iteration. This selection can be done either randomized or deterministic. In the following, four different selection methods are introduced, which are compared in Section 6.

P0 – "RANDOM PAIR SELECTION"

The simplest strategy is the Random Pair Selection method, which can be realized with any path planning algorithm. According to the current state of the MTP graph, two configurations q_i and q_j, which are yet not connected, are randomly selected. The path planner must then find a collision-free path between q_i and q_j. As this method has no knowledge about the distance between the configurations, a lot of eventually unnecessary planning runs are done, in order to find a valid sequence. In the worst case, the maximal number of planning runs R_{Max} has to be made.

P1 – "NEAREST PAIR SELECTION"

Instead of a randomized selection method, the start and goal pairs can be selected according to a specified order. For instance, the Euclidian distance between the start and goal configurations could be a possible order rule which can be easily computed at the beginning. In this strategy, the MTP Control begins with the start and goal pair, which has the shortest distance. In opposite to **P0**, the neighbored planning problems will be solved at first. But, in some MTP problems, it could happen, that a planning task seems to be very easy (short distance) at the beginning, but the direct path is blocked by an obstacle. In this case, the planning might take a long time until a collision-free can be found.

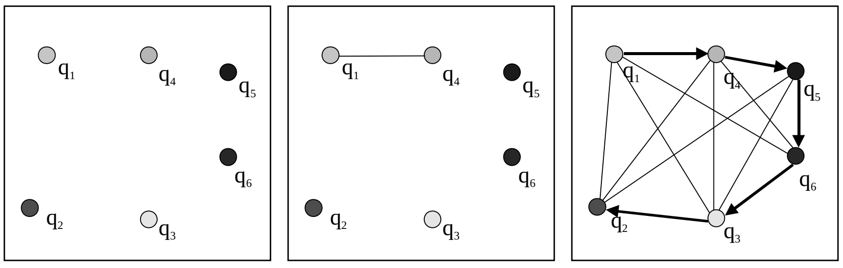

Figure 3: (a) Initial MTP Graph (b) Inserted solution edge (q_1, q_4) after one call of the *MTP Path Planning* (c) Solution sequence after 13 iterations computed by the *Shortest Sequence Planning* module

P2 – "NEAREST GOAL SELECTION"

In this strategy, the path planner must be able to cope with multiple goal configurations as for instance the MTP Path Planner, which is introduced in Section 4.

Beginning at the specified start configuration, the path planner receives all other (N-1) configurations as possible goals. As the path planner will find the shortest path at first, the nearest configuration will be selected automatically. In the next run, the planner continues with the previous goal and tries to find a collision-free path to all remaining configurations. After (N-1) runs, a valid sequence is found.

The computational amount for this strategy is linear in the number of poses, but it cannot guarantee to find the shortest sequence in terms of the total path length. This derives from the fact that it only considers the shortest path between one goal to the nearest goal and not the minimization of the total path. But the main advantage is that already after the minimum number of planning runs a first solution is available.

P3 – "ADAPTIVE PAIR SELECTION"

Is the path planner additionally able to cope with multiple start configurations as well as with multiple goal configurations, thus selecting automatically the easiest start and goal pair, then the Adaptive Pair Selection method can be applied. In this strategy, which can also be realized with the MTP Path Planner in Section 4, the planning systems receives all configurations including a list of edges (representing the missing collision-free paths), which still have to be computed.

4 MTP PATH PLANNING

The new MTP Path Planning method is based on an earlier reported PTP Path planner. The main concept consist on a best first search in the C-space [10]. The search needs an OPEN and CLOSED list to store the configurations to be investigated and the nodes which are already investigated. In every iteration, according to an evaluation function, the best node of OPEN is expanded. In order to avoid the time consuming obstacle transformations, collisions are detected in the workspace by a hierarchical distance computation [8]. In on-line provided environments with static obstacles and with an optimal discretisation [11], the planning times are only a few seconds [10].

Based on the goal switching concept introduced in [12], the search algorithm is extended to support multiple start as well as multiple goal configurations, in order to realize the MTP Path Planning.

