Czech Technical University in Prague
Faculty of Electrical Engineering

Bachelor Thesis

Asynchronous Decentralized Prioritized Planning for Cooperative Vehicles

Matěj Vavřinec

Supervisor: Ing. Martin Schaefer

Study Programme: Open Informatics

Field of Study: Computer and Information Science

May, 2015

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne ....

Podpis autora práce

Poděkování

Rád bych na tomto místě poděkoval panu Ing. Martinu Schaeferovi za velice trpělivé vedení, vstřícný přístup a podnětné návrhy, které obohatily mojí bakalářskou práci. Tímto též děkuji za příležitost podílet se na zajímavém projektu AgentDrive.

Abstract

Modern cars are being equipped with technologies enabling partial automation of driving. It is expected, that the autonomous vehicles are going to take full control from the humans in the near future. This thesis deals with modification and subsequent application of the Prioritized Planning as a coordination mechanism for models of the autonomous vehicles in the Road Traffic Domain. The proposed method is tested on a traffic simulator. The experiments show, that the method is able to control an artificial traffic on the simulators without any collisions. The model of the environment was simplified, however the results promise an interesting future research in the field of multi-agent coordination systems for autonomous vehicles.

Keywords
autonomous vehicles, coordination, prioritized planning, multi-agent pathfinding

Abstrakt

Moderní automobily jsou vybavovány technologií umožňující částečnou automatizaci řízení. Očekává se, že autonomní vozidla v blízké budoucnosti plně převezmou řízení od lidí. Tato bakalářská práce se věnuje úpravě a následné aplikaci prioritního plánování jako koordinačního mechanismu pro autonomní vozidla na doménu silničního provozu. Navržená metoda je následně otestována na dopravním simulátoru. Experimenty ukazují, že tato metoda je schopna bezkolizně řídit umělý provoz na simulátorech. Model prostředí byl sice zjednodušen, nicméně výsledky slibují zajímavý budoucí výzkum na poli multiagentních koordinačních systémů pro autonomní vozidla.

Klíčová slova
autonomní vozidla, koordinace, prioritní plánování, multiagentní hledání cest

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Matěj V a v ř i n e c

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Prioritizované plánování pro kooperativní vozidla

Pokyny pro vypracování:

1. Nastudujte Asynchronní Decentralizované Prioritní Plánování a související plánovací přístupy.
2. Diskutujte aplikaci ADPP v doméně silničního provozu.
3. Integrujte ADPP do platformy AgentDrive pro simulační validaci.
4. Navrhněte adaptaci ADPP pro aplikaci v doméně silničního provozu.
5. Poskytněte experimentální zhodnocení navržených modifikací.

Seznam odborné literatury:

[1] LaValle, S. M. (2006). Planning algorithms. Cambridge University Press.
[2] Čáp, M., Novák, P., Kleiner, A., & Selecký, M. (2014). Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots. arXiv preprint arXiv:1409.2399.
[3] Schaefer, M. Collision Avoidance of Highway Traffic, (2014). Master's Thesis. Czech Technical University in Prague, Czech Republic
[4] Čáp, M., Novák, P., Selecky, M., Faigl, J., & Vokřínek, J. (2013, November). Asynchronous decentralized prioritized planning for coordination in multi-robot system. In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on (pp. 3822-3829). IEEE.
[5] Čáp, M., Novak, P., Vokřínek, J., & Pěchouček, M. (2012). Asynchronous decentralized algorithm for space-time cooperative pathfinding. arXiv preprint arXiv:1210.6855.

Vedoucí bakalářské práce: Ing. Martin Schaefer

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 14. 1. 2015

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Matěj V a v ř i n e c

Study programme: Open Informatics

Specialisation: Computer and Information Science

Title of Bachelor Project: Asynchronous Decentralized Prioritized Planning for Cooperative Vehicles

Guidelines:

1. Research Asynchronous Decentralized Prioritized Planning and related planning approaches.
2. Discuss an application of the ADPP in road traffic domain.
3. Integrate ADPP in AgentDrive platform for validation in simulation.
4. Propose an adaptation of ADPP for road traffic domain application.
5. Provide an experimental evaluation of the proposed modifications.

Bibliography/Sources:

[1] LaValle, S. M. (2006). Planning algorithms. Cambridge University Press.
[2] Čáp, M., Novák, P., Kleiner, A., & Selecký, M. (2014). Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots. arXiv preprint arXiv:1409.2399.
[3] Schaefer, M. Collision Avoidance of Highway Traffic, (2014). Master's Thesis. Czech Technical University in Prague, Czech Republic
[4] Čáp, M., Novák, P., Selecky, M., Faigl, J., & Vokřínek, J. (2013, November). Asynchronous decentralized prioritized planning for coordination in multi-robot system. In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on (pp. 3822-3829). IEEE.
[5] Čáp, M., Novak, P., Vokřínek, J., & Pěchouček, M. (2012). Asynchronous decentralized algorithm for space-time cooperative pathfinding. arXiv preprint arXiv:1210.6855.

Bachelor Project Supervisor: Ing. Martin Schaefer

Valid until: the end of the summer semester of academic year 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 14, 2015

Contents

List of Figures

List of Tables

Chapter 1

Introduction

More than a million people die on the world's roads every year and the costs of the crash consequences runs to billions of dollars. With these statistics it is a real challenge for the automotive industry to ensure the safety of the road traffic participants. One of the possible solution, in which it is being invested by the car manufactures, are the Advanced Driver Assistant Systems (ADAS). These systems are able to help the driver to prevent accidents. The systems detect obstacles or other vehicles in a close-range using sensors and warn drivers about possible dangerous situations. Eventually the systems can act autonomously in order to prevent a collision.

