Graph-Based Motion Planning for Automated Vehicles Using Multi-Model Branching and Admissible Heuristics

Oliver Speidel, Jona Ruof, and Klaus Dietmayer

Institute of Measurement, Control and Microtechnology
Ulm University, 89081 Ulm, Germany
firstname.lastname@uni-ulm.de

Abstract
Automated driving in urban scenarios requires efficient planning algorithms able to handle complex situations in real-time. A popular approach is to use graph-based planning methods in order to obtain a rough trajectory which is subsequently optimized. A key aspect is the generation of trajectories implementing comfortable and safe behavior already during graph-search, while keeping computation times low. To capture this aspect, on the one hand, a branching strategy is presented in this work that leads to better performance in terms of quality of resulting trajectories and runtime. On the other hand, admissible heuristics are shown which guide the graph-search efficiently, where the solution remains optimal.

Index Terms—Motion Planning, Trajectory Planning, Decision-making, Automated Vehicles, Autonomous Driving

1. Introduction
Over the last years, intensive research has been carried out in the field of autonomous driving [1, 2, 3]. Thereby, motion planning is a crucial requirement and one of the most challenging aspects for automated vehicles. As early as 2007, impressive automated systems for complex urban scenarios with interacting vehicles were presented as part of the well-known Urban Challenge initiated by the Defense Advanced Research Projects Agency (DARPA) [1]. In 2013, the Mercedes S-Class Bertha was able to drive fully autonomously more than 100km from Mannheim to Pforzheim in Germany [2, 4].

A popular architecture for the motion planning system follows the idea that a behavior planning module decides for a strategic maneuver option, which is passed to a trajectory planning module where a feasible trajectory is calculated. A practicable approach for behavior planning is to generate a maneuver option rule-based using heuristics. However, this limits the capabilities for foresighted motion planning in complex environments [5].

For this reason more foresighted but still efficient behavior and trajectory planning systems are widely investigated. A popular concept for graph-based behavior planning is shown in [6]. A speed profile along a given reference path is obtained by using graph-search methods. Therefore, a graph is generated where nodes represent states and edges represent actions. The idea is to extract a rough behavior trajectory over a planning horizon $t_{\text{hor}} \approx 10\text{ s}$ in order to enable foresighted behavior planning. The action set consists of discrete acceleration values and the temporal discretization is $\Delta t = 1\text{ s}$. There exist several approaches which extend this concept of behavior planning for, e.g., short term lateral motion [7], merging behavior at highways [8] or courteous behavior at intersections [9].

In [10], closed-loop forward simulation implementing high-level policies is used in order to generate subsequent states in the graph, in contrast to discrete acceleration or velocity values. The forward simulation is done using the Intelligent Driver Model (IDM) [11] and the Pure Pursuit Controller [12]. However, due to the computational complexity of the approach, the concept is restricted to a horizon of $t_{\text{hor}} = 8\text{ s}$ seconds and large discretization of $\Delta t = 2\text{ s}$. Further, only one policy change is allowed within the planning horizon $t_{\text{hor}}$. A similar method is used in [13] where cooperative behavior for highway scenarios is generated using Monte Carlo Tree Search. In this concept, the IDM as well as pre-defined acceleration actions are employed. Driver models could also be successfully used in various other concepts to efficiently generate social compliant behavior. For example, in [14] the IDM-based MOBIL model [15] is utilized in order to decide whether a lane change is desirable.

Based on the previous discussion, in this work, a motion planning framework is developed enabling foresighted and courteous behavior using graph-search methods extending our concept presented in [9]. The main idea is to utilize different control and driver models, which are known to generate preferable actions for specific scenarios. Consequently, we are able to improve the performance of graph-based behavior planning and driven trajectories compared to related work [6, 9]. In order to still assure real-time capabilities and significantly reduce calculation times, we propose action selection strategies as well as efficient admissible heuristics, which are applicable in interactive urban scenarios.
2. METHODOLOGY

The concept follows the modular architecture of behavior and trajectory planning as shown in Figure 1. Preceding modules provide environmental data including map data as well as state estimations of other vehicles with according predictions. Thereby, a set of predicted trajectories for each other vehicle with corresponding uncertainties is received.

