2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Contingency Planning for Automated Vehicles

João Salvado, Luís M.M. Custódio, Daniel Hess

Abstract— **Automated driving is a safety critical process, which requires complex decision making. In order to validate driving decisions, it is possible to maintain at all times a contingency maneuver, which transfers the vehicle to a safe standstill, if other decision making processes fail. In this paper we present a motion planner, which computes contingency maneuvers for an automated vehicle in a 0.1[s] time frame. A discrete set of motion primitives is assembled in a heuristic best-first search. In order to speed up the search, an obstacle sensitive heuristic is applied, which maintains properties of bounded sub-optimality and completeness. A run-time comparison with and without the obstacle sensitive heuristic is presented on two exemplary collision avoidance scenarios.**

Index Terms - **Autonomous Agents; Motion and Path Planning; Motion and Trajectory Generation.**

I. INTRODUCTION

Automated vehicles face several challenges when planning their actions in an uncertain and partially known environment. Creating optimal and feasible paths in those scenarios in a reduced runtime is a difficult problem. A planner adapted to a cluster of scenarios and integrated in a hierarchical planning system with other modules is a viable solution.

We focus on combinatorial search methods for real-time planning and control state sampling by using precomputed motion primitives.

The state of the art is divided into two subsections, where the motion primitives I-A correspond to the offline phase and the search methodologies I-B to the real-time phase in our approach.

A. Motion Primitives

A grid-based approach overlays a network of ”points” on the state/control space continuum. At each grid point, the vehicle can move to neighbour points if the path between them is enclosed by $\mathscr{C}_{Free}$, collision-free space. Some examples of this approach are a simple N-order grid, Barroque Latombe motion primi- tives [1] and Mihail Piv- toraiko Lattices [2]. As a result of the discretization of the state space, one can only assure resolution completeness. On the other hand, the quality of the path sets generated is high, since the computation of the trajectories is made offline, which allows the solution of a boundary value problem (BVP) by using numerical methods, when there is no closed form solution and the state space is sampled instead of the control space. Even when it is not necessary to solve a BVP, offline control space sampling can have a positive effect on guiding the search by careful selection of samples, as described in [3].

Since an increase in the number of the motion primitives increases the number of expansions during the online phase, several strategies were developed in order to reduce motion primitives by following certain selection criteria. Some of the metrics used during the sampling of the state space are discrepancy, dispersion or diversity [3], [4]. There are deterministic approaches exclusive to grid-based methods, e.g. Primitive Set Decomposition [5], which prunes newly created trajectories by accessing previous lattices and comparing them within a certain adjustable boundary factor.

B. Search Methodologies

The work in [6] presents an overview of several combinatorial methods going from the fundamentals with the classic A* [7] to more recent approaches. For example, D*Lite is an A* alike method, which allows replanning cycles. ARA* gives anytime characteristics to the planner [8]. LPA* is able to repair the search tree with a changing environment [9]. Finally AD* combines the anytime characteristics of ARA* and the repairing features of the LPA* algorithm.

Anytime search is a pragmatic approach for trading solution cost and planning time. It can also be used for solving problems within a time bound. Three frameworks for constructing anytime algorithms in a bounded suboptimal search have been proposed: continuing search, repairing search and restarting search [10]. Continuing is a bounded suboptimal search, which after encountering a solution, continues by iterating the process, allowing ever improving solutions. The OPEN list is maintained in each cycle. This approach was initiated in [11]. Repairing searches have a different manner of handling duplicate states by creating three lists (OPEN, CLOSED and ICONS), where all inconsistent nodes are stacked, rather than immediately expanded, until the next iteration of the repairing cycle. This method is valuable in situations where an agent finds a solution initially and while moving to the goal perceives a changed environment, because only the part of the graph where the environment changed is repaired, as [12] describes. A second difference is related to the fact that in each cycle the search starts from the initial node instead of a node with the lower evaluation function value in the OPEN list. Restarting search is similar to continuing search. A suboptimal bound is tightened in each iteration, allowing the improvement of the solution in each cycle. The main difference is that the search restarts from the initial node. This approach eliminates the low-h-bias in the initial phase of the search. Since the effect of the

