Proceedings of the 1997 IEEE
International Conference on Robotics and Automation
Albuquerque, New Mexico - April 1997

A Two-level Search Algorithm for Motion Planning

Pekka Isto
Laboratory of Information Processing Science
Helsinki University of Technology
Otakaari 1, FIN-02150, Espoo, Finland
evp@cs.hut.fi

Abstract

A two-level search algorithm for motion planning is presented in this paper. The algorithm combines a multiheuristic local search algorithm with a subgoal graph based global guidance. A novel feature of the planner is that it can adjust the balance between local and global planning. As the experimental data suggests that the optimal balance depends on the problem, a scheduling mechanism is added to the algorithm to adjust the balance during planning. The resulting motion planner is capable of solving very difficult motion planning problems.

1. Introduction

In the basic variation of motion planning, the task is to generate a collision-free path for a movable object among known and static obstacles. A considerable amount of research has been conducted during the last two decades, but because of the hardness of the problem, no single overall method for motion planning is available. Therefore, research in robot motion planning remains as one of the important fields of study in the task of building autonomous or semi-autonomous robot systems. One of the most successful algorithms is the Randomized Path Planner (RPP) [1] which has been used in several applications [2,3]. However, the randomized technique that RPP uses to escape local minimum wells is known to work poorly in certain cases [4].

An alternative way to deal with local minimum wells is to decompose the problem to subproblems that avoid deep local minimum wells and solve those subproblems. The motion planning algorithm presented in this paper uses the subgoal graph approach to decompose the problem and an efficient heuristic search algorithm to solve a sufficient set of the subproblems. The algorithm operates on two levels. As more subgoals are generated, the global planner increases the capability of the local planner by adjusting the balance between local and global planning.

Several two-level algorithms have been presented in the literature before, but the algorithm described in this paper is unique, since the local method can be very powerful. Furthermore, the balance between the local and the global planner is easy to modify by adjusting a single parameter. One of motivations for this research was to study whether one should have a weak or a powerful local planner in a two-level algorithm. As the results indicate that the optimal balance depends on the problem, a scheduling mechanism was added to the global planner to adjust the balance during planning.

The following section will review some of the previous work on motion planning. There is a rather large body of research reported in the literature and good surveys are available [5,6]. Therefore, the following review is restricted to topics that are related to the presented algorithm. Sections 3 and 4 discuss the local and global planners respectively, and section 5 will present experimental results obtained with an implementation of the algorithm. The final section presents the conclusions.

2. Previous Related Work

Like most of the current algorithms for motion planning, the presented algorithm is formulated in the configuration space of the robot. Configuration space (*C-space*) [7] is the set of values for the parameters that define the position of every point of the robot. For manipulators, the values of the joints of the robot usually define a configuration. The benefit of formulating the problem in the *C-space* is that the problem becomes equivalent to navigating a point object. This means that the specifics of the robot kinematics, the robot geometry and the geometry of the environment have to be taken into consideration only indirectly through collision checking.

The local planner of the algorithm is based on cell decomposition approach [5]. The idea is to calculate a decomposition of the *C-space* and search it for a connected set of collision-free regions from the start configuration to the goal configuration [8]. To conserve memory, a hierarchical representation of the decomposition is often preferred [9]. If the robot Jacobian is available, it is possible to compute blocks of free *C-space* and search a 2^d-tree representation of the free *C-space* for a solution [10]. Because of the size of

the *C-space*, computing the complete *C-space* decomposition becomes infeasible for robots that have more that four degrees-of-freedom (DOF). In such a case some kind of heuristic approach is needed. Kondo presented an efficient heuristic search method for searching a rectangular grid representation of the *C-space* for a solution [11]. The local planner of the algorithm presented here is based on an improved version of Kondo's algorithm [12].

The global planner of the presented algorithm is based on the subgoal graph or landmark approach. The idea is to place subgoal or landmark configurations in the free *C-space* of the robot. A local planner is then used to connect the start, the goal and the subgoal configurations until a solution to the original problem is found. The efficiency of the approach comes from the fact that the original problem can often be decomposed to a set of subproblems that are more easily solvable than the original problem. If the local planner succeeds to connect start and goal configurations via the subgoals, the subpaths obtained with the local planner can be combined to yield a solution to the original problem. However, often an additional postprocessing or smoothening algorithm needs to be run on the combined solution, especially if a randomized algorithm was used to produce it.