The path planner is controlled by a *Task Table TT(i,j)* with $i,j=[1,\ldots,K]$ and $K=\sum_{i=1}^{N} n_i$, which is shown in Table 1.

Goal / Start	$q_{1,1}$	$q_{2,1}$	$q_{2,2}$	$q_{3,1}$
$q_{1,1}$	-1	0	0	1
$q_{2,1}$	0	-1	1	1
$q_{2,2}$	0	1	-1	0
$q_{3,1}$	1	1	0	-1

Table 1: Task Table TT for a MTP problem with one start configuration $q_{1,1}$, one pose with two configurations $q_{2,1}, q_{2,2}$ and one final goal configuration $q_{3,1}$. If for instance, a path from $q_{2,2}$ to $q_{3,1}$ is found, then in the next run, the corresponding table elements *TT*(3,4), *TT*(4,3) are set to 1.

TT(i,j) = 0 means that a collision-free path from configuration q_i to configuration q_j must be found. If *TT(i,j)* = 1, then no collision-free path has to be found between the configurations q_i and q_j. The table elements *TT(i,j)* with *i=j* are set to -1, since it is not necessary to find a path from one configuration to itself. Thus, the Task Table corresponds to the adjacent matrix of the MTP graph.

At the beginning of a path planning, the path planner can be initialized by the Task Table. Thus, if one table element in column *j* is equal to 0, the corresponding configuration q_j is inserted as a goal configuration into the hashing table CLOSED of the search algoithm. Analogue, if any element in row *i* is equal to 0, the corresponding

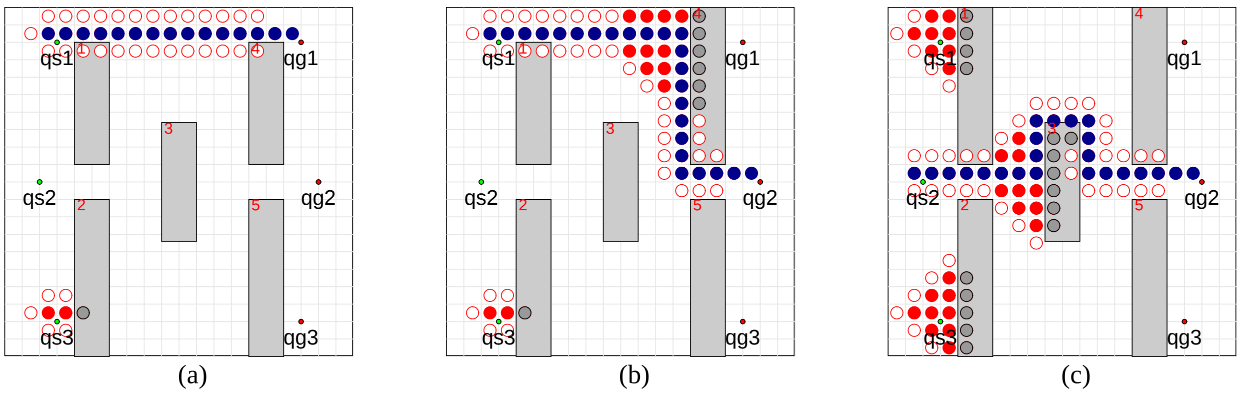

Figure 4: Three different examples in the two-dimensional C-Space with dynamic start and goal switching

configurations q_i is inserted as a start configuration into the OPEN list of the search algorithm. Thus, the Task Table is a flexible representation to specify the problem for the path planner.

The functionality of the dynamic start and goal switching is explained by three different examples, which are shown in Figure 4. All configurations on the left side are possible start configurations and all configurations on the right side are possible goal configurations. In all three examples, the search starts at the third configuration. As the search front hits the upright obstacle, the dynamic start switching leads to an expansion of the first configuration. In case (a), the search front can directly move to the first goal. In case (b), the obstacle in front of the first goal blocks the search, thus the dynamic goal switching leads to an automatic switch to the second goal. As in Figure (c), the first goal is blocked much earlier, the dynamic start switching activates a third search front at the second start configuration. This last search front slides around the obstacle in the middle and reaches the second goal without any further problems.