Daniel Hess is with Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), Institut für Verkehrssystemtechnik, Lilienthalplatz 7, 38108 Braunschweig Germany, Daniel.Hess@dlr.de

João Salvado and Luís M.M. Custódio are with Institute for Systems and Robotics (ISR/IST), LARSyS, Instituto Superior Técnico, Univ Lisboa, joaosalvado@ist.ult.pt and lmmc@isr.ist.utl.pt

inflated part of the heuristic is bigger at lower depths, final states would have lower evaluation function values, which would be responsible for having similar solutions in the following anytime cycles. To overcome the low-h-bias the cost of each node in the OPEN list is recomputed considering the true cost to the goal from the previous iteration, as reported in [13].

Concerning heuristics, several methods allow the usage of nonadmissible heuristics with an anchor admissible heuristic. Such a combination might allow the maintenance of the anchor heuristic properties, e.g. the work developed in [14], where the heuristic is the minimum between the anchor admissible and the nonadmissible heuristic (*pathmax* strategy). A multi-heuristic search with uncalibrated heuristics was created in [15], where several OPEN lists are utilised, each one with an uncalibrated heuristic, and one of the OPEN lists has an anchor heuristic to assure admissibility. With this strategy, several heuristics could be generated for different situations, which will make the overall planner more adaptable to the environment features.

Motivated by these recent developments, a new contingency planner is proposed in this paper. A variety of possible methods for control space sampling and search methodologies could be applied and tuned for emergency scenarios. This work evaluates a set of strategies for its applicability to the specific problem of finding a solution in a 0.1[s] time frame.

The article is organized as follows: first the problem statement is introduced in section II, this is followed by the approach in section III, in section IV experiments and simulation results are shown and discussed, finally in section V conclusions are presented, contributions are listed and suggestions for future work are given.

II. PROBLEM STATEMENT

The goal is to implement a contingency planner for an emergency situation, which finds a feasible and safe maneuver in a 0.1[s] time frame. In order to accomplish such a goal the planner was divided into two phases.

In the offline phase, a set of motion primitives are generated, pursuing a nonlinear system model, bounded by physical constraints, which takes the burden of feasibility verification from the online phase, where time has a higher cost.

The online/real-time phase is the search phase, in which a graph of maneuver segments is constructed taking advantage of the precomputed motion primitives created in the offline phase. During this phase, collision tests with moving obstacles and road boundaries are conducted.

Fig. 1 describes how the contingency planner fits in the scope of the complete DLR planner. Here, the route planner lies on the spectrum of long-range and low-fidelity planning and the trajectory tracking planner focuses on short-range and high-fidelity.

In this work, we present a new grouping and organization scheme for outgoing edges, with a combination of an obstacle-sensitive heuristic, while maintaining completeness and sub-optimality properties. The new scheme allows reducing runtime for collision detection and heuristic computation due to the descendant nodes grouping. A control space sampling scheme is constructed and tuned for emergency maneuver planning.

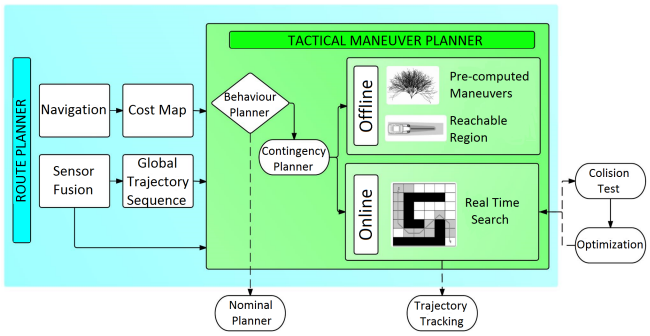


Fig. 1. Hierarchical Planner of the overall project, contemplating several modules/planners and its iterations

