Towards Learning Path Planning for Solving Complex Robot Tasks

Thomas Frontzek, Thomas Navin Lal, Rolf Eckmiller

Department of Computer Science VI, University of Bonn
Römerstraße 164, D – 53117 Bonn, F. R. Germany
E-mail: {frontzek,lal,eckmiller}@nero.uni-bonn.de

Abstract. For solving complex robot tasks it is necessary to incorporate path planning methods that are able to operate within different high-dimensional configuration spaces containing an unknown number of obstacles. Based on Advanced A*-algorithm (AA*) using expansion matrices instead of a simple expansion logic we propose a further improvement of AA* enabling the capability to learn directly from sample planning tasks. This is done by inserting weights into the expansion matrix which are modified according to a special learning rule. For an examplary planning task we show that Adaptive AA* learns movement vectors which allow larger movements than the initial ones into well-defined directions of the configuration space. Compared to standard approaches planning times are clearly reduced.

1 Introduction

The A*-algorithm is an established method for planning tasks. In the literature one can find theoretical studies, mainly dated from the late 60s to the early 80s ([6][5][8]), as well as lots of variations of A*, dated from the 80s until now ([7][10][3][1][4][9]). Most of the applications focus on computing a (slightly suboptimal) path as quickly as possible for limited types of configuration spaces.

In contrast to these approaches we conserve the optimality of computed paths. Furthermore by introducing expansion matrices we make A* more flexible for planning within different types of configuration spaces, and by adding weights into an expansion matrix we enable the capability of learning from sample planning tasks. These capabilities are absolutely needed to implement a flexible and robust tool for complex planning applications, e.g. for path searching in robot control, and for avoidance of self collision and collisions with obstacles in manipulator control.

2 Learning Path Planning with Adaptive AA*-Algorithm

Based on standard A* we explain our concept of Adaptive Advanced A*-algorithm for learning path planning. First, we modify the collision check by abstracting from concrete cell extents. Then we introduce expansion matrices replacing the simple expansion logic of A*. At last we further improve the capabilities of expansion matrices by making them adaptive using additional weights and a special learning rule.

2.1 Standard A*-Algorithm

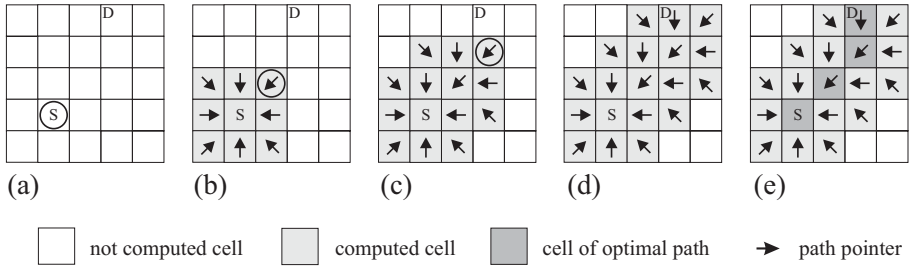

Fig. 1. Proceeding of standard A* from start cell S to goal cell D. First, S is expanded to its adjacent cells (a) and path pointers from these cells to S are set (b). Until reaching D the cell of current minimum costs is expanded (marked with a circle). Finally, an optimal path P_{opt} linked from D to S is computed (e).

Applying standard A*-algorithm the configuration space representing the actual world model is subdivided into entities called *cells* which usually have the same size. A* searches for an optimal path P_{opt} through the free space from a given start cell S (start) to a given goal cell D (destination). For every considered cell C the costs $f(C,S,D) = g(S,C) + h(C,D)$ are computed, with $g(S,C)$ as costs for the movement from start cell S to the actually considered cell C and $h(C,D)$ as heuristic estimation of the costs from cell C to the goal cell D. Additionally, for every considered cell a check is performed whether the movement to this cell is permitted or not. In real-world scenarios a collision check between the object to be moved and some obstacles has to be performed. Fig.1 visualizes the expansion logic and construction of path pointers of A*.