The integration of the dynamic start switching into the basic path planner needs no significant modifications. As the OPEN list of the search algorithm is per definition able to cope with multiple configurations during a search, only the initialization has to be adapted. Instead of inserting only one start configuration, all configurations according to the present state of the Task Table are added to the OPEN list. The search then starts with that configuration which has the cheapest evaluation value.

In order to realize the dynamic goal switching, the calculation of the evaluation function has to be adapted. Instead of computing the estimated costs of the current configuration q_i only to one goal configuration, it has to be computed to all configurations q_j, where TT(i,j) = 0.

5 SHORTEST SEQUENCE PLANNING

Finding the shortest sequence is very similar to the Traveling Salesman Problem (TSP) [15]. In the TSP, a salesman has to visit a number of towns at least once and as traveling costs time and money, the salesman is interested in traveling on the shortest tour starting and ending at his home town. Adapted to the Multi-Goal Path Planning, an industrial robot has to reach a number of configurations at least once. In contrast to the TSP, the robot usually do not have to return to its starting pose. Therefore, we call this problem the *Shortest Sequence Problem* (SSP).

The input of a TSP or SSP solver (xSP solver) is usually a graph with nodes and edges. In our application, the nodes represent the goal configurations and the edges represent a collision-free path between these configurations. Every edge has a cost value according to its path length. The path length of a collision-free path is measured in the number of via configurations. As a given pose may be reached by several different configurations, the graph contains groups of nodes, representing the different poses. In this case, a valid sequence consists of one node of each group, in order to reach every pose at least once. The similar TSP scenario would provide different suburbs for each town, and the salesman has to visit only one suburb of every town. We call these extended problems TSP++ and SSP++.

5.1 CONNECTIVITY TEST

In order to find a shortest sequence in a graph, the considered graph must be connected. While in the basic xSP cases the standard connectivity test is sufficient (see e.g. [15]), it has to be extended for the xSP++ cases. For these cases, the standard test could of course be applied, but it may happen, that a feasible xSP++ solution could be recognized much later. In Figure 5(a) and (b) two different examples are shown, each with one start configuration, two middle poses with three alternative configurations and one goal configurations. After four planning runs, in both cases a standard connectivity test would fail, although the example in Figure 5(b) is “xSP++ connected”.

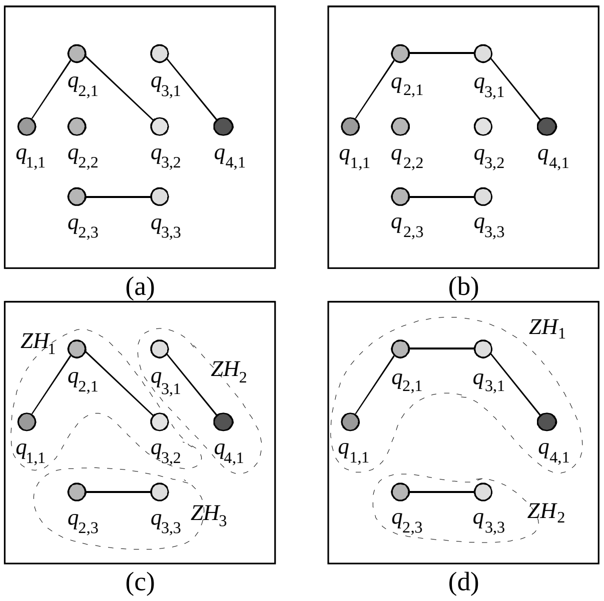

Figure 5: (a,b) MTP Graphs for two different MTP problems, (c,d) Connected components ZH_1, ZH_2 and ZH_3

In order to avoid this disadvantage, the xSP++ connectivity is checked in the following way: At first, a standard algorithm computes the connected components ZH_i in the MTP Graph [15]. Then it will be tested whether one component contains the start configuration (1), at least one configuration of every pose (2) and at least on configuration of the final pose (3). If one component fulfills these conditions, the MTP Graph is called *xSP++ connected*.

