Planning Cooperative Motions of Cognitive Automobiles Using Tree Search Algorithms

Christian Frese[1] and Jürgen Beyerer[2,1]

[1] Karlsruhe Institute of Technology,
Institute for Anthropomatics, Vision and Fusion Laboratory,
76128 Karlsruhe, Germany, `http://www.ies.uni-karlsruhe.de`
[2] Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB, Fraunhoferstraße 1, 76131 Karlsruhe, Germany

Abstract. A tree search algorithm is proposed for planning cooperative motions of multiple vehicles. The method relies on planning techniques from artificial intelligence such as A* search and cost-to-go estimation. It avoids the restrictions of decoupling assumptions and exploits the full potential of cooperative actions. Precomputation of lower bounds is used to restrict the search to a small portion of the tree of possible cooperative actions. The proposed algorithm is applied to the problem of planning cooperative maneuvers for multiple cognitive vehicles with the aim of preventing accidents in dangerous traffic situations. Simulation results show the feasibility of the approach and the performance gain obtained by precomputing lower bounds.

1 Introduction

Wireless communication between cognitive vehicles bears a potential for increasing traffic safety by cooperative actions of multiple vehicles. In many dangerous situations, it is possible to prevent or mitigate an accident by performing an automated cooperative maneuver, whereas the drivers of the individual vehicles cannot achieve this on their own because of reaction times and lack of coordination possibilities [1]. Therefore, a safety system is proposed which intervenes in dangerous situations by autonomously executing cooperative maneuvers within a group of cognitive vehicles equipped with sensors and actuators [6]. This paper is concerned with an algorithm for the planning of cooperative maneuvers, which is an important component of such a system.

Most previous algorithms to plan cooperative motions for multiple vehicles or robots rely on decoupling assumptions to simplify the problem. One possibility is the path-velocity decomposition [10, 7]: paths are planned separately for each vehicle, and a subsequent coordination of velocities along fixed paths is used to achieve collision-free motions. This approach has been extended to the coordination of motions within a roadmap [17]. The other common simplifying assumption leads to prioritized planning [3, 13]. The motion of the highest-priority vehicle is planned without considering the other vehicles. For the second vehicle, a motion plan is derived which avoids the obstacles and the previously planned trajectory of the first vehicle, but ignores all remaining vehicles, and so on.

The decoupling approaches are appropriate for systems with negligible dynamics. However, in the present context of planning cooperative maneuvers in emergency situations, the cognitive automobiles have significant velocities and dynamics. This implies that many possible solutions are ruled out by the decoupling assumptions, and thus, accidents cannot be prevented in some situations although a feasible cooperative motion exists [5]. This contribution therefore investigates a cooperative motion planning algorithm which does not make these simplifying assumptions. In theory, it is well known that planning with the full freedom of cooperative actions is possible in the composite configuration space [11]. However, these so-called centralized methods have rarely been used in practice due to their high computational complexity [15]. In this paper, artificial intelligence algorithms such as A* search, branch and bound methods, and cost-to-go estimation are applied to cooperative motion planning, and it is shown that they can reduce computing times significantly compared to the naive centralized approach.

Section 2 formulates the cooperative motion planning task as a search problem in a tree of possible actions. In Section 3 it is shown that some of the information computed during the tree search can be reused in subsequent stages of the search. Methods to obtain lower bounds for subtrees are described in Section 4. Section 5 presents and compares two possible search strategies, namely A* search and depth-first branch and bound search. Simulation results and computation times are shown in Section 6, and conclusions are presented in Section 7.

2 Tree Search Formulation of Cooperative Motion Planning

In the proposed approach, both time t and action space $\mathcal{A}$ are discretized. Within the planning horizon, there are T decision points at times $t_0, \ldots, t_{T-1}$. At each decision point, an action $\mathbf{a}$ is chosen out of the finite action set $\mathcal{A}$ and is constantly applied until the next decision point is reached. In the domain of cognitive automobiles, time discretization is justified because actuator limits prohibit a quick change of actions. The discrete action set should contain extreme actions like maximum braking and maximum steering admissible by the dynamics of the vehicle in its current state. These extreme actions are known to be optimal for a single vehicle evading an obstacle [16, 9].