These motion planning algorithms can be further divided into two types. One-shot planners do not store information from one run to the next. Learning and preprocessing motion planners try to capture the connectivity of the *C-space* and utilize that knowledge to improve the efficiency of the subsequent planning tasks. Faverjon and Tournassoud introduced the idea of two-level approach [13] and presented a learning algorithm to capture the connectivity of the *C-space* in the form of transition probabilities [14].

A one-shot algorithm developed by Glavina combines subgoal graph approach with a simple sliding local planner in the *C-space* [15, 16]. The subgoals are randomly generated. Since the local planner is rather weak and there is no heuristics for the subgoal placement, it is likely that this algorithm will have difficulties with problem instances that involve passing through narrow passages. Chen and Hwang have presented motion planners for manipulators and rigid objects [17,18]. The planners are based on a search strategy they call SANDROS. SANDROS strategy is a two-level, non-uniform search technique that uses a simple local planner to connect heuristically generated subgoal sequences. They claim that a significant improvement over their algorithm is only possible with radically different approaches like parallel processing or knowledge-based algorithms [17,6]. Baginski's algorithm uses a local planner based on a shrink measure to connect randomly generated subgoals [19]. The shrink measure is obtained by shrinking the colliding links of the robot, and it is used to modify the path to make it collision-free. The algorithm is currently defined only for robots that are linear chains of convex parts.

Since the subproblems are independent, they can be solved simultaneously with a parallel computer. Bessiére *et. al.* have presented a motion planner that is based on genetic algorithms [20]. One algorithm is used to place landmarks evenly in the *C-space*, and the other algorithm is used to connect these landmarks. A parallel computer built from 128 transputers was used to utilize several levels of parallelism in the motion planner. Another parallel algorithm is presented by Qin and Henrich [21]. They use a network of workstations to run copies of a rule-based local planner. Again, these local planners are used to connect randomly generated subgoals until a solution is found.

Preprocessing algorithms perform a learning phase to capture the connectivity of the *C-space* in the form of a roadmap [4,22,23]. Once the roadmap is built, motion planning tasks can be solved rapidly by connecting the start and goal configurations to the roadmap and searching the roadmap for a solution. While one-shot planners typically use a moderate number of subgoals, preprocessing algorithms use thousands of subgoals to build the roadmap. The subgoals are generated randomly, but heuristically placed subgoals may be used to better capture the connectivity [4,22]. The local planners are rather simple, since the actual paths between the nodes of the roadmap graph are not stored but are reproduced with the local planner as needed. A more powerful local planner can be used to connect components of the roadmap that the simple local planner fails to connect [4,23]. Preprocessing algorithms can be very efficient, if the cost of building the roadmap can be amortized over many planning tasks.

3. The Local Search Algorithm

The local planner of the motion planner presented in this paper is an improved version of the A^* based motion planner presented by Kondo. The algorithm uses several heuristics to guide searching in a grid representation of the *C-space*. The search algorithm generates a portion of the grid and stores the configurations as nodes and the adjacency relationships between those configurations as links between the nodes. If the start and the goal configurations become connected in the collision-free part of the representation, a solution to the problem is found. As the heuristics are executed they are also

evaluated for efficiency and more efficient heuristics guide the search more than the less efficient ones.

Kondo's algorithm is capable of solving many problems, but there is room for several improvements [12]. In order to reduce the number of collision checks performed during searching only those configurations that are considered to become a part of the solution are checked. This means that not all of the nodes that are generated are tested, but only those nodes that are further expanded by generating the neighbouring configurations.

The heuristics are executed sequentially in a round-robin fashion. After the j th round, an evaluation value

$$P_t(j) = \frac{\sum_{(for\ \ last\ \ Q\ \ nodes)} p_t(C)}{Q}$$

is calculated for each heuristics $t = 1,...,T$, where T is the number of heuristics and a constant Q has the value of *20*. The value of $p_t(C)$ is calculated for each opened node C by the following equation:

$$p_t(C) = \frac{g(C)^{DOF}}{F_t(C)},$$

where $g(C)$ is the distance from the start node to the current node C in grid steps and $F_t(C)$ is the total number of nodes opened by the tth heuristics until the expansion of node C.