III. APPROACH

A. Motion Primitives for an Automated Vehicle in an Emergency Situation

A initial value problem is solved by sampling the control space, initial condition, and solving the ordinary differential equations afterwards. The control space $\mathbb{R}^2$ was discretized as a two-dimensional grid with lateral acceleration and longitudinal velocity and was constrained by physical boundaries, which are maximum and minimum velocity, lateral acceleration and curvature. A time interval constraint is also applied. The grid with the constraints previously depicted will be referred to as as constraint graph for simplicity's sake.

Emergency situations are characterized by having a high collision risk. A special set of motion primitives was designed with the following assumptions.

$$a_x = sign(v_{x1} - v_{x0})\sqrt{g^2 - max(|a_{y0}|, |a_{y1}|)^2} \quad (1)$$

$$\Delta t = \frac{v_{x1} - v_{x0}}{a_x} \quad (2)$$

$$\frac{da_y}{dt} = \frac{a_{y1} - a_{y0}}{\Delta t} \quad (3)$$

$$a_y{}^2 + \left(\frac{v_{x1} - v_{x0}}{\Delta t}\right)^2 \leq g^2 \ : \ 0.5 \leq \Delta t \leq 2.5[s] \quad (4)$$

The longitudinal acceleration (braking) a_x is defined according to the chosen lateral acceleration and the friction circle constraint, in equation 1, where g is the gravitational constant. The sets of velocity and lateral acceleration (v_{x0}, a_{y0}) (v_{x1}, a_{y1}) represent the initial and final state, respectively. By having a constant a_x, the trajectory time Δt is computed as in equation 2. The lateral acceleration derivative is assigned as constant, in equation 3. Finally, trajectories created last between 0.5 and 2.5 seconds, in equation 4. The creation of the motions primitives requires a tuning of the trajectories duration. The time frame chosen yields an acceptable balance

between precise maneuvering with short trajectories and depth first search exploration using long trajectories. The aim behind the assumptions is to utilize the remaining potential of the friction circle, after the desired lateral acceleration has been chosen, while maximizing braking.

We consider a model for the vehicle motion defined by the following ordinary differential equations

$$\begin{aligned} \dot{x} &= v_x \cos(\theta) \\ \dot{y} &= v_x \sin(\theta) \\ \dot{\theta} &= a_y v_x^{-1} \\ \dot{v_x} &= a_x = u_1 \\ \dot{a_y} &= u_2 \end{aligned} \tag{5}$$

This formulation specifies the system state as (x, y, θ, v_x, a_y), with (x, y) the vehicle position in euclidean coordinates and θ the vehicle heading. The control inputs are $(a_x, \dot{a_y})$.

The reference system x-y is centered on the car gravitational center. The vehicle x-axis is assumed to be parallel to the trajectory. The conjunction of the assumptions depicted in equations 1-4 and ODE's in 5 allows the construction of motion primitives, which can be seen in figure 2.

Furthermore, one has to consider that a higher resolution of the constraint graph means a geometric increase in motion primitives. This fact generates two major issues. Firstly, the data file of motion primitives can quickly reach a size that is not acceptable. And secondly the search branching factor also increases. To overcome this, an empirical pruning strategy was utilized, where instead of creating all the trajectories that have an end state inside the bounding ellipses, defined in the equation 4, only trajectories within a grid cell distance from the bounding ellipses are utilized. Figure 2 represents the constraint graph with the pruning strategy and the respective trajectories is depicted in figure 2.