The state of vehicle i is denoted by $\mathbf{x}_i = (x_i, y_i, \varphi_i, v_i)^{\mathrm{T}}$, where (x_i, y_i) is the planar position of the vehicle centroid, φ_i is the yaw angle, and v_i is the scalar velocity of the vehicle. A kinodynamic vehicle model f_i is used to compute the state $\mathbf{x}'_i$ at time t_{k+1} which is reached by executing action a_i starting from state $\mathbf{x}_i$ at time t_k: $\mathbf{x}'_i = f_i(\mathbf{x}_i, a_i, t_k)$. The composite state vector $\mathbf{x}$ of the cooperative vehicles is obtained by concatenating the individual state vectors $\mathbf{x}_i$. A cooperative action is denoted as $\mathbf{a} = (a_1, \ldots, a_m)^{\mathrm{T}}$, where $a_i \in \mathcal{A}_i$ is the action of the ith vehicle. When action $\mathbf{a}$ is applied from t_k until t_{k+1} starting from

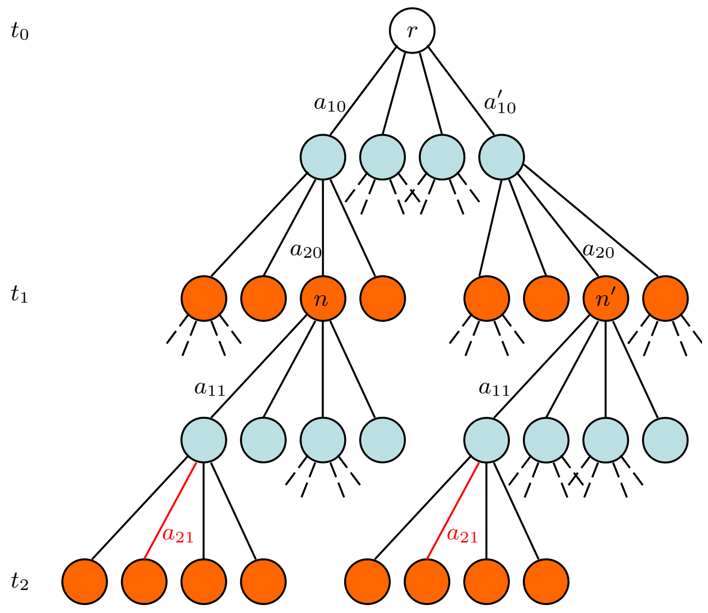

Fig. 1. Tree $\mathcal{T}$ of cooperative actions for $m = 2$ vehicles.

state $\mathbf{x}$, a loss $L(\mathbf{x}, \mathbf{a}, t_k) \geq 0$ is incurred. The loss function rates collisions among vehicles or with obstacles, road departure, and control effort. Loss is a notion used in utility theory to determine preferences between alternative actions [2].

The possible action sequences of one vehicle can be arranged in a tree $\mathcal{T}_i$ having A_i^T leaves, with $A_i := |\mathcal{A}_i|$ being the number of actions. When planning cooperative motions of m vehicles, each action of one vehicle can be combined with all possible actions of the remaining vehicles, yielding the composite action set $\mathcal{A} := \mathcal{A}_1 \times \ldots \times \mathcal{A}_m$. If the action sets $\mathcal{A}_i$ are of equal size $A_0 = |\mathcal{A}_i|$ for all vehicles, the composite action set contains $A := |\mathcal{A}| = A_0^m$ actions. Therefore, the tree $\mathcal{T}$ of cooperative decisions has $A^T = A_0^{mT}$ leaves (see Fig. 1).

This results in large trees even for moderate values of A_0 and T. A naive search for the minimum-loss action sequence visiting every tree node cannot be conducted efficiently. There are two possibilities for speeding up the search while retaining the guarantee of obtaining the optimal solution: firstly, the computing time per tree node can be reduced, e. g., by avoiding some redundancies, and secondly, the number of explored nodes can be reduced by pruning subtrees that cannot contain the optimal solution. Both approaches are detailed in the following sections.