For the first round, each heuristics is allocated $E_{init} = 25$ node expansions. For the subsequent rounds each heuristics is allocated

$$E_t(j+1) = \max\left[E_{init} \frac{P_t(j)}{\max\left[P_1(j),...,P_T(j)\right]}, 1 \right]$$

node expansions. The outer maximum operation guarantees that at least one expansion is always allocated.

A second evaluation value is calculated for each opened node according to the equation:

$$O_t(C) = \frac{\sum F_t(C)}{g(C)}.$$

If the evaluation value $O_t(C)$ for the tth heuristics decreases below a threshold value O_{th}, the execution of the heuristics is discontinued for that round. However, the discontinued heuristics may become active again in some later round of the search, if one of the other heuristics expands nodes that move the discontinued heuristics to a better area in the search space. If all heuristics are discontinued, the local planner fails.

The search is bi-directional. After each round, the search direction with the smaller OPEN set is selected for the next round [24]. The effect is that the search proceeds from the cluttered space to the open space.

The heuristic evaluation of a node is performed with the function $f(C) = g(C) + h(C)$, where $g(C)$ was given above and $h(C)$ is weighted manhattan distance between the evaluated configuration and the goal configuration. The following four weight sets are used to define four heuristics.

Manipulator heuristics:

$$a_i = \begin{cases} 9-i, & i = 1,...,7 \\ 1, & i = 8,...,DOF \end{cases}$$

Position heuristics:

$$a_i = \begin{cases} 9, & i = 1,...,d \\ 1, & i = d+1,...,DOF \end{cases}$$

Rotation heuristics:

$$a_i = \begin{cases} 1, & i = 1,...,d \\ 9, & i = d+1,...,DOF \end{cases}$$

Even heuristics:

$$a_i = 5, \quad i = 1,...,DOF.$$

$d = \lfloor (DOF + 0.5)/2 \rfloor$.

This set of weights was previously tested with several devices of different kinematic structure [12]. The results indicated that the combination has a better performance than any one of the heuristics alone. This set exhibited also better performance than the average performance of randomly generated weight sets.

An additional multiplier term $A=3$ and a tie-breaking term ρ are added to the full expression of $h(C)$:

$$h(C) = A\left(\sum_{i=1}^{DOF} a_i D_i(C,G) - \rho a_j \right),$$

$D_i(C,G)$ is the distance between the current node and the goal node G in grid steps along axis i. The tie-breaking term has a value of 0.5 if the evaluated node was expanded in the same direction as the parent node along the axis j or 0 otherwise.

4. The Global Planner

The local planner is a resolution complete motion planner as it will eventually expand the whole *C-space* and find a solution, if one exists. The local planner is efficient enough to be used alone in many cases, but it is rather susceptible to local minimum wells. If the local planner encounters a region of the search space where all of the heuristics guide it to a dead-end, it has to examine and test for collisions all of the configurations residing in the region. This can be computationally very costly and in fact renders the algorithm impractical for more difficult problems.

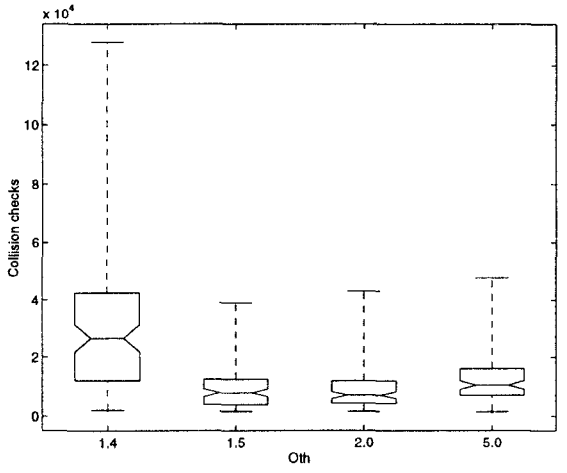


Figure 1: Box plots of number of performed collision checks on various O_{th} values on the task of figure 3. Note the non-uniform distribution of O_{th} values.