Another possible solution are the driver-less or autonomous cars. These cars, as the name states, can act completely autonomously without any help from the human driver. These vehicles could potentially decrease significantly the collision rate, as almost 95% of all accidents are caused by the human factor[1]. The benefit of the fully autonomous vehicles is the complete elimination of the human factor. However the challenge is to provide a collision-free autonomous vehicle able to drive through the existing road traffic. Development and research for these types of vehicles is motivated by the DARPA[2] challenges. All these vehicles are equipped with a collision avoidance system, that is able to react to every possible situation in the traffic and prevent any collision by steering the vehicle safely.

The collision avoidance systems can be divided into two main approaches, the single-vehicle collision avoidance based on observation of the surroundings and the multi-vehicle coordination systems based on the Vehicle-to-Vehicle (V2V) communication and plan sharing. The former approach is core of a robust vehicle coordination. Fast, reactive methods depending solely on the information from the local sensors

[1] <http://www-nrd.nhtsa.dot.gov/Pubs/811059.PDF>
[2] <http://www.theroboticschallenge.org>

about the surroundings are irreplaceable for the low level collision avoidance (e.g. DRCA [Lalish et al., 2008], ORCA [Van Den Berg et al., 2011] or a method based on the safe-distance [Schaefer, 2014]).

With the presence of communication, the individual vehicle are able to share information, such as their speed, heading but also their following intentions i.e change lane, go left on a junction. The Vehicle-to-Vehicle communication mechanism is an already implemented solution, that is being adopted by the automotive industry [Ye et al., 2008]. The communication and plan sharing enables more sophisticated coordination of the vehicles and methods utilising these advantages are able to solve more planning problems, than the reactive collision avoidance methods [Čáp et al., 2014].

In this thesis we present a modification of the Asynchronous Decentralized Prioritized Planning [Čáp et al., 2014] algorithm as the coordination mechanism for the autonomous vehicles. We assume an existing communication system used by the vehicles, that is able to share plans. The modified version of the algorithm is then evaluated in a set of scenarios with an artificially generated traffic.

Chapter 2

Related Work

2.1 Multi-Agent Systems

The Multi-Agent System (MAS) is a system composed of multiple intelligent autonomous entities (agents) within an environment. These agents can obtain information about the environment using sensors and perform number of actions using actuators. The actuators change the environment according to the action performed and this way the agents can solve complex tasks or coordinate their actions.

The system has several important properties:

- **Autonomy**: the agents in the system are at least partially independent.
- **Locality**: the agents are limited in the amount of information they can obtain from the environment using sensors.
- **Decentralization**: there is no central controlling entity.

2.1.1 Agents

The MAS consists of multiple agents. A single agents is, to some extent, an autonomous entity as a component of the MAS. The definition of the agent according to [Ferber, 1999] follows:

> An agent can be a physical or virtual entity that can act, perceive its environment (in a partial way) and communicate with others, is autonomous and has skills to achieve its goals and tendencies. It is in a multi-agent system (MAS) that contains an environment, objects and agents (the agents

being the only ones to act), relations between all the entities, a set of operations that can be performed by the entities and the changes of the universe in time and due to these actions.

2.2 Principles of the Asynchronous Decentralized Prioritized Planning

The Asynchronous Decentralized Prioritized Planning (ADPP) [Čáp et al., 2014] algorithm is based on the Centralized Prioritized Planning (PP) [Erdmann and Lozano-prez, 1987]. In the classical prioritized planning each agent is given a unique priority $p \in \mathbb{N}$. Then the agents are sorted by the their priority either in ascending or descending order depending whether the highest priority is set to be the highest number or the lowest. Planning proceeds by taking the higher priority agents first. Every agent consider the higher priority agents to be moving obstacles. This ensures, that the resulting trajectory for all agents will be conflict-free.

2.2.1 ADPP as a Modification of Prioritized Planning

In the Asynchronous Decentralized version of the Prioritized Planning (ADPP) the former principles are extended with two features, particularly decentralization and asynchronism. In the road traffic domain, decentralization is a given natural property of the environment, although both these features might be beneficial for overall computational time.

The benefit of the decentralization is, that all agents can start planning simultaneously, whereas with one centralized planner, the planning is usually sequential because the planner can process one agent at time[1]. Since all the agents start planning at the same time, the prioritized approach is violated. To preserve it a system of plan sharing is introduced [Čáp et al., 2014, Velagapudi et al., 2010]. Each agent has the ability to *broadcast* and *receive* a plan. With this two abilities, the agents can share each other's plans and preserve the prioritized principle. From the definition of the prioritized planning, one can easily see, that it is sufficient to broadcast a plan only to the lower priority agents. Also it sufficient to receive plans only from the higher priority agents. In the synchronized version every agent broadcasts his

[1]The centralized planner can process multiple agents at time, but the planner has to divide it's resources between the individual agent, whereas in the decentralized version, the agents do not share any resources, because they are completely separate entities.

plan not before all other agents have finished planning. Once an agent receives all new plans from higher priority agents, he starts new replanning round. If the new replanned plan differs from the previous one, it is broadcast in the next round and the whole process starts again. This may lead to many replanning tasks, but as it is shown in the experimental results in [Čáp et al., 2014], these replanning demands don't happen too often and the decentralized version still outperforms the centralized one while keeping the same properties.

Asynchronism also modifies the algorithm. The improvement is not to wait for every agent to finish planning and then start resolving the received plans, but instead to resolve the received plan as soon as they're obtained. This speeds up the whole planning process by getting rid of the synchronization delay. Also if an agent has relatively long planning compared to the agents, but his trajectory is not in collision with any other or few other agents, this slows down the planning time, since all the other agents have to wait for this slow-computing agent to finish his planning round. In the asynchronous version however, they can start resolving the new plans while this slow agent is still planning and speed up the whole planning process. Figure 2.1 shows a diagram of comparison of these three different versions of the Prioritized Planning algorithm.