The possibility to use explicit vehicle, loss, and action models is an important advantage of the algorithm. This allows to consider geometry, kinematics, and dynamics in the required level of abstraction. More details on the models currently used are described in [4].

3 Reuse of Computed Information

A naive traversal of the tree $\mathcal{T}$ causes a lot of redundant computations to be performed. For example, let the action sequences $((a_{10}, a_{20})^{\mathrm{T}}, (a_{11}, a_{21})^{\mathrm{T}})$ and $((a'_{10}, a_{20})^{\mathrm{T}}, (a_{11}, a_{21})^{\mathrm{T}})$ pass through the nodes n and n' of $\mathcal{T}$, respectively.[3] If $a'_{10} \neq a_{10}$, $n \neq n'$ results (Fig. 1). However, the state of the second vehicle is the same in both nodes:[4] $\mathbf{x}_2(n) = \mathbf{x}_2(n') = f_2(\mathbf{x}_2(r), a_{20}, t_0)$, where r is the root node of $\mathcal{T}$. Therefore, the identical computation $f_2(\mathbf{x}_2(n), a_{21}, t_1)$ is executed both for n and for n' in order to determine the state of the respective successor node.

Hence, it would be beneficial to store the vehicle states in a suitable look-up data structure, avoiding redundant state computations. This can be achieved by annotating the single-vehicle trees $\mathcal{T}_i$, $i = 1, \dots, m$ with the state information. The single-vehicle trees can be kept in memory as they are much smaller than $\mathcal{T}$ (see Section 2). Mappings $n_i : \mathcal{T} \mapsto \mathcal{T}_i$ can be defined which yield for each node n of the cooperative search tree $\mathcal{T}$ the corresponding nodes $n_i(n)$ in the single-vehicle trees $\mathcal{T}_i$. In the above example, $n_2(n) = n_2(n')$, and therefore the successor state can be stored in $\mathcal{T}_2$ when n is visited and looked up again when n' is visited.

Additionally, some components of the loss function only depend on the action sequence of one vehicle. In particular, these are the terms accounting for collisions with obstacles, road departure, and control effort. In order to avoid redundant computations of loss terms, their sum $L_i(\mathbf{x}_i, a_i, t_k)$ for each applied action $a_i \in \mathcal{A}_i$ can be stored in $\mathcal{T}_i$ in the same way as the vehicle state.

The only remaining component of the loss function is the term for collisions among cooperative vehicles. This term does not depend on the full cooperative action sequence either: it only depends on the actions of a pair of vehicles [12]. Therefore, similar issues with redundant computations arise during a cooperative search with $m > 2$, and the same principle can be applied to store the already computed loss values. Precisely, the two-vehicle trees $\mathcal{T}_{ij}$, $1 \leq i < j \leq m$, can be annotated with the collision loss $L_{ij}(\mathbf{x}_i, \mathbf{x}_j, a_i, a_j, t_k)$, and mappings $n_{ij} : \mathcal{T} \mapsto \mathcal{T}_{ij}$ can be constructed to retrieve the stored values.

4 Precomputing Lower Bounds

During tree search, it is helpful to have a cost-to-go estimate for the optimal loss within a subtree of $\mathcal{T}$. These estimates can be used as a heuristic to guide the search and to avoid subtrees with high loss. In particular, when a lower bound

[3] In the following, $\mathbf{x}_i(n)$ and $t(n)$ denote the state of the ith vehicle at n and the decision time of the node, respectively.

[4] Strictly speaking, this is true only if no collisions have occurred yet. After a collision of two cooperative vehicles, the state of one vehicle is no longer independent of the other vehicle's actions. Currently, this is not considered by the proposed tree search method. Note, however, that only the states in the subtree below the point of collision would have to be recomputed.

for the loss within a subtree is known, it can be proven under certain conditions that the optimal action sequence cannot visit this subtree.