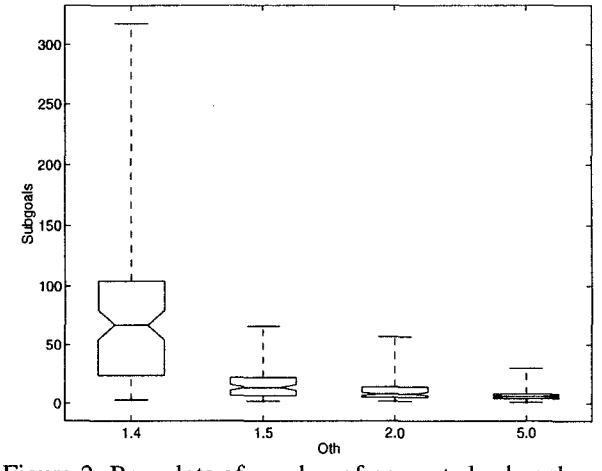


Figure 2: Box plots of number of generated subgoals on various O_{th} values on the task of figure 3.

A related problem is that the local planner can consume excessive amounts of memory to represent the *C-space* for the underlying A^* search algorithm. Actually, the amount of memory needed sets an upper limit for the difficulty of the problems that the local planner alone can practically deal with. The test problems presented in this paper are not practically solvable by multiheuristic search only, since the search algorithm will exhaust the main memory of the workstation. The resulting swapping effectively stops the search algorithm before a solution is found.

A solution is to use the A^* based search algorithm as a lower component of a two-level motion planner and set a limit on how hard subproblems will be solved with the search algorithm. The upper or global component of the two-level planner decomposes the original problem to subproblems and uses the local planner to solve them until a solution to the original problem can be composed. As the global planner can gather information over many runs of the local planner during the planning, it can adjust the behaviour of the local planner to the problem being solved. The difficulty of the subproblems the local planner is trying to solve is estimated by the minimum of $O_t(C)$ values.

There are conflicting views whether one should use a weak [22] or a powerful [19] method as the local planner. The presented algorithm is flexible in this respect, since the capability of the local planner can be controlled by the parameter O_{th}. While it is difficult to select an *a priori* value for O_{th}, it is possible to set a schedule for it. Each time a subgoal is generated the value of O_{th} is updated according to equation:

$$O_{th} = O_{th}^{'} R^{N_s} .$$

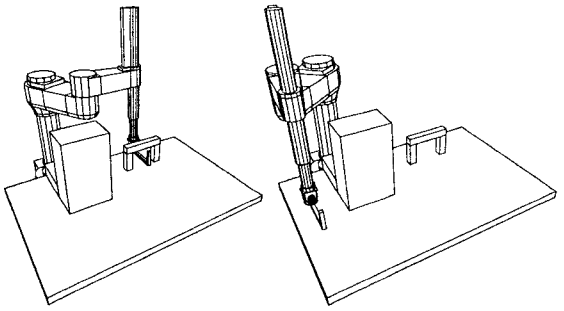
N_s is the current number of subgoals. $O_{th}^{'}$ and R are constants. The rationale is that if many subgoals are needed, the local planner is too weak and should be made more powerful.

In addition to the direct search from the start configuration to the goal configuration, the global planner tries to connect the start and goal configurations to the subgoal configurations. The connections between subgoal configurations are searched only for pairs of start connected and goal connected subgoals. This means that there can be up to two subgoals in the solution. The subgoals are generated randomly.

5. Experimental Results

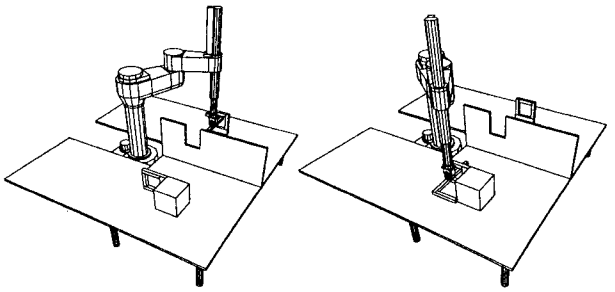
Two sets of experiments are performed to characterize the performance of the motion planner. First, motion planning for the test problems is performed with several different values of O_{th} in order to study the effect of the balance between local and global planning. The number of executed collision checks is taken as the measure of performance. Once the values for $O_{th}^{'}$ and R are determined, a second set of motion planning experiments is performed to qualify the efficiency of the planner. Since the algorithm is randomized, a distribution of results is produced for each problem.