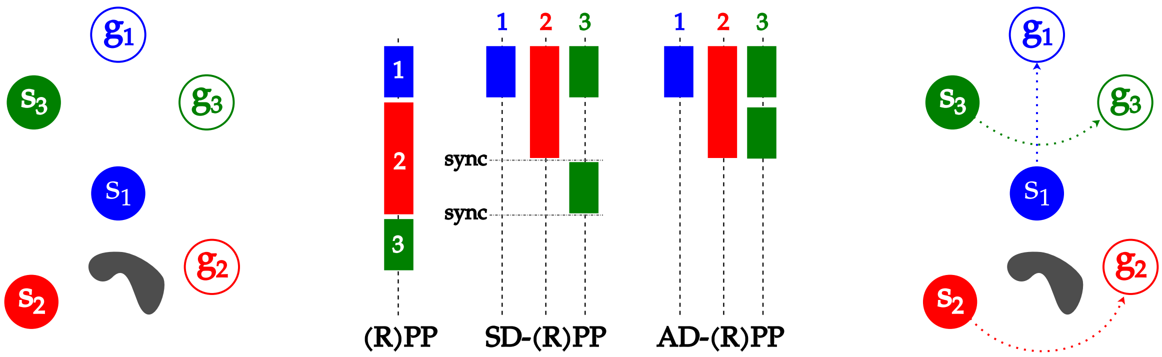


Figure 2.1: Comparison of the overall planning times of the Prioritized Planning, Synchronized Decentralized Prioritized Planning and Asynchronous Decentralized Prioritized Planning on a simple scenario [Čáp et al., 2014].

2.2.2 Revised Version of the Prioritized Planning

A key requirement of any multi-agent path planning algorithm is to ensure that there always exists a collision-free plan for every agent in every situation. Unfortunately, in the classical Prioritized Planning this requirement is difficult to guarantee, since the higher-priority agents completely ignore the trajectories of the lower-priority agents when planning their trajectories. One way to ensure that there will always be a collision-free trajectory for every agent is to allow all lower-priority agents to perform a safe-maneuver in all situations, such as to fully stop. As result, the lower-priority agents then appear as obstacles for the higher-priority agents. This version of Prioritized Planning is proposed in [Čáp et al., 2014] and called Revised Prioritized Planning (RPP).

2.2.3 Collision-free Trajectory Generation

The ADPP algorithm is an abstract tool providing fast, efficient and reliable approach of coordinating multiple autonomous agents with collision-free trajectories. However it doesn't explicitly specify how to obtain such collision-free trajectory for the individual agents. The trajectory generator works as a "*black-box*" for the ADPP algorithm.

The algorithm for the collision-free trajectory generation must have one property defined as follows:
Given a

- Description of the environment E
- Initial and goal position I, G
- Set of static obstacles O
- Set of trajectories of the higher-priority agents T

The algorithm must provide a trajectory from the initial position I to the goal position G avoiding both the static obstacles O and the dynamic obstacles T in the form of trajectories of the higher-priority agents. With revised principle described in the Section 2.2.2, the trajectory must also avoid the initial positions of the lower-priotity agents. Any algorithm satisfying this condition can be used as the trajectory generator for the individual agents in the ADPP algorithm.

In this thesis we have adopted the A^* algorithm[2] used in the code [Čáp et al., 2014] as the trajectory generator. The A^* algorithm works on a general weighted graph which is convenient for the road traffic domain, since the road network can be easily converted to a directed weighted graph as described in the Section 3.2.1.

However, the spatial graph is not sufficient when dealing with collision avoidance of moving objects throughout time and space. For this reason, the spatial graph is

[2] <http://en.wikipedia.org/wiki/A*_search_algorithm>

extended by adding an additional dimension representing the time axis. The resulting graph is still a directed graph with additional nodes in the extended time-space. Figure 2.2 shows two trajectories in the extended time-space with the underlying spatial graph.

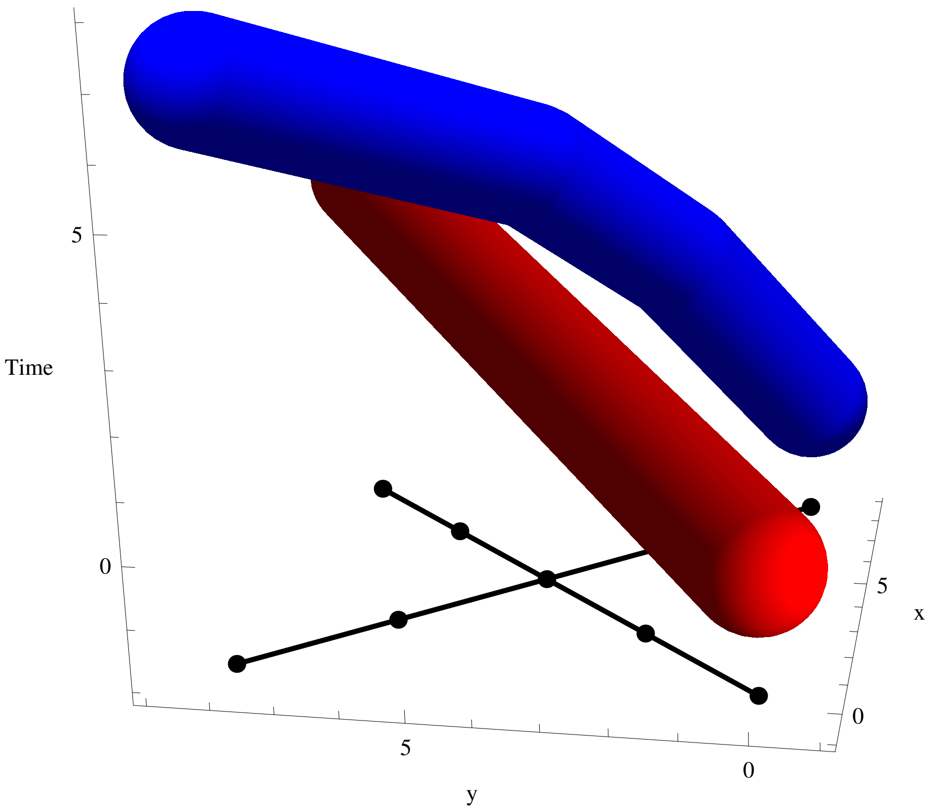


Figure 2.2: Two trajectories in the extended time-space. The space of the tubes reflect the shapes of the moving objects – moving circles. The underlying spatial graph is depicted in black.