The example in Figure 5(a) contains three connected components, but none of them fulfills the three conditions, thus this example is not xSP++ connected. In opposite, as the connected component ZH_1 of the second example fulfills all three conditions, this example is xSP++ connected.

5.2 FINDING SHORTEST SEQUENCE

As a connected graph is not necessarily complete, a standard "Shortest-Path-Algorithm" as e.g. the Dijkstra Algorithm in [4] is used to compute all missing edges, which are inserted as *virtual* edges into the MTP graph.

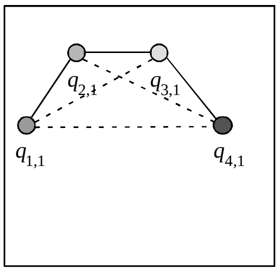

Figure 6: Completed MTP graph by adding the virtual edges (dashed line) for the connected component ZH_1 in Figure 5(d). A virtual edge between $q_{2,1}$ and $q_{4,1}$ represents a shortcut via the configuration $q_{3,1}$, computed by the shortest path algorithm.

Based on this completed graph, a xSP-solver can find the shortest tour or sequence. While solution methods for TSPs are already deeply investigated (see e.g. [5]: 120 cities; [18]: 532 cities; [6]: 666 cities; [1]: 13.509 cities), no methods of xSP++ tasks have yet been considered to our knowledge. By extending the available implementation of a TSP solver [19], we have developed a new xSP++ solver.

Pederson has used genetic algorithms (GA) to solve the traveling salesman problem. A simple introduction in the theory of GA can be found in [13]. Adapted to MTP problems, a *gene* stands for a goal configuration, a *chromosome* represents a valid sequence and a *population* is a set of multiple chromosomes. Every chromosome in one population is rated by a *fitness evaluation* function, thus, the best chromosome representing the best sequence can be found.

Analogue to nature, a population can grow by applying the standard GA functions crossover and mutation. In our approach, the *crossover* randomly selects two chromosome (parent P1 and parent P2) and generates a new chromosome (child C) by exchanging random parts of the parents (see Figure 7). The child will then be changed in the *mutation* function, by a permutation of a random selected part (see Figure 8).

In order to handle also group of nodes (multiple configurations of a TCP), we have introduced an additional layer into the GA, which we call a *gene group*. Thus, a chromosome consists no longer of genes, but of gene groups. Every gene group consists of at least one gene, if it only represents a single configuration or of multiple genes according to the number of nodes in the group under consideration. As a chromosome represents a valid sequence, every gene group selects one gene as the current member of the sequence.

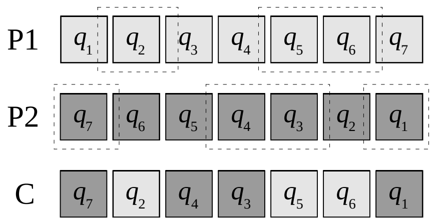
P1
q_1 q_2 q_3 q_4 q_5 q_6 q_7
P2
q_7 q_6 q_5 q_4 q_3 q_2 q_1
C
q_7 q_2 q_4 q_3 q_5 q_6 q_1

Figure 7: *Crossover* function of our genetic algorithm: The genes of parent P1 are selected and copied to the child at exactly the same location as they appear in the parent chromosome. Thus, both order and position of the genes in parent P1 are preserved. The genes that are not selected from parent P1 are copied form parent P2 to fill in the empty spaces in the child C. The order of the genes in parent P2 is preserved but the exact location are for obvious reasons, not.

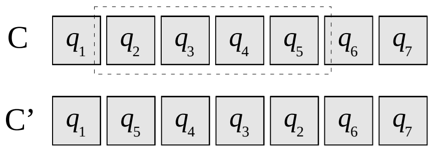
C
q_1 q_2 q_3 q_4 q_5 q_6 q_7
C'
q_1 q_5 q_4 q_3 q_2 q_6 q_7

Figure 8: *Mutation* function of our genetic algorithm: One at random selected part (consisting of consecutive genes) of the chromosome is permuted. In practice, this is done by switching the location of the gene within the selected part.