2.2 Collision check

Abstracting from concrete cell extents we consider a reference point r_C for each cell C, forming a grid within the configuration space. In addition, a reference point r_O for the object O to be moved is determined. For the collision check first r_O is mapped to the actually considered r_C; then the collision check with respect to the actually used configuration space is computed (fig.2). The underlying world model is only necessary for computing the collision check. Hence, the expansion logic of A* is completely detached from the world model.

2.3 Expansion matrices

We replaced the expansion logic of standard A* by our novel concept of expansion matrices, forming the Advanced A*-algorithm (AA*) ([4]). This was designed to overcome the limitations of equally sized cells resp. equidistant reference points and the restriction of symmetric path costs.

An *expansion matrix*:

$$\mathcal{E} := \begin{pmatrix} \Delta x_{11} & \cdots & \Delta x_{1k} & c_1 \\ \vdots & & \vdots & \vdots \\ \Delta x_{m1} & \cdots & \Delta x_{mk} & c_m \end{pmatrix}$$

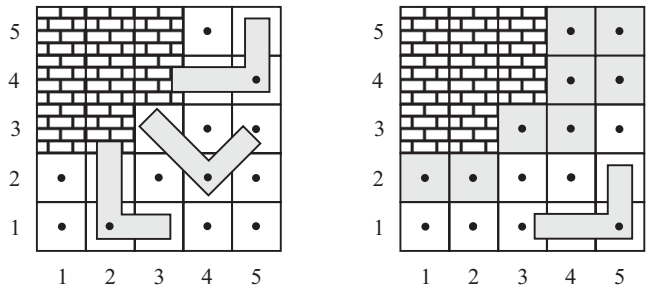

Fig. 2. Collision check for geometric objects considering reference points. The left figure shows an "**L**"-shaped object having orientations of 0°, 45° and 90°. For example cell $(4, 2, 45°)$ is not occupied, cells $(2, 1, 0°)$ and $(5, 4, 90°)$ are defined to be occupied by an obstacle, because at these positions the object intersects an obstacle. The right figure visualizes all cells defined to be occupied by an obstacle (stonewalled and grey cells) given an object orientation of 90°.

is an $m \times n$-matrix where m denotes the number of expansion steps and $n = k+1$ with k denoting the number of dimensions of the configuration space $\mathcal{C}$. $c_i \in \mathbb{R}^+$ indicates the costs of expansion step i, and $\Delta x_{ij} \in \mathbb{R}$ indicates the movement difference within dimension j during the expansion step i.

Four parameters of an expansion matrix $\mathcal{E}$ have to be determined:

- *the number of rows*
- *the expansion sequence by permuting rows of* $\mathcal{E}$
- *the length and direction of each movement vector* $(\Delta x_{i1}, \cdots, \Delta x_{ik})$ *for every row* i *of* $\mathcal{E}$
- *the costs* c_i *for every expansion step to model symmetric as well as asymmetric costs*

Starting a new AA* planning process an expansion matrix $\mathcal{E}_{init}$ is initialized appropriating knowledge about the world model. During a single planning process this expansion matrix remains unchanged. With this concept a variety of complex environments and difficult planning tasks can easily be modeled, like different cell extensions resp. distances of reference points or the effect of flow (for example movements with the flow cost nothing, whereas movements against the flow are very expensive).

2.4 Adaptive expansion matrices

By adding a weight w_{ij} for every entry of an expansion matrix $\mathcal{E}$ we further improve the capabilities of our concept.

An *adaptive expansion matrix*:

$$\mathcal{A} := \begin{pmatrix} w_{11} \cdot \Delta x_{11} & \cdots & w_{1k} \cdot \Delta x_{1k} & w_{1n} \cdot c_1 \\ \vdots & & \vdots & \vdots \\ w_{m1} \cdot \Delta x_{m1} & \cdots & w_{mk} \cdot \Delta x_{mk} & w_{mn} \cdot c_m \end{pmatrix}$$

is an $m \times n$-matrix using the same notations as $\mathcal{E}$ (see section 2.3). $w_{ij} \in \mathbb{R}^+$ indicates the weight for an entry of $\mathcal{A}$.