Lower bounds for the cooperative motion planning problem are obtained as follows. The single-vehicle search trees $\mathcal{T}_i$ and the corresponding loss values L_i are precomputed. Then, lower bounds h_i can be established by propagating loss information upwards the trees recursively:

$$h_i(\mathbf{x}_i, t_k) := \begin{cases} 0 & \text{if } k = T \\ \min\limits_{a_i \in \mathcal{A}_i} \{L_i(\mathbf{x}_i, a_i, t_k) + h_i(f(\mathbf{x}_i, a_i), t_{k+1})\} & \text{if } k < T \end{cases} \tag{1}$$

A lower bound for the subtree rooted at node n of the cooperative search tree $\mathcal{T}$ can be computed as follows:

$$h_{\mathrm{SV}}(n) := \sum_{i=1}^{m} h_i(\mathbf{x}_i(n_i(n)), t(n)) \tag{2}$$

In a similar way, lower bounds $h_{\mathrm{coll}}(n)$ are obtained by precomputing the two-vehicle trees $\mathcal{T}_{ij}$ and the collision loss values L_{ij}.

Lower bound precomputation (P) and storage (S) of possibly reusable information can either be performed for the entire problem or only up to certain tree depths $0 \leq P_{\mathrm{SV}} \leq S_{\mathrm{SV}} \leq T$ and $0 \leq P_{\mathrm{coll}} \leq S_{\mathrm{coll}} \leq T$. $h_{\mathrm{SV}}(n)$ and $h_{\mathrm{coll}}(n)$ are defined to be zero if no precomputation has been performed for node n.

5 Search Strategies

The A* algorithm makes use of a so-called heuristic function to search graphs efficiently [8]. If the heuristic function $h(n)$ yields lower bounds of the true loss values from n to a goal node, the search is guaranteed to find the optimal solution [14]. As the precomputed values $h_{\mathrm{SV}}(n)$ and $h_{\mathrm{coll}}(n)$ fulfill this requirement, A* can be used to search the tree of cooperative actions $\mathcal{T}$. Starting with the root r, nodes are inserted into a priority queue sorted by ascending value of $g(n) := L(r, n) + h_{\mathrm{SV}}(n) + h_{\mathrm{coll}}(n)$, where $L(r, n)$ is the loss accumulated on the path from r to n in $\mathcal{T}$. The minimum element of the queue is chosen for processing, and the children of this node are inserted into the queue. The search terminates with the optimal action sequence as soon as a leaf node is reached.

Alternatively, the branch and bound (BB) search strategy can be used. It explores the tree in depth-first order, remembering the best solution found so far and its loss value L^*. Subtrees are excluded from the search if it can be proven that they cannot contain any part of the optimal solution. This is the case if $g(n) > L^*$ for the root node n of the subtree. Then, the total loss of any path passing through the subtree must be greater than L^*.

When compared to BB, the A* strategy has the drawback of an additional logarithmic time complexity for the operations of the priority queue. Moreover, the memory requirements of the priority queue can become problematic, as the queue may contain every leaf node in the worst case. By contrast, the BB strategy has very low memory requirements. Only the current path of the depth-first search and the best action sequence found so far need to be remembered.

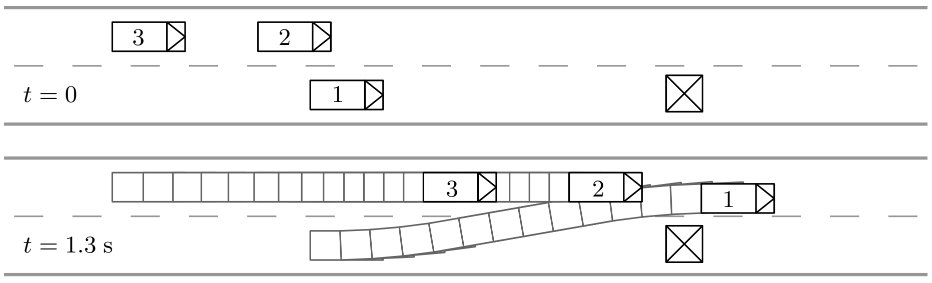

Fig. 2. Example of a cooperative merging maneuver planned with the proposed algorithm. While vehicle 1 changes to the left lane due to the obstacle, vehicles 2 and 3 brake in order to allow the merging. The full planning horizon has been 2.5 seconds.