With the added dimension, the resulting graph can be much larger than the spatial graph. Storing the whole graph in memory would be nearly impossible. Also the graph can be really complex. For this reason, the time-space graph is generated on-demand. The expansion algorithm is given a current node as tuple $n = (x, y, t)$, where x and y are the spatial coordinates of the node and t is the time coordinate. Also it is given a set of velocities V available for the current agent. The time-space node is expanded by applying all possible velocities from the set V and a "waiting-edge". Algorithm 1 shows the expansion of the one time-space node.

```
Input: Directed spatial graph G, finite set of velocities S, set of dynamic
       obstacles O, time-space vertex v, time step t_s
Output: Set of time-extended vertices V
V ← ∅;
// Get all outgoing spatial edges of the vertex
foreach e ∈ OutgoingEdges(v, G) do
    foreach s ∈ S do
        // Calculate time to reach the end vertex of the current
        // edge with the current speed
        t ← Time(v) +||e||_2/s;
        v' ←TimeVertex(End(e), t);      // Create time-extended vertex
        if SatisfyContraints(v, v', O) then
            V ← V ∪ {v'};
        end
    end
    v_wait ←TimeVertex(v, Time(v) +t_s);         // Create a waiting edge
    if SatisfyContraints(v, v_wait, O) then
        V ← V ∪ {v_wait};
    end
end
return V
```

Algorithm 1: Pseudo code of the time-extension algorithm.

The time step constant t_s can be set to an arbitrary value. Smaller values lead to longer plans and for that longer planning times. On the other hand smaller values can produce more precise plans.

Chapter 3

Problem Specification

This chapter describes the problem of multi-agent path planning. Also the various specific properties of the road traffic domain are introduced. Finally the model of the road traffic environment is covered in the last section.

3.1 Multi-agent Path Planning

The problem of multi-agent path planning is a reasonable formalization of coordination mechanism for road vehicles. In the previous sections we introduced (semi) autonomous road vehicles equipped with a vehicle-to-vehicle communication. These types of vehicles enables more sophisticated methods for coordination, than a simple collision checking.

The problem of multi-agent path planning can be formally stated as follows: given a set of n agents $A = 1, 2, \ldots, n$ and a planning space $P \subseteq \mathbb{R}^m$, $m \in \mathbb{N}$ usually a $\mathbb{R}^2$ or $\mathbb{R}^3$, find a trajectory $t_i : [0, \infty) \rightarrow P$ from starting state $S_i \in P$ to a goal state $G_i \in P$ for each agent i, so that the trajectory does not collide with any other trajectory t_j, $j \neq i$.

3.2 Road Traffic Domain

The Road traffic domain has a set of specific properties, that needs to be taken into account when dealing with designing a reliable coordination mechanism.

One of the most important properties of the road traffic domain, is decentralization. When dealing with coordination mechanisms in this domain, there cannot be any centralized authority responsible for coordination. Instead all vehicles in the traffic act as autonomous entities.

The road traffic domain is also a very heterogeneous environment. In a city traffic, there are not only vehicles, but also pedestrians and trams sharing the same roads. Also with the presence of the autonomous vehicles, there can be a number of different types of coordination mechanisms controlling these vehicles. The reliable coordination system must be able to manage this very heterogeneous environment and provide a collision-free trajectory in every possible situation.

3.2.1 Environment Model

In the road traffic domain, a valid trajectory respects the underlying road network. To be able to create such a trajectory, a model of the road network is used as base for the path planning algorithm.

In the *AgentDrive* project the Simulator for Urban Mobility[1] (SUMO) is used heavily for operations with the road network. The SUMO project is a road traffic simulator designed for simulating large road traffic networks in the size of an entire city. It contains a tool for importing a road network structure data from a map data (e.g., OpenStreetMaps[2]).

In the SUMO road network model, each road is represented as an `edge`. Each edge has multiple or single `lane` representing an actual lane in the road e.g. at highway. `Edge`s can be connected either directly to each other, to form more complex structures such as curved roads, or via `junction`s. At `junction`, each incoming `lane` is connected to one or more outgoing `lane`s. Figure 3.1 shows the diagram of a SUMO `junction`.

Although the ADPP algorithm works on a general graph and the SUMO road network is a graph, the resolution is not sufficient for a precise collision-free trajectory planning. The road network allows finding a path as a sequence of roads to follow, similar to a turn-by-turn navigation[3]. However this type of planning doesn't allow to build more precise plan, that considers also local collision avoidance.

To be able to create such plan, the SUMO road network is discretized by an arbitrary constant and then a directed graph is build on top of this discretized space. The constant can be set to a larger value to create more sparse graph for more

[1] <`http://sumo.dlr.de/wiki/Main_Page`>
[2] <`http://www.openstreetmap.org`>
[3] <`http://en.wikipedia.org/wiki/Turn-by-turn_navigation`>