Separating a row i of $\mathcal{A}$ we get the *actual movement vector* $\boldsymbol{v}_{act}$ of an expansion matrix:

$$\boldsymbol{v}_{act} := \mathcal{A}^T \cdot \boldsymbol{e}_i \quad with\ \boldsymbol{e}_i\ unit\ vector\ i$$

$\boldsymbol{v}_{act}$ contains the weights $w_{i1}, ..., w_{in} =: \boldsymbol{w}_{act}$.

Inserting $\mathcal{A}$ instead of $\mathcal{E}$ into the expansion logic of AA* the capability of learning from sample planning tasks is enabled by modifying w_{ij}.

At the beginning of a planning process all weights w_{ij} of the expansion matrix $\mathcal{A}$ are initialized with 1.0. Throughout the planning process for every expansion step AA* gets row by row $\boldsymbol{v}_{act}$ of $\mathcal{A}$ describing the movement from the actual considered reference point r_C to a next, possibly new one. After the planning process the computed optimal path P_{opt} is analyzed: Depending on the contribution of the respective $\boldsymbol{v}_{act}$ forming P_{opt} the corresponding weight vector $\boldsymbol{w}_{act}$ is increased by a constant vector. This is done with the aim to strengthen successful movements that is to say movements used forming P_{opt}.

During a single planning process $\mathcal{A}$ stays fixed. After terminating the actual planning process $\mathcal{A}$ is modified taking into account all changes for every weight vector, forming $\mathcal{A}'$. Then $\mathcal{A}'$ is used for subsequent planning processes.

Another variant of our concept, inspired by the paradigm of evolutionary algorithms, is pruning within the adaptive expansion matrix: if a weight vector falls below a certain threshold Φ, the corresponding row of the expansion matrix is substituted by a new, randomly initialized one.

3 Results

We embedded our novel Adaptive Advanced A*-algorithm into a simulation environment which allows us to plan within n-dimensional configuration spaces. First simulations, available for $3D$-configuration spaces ($2D$-position and $1D$-orientation) as well as for $6D$-configuration spaces (6 joint angles of an industrial robot), show encouraging results.

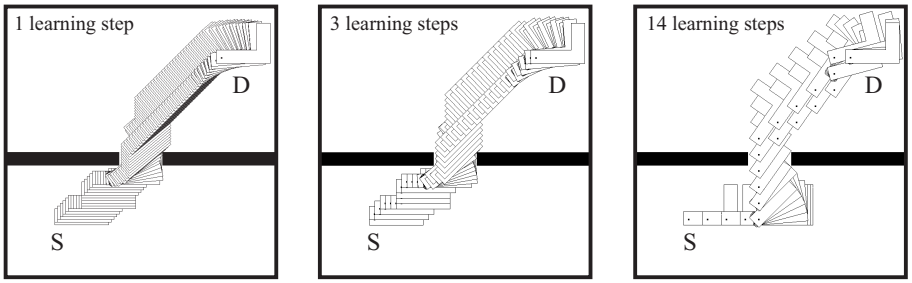

Fig. 3. Visualization of optimal paths computed by Adaptive AA* after one (left), three (middle), and 14 (right) learning steps. The planning task was to move an "**L**"-shaped object from S to D through a bottleneck. The length of the computed optimal path is 133 (left), 77 (middle), and 33 (right) reference points.