The goal of the graph-based behavior planning is to obtain a rough behavior trajectory. In general, planning is done relative to the center line of the current road lane. Therefore, a node in the graph is represented by a state vector

$$\mathbf{x}_k = [s_k, d_k, \theta_k, \kappa_k, v_k, a_k]^T, \quad k \in \{0, \ldots, T\},$$

where \(s\) is the longitudinal position along the lane, \(d\) the lateral distance to the lane, \(\theta\) the orientation, \(\kappa\) the curvature, \(v\) the velocity, \(a\) the acceleration and \(k\) the corresponding time step. The end of the planning horizon \(t_{hor}\) is denoted by the index \(T\). The lane relative position \([s, d]\) can be transformed to the classic representation \([x, y]\) in Cartesian coordinates and vice versa. For further details, the reader is referred to [16].

The expansion of a node, i.e. the generation of possible subsequent states, is done using different models which will be presented in Section 2.2.1. The time discretization of subsequent nodes is \(\Delta t = 1s\). Beginning from the root node, i.e. the current state, a graph is generated up to the planning horizon \(t_{hor} \approx 10s\). The graph structure is exemplary shown in Figure 2. The optimal behavior trajectory is extracted using the A*-search algorithm, where the search is guided by the admissible heuristic functions shown in Section 2.2.2. The generation of the resulting trajectory in the trajectory planning module is based on the approach presented in [9]. The general idea is to use polynomials in order to interpolate between the behavior trajectory states. In contrast to our previous work, in this work we also regard lateral optimization. In the end, the resulting trajectory is passed to the controller which generates the input for the actuators.

2.1. Branching Strategy

The idea of the branching strategy is to combine the advantages of pre-defined acceleration actions and model-based action which generate preferable behavior for different scenarios, inspired by [13]. In order to omit the expansion of all actions, only a subset of actions is expanded at each node. In the following, this process of choosing which action to expand at which node is also referred to as action selection. In general, the ideas of [13] are extended by additional control models as well as more sophisticated action selection strategies. Further, in this work, the behavior planning is embedded into a holistic framework generating comfortable trajectories. In addition, the solution of behavior planning guarantees the existence of a feasible solution in the trajectory planning module, as all kinematic and collision constraints are considered during forward simulation.

In general, longitudinal actions \(A_{lon}\) and lateral actions \(A_{lat}\) can be distinguished, where the resulting action set is \(A = A_{lon} \times A_{lat} = \{\alpha^{(0)}, \ldots, \alpha^{(n)}\}\). Hereafter, the different actions and corresponding action selection strategies are presented.

**Longitudinal actions**: For longitudinal action generation, acceleration and velocity targets are distinguished.

First the acceleration targets are discussed. These consist of pre-defined accelerations \(a^{(i)} \in \{-2, -1, 0, 1, 2\} m/s^2\) and the acceleration according to the IDM \(a^{(idm)}\). The \(a^{(idm)}\) is expanded during car-following scenarios, as it is able to model comfortable and human-like following behavior. In order to omit expansions of similar states, in car-following scenarios only generic acceleration targets \(a^{(i)}\) with \(|a^{(i)} - a^{(idm)}| > 0.5 m/s^2\) are expanded. Further, we require \(|a_{k+1} - a_k| \leq 1.9 m/s^2\).

The longitudinal state transition model for acceleration
targets is defined by

\[
\begin{bmatrix}
    s_{k+1} \\
    v_{k+1} \\
    a_{k+1}
\end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{2} \Delta t^2 \\
    0 & 1 & \Delta t \\
    0 & 0 & 1 \end{bmatrix} \begin{bmatrix}
    s_k \\
    v_k \\
    a_k
\end{bmatrix} + \begin{bmatrix} \frac{1}{2} \Delta t^3 \\
    \frac{1}{2} \Delta t^2 \\
    \Delta t \end{bmatrix} \hat{a}_k,
\]

where \( \hat{a}_k = (a^{(i)} - a_k) / \Delta t \) and, as a result, \( a_{k+1} = a^{(i)} \).

The velocity targets are defined by the desired velocity \( v_d \) and stillstand \( v_\text{th} = 0 \), where the acceleration of the targets is constrained to 0. The expansion of these actions is triggered if the target velocity is reachable within \( \Delta t \). As state transition model, the concept of C1-continuous time optimal trajectories summarized in [17] is employed. Thus, comfort is ensured by restricted and continuous jerk. In general, C1-continuous time optimal trajectories also allow emergency maneuvers at kinematic limits. However, in this work, we limit our scope to non-safety critical scenarios. For further details, the reader is referred to [17].