method	P_{SV}	S_{SV}	P_{coll}	S_{coll}	precomputation time		total time		nodes visited	
					average	max	average	max	average	max
A*	0	0	0	0	0.000	0.000	0.390	1.679	7 332	33 796
A*	0	4	0	0	0.000	0.000	0.078	0.371	7 332	33 796
A*	0	4	0	3	0.000	0.000	**0.026**	0.098	7 332	33 796
A*	3	4	0	3	0.017	0.020	0.033	**0.090**	4 638	25 438
A*	4	4	4	4	31.955	37.791	31.972	37.807	**477**	**2 303**
BB	0	0	0	0	0.000	0.000	33.151	380.278	2 840 040	34 087 340
BB	0	4	0	0	0.000	0.000	0.189	1.144	16 915	114 761
BB	3	4	0	3	0.017	0.020	**0.094**	0.832	6 467	79 242
BB	4	4	0	3	0.148	0.171	0.156	**0.196**	377	1 725
BB	4	4	4	4	31.581	37.422	31.598	37.439	**97**	**525**

Table 1. Computing times in seconds averaged over 20 different instances of the merging problem of Fig. 2, with $m = 3$ and $T = 4$. The best results for A* and BB, respectively, are highlighted. Also shown is the number of tree nodes visited. Search is restricted to a small fraction of the tree: the total number of leaf nodes is $1.3 \cdot 10^{10}$ in this example.

6 Results

The proposed algorithms have been implemented in C++ and integrated with the traffic simulator described in [18]. They have been tested in various obstacle avoidance, merging, overtaking, and intersection scenarios [4]. Fig. 2 depicts a cooperative maneuver planned with the proposed algorithm.

Tables 1 and 2 list the computing times measured for different variants of the algorithm on an off-the-shelf desktop PC. The best performing variants of each A* and BB are shown together with the respective baseline algorithms. On average, A* performed better than BB, but there are some instances of the problem for which BB was significantly faster. For example, computing times of 0.33 s (BB) and 1.2 s (A*) have been observed for a specific overtaking problem involving four vehicles. Concerning the different storage and precomputation variants, reuse of single-vehicle information was always useful. Precomputation of single-vehicle lower bounds and reuse of collision loss values were advantageous up to certain tree depths P_{SV}, S_{coll}. For larger S_{coll} the overhead for maintaining the data structures dominates the gain of avoiding redundant collision checking. Precomputation of collision loss was not beneficial in most situations: while it reduced the number of visited nodes even further, the precomputation effort usually more than outweighed the performance gain during the search (Table 1).

In the most difficult setting considered ($m = 4$, $T = 4$) none of the variants with $P_{\mathrm{SV}} < T$ was feasible: BB showed runtimes in the order of hours, and A* aborted with out of memory error. With precomputation, the algorithm has potential for real-time use in all simulated instances of the considered tasks (Table 2). This result highlights the effectiveness of the presented lower bound precomputation scheme, although no guarantees on the number of explored nodes can be given for the theoretical worst case.

7 Conclusions

A new algorithm for cooperative motion planning has been proposed which exploits the full potential of cooperative actions. It has been shown that precomputation of lower bounds is essential for the computational feasibility of the tree search approach. The method has been successfully applied to simulate cooperative maneuvers in dangerous traffic situations. Although A* performed better on average, BB and A* search each have advantages in specific problem instances so that no clear preference between the two search strategies could be derived. Further research is necessary to gain deeper insight into this issue, and possibilities for combining both search strategies will be evaluated.

Acknowledgements

The authors gratefully acknowledge support of this work by Deutsche Forschungsgemeinschaft (German Research Foundation) within the Transregional Collaborative Research Center 28 “Cognitive Automobiles”. Furthermore, the authors would like to thank all project partners contributing to the development of the simulation system.