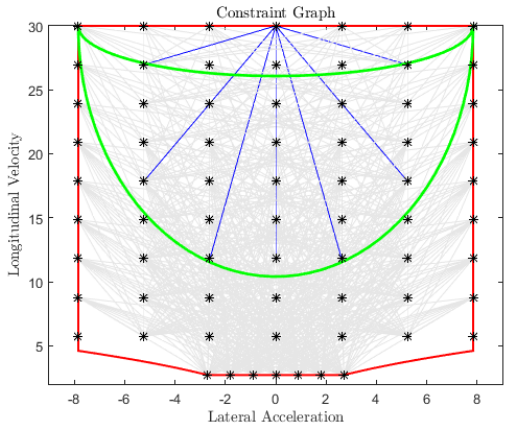


Fig. 2. Constraint graph delimited by curvature, longitudinal velocity and lateral acceleration constraints, in red. Bounding ellipses computed by equation 4, in green. All possible trajectories having an initial maximum velocity and zero lateral acceleration are shown in blue, such group is assigned as trajectory superset. In light gray are represented all motion primitives.

An important step in our approach is the creation of a hierarchical structure of motion primitives. For each discrete vehicle state in the v_x-a_y grid, the available (outgoing) motion primitives are grouped into $k = 3$ sets, according to their final lateral acceleration, see figure 3. The clustering of motion primitives allows to speed up collision detection and the computation of the obstacle-sensitive heuristic by set-based precomputations. In the evaluation of each individual motion primitve, only those objects are considered, which intersect the convex hull of its motion primitive set.

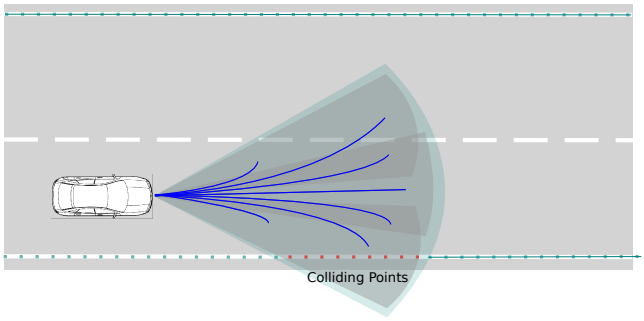


Fig. 3. Simplified representation of the division of trajectories into a hierarchical structure such that a group of trajectories belong to a trajectory set (three arc sections in grey) and the group of trajectory sets is a trajectory super set (arc section in light blue). The trajectories are represented in blue.

B. Continuing Anytime Search for an Automated Vehicle in Urban Traffic

A search problem can be defined formally by initial state, nodes and edges configuration, transition model, goal test and path cost [16]. The initial state/root-node is in the configuration $(x = 0, y = 0, \theta, v_x, a_y, t = 0)$. The pre-computed motion primitives depicted in subsection III-A correspond to the edges in the real time search. The transition model employs the ODE's in (5). The goal node is $(x, y, \theta, v_x = 0, a_y, t)$ when the vehicle is stationary. A solution trajectory is composed of the motion primitives leading from the root of the tree, which is the current vehicle state, to a goal node. Nodes/states have a six-dimensional representation $(x, y, \theta, v_x, a_y, t)$. The path cost is the total duration of the trajectories.

Since we intend to find a solution in a 0.1[s] time frame, the core algorithm here proposed is an epsilon-bounded anytime search and the approach followed is a continuing anytime search, as reported in [10]. Anytime methods find sub-optimally bounded solutions fast and allow ever improving solutions until time runs out [8].

Node cost $c(s)$ is defined as the total trajectory time $\Delta t_{\text{incoming trajectory}}$ in equation (6), since it is commonly accepted that a faster-braking vehicle has its damages minimized, in case a collision happens. The true cost of going from the initial state/node to the final state/node s is represented as $g(s)$, in equation (7). For the anchor heuristic $h(s)$, an admissible heuristic was constructed, being the time required to come to a standstill state when applying maximum longitudinal acceleration, in equation (8). Since a_{max} is assigned to be equal to 9.81 [m/s^2] the cost to the

goal is always underestimated, which allows the maintenance of the heuristic admissibility properties.