Lateral actions: The lateral action set is given by the different road lanes which can be targeted to drive on. Therefore, \( \mathcal{A}_{\text{lat}} = \{l_1, r_c, r_r\} \) where \( l_1 \) represents the lane to the left, \( r_c \) the current lane and \( r_r \) the lane to the right. Using \( r_c \), the motion can be modeled purely longitudinal along the center line of the current lane. In order to perform a lane change to \( r_1 \) or \( r_r \), the Pure Pursuit Controller is employed as lateral transition model and, consequently, the vehicle state is regarded in Cartesian Coordinates. The steering behavior defined by the Pure Pursuit Controller is combined with different acceleration targets for longitudinal behavior. This allows to restrict \( \kappa, \dot{\kappa} \) and the absolute acceleration \( a_{\text{ab}} \) already during behavior planning which ensures feasible solutions in the trajectory planning module. The context in which a lane change is explored is defined by the MOBIL model [15], which is known to generate human-like decision-making for lane change behavior [14]. Thereby, it is estimated if a lane change is favorable for the combined costs of all involved vehicles.

2.2. Cost and Heuristic Functions

The costs attributed to a node are defined by

\[
J = w_{1,i} j_l + w_{c} j_c + w_{v} j_o + w_{a} j_a + w_{\dot{a}} j_{\dot{a}} + w_{c} j_{c},
\]

where \( j_l \) represents costs for the spatio-temporal distance to the vehicle in front. \( j_c \) are courtesy costs that arise if the ego vehicle pulls out or drives in front of another vehicle [9]. The difference to the desired velocity is regarded by \( j_o \), and the comfort is optimized by costs \( j_a \) and \( j_{\dot{a}} \) for larger absolute values of \( a \) and \( \dot{a} \). Further, costs \( j_{c} \) arise for lane changes. The single cost terms can be weighted with the according cost weighting \( w \). In order to generate courteous and safe behavior, a set of predicted trajectories for each other vehicle is considered, where the corresponding uncertainties are incorporated by the single cost terms.

To further improve the runtime, admissible heuristics are developed which can be calculated online. The idea is to use a linear combination of heuristic terms rather than model directly one overall heuristic. If the heuristics for the single cost terms are admissible, the linear combination remains admissible [18]. Therefore, the heuristic is given with

\[
h_{\text{alt}} = \sum_{i=k+1}^{T} w_{c} j_{c,i}^{\min} + w_{v} j_{o,i}^{\min} + w_{a} j_{a,i}^{\min} + w_{\dot{a}} j_{\dot{a},i}^{\min},
\]

where \( j_{c,i}^{\min} \) and \( j_{a,i}^{\min} \) are the minimal costs for the corresponding cost term that arise at time \( i \) originating from the currently expanded node \( x_k \). In the following, the calculation of the single minimal costs terms is explained. The terms \( j_{c,i}^{\min} \) and \( j_{a,i}^{\min} \) are determined by the minimal necessary jerk and acceleration to avoid a collision with the vehicle in front. The term \( j_{o,i}^{\min} \) can be estimated by calculating the maximum possible distance to the vehicle in front at \( i \). The same applies for \( j_{\dot{a},i}^{\min} \), where the maximum possible distance to the vehicle behind is calculated. The minimal arising velocity costs \( j_{v,i}^{\min} \) are given if the ego vehicle accelerates with maximum acceleration to the desired speed. Even though the single minimal costs terms result in low estimated heuristic costs, the evaluation shows that the combination of all heuristics leads to a significant reduction of calculation times, while the solution remains optimal.

3. EVALUATION

The evaluation is done using real world map data contained in a high-precision digital map of Ulm (Germany) including lane-changes, intersections, roundabouts and on-ramp scenarios [3]. The concept is implemented in C++ using the A*-search algorithm of the DOSL library [19]. Runtimes are obtained using an Intel XEON E5-1660 v4 CPU with 3.2 GHz utilizing a single thread. Other vehicles are simulated with random acceleration uniformly distributed between \([-1 \text{ m s}^{-2}, 1 \text{ m s}^{-2}]\) in each time step. In general, for each of the evaluations about 250 scenarios were analyzed, including lane change, highway on-ramp, roundabout and intersection scenarios.