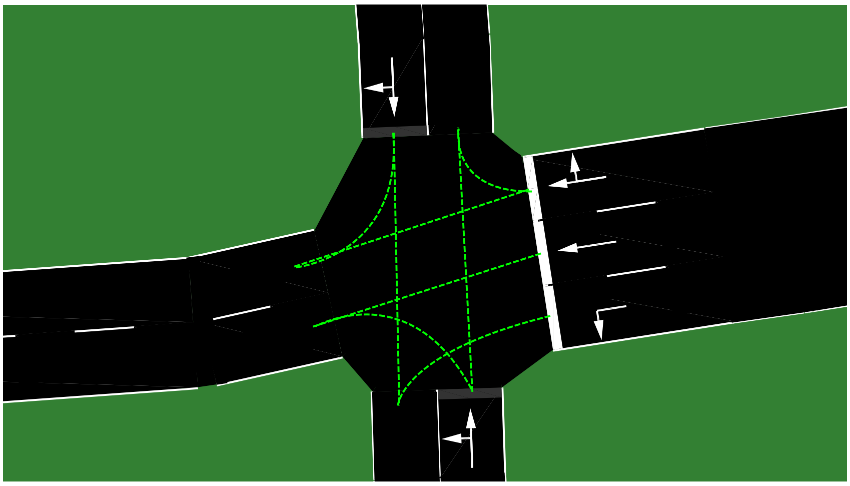
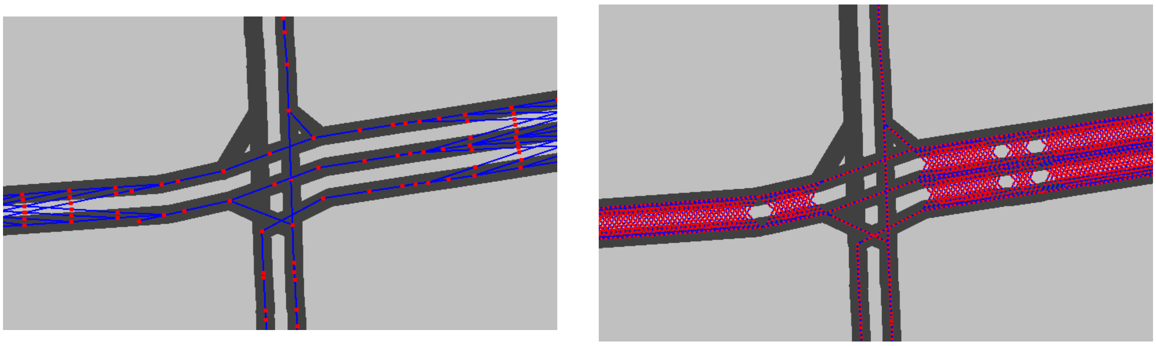


Figure 3.1: Junction as represented by SUMO. The green dashed lines represent connections from the incoming lanes to the outgoing ones. The visualization is exported from the graphical user interface of the SUMO toolkit.

approximate plans or to a smaller value for plans with precisions in fractions of metres. The directed spatial graph is built by doing a BFS search starting from the agent's initial position and proceeding in the discretized space. This guarantees, that the resulting detailed graph contains only reachable nodes for the agent. Also if an `edge` contains multiple parallel `lanes`, a lane-changing edges are added to the resulting graph. Figure 3.2 shows some of the mentioned features of the discretization.



(a) Discretization of the SUMO road network. The discretization factor is set to a large value, i.e. number of metres.

(b) The discretization factor is approximately 10 times smaller than in the Figure 3.2a.

Figure 3.2: The *SUMO* road network model and the directed spatial graph built on top of the road network model. One can see, that there are no nodes in the top left lane, because this lane is not reachable for the current agent from his starting position.

Chapter 4

Application of the ADPP Algorithm to the Road Traffic Domain

This chapter describes the process of application the ADPP algorithm to the road traffic domain. In the first sections the proposed modifications that enable the application, are described. In the last section the problems arisen with the modifications are described.

4.1 Proposed Modifications

This section describes the proposed modification of the original ADPP algorithm. These modifications were necessary to introduce to be able to meet the requirements of the road traffic domain described in the Section 3.2.

4.1.1 Dynamic Constraints

In the time-expansion Algorithm 1, the dynamic model of the agent is represented very simply as a finite set of velocities V. This is sufficient if the dynamic model is not important for the application of the algorithm or if the set V contains only few velocities. Otherwise it can be easily seen, that with this model the agent is able accelerate or decelerate between any two velocities from the set V. Another problem is that every node in the time-space graph has the waiting-edge. This doesn't model the road traffic domain well, because the acceleration and deceleration of the individual vehicles is limited. Also the simplification that the vehicle is able to fully stop from generally any velocity and remain stationary for one time step t_s is not desirable.

To have more realistic dynamic model reflecting the road traffic domain the Algorithm 1 is modified. Two constants are introduced:

- Acceleration factor a_{acc}
- Deceleration factor a_{dec}

These two constants are used to model a simplified version of the vehicle physics model with constant acceleration as well as deceleration. However with the current planning space as a directed graph with nodes in a 3-dimensional space[1], these constraints cannot be directly applied. The graph nodes lacks the information about the speed, which is necessary for the application of the constant acceleration constraints. The planning space needs to be extended with an additional dimension providing the information about the velocity.

The modified version of the time expansion algorithm is proposed in the Algorithm 2.

The Algorithm 2 differs from the original Algorithm 1 in three major things.

1. The dimensionality of the resulting directed graph has changed. In the modified version, each vertex in the graph represents a point in the 4-dimensional space $S_{planning} \subseteq \mathbb{R}^4$. Each vertex can be represented by a tuple $v = (x, y, s, t)$, where x and y are the spatial coordinates of the point, s is the velocity and t is the time of the vertex.

2. On the line 3 the current vertex is expanded in accordance to the dynamic constraints. The expansion function works as follows. Instead of expanding all possible velocities as the original Algorithm 1, the current vertex $v = (x, y, s, t)$ is expanded to the spatial vertex $v_s = (x', y')$ using the following 3 options:

   (a) The current velocity s is kept.

   $$v' = (x', y', s, t + \frac{||E_s(v, v_s)||_2}{s}) \tag{4.1}$$

   $E_s(v, v')$ denotes the spatial edge from the vertex v to the vertex v'.