$$c(s) = \Delta t_{\text{incoming trajectory}} \tag{6}$$
$$g(s) = c(s) + g(\text{parent}) \tag{7}$$
$$h(s) = \frac{v_x}{a_{\max}} \tag{8}$$
$$f_{\text{uninflated}}(s) = g(s) + h(s) \tag{9}$$
$$f_{\text{inflated}}(s) = g(s) + \varepsilon \cdot h(s) \tag{10}$$

We also compute an inflated and uninflated evaluation function ($f_{\text{inflated}}(s), f_{\text{uninflated}}(s)$) in equations (9) and (10), for the adaptative strategy that is responsible for recomputing the ε value.

The anchor heuristic does not have any information about the environment, so an environment-sensitive heuristic was created. Note that the inflated evaluation function has an uninflated and inflated part, as equation (11) shows:

$$\begin{aligned} & f_{\text{inflated}}(s) = g(s) + (\varepsilon + 1 - 1)h(s) \\ \equiv\ & f_{\text{inflated}}(s) = f_{\text{uninflated}}(s) + (\varepsilon - 1)h(s) \\ \equiv\ & f_{\text{inflated}}(s) = f_{\text{uninflated}}(s) + I(\varepsilon, s) \end{aligned} \tag{11}$$
$$I(\varepsilon, s) = \alpha(\varepsilon - 1)h(s) : \alpha \in [0,1] \tag{12}$$

where I is a function of ε and s that computes the inflated part of the evaluation function.

We intend to tune the inflated part by multiplying it by a factor α, in equation (12). Considering that α is between zero and one, the true cost to the goal is always being underestimated, allowing the maintenance of a sub-optimally bounded solution and completeness properties. Such tuning can be seen the in figure 4.

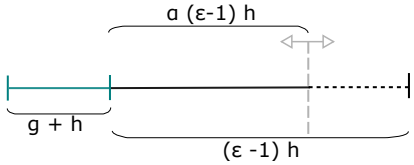


Fig. 4. Representation of the utilization of the inflated part by changing α parameter

The strategy of expanding *versus* reducing the inflated part I is a framework for a wide range of situations that could be specialized for a cluster of scenarios by computing a suitable α factor. For emergency situations the α factor can be computed as follows:

$$\alpha_{s_t=i} = \frac{\max_{s_t \in \mathscr{S}_t}(\overline{d}_{os_t}) - \overline{d}_{oi}}{\max_{s_t \in \mathscr{S}_t}(\overline{d}_{os_t})} : o \in \mathscr{O} \tag{13}$$

where $\overline{d}_{oi}$ is the average distance between the center of all obstacles $o \in \mathscr{O}$ and the center of a trajectory set $i \in \mathscr{S}_t$. $\mathscr{O}$ is the universe of all the objects detected by the ego vehicle and $\mathscr{S}_t$ the group of all trajectory sets in a trajectory super set, of which in this implementation there are three.

As a result, trajectories inside each trajectory set have the same inflated part. Nevertheless, each trajectory has a unique uninflated part. It is therefore possible to distinguish clusters of trajectories (trajectory sets) by tuning the inflated part of the evaluation function, which allows a prioritization of trajectory sets that are far from the obstacles.

When a solution is found a new anytime cycle starts, where the ε bound is updated following the work in [8], adapted to a continuing anytime search, in equation (14). Afterwards, nodes in the Open list are reordered considering the recomputed evaluation function values.

$$\varepsilon_{new} = \min_{s \in OPEN}\left(\varepsilon_{old}, \frac{\bar{f}}{g(s) + h(s)}\right) \tag{14}$$

where $\bar{f}$ represents the true cost of the solution found.

The overall contingency planner diagram is shown in figure 5.