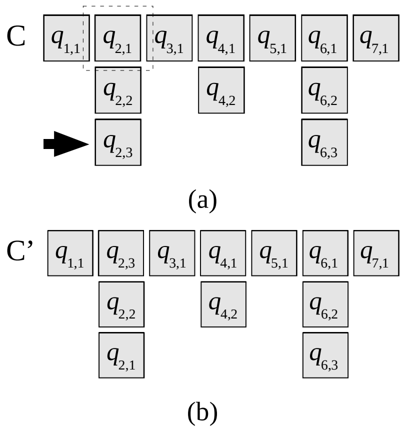
C
$q_{1,1}$ $q_{2,1}$ $q_{3,1}$ $q_{4,1}$ $q_{5,1}$ $q_{6,1}$ $q_{7,1}$
$q_{2,2}$ $q_{4,2}$ $q_{6,2}$
$q_{2,3}$ $q_{6,3}$
(a)
C'
$q_{1,1}$ $q_{2,3}$ $q_{3,1}$ $q_{4,1}$ $q_{5,1}$ $q_{6,1}$ $q_{7,1}$
$q_{2,2}$ $q_{4,2}$ $q_{6,2}$
$q_{2,1}$ $q_{6,3}$
(b)

Figure 9: New exchange function for our genetic algorithm: (a) One gene group is randomly selected (dashed rectangle). If the group contains more than one gene, the currently selected gene is exchanged by another randomly selected gene (arrow). (b) The resulting chromosome.

The *crossover* as well as *mutation* can therefore still be applied on these new chromosomes, but now working on gene groups instead of genes. The new function *exchange* was added in order to exchange a random selected gene of a random selected gene group, thus, modifying a chromosome (see Figure 9). As this function is applied only on one chromosome, it is very similar to the mutation function and could of course be integrated in it. But for a better modularity, we explain it as a separate method.

6 EXPERIMENTAL RESULTS

We have realized this MTP approach on a workstation (Pentium PC with 350 MHz and 128 MByte memory) running under LINUX operation system. The MTP path planner is implemented in C language and runs as a server process.

The MTP control unit is written in C++ using the LEDA-library [17] and communicates via the parallel virtual machine (PVM) interface with the path planner. For comparing the four goal selection methods, we have tested this new MTP approach on several industrial MTP problems. In Figure 10 and 11 the two MTP problems PIN ASSEMBLY and SPOT WELDING are shown.

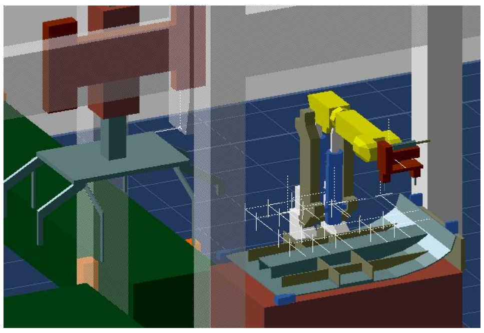

Figure 10: MTP problem PIN ASSEMBLY with 1 start configuration, 20 sub goal configurations and 1 final goal configuration.

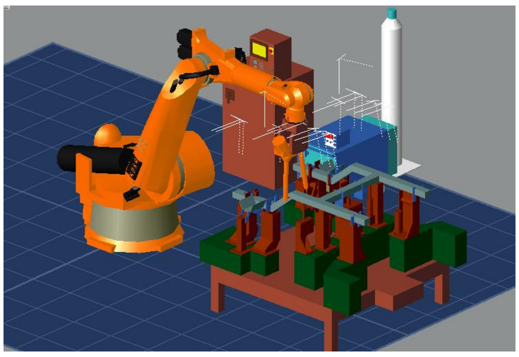

Figure 11: MTP problem SPOT WELDING with 1 start configuration, 16 sub goal configurations and 1 final goal configuration.

For each problem, we have solved the maximum number of planning runs (PIN ASSEMBLY: 231, SPOT WELDING: 153) ten times, in order to show the averaged performance of the different strategies (see Figure 12 and 13). It can be clearly seen, that in both problems, the randomized strategy **P0** shows the worst results.