   (b) The maximal acceleration a_{acc} is applied. The time t' needed for the vehicle to travel is calculated from the Newton second law of motion:

   $$d_s = s \cdot t' + \frac{1}{2} a_{acc} \cdot t'^2 \tag{4.2}$$

[1] Two dimensions representing space axes and one representing time axis

Input: Directed spatial graph G
Finite set of velocities S
Set of dynamic obstacles O
Time-velocity-space vertex v
Time step t_s
Output: Set of time-extended vertices V

```
V ← ∅;
// Get all outgoing spatial edges of the vertex
foreach e ∈ OutgoingEdges(v, G) do
    // Expand the current vertex according to the dynamic
    // constraints
    V' ← DynamicConstraints(v, End(e));
    foreach v' ∈ V' do
        if SatisfyContraints(v, v', O) then
            V ← V ∪ {v'};
        end
    end
    // Create a waiting-edge iff the agent is stationary
    if Velocity(v) = 0 then
        v_wait ←TimeVertex(v, Time(v) +t_s);
        if SatisfyContraints(v, v_wait, O) then
            V ← V ∪ {v_wait};
        end
    end
end
```

Algorithm 2: Pseudo code of the modified time-extension algorithm with the improved dynamic model.

where $d_s = ||E_s(v, v_s)||_2$ is spatial Euclidean distance from the vertex v to v'. After obtaining the time t, the end velocity s' is computed from the equation:

$$s' = a_{acc} \cdot t' + s \tag{4.3}$$

If the end velocity is greater than the maximal velocity s_{max} of the vehicle, the time t is recomputed using the following equations:

$$t' = \frac{2||E_s(v, v_s)||_2}{s_{max} + s} \tag{4.4}$$

$$s' = s_{max} \tag{4.5}$$

Then the expanded vertex can be constructed as follows:

$$v' = (x', y', s', t + t') \tag{4.6}$$

(c) The maximal deceleration a_{dec} is applied. The time t' needed for the vehicle is again computed from the Equation 4.2, the only difference is that the positive acceleration a_{acc} is replaced by the negative deceleration a_{dec}. After the time t' is obtained the end speed s' is again computed from the Equation 4.3.

Since the deceleration $a_{dec} < 0$, the solution to the quadratic equation can be a complex number if the discriminant $D = s^2 + 2a_{dec} \cdot t' < 0$. This happens when the deceleration is too large compared to the speed so that the vehicle would never reach the destination vertex. If the $D < 0$, then the end speed is computed from the following equation:

$$t' = \frac{2||E_s(v, v_s)||_2}{s} \tag{4.7}$$

$$s' = 0 \tag{4.8}$$

The expanded vertex is again constructed from the Definition 4.6.

3. To prevent the vehicle to be able to stop from any possible speed a condition is introduced on the line 9. This condition allows the vehicle to remain stationary on the current position if and only if the vehicle is not moving.

4.1.2 Planning horizon

The original ADPP algorithm was designed to solve problems like navigating a set of robots from their starting positions to their respective goal positions in a closed environment such as a warehouse. Figure 4.1 shows such a problem. In this problem instance, the overall planning time is not the most important thing. The highest priority is to provide collision-free trajectory for all agents from their start to their goal positions and thereby complete the task successfully. Also the agents are able to plan the whole trajectory and there is no need to replan it during the plan execution, since the environment is static and doesn't change except for the positions of the individual agents.

In the road traffic domain however, the situation is much different. The environment is dynamic and very large. The environment here represents a whole or a part

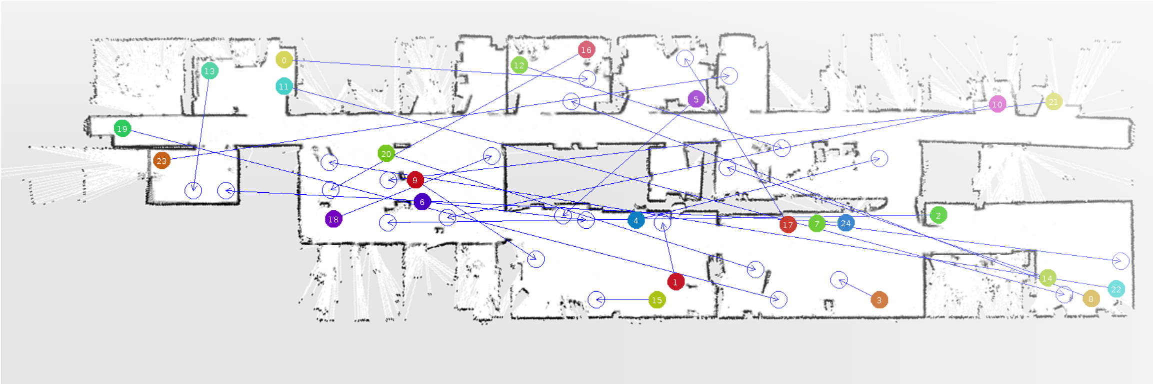


Figure 4.1: Office corridor scenario [Čáp et al., 2014]

of a whole country. Planning whole collision-free trajectory in this very large environment is a very time-demanding task, especially given the trajectory should planned with precision in metres. In the road traffic domain, the collision-free trajectory should be planned within hundreds of milliseconds or less to ensure that the vehicle coordination system is able to react to a new situation in time and act accordingly. Also the assumption that an agent can communicate with every other agent in the environment is generally not possible. V2V communications are without an exception a local communication solutions – connecting only local vehicles in the defined communication range. Considering this limitations of the environment and communication technology, planning with a limited time horizon is a reasonable concept. The horizon must be set according to the communication range, so that no vehicle outside the communication range can be in collision with the agent's plan with the limited planning horizon.

Since we're using A^* as the algorithm used for finding the collision-free trajectory on a directed graph, there needs to a defined a goal vertex for the algorithm to finish. In the original ADPP algorithm without the planning horizon, the goal vertex is easily defined as the position of the goal for each agent. However, when the planning horizon is introduced this goal state couldn't be generally reached, because the planning horizon can be smaller than the time needed for the agent to travel from the starting position to the goal position. Instead a new intermediate goal state is introduced to model the planning horizon.