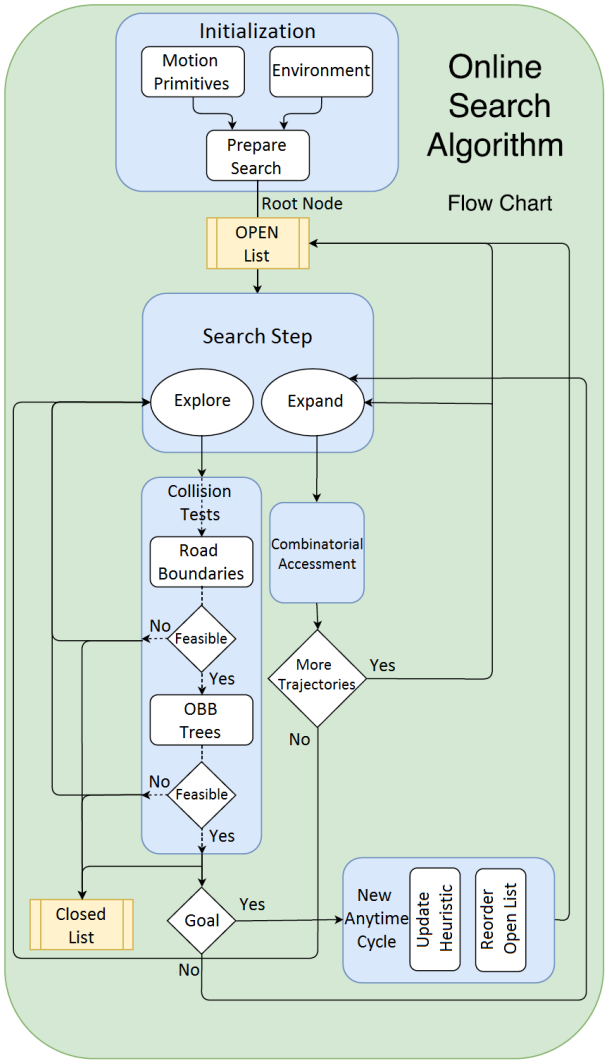


Fig. 5. Online Search Algoritm flow diagram

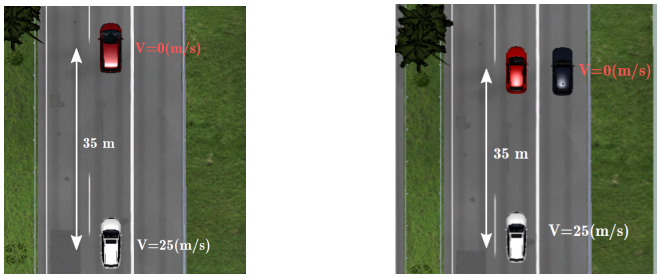
In the initialization phase, motion primitives are loaded from a data file and the environment is sensed. Afterwards a prepare search step occurs, where a root node is initialized

with the current vehicle state, v_x and a_y rounded to the nearest grid-point. During the exploration step a hierarchical collision test strategy is implemented. The expansion step is responsible for the combinatorial accessment of each node following equations (6-13). When the search reaches a goal node, an anytime cycle starts, the ε bound is updated as in equation 14 and the OPEN list is reordered.

IV. SIMULATIONS AND RESULTS

The integration of the motion primitives into the anytime online search described in the previous sections allowed the construction of two strategies that will be compared and evaluated in this section. One is an anytime, sub-optimally bounded search, with a trajectory clustering strategy, and environment-sensitive heuristic, SHA* - Sensitive Heuristic A*. The other is an anytime, sub optimally bounded, standard search, AWA*- Anytime Weighted A*. In order to conduct the experiments the "DLR Automotive Motion Planner" was utilized.

The strategy SHA* has a heuristic that requires a higher computation time *per* node, nevertheless we expect a better informed search. As a result, we intended to create tests and evaluate the results such that one could answer if one effect compensates the other and what are the advantages and disadvantages of each method. A couple of test scenarios is depicted in figure 6 and a discussion of the results will follow.