The tests are run on a SGI Indigo^2 workstation with a 250 MHz R4400 processor and 128 MB of main memory. TELEGRIP software is used for object modeling and collision checking [25]. The reported times are CPU times on this system. Each joint is evenly discretized to have 100 positions. The random number streams for subgoal generation are synchronized so that the same sequences of subgoals are produced for the runs with different value of O_{th}.



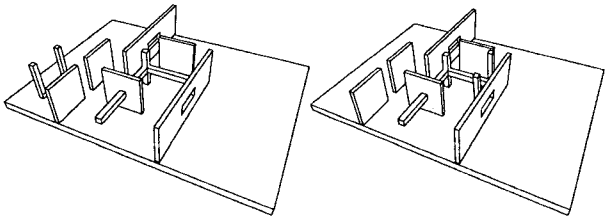
(a) (b)

Figure 3: A version of the benchmark problem proposed by Hwang and Ahuja [6].



(a) (b)

Figure 4: A hard motion planning problem for a SCARA robot. The robot has to bend the wrist to get the payload through the opening in the wall or between the wall and the robot body.



(a) (b)

Figure 5: A very hard motion planning problem for a 6 DOF rigid body.

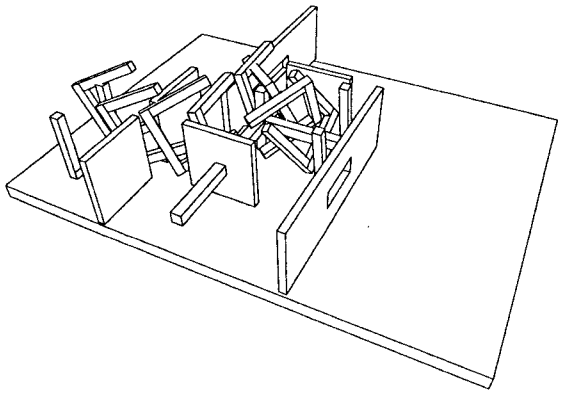


Figure 6: A solution for the problem in figure 5.

The first problem case is similar to the benchmark problem proposed by Hwang and Ahuja [6]. The SCARA-type robot has 5 degrees-of-freedom. Figure 3 shows the start and goal configurations of the robot. Figure 4 shows the second test problem. The task is more difficult because the robot has to go through the small passage in the wall. Since the passage is in a flat wall, there are no obstacles in the workcell guiding the robot towards the passage and a considerable amount of searching is needed in order to find the passage. The third test case is a task for a 6 DOF rigid body. The task is very difficult since the U-shaped body can pass through the passages only in certain orientations. Furthermore, the U-shaped body gets easily trapped by the pipe-like construction in the model.

The results of the first experiment are presented in the figures 1 and 2 in as MATLAB box plots [26] of a sample of 100 runs. The lower and upper lines of the box of the plot are the 25th and 75th percentile of the data. The line in the middle of the box is the median of the data. The lines extending above and below the box give the range of the data. The "notches" in the box show the 95% confidence interval about the mean of the data.

O_{th}	1.9	2	5	10	15
CC	144550	135470	82751	95024	76716
SG	180	141	32	16	10

Table 1: The average numbers of performed collision checks and generated subgoals on various O_{th} values on the task of figure 4.

O_{th}	3	5	10	15	20
CC	543038	261189	252048	261103	356191
SG	245	70	29	22	22

Table 2: The average numbers of performed collision checks and generated subgoals on various O_{th} values on the task of figure 5.