3.1. Branching and Heuristic Functions

At first, the action selection strategy is investigated. The corresponding findings are summarized in Table[1]. In order to measure the comfort of resulting trajectories, both average squared acceleration \( \varnothing a^2 \) and jerk \( \varnothing \dot{a}^2 \) are regarded, as they are also incorporated into the cost function during trajectory planning. The results show that our model-based action selection strategy only has minor influence on the quality of resulting trajectories. This emphasizes that during car-following scenarios exploration of similar states is omitted and that the MOBIL model yields well suited decision making for lane changes when integrated into the graph-based
framework. Thereby, the runtime is reduced by 90% compared to a more passive action selection strategy similar to the defined preconditions in [13]. Consequently, the proposed action selection strategy enables the usage of the extended action set for real-time application, without increased trajectory costs.

Further, the heuristic functions as well as the overall branching strategy were evaluated, where Table 2 shows corresponding results. As baseline the concepts [9, 6] are used, which implement a similar branching strategy. It is shown that comfort is much higher for the proposed approach, as $\varnothing a^2$ is slightly reduced and $\varnothing a^2$ is nearly 25% lower. These results emphasize the idea that model knowledge of driver and control models can be effectively used in order to improve comfort, in the trade-off against longer runtimes. However, the proposed heuristic function are able to effectively reduce the runtime. It is demonstrated that the average runtime can be reduced by about 17%. The maximum runtime is even improved by about 60%, from 158.3 ms to 62.67 ms, while the solution of behavior planning remains optimal. Slight divergences of driven trajectories occur due to numerical issues. As a result, runtimes for behavior planning are significantly reduced even compared to [9, 6], while the quality of driven trajectories is improved using the proposed branching strategy.

### 3.2. Motion Planning Framework

In order to give an insight to the overall performance and the resulting trajectories of the motion planning framework, an exemplary scenario is depicted in Figure 3. In general, an urban left turn scenario is regarded without right-of-way. At time $\varnothing$, the ego vehicle $\bigcirc$ slowly approaches the intersection, while vehicle $\varnothing$ crosses it. Afterwards, vehicle $\varnothing$ and $\varnothing$ approaching from the right have to be considered. Taking the turn in front of vehicle $\varnothing$ would cause to much courtesy

### Table 1: Evaluation of the action selection strategy, including strategies for car-following and lane changing using the MOBIL model. Thereby, the performance of the proposed action selection strategy is compared with a more passive, i.e. less restrictive, one similar to the preconditions presented in [13]. Runtimes for behavior planning as well as average squared jerk and acceleration of driven trajectories are shown. No heuristic functions ($h_0$) are used for the comparison.

|                | proposed ($h_0$) | passive
|----------------|------------------|--------|
| $\varnothing$ runtime [ms] | 18.09            | 18.61  |
| max runtime [ms]       | 158.33           | 158.67 |
| $\varnothing a^2$ [(m/s^2)^2] | 0.056            | 0.056  |
| $\varnothing a^2$ [(m/s^2)^2] | 0.39             | 0.39   |

### Table 2: Evaluation of the proposed behavior planning module, where runtimes for behavior planning as well as average squared jerk and acceleration for driven trajectories are shown. The evaluation scenarios were investigated using the proposed branching strategy and the branching strategy implemented in [6, 9]. Further, the proposed approach is shown with heuristic functions $h_{all}$ and without usage of heuristic functions $h_0$.

|                | proposed ($h_0$) | proposed ($h_{all}$) [6, 9] |
|----------------|------------------|-------------------------------|
| $\varnothing$ runtime [ms] | 18.09            | 14.96                         |
| max runtime [ms]       | 158.33           | 62.67                         |
| $\varnothing a^2$ [(m/s^2)^2] | 0.056            | 0.055                         |
| $\varnothing a^2$ [(m/s^2)^2] | 0.39             | 0.40                          |

Fig. 3: Evaluation scenario, where the ego vehicle performs a left turn, while maintaining comfortable behavior and courtesy towards other traffic participants. In the first row, a top view of the scene is depicted for three different points in time. The planned trajectory is represented by colored dots where red presents the planned position at $T$. The ego vehicle is red and other vehicles are blue. The center lines of the road lanes are depicted in black. The second row shows the velocity and acceleration of the driven trajectory.
costs, thus the ego vehicle merges between vehicle \(\odot\) and \(\odot\) at time \(\odot\).