References

1. Batz, T., Watson, K., Beyerer, J.: Recognition of dangerous situations within a cooperative group of vehicles. In: Proc. IEEE Intelligent Vehicles Symposium. pp. 907–912. Xi’an, China (Jun 2009)
2. Berger, J.O.: Statistical decision theory and Bayesian analysis. Springer series in statistics, Springer, New York, 2nd edn. (1993)
3. Erdmann, M., Lozano-Pérez, T.: On multiple moving objects. Algorithmica 2, 477–521 (1987)
4. Frese, C.: Cooperative motion planning using branch and bound methods. Tech. Rep. IES-2009-13. In: Beyerer, J., Huber, M. (eds.): Proceedings of the 2009 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory, pp. 187–201, KIT Scientific Publishing (2010)

m	T	method	P_{SV}	S_{SV}	P_{coll}	S_{coll}	precomp. time		total time		nodes visited	
							average	max	average	max	average	max
2	5	A*	0	0	0	0	0.000	0.000	0.136	0.775	2 927	17 521
2	5	A*	2	5	0	0	0.002	0.003	0.021	0.073	752	4 471
2	5	BB	0	0	0	0	0.000	0.000	0.933	4.028	73 533	306 936
2	5	BB	2	5	0	0	0.002	0.003	0.042	0.091	1 731	7 186
4	3	A*	0	0	0	0	0.000	0.000	0.624	10.547	5 718	163 387
4	3	A*	2	3	0	3	0.005	0.006	0.032	0.171	501	5 973
4	3	BB	0	0	0	0	0.000	0.000	733.349	6499.990	31 882 148	249 583 950
4	3	BB	3	3	0	3	0.048	0.054	0.085	0.781	413	11 643
4	4	A*	4	4	0	3	0.314	0.373	0.329	0.522	3 154	68 936
4	4	BB	4	4	0	4	0.313	0.374	1.497	64.134	13 373	926 700

Table 2. Average and maximum computing times in seconds for 150 different intersection situations.

5. Frese, C., Batz, T., Beyerer, J.: Kooperative Bewegungsplanung zur Unfallvermeidung im Straßenverkehr mit der Methode der elastischen Bänder. In: Dillmann, R., et al. (eds.) Autonome Mobile Systeme. pp. 193–200. Springer (Dec 2009)
6. Frese, C., Beyerer, J., Zimmer, P.: Cooperation of cars and formation of cooperative groups. In: Proc. IEEE Intelligent Vehicles Symposium. pp. 227–232. Istanbul (Jun 2007)
7. Ghrist, R., LaValle, S.: Nonpositive curvature and pareto optimal coordination of robots. SIAM Journal on Control and Optimization 45(5), 1697–1713 (Jan 2006)
8. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on System Science and Cybernetics 4(2), 100–107 (Jul 1968)
9. Hillenbrand, J.: Fahrerassistenz zur Kollisionsvermeidung. Dissertation, Universität Karlsruhe (TH) (2007)
10. Kant, K., Zucker, S.: Toward efficient trajectory planning: The path-velocity decomposition. Int. J. Robotics Research 5(3), 72–89 (1986)
11. LaValle, S.: Planning Algorithms. Cambridge University Press, 1st edn. (2006)
12. LaValle, S., Hutchinson, S.: Optimal motion planning for multiple robots having independent goals. IEEE Transactions on Robotics and Automation 14(6), 912–925 (Dec 1998)
13. Regele, R., Levi, P.: Kooperative Multi-Roboter-Wegplanung durch heuristische Prioritätenanpassung. In: Autonome Mobile Systeme. pp. 33–39. Springer (2005)
14. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, 2nd edn. (2003)
15. Sánchez, G., Latombe, J.C.: On delaying collision checking in PRM planning: Application to multi-robot coordination. Int. J. Robotics Research 21(1), 5–26 (2002)
16. Schmidt, C., Oechsle, F., Branz, W.: Research on trajectory planning in emergency situations with multiple objects. In: Proc. IEEE Intelligent Transportation Systems Conference. pp. 988–992 (Sep 2006)
17. Švestka, P., Overmars, M.: Coordinated path planning for multiple robots. Robotics and Autonomous Systems 23, 125–152 (1998)
18. Vacek, S., Nagel, R., Batz, T., Moosmann, F., Dillmann, R.: An integrated simulation framework for cognitive automobiles. In: Proc. IEEE Intelligent Vehicles Symposium. pp. 221–226. Istanbul (Jun 2007)