As seen in the figure 1, the cost of planning decreases slowly as the O_{th} value is decreased from 5 to 1.5. However, near the value of 1.4 the local planner becomes too weak and the cost increases sharply. The number of generated subgoals (figure 2) parallels the distribution of work between the local and global planner. As the local planner is made weaker, more subgoals are needed. When the local planner becomes too weak, a substantial increase in the number of generated subgoals is observed. As seen in the tables 1 and 2, a similar pattern of behaviour was also observed with the other problem cases. One can also observe that the optimal balance between local and global planning is

different for the tasks. The averages in tables 1 and 2 are calculated for 50 runs.

The experimental data suggests that a too weak local planner has a much more detrimental effect on the efficiency of the motion planner than a too powerful local planner. Based on the results, the value of 2 was selected for $O_{th}^{'}$, and the constant R was given the value of 1.05.

Task	25%	50%	75%
3a→3b	5136	8370	12037
4a→4b	31917	51882	94001
5a→5b	106433	172138	412630

Table 3: Percentiles of number of performed collision checks.

Task	25%	50%	75%
3a→3b	16	26	37
4a→4b	105	158	262
5a→5b	260	408	895

Table 4: Percentiles of running time in CPU seconds.

The results of the second set of experiments are presented in tables 3 and 4. The performance is characterized by presenting 25th percentile, median and 75th percentile of a sample of 100 runs. The table 3 gives the percentiles of performed collision checks, and table 4 gives the percentiles of running times. As the test problems are very difficult, the performance of the motion planner is satisfactory.

The test results demonstrate the ability of the algorithm to solve difficult motion planning tasks for robots with up to 6 degrees-of-freedom. The suitability of the algorithm for problems involving robots with many degrees-of-freedom remains open. The memory consumption of the local planner grows exponentially as the number of degrees-of-freedom is increased. The actual memory requirement can be somewhat controlled by the parameter O_{th}, but with the risk of making the local planner too weak. More experiments are needed to evaluate if the algorithm is suitable for practical problems involving highly redundant robots or multiple manipulators.

6. Conclusions

An efficient motion planning algorithm was presented in this paper. The motion planner succeeds by using the most efficient heuristics to solve the easier subproblems until a solution to the original problem is found. Instead of using some fixed search strategy, the motion planner dynamically adjusts many of its parameters to the problem at hand.

While the presented algorithm has good performance, several enhancements are possible. Currently, the order in which the global planner examines the subgoals is arbitrary, but a heuristics could be used to select the more promising subgoals first. The random generation of subgoals is satisfactory for many cases, but it has the drawback of introducing random behavior to the algorithm. Each run of the motion planner produces a different solution to the problem and requires different amount of computation to produce the solution. A trivial solution to eliminate the variation is to use the same sequence of subgoals for every run of the motion planner. The search algorithm would then select the best subgoals for the problem at hand. Because of the potentially very powerful local planner, the quality of the subgoals is not crucial for good performance. Therefore, this approach should perform well for most cases.

Acknowledgments

Mr. Johannes Lehtinen provided the model in figure 3.

References

[1] J. Barraquand, J.C. Latombe, Robot Motion Planning: A Distributed Representation Approach, The International Journal of Robotics Research 10, no. 6, Dec. 1991, 628-649.

[2] Y. Koga, K. Kondo, J. Kuffner, J.L. Latombe, Planning Motions with Intentions, Proceedings of SIGGRAPH 94, ACM SIGGRAPH, 1994, 395-408.

[3] H. Chang, T.Y. Li, Assembly Maintainability Study with Motion Planning, Proceedings of the 1995 IEEE International Conference on Robotics and Automation, IEEE Press, 1995, 1012-1019.

[4] L. Kavraki, J.C. Latombe, Randomized Preprocessing of Configuration Space for Fast Path Planning, Proceeding of the 1994 IEEE International Conference on Robotics and Automation, IEEE Press, Los Alamitos, 1994, 2138-2145.

[5] J.C. Latombe, Robot Motion Planning. Kluwer Academic Publishers, Norwell 1991.

[6] Y. K. Hwang, N. Ahuja, Gross Motion Planning - A Survey. ACM Computing Surveys, Vol. 24, No. 3, September 1992, 219-291.

[7] T. Lozano-Pérez, M. Wesley, An Algorithm for Planning Collision-free Paths among Polyhedral Obstacles. Communications of the ACM, Vol. 22, No. 10, October 1979, 560-570.