For an exemplary $3D$-planning task, moving an "**L**"-shaped object through a bottleneck, several iterations of our learning procedure were computed (fig. 3). Initialized with a standard expansion matrix

$$\mathcal{A}_0 = \begin{pmatrix} -1.0 & 0.0 & 0.0 & 1.0 \\ +1.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & -1.0 & 0.0 & 1.0 \\ 0.0 & +1.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & -1.0 & 1.0 \\ 0.0 & 0.0 & +1.0 & 1.0 \end{pmatrix}$$

and all weights w_{ij} set to 1.0 Adaptive AA* gets the same training planning task for every iteration together with the last updated expansion matrix. Output of the algorithm is an optimal path if one exists, and a modified expansion matrix. Column 1 of $\mathcal{A}_0$ describes the movement along dimension 1 (positions *left* and *right*), column 2 along dimension 2 (positions *down* and *up*), column 3 along dimension 3 (in 5°-steps; orientations *counterclockwise* and *clockwise*); every movement has the same costs 1.0 (column 4 of $\mathcal{A}_0$).

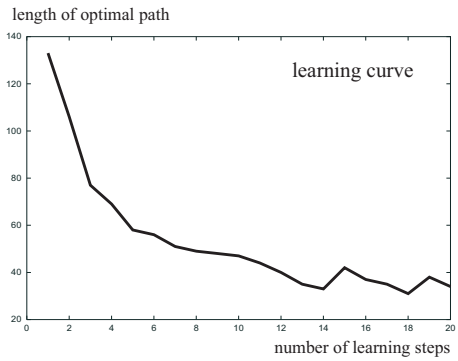

Fig. 4. Learning progress for our sample planning task (see fig. 3). Measure for the proceeding of learning is the length of P_{opt} resp. the number of reference points forming P_{opt}, which decreases from 133 to 33 reference points after 14 learning steps. Steps 15 to 20 generate unsteady results, so in this example the best result is achieved using the expansion matrix learned after step 14.

Figure 4 shows the learning progress for our example. The length of the encountered optimal path decreases from 133 reference points using $\mathcal{A}_0$ to 33 reference points after 14 learning steps; the following steps > 14 generate unsteady results. After 14 learning steps the expansion matrix

$$\mathcal{A}_{14} = \begin{pmatrix} -1.0 & 0.0 & 0.0 & 1.0 \\ +7.0 & 0.0 & 0.0 & 7.0 \\ 0.0 & -1.0 & 0.0 & 1.0 \\ 0.0 & +6.0 & 0.0 & 6.0 \\ 0.0 & 0.0 & -2.0 & 2.0 \\ 0.0 & 0.0 & +2.0 & 2.0 \end{pmatrix}$$

was learned by Adaptive AA*. $\mathcal{A}_{14}$ shows that a better partition of the configuration space than the initial one exists: one can solve the planning task

moving clearly larger steps *right* and *up*, and moving slightly larger steps *counterclockwise* and *clockwise*. Figure 5 shows the computed optimal paths within the $3D$-configuration space in detail.

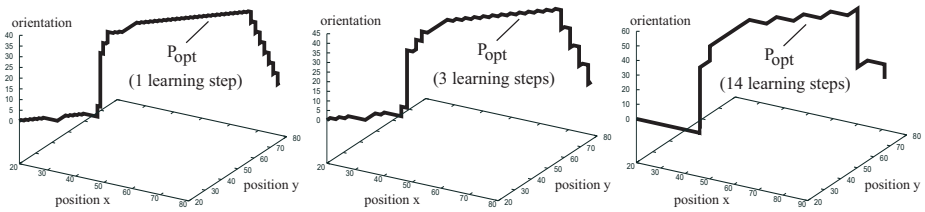

Fig. 5. Visualization of P_{opt} within the $3D$-configuration space (dimensions: position x, position y, orientation) after one (left), three (middle), and 14 (right) learning steps.

Comparison with breadth-first-search and standard A* shows the improvements of learning path planning performed by Adaptive Advanced A*-algorithm (see table 1). The increase of $\#r_C$ after 14 learning steps is due to the difficulty of moving through the bottleneck using larger steps *right* and *up* (see figure 3); nevertheless $\#P_{opt}$ decreases.