Given a current state in the planning state space $\mathbb{R}^4$ as a tuple: $s = (x, y, v, t)$, where x and y is a spatial position of the state, v is the speed in that state and t is the time of the state. Also let h be a time horizon i.e. the time that the algorithm should plan ahead. Then the intermediate goal can be defined as a predicate $G : s \to \{0, 1\}$ as follows:

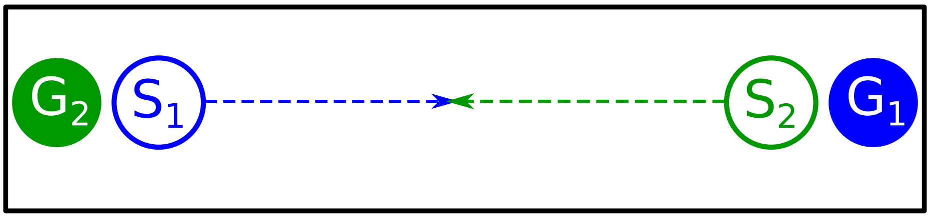
$$G(s) = \begin{cases} 1 & \text{if } time(s) >= h \vee goal(s) \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

where $time : s \rightarrow \mathbb{R}$ returns the time of the state s. The $goal : s \rightarrow \{0, 1\}$ is the predicate from the original ADPP algorithm, choosing only states with the spatial position equal to the goal position. This goal predicate is sufficient for finding the intermediate goals as well as the global goal.

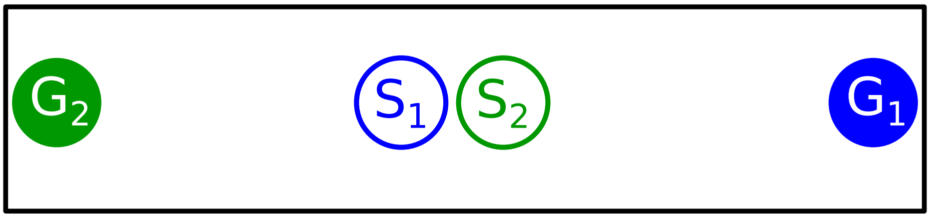
4.1.3 Safe maneuver

As mentioned in the Section 2.2.2, the classical Prioritized Planning algorithm doesn't guarantee, that there will always exist a valid trajectory for every agent. The solution for this problem in the original ADPP algorithm is proposed in [Čáp et al., 2014] by introducing the revised version of the Prioritized Planning. In the revised version, each agent has to avoid the starting positions of the agents with lower priority and the goal positions of the agents with higher priority. In other words, each agent consider the starting positions of the lower-priority agents and the goal positions of the higher-priority agents as static obstacles. It can be proved, that if there exists a trajectory avoiding these positions for each agent, there exists a conflict-free trajectory for every agent.

However when using the planning horizon, the proposition mentioned above doesn't hold anymore. Figure 4.2 shows a situation when using the original revised version with the planning horizon fails to find a conflict-free trajectory. The principle of the revised prioritized planning must be adapted considering also the planning horizon. One way to ensure, that there will always be a collision-free trajectory available for every agent is to always stop the planning process in a state, where it is guaranteed, that there will be at least one collision-free trajectory available for the next planning round. In this instance of the problem it can be seen, that if an agent can stay at his current position for time equal to the planning horizon, the intermediate goal in the Equation 4.9 will be achieved. Now the revised principle of the prioritized planning must be modified, so that the higher-priority agents allow the lower-priority ones to perform the full-stop maneuver and to stay on the position for unlimited time. Since the planning horizon was introduced, the goal predicate from 4.9 must be modified to reflect the newly constructed conditions. Because the agent must always end his plan in a full-stop state, the modified goal predicate $G_{stop} : s \rightarrow \{0, 1\}$ can be defined as follows:



(a) A scenario where the original Revised principle with the planning horizon fails to provide a collision-free trajectory. In the first planning round each agent plans his trajectory as depicted by the respective arrows. The evolved situation is described in the Figure 4.2b.



(b) The evolved situation from the Figure 4.2a. The agents have executed their plans, but are in the situation where no plan exist, because each agent has some velocity $v > 0$.

Figure 4.2: Fail scenario of the original RPP with the planning horizon.

$$G_{stop}(s) = \begin{cases} 1 & \text{if } (time(s) >= h \vee goal(s)) \wedge velocity(s) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

where *time* and *goal* are the same functions as in 4.9 and $velocity : s \rightarrow \mathbb{R}$ returns the velocity of the state s. With this modification, there will always exist at least one collision-free plan for the next planning horizon for every agent, i.e. stay on his current position.