[8] T. Lozano-Pérez, Automatic Planning of Manipulator Transfer Movements, IEEE Transactions

on Systems, Man and Cybernetics, Vol. 11, No. 10, 1981, 681-698.

[9] R. A. Brooks, T. Lozano-Pérez, A Subdivision Algorithm in Configuration Space for Findpath with Rotation, Proceedings of the 8th International Conference on Artificial Intelligence, Kaufmann, Los Altos 1983, 799-806.

[10] B. Paden, A. Mees, Path Planning Using a Jacobian-Based Freespace Generation Algorithm, Proceeding of the 1989 IEEE International Conference on Robotics and Automation, IEEE Press, 1989, 1732-1737.

[11] K. Kondo, Motion Planning with Six Degrees of Freedom by Multistrategic Bidirectional Heuristic Free-Space Enumeration. IEEE Transactions on Robotics and Automation, Vol. 7, No. 3, June 1991, 267-277.

[12] P. Isto, Path Planning by Multiheuristic Search via Subgoals, Proceedings of the 27th International Symposium on Industrial Robots, CEU, Milan 1996, 721-726.

[13] B. Faverjon, P. Tournassoud, A Local Based Approach for Path Planning of Manipulators with a High Number of Degrees of Freedom. Proceedings of the 1987 IEEE International Conference on Robotics and Automation, IEEE Press, Washington 1987, 1152-1159.

[14] B. Faverjon, P. Tournassoud, A Practical Approach to Motion-Planning for Manipulators with Many Degrees of Freedom. Robotics Research 5, H Miura, S. Arimoto (Eds.), MIT Press, Cambridge, MA 1990, 425-433.

[15] B. Glavina, Solving Findpath by Combination of Goal-Directed and Randomized Search. Proceeding of the 1990 IEEE International Conference on Robotics and Automation, IEEE Press, Piscataway 1990, 1718-1723.

[16] B. Glavina, A Fast Motion Planner for 6-DOF Manipulators in 3-D Environments. Proceedings of the Fifth International Conference on Advanced Robotics, IEEE Press, 1991, 1176-1181.

[17] P. C. Chen, Y.K. Hwang, SANDROS: A Motion Planner with Performance Proportional to Task Difficulty, Proceedings of the 1992 IEEE International Conference on Robotics and Automation, IEEE Press, Los Alamitos 1992, 2346-2353.

[18] Y.K. Hwang, P. C. Chen, A Heuristic and Complete Planner for the Classical Mover's Problem, Proceedings of the 1995 IEEE International Conference on Robotics and Automation, IEEE Press, Los Alamitos 1995, 729-736.

[19] B. Baginski, Local Motion Planning for Manipulators Based on Shrinking and Growing Geometry Models. Proceeding of the 1996 IEEE International Conference on Robotics and Automation, IEEE Press, Piscataway 1996, 3303-3308.

[20] P. Bessiére, J.M. Ahuactzin, E.G. Talbi, E. Mazer, The "Adriane's Clew" Algorithm: Global Planning with Local Methods, Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE Press, 1993, 1373-1380.

[21] C. Qin, D. Henrich, Randomized Parallel Motion Planning for Robot Manipulators, Technical Report 5/96, Computer Science Department, University of Karlsruhe, 1996.

[22] M. H. Overmars, P. Svestka, A Probabilistic Learning Approach to Motion Planning, Technical Report UU-CS-1994-03, Department of Computer Science, Utrecht University, The Netherlands, January 1994.

[23] Th. Horsch, F. Schwarz, H. Tolle, Motion Planning with Many Degrees of Freedom - Random Reflections at C-Space Obstacles, Proceeding of the 1994 IEEE International Conference on Robotics and Automation, IEEE Press, Los Alamitos, 1994, 3318-3323.

[24] I. Pohl, Bi-directional Search, Machine Intelligence 6, Edinburgh University Press, Edinburgh, 1971, 127-140.

[25] TELEGRIP User Manual. Deneb Robotics Inc. 1994.

[26] B. Jones, Statistics Toolbox User's Guide, The MathWorks Inc, Natick, 1994.