(a) Scenario 1 (b) Scenario 2

Fig. 6. Scenario 1 contemplating a stationary obstacle vehicle in red and the ego vehicle with a velocity of 25 (m/s). Following the work of [17] and [18] the risk of collision in this scenario is 97%. Scenario 2 another stationary vehicle is added to the right lane.

In table I the results for both methods and scenarios can be consulted.

TABLE I
RESULTS FOR BOTH SCENARIOS AND SEARCH METHODS

	Senario 1				Senario 2			
	AWA*		SHA*		AWA*		SHA*	
Solution	1st	Last	1st	Last	1st	Last	1st	Last
Time [ms]	18	21	3	23	31	51	3	36
# Invalid Nodes	179	240	3	77	599	755	3	222
Memory	42	61	563	723	58	108	387	548
Epsilon Value	4	1.22	4	1.10	4	1.43	4	1.20

The results show a major improvement in **runtime** when comparing the first solution of both methods in the two scenarios. The approximation of the final solution is similar, which is expected, since both methods converge to an A^* by reducing the ε value to almost unity. N.B. the aim of the planner is to find a solution as soon as possible.

Concerning **invalid explorations**, the superiority of the SHA* method is straightforward, with the number of invalid explorations being smaller in both the first and final solution found. In fact since the SHA*is environment-sensitive, one expects fewer invalid explorations to be made. In scenario 2 the difference increases since both central and right trajectory set is highly penalized, which allows a prioritization of the left lane trajectory set.

The number of expanded nodes which stay in **memory**, in the SHA* strategy, is an order of magnitude bigger than in the standard AWA* method. Looking at the time interval until a solution is found, it is possible to conclude that expansions don't have a major role in that respect. The elevated number of nodes in memory can also be explained by the fact that with a SHA* strategy more jumps between sets are expected, and even though a trajectory superset is divided into three trajectory sets, all of the nodes inside a trajectory set are expanded.

An important characteristic of both methods is the convergence of the **epsilon value**. The final value of ε in the SHA* method is closer to unity than in the AWA*, which means that SHA* has a better convergence to A^*. N.b. the initial epsilon value was chosen to be 4. Since the final solution was found in a time lower than 0.1[s] this means the search stopped in both cases because it found a resolution-optimal solution. Graph trees for both scenarios and methods are presented in figure 7.

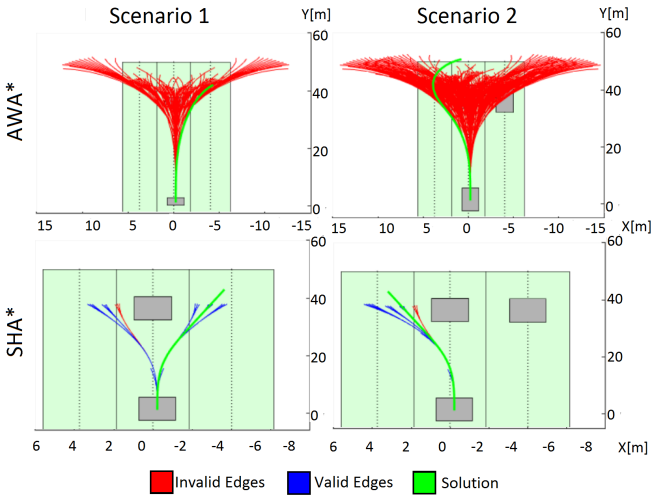


Fig. 7. Graph trees generated in the first solution found for both scenarios and methods

Note that a grouping of trajectories into trajectory sets is also responsible for a reduction in the collision tests runtime. In fact, we compute a preliminary collision test where a circle segment geometry enclosing a trajectory set is tested against moving obstacles and road boundaries.

We conclude that SHA* is better in terms of runtime

for the first solution and reduced number of invalid explorations. Nevertheless, it has a higher memory burden. So, one could agree that computing a computationally complex environment-sensitive heuristic allows a better informed search.