In addition, it is worth noting that during the analysis of all 250 scenarios, the maximum measured overall calculation time for motion planning was 89.33 ms using the presented approach. Further, none of the scenarios led to any collisions despite random behavior of other traffic participants. This emphasizes the capability of the framework to handle complex urban scenarios.

4. CONCLUSION

In this work, we presented a motion planning framework for autonomous vehicles in urban environments utilizing graph-search methods. The proposed branching strategy and admissible heuristic functions yield trajectories attributed with lower costs, while the runtime is reduced significantly compared to related work. Therefore, the implementation of the concept on the research vehicle of Ulm University and according validations in real-world public traffic is part of our future work.

5. REFERENCES

[1] C. Urmson et al., “Autonomous driving in urban environments: Boss and the urban challenge,” Journal of Field Robotics, vol. 25, no. 8, pp. 425–466, 2008.

[2] J. Ziegler et al., “Making bertha drive—an autonomous journey on a historic route,” IEEE Intelligent Transportation Systems Magazine, vol. 6, no. 2, pp. 8–20, 2014.

[3] F. Kunz et al., “Autonomous driving at ulm university: A modular, robust, and sensor-independent fusion approach,” in 2015 IEEE Intelligent Vehicles Symposium (IV), June 2015, pp. 666–673.

[4] J. Ziegler, P. Bender, T. Dang, and C. Stiller, “Trajectory planning for bertha 2014; a local, continuous method,” in 2014 IEEE Intelligent Vehicles Symposium Proceedings, June 2014, pp. 450–457.

[5] J. Ziegler, “Optimale bahn-und trajectorienplanung für automobile,” 2015.

[6] C. Hubmann, M. Aeberhard, and C. Stiller, “A generic driving strategy for urban environments,” in 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Nov 2016, pp. 1010–1016.

[7] W. Zhan, J. Chen, C. Chan, C. Liu, and M. Tomizuka, “Spatially-partitioned environmental representation and planning architecture for on-road autonomous driving,” in 2017 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2017, pp. 632–639.

[8] E. Ward and J. Folkesson, “Towards risk minimizing trajectory planning in on-road scenarios,” in 2018 IEEE Intelligent Vehicles Symposium (IV), June 2018, pp. 490–497.

[9] O. Speidel, M. Graf, T. Phan-Huu, and K. Dietmayer, “Towards courteous behavior and trajectory planning for automated driving,” in 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Oct 2019, pp. 3142–3148.

[10] L. Zhang, W. Ding, J. Chen, and S. Shen, “Efficient uncertainty-aware decision-making for automated driving using guided branching,” arXiv preprint arXiv:2003.02746, 2020.

[11] M. Treiber, A. Hennecke, and D. Helbing, “Congested traffic states in empirical observations and microscopic simulations,” Physical review E, vol. 62, no. 2, pp. 1805, 2000.

[12] R. C. Coulter, “Implementation of the pure pursuit path tracking algorithm,” Tech. Rep., Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[13] D. Lenz, T. Kessler, and A. Knoll, “Tactical cooperative planning for autonomous highway driving using monte-carlo tree search,” in 2016 IEEE Intelligent Vehicles Symposium (IV), June 2016, pp. 447–453.

[14] M. Graf, O. Speidel, and K. Dietmayer, “A model based motion planning framework for automated vehicles in structured environments,” in 2019 IEEE Intelligent Vehicles Symposium (IV), June 2019, pp. 201–206.

[15] A. Kestinge, M. Treiber, and D. Helbing, “General lane-changing model mobil for car-following models,” Transportation Research Record: Journal of the Transportation Research Board., no. 1999, pp. 86–94, 2007.

[16] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, “Optimal trajectory generation for dynamic street scenarios in a frenet frame,” in 2010 IEEE International Conference on Robotics and Automation, May 2010, pp. 987–993.

[17] K. L. Knierim and O. Sawodny, “Real-time trajectory generation for three-times continuous trajectories,” 2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 1462–1467, 2012.

[18] S. J. Russell and P. Norvig, Artificial intelligence: a modern approach, Malaysia; Pearson Education Limited., 2016.

[19] S. Bhattacharya, “Discrete optimal search library (dosl): A template-based c++ library for discrete optimal search,” 2017, Available at https://github.com/subh83/DOSL.