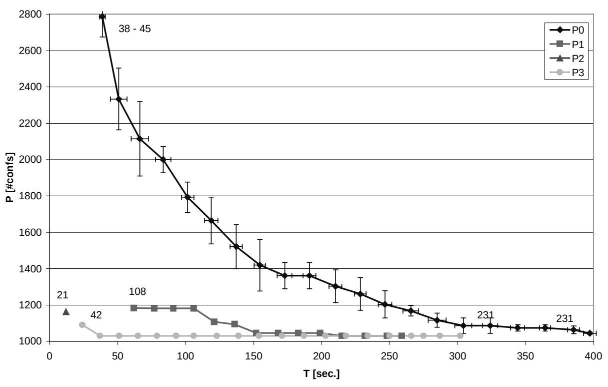

Figure 13: Total sequence length *P* depending on the running time *T* for the four different stratetgies **P0** to **P3** for the MTP problem PIN ASSEMBLY.

In the fastest run for the MTP problem PIN ASSEMBLY, the strategy **P0** is able to find a first sequence after 38 iterations, but with a total sequence length of P = 2786, which is far from the optimum with P = 1030. **P1** can make use of the distance information between the configurations as the first sequence has a total length of P = 1183. But as a lot of configuration are very close to each other, **P1** needs 108 iterations for finding this first solution. In contrast, the strategy **P3** computes the first solution already after 42 iterations (P = 1090) and the optimum after 50 iterations. The fastest results can be achieved with the strategy P2 after 21 iterations with a sequence length of P = 1164.

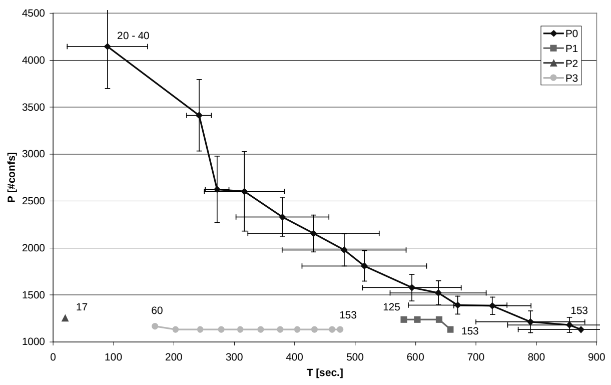

Figure 13: Total sequence length *P* depending on the running time *T* for the four different stratetgies **P0** to **P3** for the MTP problem SPOT WELDING.

In the second MTP problem SPOT WELDING the results are sligthly different. **P0** achieves still the worst results. But as the configurations can be divided into two groups (left and right bar), the strategies **P1** and **P3** calculates a lot of collision-free paths between the configurations of each group in comparison to **P0**. Thus, **P0** is per chance able to compute those collision-free paths, which are needed for planning a sequence. Therefore, **P0** can provide already after 20 iterations a first solution in its fastest run. But this solution with a total sequence length of $P = 4145$ is far away from the optimum ($P = 1132$). For **P1**, the clustering of the configurations leads to the fatal number of 125 iterations for providing the first solution. Additionally, the optimum can be found only after the maximum number of runs. In contrast, **P3** calculates its first sequence after 60 iterations with a length of $P = 1166$. The best performance shows strategy **P2**, with a sequence of $P = 1254$ after 17 planning runs.

7 CONCLUSIONS

In this paper, we have presented the new Multi-Goal Path Planning (MTP) approach as the first powerful support tool for human programmers to solve MTP problems in off-line programming tasks. We have introduced four different goal selection methods and compared them for two industrial MTP problems. Based on an efficient path planner in combination with Nearest Goal and Adaptive Pair Selection, the simplest path segment are solved first. With every additional path planning, the total sequence path length can be further reduced, thus the Multi-Goal Path Planning can be interrupted at any time, after a first sequence is available. With all other available path planners, the MTP problem can only be solved with Random Pair or Nearest Pair Selection. In order to find the shortest sequences, a new GA based xSP++-solver and a modified MTP graph connectivity test have been developed.