V. CONCLUSIONS AND OUTLOOK

The proposed contingency planner with an anytime suboptimally bounded search using a triple set organization and an environment-informed heuristic with the help of the precomputed motion primitives can generate reasonable and fast solutions for the tested scenarios, improving upon existing anytime methods. It also fulfils the project objective of finding a solution in a time frame of 0.1 [s], furthermore finding a resolution-optimal solution.

The major achievements of the contingency planner developed are the following:

- The creation of an anytime search informed heuristic with environment information, which maintains bounded suboptimality and resolution completeness properties.
- The construction of a more computationally complex heuristic, which, although more computationally expensive, leads to an overall decreased computation time, due to its tendency to explore free space.
- The construction of motion primitives that are suited for the emergency situations in the tested scenarios and that are not a burden in terms of memory to the online search.

Concerning the combinatorial accessment, a new formulation of the parameter α could be utilized for a different cluster of scenarios. It would be interesting to evaluate whether an increased number of samples would accelerate the search and at which point the computation of the inflated part does not compensate a reduction in the number of invalid explorations and runtime. Furthermore, it would be interesting to evaluate, how the increased computational load of the obstacle sensitive heuristic balances against a decreased runtime resulting from the reduction of invalid explorations, when the branching factor is increased.

VI. ACKNOWLEDGEMENTS

The authors gratefully acknowledge financial support by the European Commission project UnCoVerCPS under grant number 643921.

This work was supported by the FCT project (VID/EEA/50009/2013).

The experiments were conducted using "DLR Automotive Motion Planner". Existing software packages from DLR were utilised, anytime planning with AWA*, collision detection with OBBs, interfaces for sampling and cost strategy.

REFERENCES

[1] J. Barraquand and J.-C. Latombe, "Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles," *Algorithmica*, vol. 10, no. 2-4, pp. 121–155, 1993.

[2] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[3] R. Knepper, M. T. Mason *et al.*, "Path diversity is only part of the problem," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 3224–3229.

[4] L. H. Erickson and S. M. LaValle, "Survivability: Measuring and ensuring path diversity," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2068–2073.

[5] M. Pivtoraiko and A. Kelly, "Kinodynamic motion planning with state lattice motion primitives," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 2172–2179.

[6] D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," in *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*, 2005, pp. 9–18.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[8] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," in *Advances in Neural Information Processing Systems*, 2003, p. None.

[9] M. Likhachev and S. Koenig, "A generalized framework for lifelong planning a* search." in *ICAPS*, 2005, pp. 99–108.

[10] J. T. Thayer and W. Ruml, "Anytime heuristic search: Frameworks and algorithms," in *Third Annual Symposium on Combinatorial Search*, 2010.

[11] E. A. Hansen and R. Zhou, "Anytime heuristic search." *J. Artif. Intell. Res.(JAIR)*, vol. 28, pp. 267–297, 2007.

[12] S. Koenig, M. Likhachev, Y. Liu, and D. Furcy, "Incremental heuristic search in ai," *AI Magazine*, vol. 25, no. 2, p. 99, 2004.

[13] S. Koenig and M. Likhachev, "Real-time adaptive a*," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM, 2006, pp. 281–288.

[14] J. T. Thayer, W. Ruml, and E. Bitton, "Fast and loose in bounded suboptimal heuristic search," in *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08)*, 2008.

[15] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic a*," in *Seventh Annual Symposium on Combinatorial Search*, 2014.

[16] S. Russell, P. Norvig, and A. Intelligence, *A modern approach*. Citeseer, 1995, vol. 25.

[17] S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, and L. Nouveliere, "Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, no. 3, pp. 589–606, 2010.

[18] M. Bahram, A. Wolf, M. Aeberhard, and D. Wollherr, "A prediction-based reactive driving strategy for highly automated driving function on freeways," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 400–406.