Algorithm	$\#P_{opt}$	$\#r_C$	time [s]
breadth-first search	137	455989	4422
standard A*	137	153811	2190
AA* (1 learning step)	133	3651	2
AA* (3 learning steps)	77	3480	1
AA* (14 learning steps)	33	8723	7

Table 1. Comparison of breadth-first search, standard A*, and Adaptive Advanced A* after 1, 3, 14 learning steps; $\#P_{opt}$ denotes the length of the encountered optimal path, $\#r_C$ denotes the number of computed reference points during the whole planning process. The $3D$-configuration space consists of 1000000 reference points.

Performing Adaptive AA* we also computed optimal paths within $6D$-joint angle-space for our industrial 6DOF-manipulator *manutec r2*. Here the collision check had to be expanded by coding limitations of the joints as obstacles and by a self-collision check. The results for Adaptive AA* after learning were similar to those specified for our exemplary planning task. In addition to software tests we executed optimal paths generated by our algorithm with the manipulator *manutec r2*.

Altogether our simulation results show that Adaptive AA* can learn from sample planning tasks using adaptive expansion matrices. For our exemplary planning task, moving an "**L**"-shaped object through a bottleneck, a better partition of the configuration space than the initial one was learned. Current investigations focus on a learning rule that substitutes rarely used movement vectors by new, randomly initialized ones.

4 Conclusions

Based on Advanced A*-algorithm we proposed a novel concept of adaptive expansion matrices for learning path planning. Substituting the standard expansion logic of A* we enabled the capability to overcome the limitations of equidistant reference points as well as to learn from sample planning tasks. Adaptive AA* keeps the feature of A* to compute an optimal path, with respect to the defined costs and the actual expansion matrix.

Results obtained by solving exemplary planning tasks underlined the power of AA* using adaptive expansion matrices: movement vectors, which allow larger movements than the initial ones into well-defined directions of the configuration space, were learned. Learning efficient movements within high-dimensional configuration spaces leads to the ability of Adaptive AA* to plan complex robot tasks. This can be done either by using Adaptive AA* as global planner (with some restrictions regarding the "curse of dimensions") or as very flexible local planner within a probabilistic roadmap frame.

Acknowledgments

Parts of this work have been supported by the Federal Ministry for Education, Science, Research, and Technology (BMBF), project LENI. Thanks to Jürgen A. Donath, Nils Goerke, and Stefan Hoffrichter for useful comments.

References

1. N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'98) Leuven, Vol.1*, pages 630–637, 1998.
2. S. E. Dreyfus. An appraisal of some shortest-path algorithms. *Operations Research*, 17:395–412, 1969.
3. T. Frontzek, N. Goerke, and R. Eckmiller. A hybrid path planning system combining the A*-method and RBF-networks. In *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN'97) Lausanne*, pages 793–798, 1997.
4. T. Frontzek, N. Goerke, and R. Eckmiller. Flexible path planning for real-time applications using A*-method and neural RBF-networks. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'98) Leuven, Vol.2*, pages 1417–1422, 1998.
5. D. Gelperin. On the optimality of A*. *Artificial Intelligence*, 8:69–76, 1977.
6. P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. In *IEEE Transactions on Systems, Man, and Cybernetics*, volume 2, pages 100–107, 1968.
7. R. E. Korf. Real-time heuristic search. *Art. Int.*, 42:189–211, 1990.
8. J. Pearl. Knowledge versus search: A quantitative analysis using A*. *Artificial Intelligence*, 20:1–13, 1983.
9. L. Shmoulian and E. Rimon. $A^*_\epsilon - DFS$: an algorithm for minimizing search effort in sensor based mobile robot navigation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'98) Leuven, Vol.1*, pages 356–362, 1998.
10. C. W. Warren. Fast path planning using modified A* method. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 662–667, 1993.