In our future work, we will investigate on an automatic strategy selection method, in order to additionally improve the total performance. Furthermore, we will combine the Nearest Goal and the Adaptive Pair Selection, because the first method can already provide a valid solution after the minimum number of planning runs. Although the Nearest Goal Selection cannot guarantee the optimal sequence, this first solution can be used as a basis for the Adaptive Pair Selection method.

8 REFERENCES

[1] Applegate, D.; Bixby, R.; Chvátal, V. and Cook, W.: „On the Solution of Traveling Salesman Problems“, Doc.Math.J.DMV Extra Volume ICM III, pp. 645-656, 1998.

[2] Baginski, B.: „Motion Planning for Manipulators with Many Degrees of Freedom - The BB-Method – “, Dissertation, Fakultät für Informatik, Technische Universität München, 1998.

[3] Chen P.; Hwang, Y.K.: „A Dynamic Graph Search Algorithm for Motion Planning“, IEEE Transaction on Robotics and Automation, vol. 14, 1998.

[4] Dijkstra, E.W.: „A note on two problems in connexion with graphs. Numer. Math. 1, pp. 269-271, 1959.

[5] Grötschel M.: “On the symmetric travelling salesman problem: Solution of a 120-city problem”, Math. Progr. Studies 12, pp. 61-77, 1980.

[6] Grötschel, M.; Holland, O.: “Solution of large-scale symmetric traveling salesman problems”, Mathematical Programming 51, pp. 141-202, 1991.

[7] Hart P.E., Nilsson N.J., Raphael B.: „A formal basis for the heuristic determination of minmum cost paths“, IEEE Trans. Syst. Sci. Cybern, pp. 100-107, 1968 -> A*, Ref. aus [Korf90].

[8] Henrich D., Cheng X., "Fast Distance Computation for On-line Collision Detection with Multi-Arm Robots", IEEE International Conference on Robotics and Automation, Nice, France, May 10.-15., pp. 2514-2519, 1992.

[9] Henrich D., Gontermann St., Wörn H.: „Schnelle Kollisionserkennung durch parallele Abstandsberechnung“ In: 13. Fachgespräch Autonome mobile Systeme (AMS´97), Stuttgart, 6. + 7. Oktober, Springer-Verlag, Reihe „Informatik Aktuell“, 1997.

[10] Henrich D., Wurll Ch., Woern H.: "6 DOF path planning in dynamic environments - A parallel on-line approach". In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA-98), Leuven, Belgium, May 16-21, 1998.

[11] Henrich D., Wurll Ch., Wörn H.: "On-line path planning with optimal C-space discretisation", In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98), Victoria, Canada, Oct. 12-16, 1998

[12] Henrich D., Wurll Ch., Woern H.: " Multi-directional search with goal switching for robot path planning ". In: The 11th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE'98), Castellan, Spain, July 1-4, 1998.

[13] Holland, J.H.: “Adaption in Natural and Artificial Systems”, The University of Michigan Press, Ann Arbor, 1975.

[14] Hwang Y. K., Ahuja N., "Gross motion planning – A survey", ACM Computing Surveys, vol 24, no 3, Sept. 1992.

[15] Jungnickel, D.: "Graphs, networks and algorithms”, Springer-Verlag, 1999

[16] Kamal L., Gupta K., del Pobil, A.P.: „Practical motion planning in robotics: Current approaches and future directions“, IEEE Robotics & Automation Magazine, Dec. 1996.

[17] Mehlhorn, K.; Näher, St.: “The LEDA Platform of Combinatorial and Geometric Computing”, to appear with Cambridge University Press, 1999. (see also http://www.mpi-sb.mpg.de/LEDA/)

[18] Padberg, M.; Rinaldi, G.: “Optimization of a 532-city symmetric traveling salesman problem by branch and cut”, Operations Research Letters 6, pp. 1-7, 1987.

[19] Pederson, Th.: “GATSS – A Genetic based solver of a Traveling Salesman problem”, http://www.acc.umn.se/~top/travel_information.html, 1995.