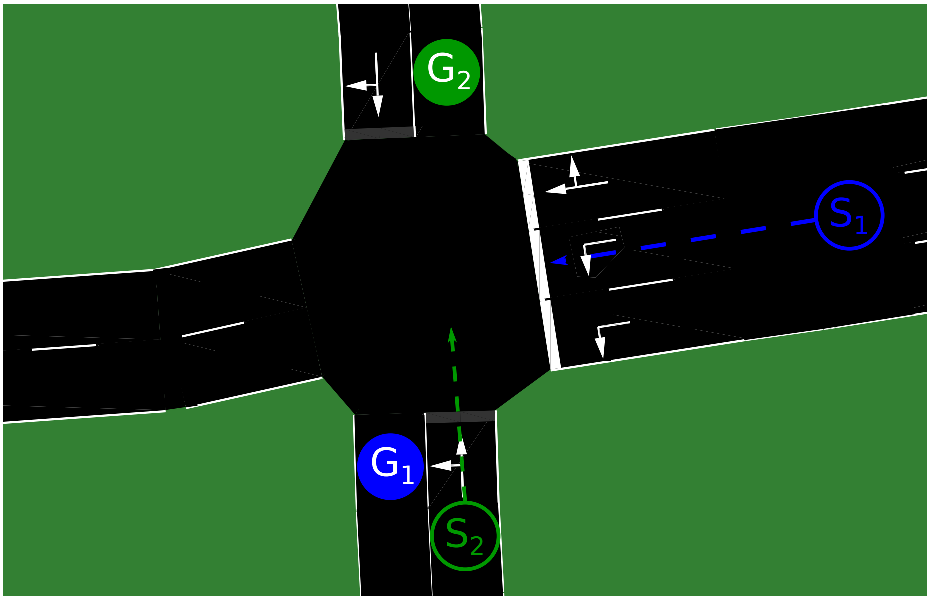
4.1.4 Deadlock Prevention

With the modifications introduced in 4.1.2 and 4.1.3 another problem arises. Consider a situation depicted in Figure 4.3a. In this situation, the blue agent with higher priority plans his trajectory to end in the full-stop state, with respect to the modification proposed in 4.1.3, just in front of the center junction. The green agent with lower priority plans his trajectory to end also in the full-stop, but right in the middle of the junction. In the next planning round, as seen in the Figure 4.3b, the blue agent plans his trajectory to stop right in front of the green agent, leaving him no option but to stay put on his position. This situation results in a deadlock, as both agents have only one collision-free trajectory available, namely to stay on their positions.

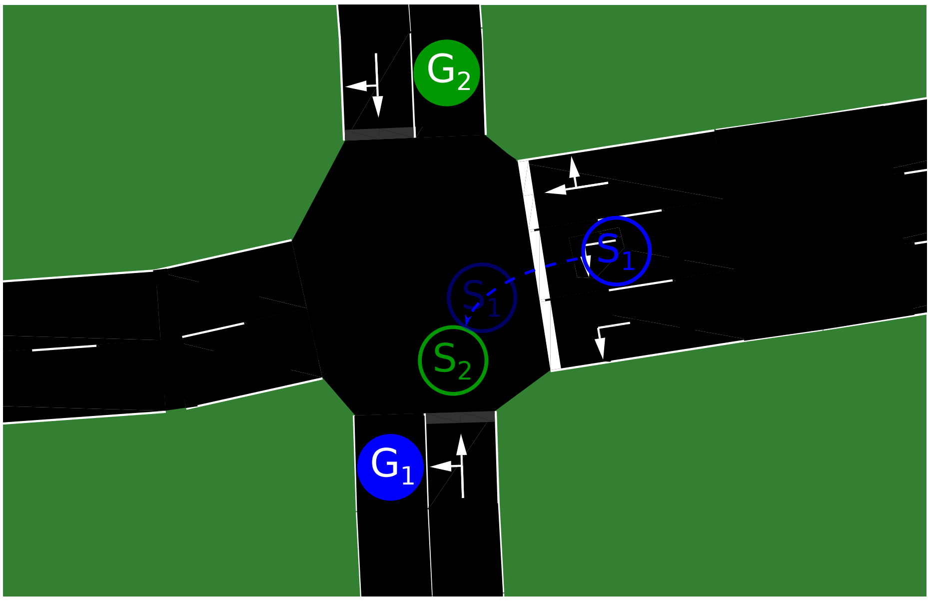
After a quick look at the whole situation again, one can easily see, that the deadlock arises when the lower-priority agent planned his full-stop safe-maneuver to end in the middle of the junction. There he blocks the other agent, because he has to avoid his position, thanks to the revised prioritized planning principle. The obvious solution to this deadlock problem is to prevent the agent from planning the trajectory so that the end state will be inside a critical area i.e. a junction. For this the goal predicate 4.10 needs to be adapted as follows:

$$G_{deadlock}(s) = \begin{cases} 1 & \text{if } ((time(s) >= h \land \neg critical(s)) \lor goal(s)) \land velocity(s) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

where $G_{deadlock} : s \to \{0, 1\}$ is the modified intermediate goal predicate, *time*, *goal* and *velocity* are the same functions as in 4.10 and $critical : s \to \{0, 1\}$ determines whether a state s lies in a critical section. This condition prevents the deadlock situation described earlier, while allows an agent to stop shortly in the critical section, provided he than immediately leaves the critical position. This models for example the situation on a junction during a left turn.



(a) The situation leading to the deadlock. The blue agent has higher priority than the green agent.



(b) Once the blue agents plans his trajectory, the situation leads to deadlock. Both agents are left with no option, but to stay put on their current positions.

Figure 4.3: Deadlock scenario with the planning horizon and the safe-maneuver.

Chapter 5

Evaluation

This chapter describes the tests of the proposed and implemented modification of the ADPP algorithm. The purpose of the testing is to check whether the proposed algorithm can serve as a reliable coordination mechanism for the road traffic domain. The environment and physical model is simplified to obtain preliminary results. The outcome of the tests is discussed after every test description.

5.1 Simulation

To obtain meaningful experimental results without deploying the algorithm to a real world car, the tests were performed on a simulator. For the purpose of the simulation and evaluation, the algorithm was integrated into the existing *AgentDrive* project [Schaefer and Vokřínek, 2015]. This project provides a platform for the simulation of the multi-agent based vehicle-coordination methods.

The architecture of the *AgentDrive* project is depicted in the Figure 5.1. The core of the platform is depicted in red. This consists of the scenario, coordination module, basic simplified physical simulation and simple 2D visualization module. The parts in blue represent an external physical simulator. Because the system is divided into this two major modules (the core and the external simulator), there are several advantages. The core itself is capable of a basic simulation with perfect execution of the plans. This is useful for the initial development of new coordination mechanism, because the developer is able to see the exact outcome of the coordination method. When the method is fully tested on the simplified simulation, a more complex simulator with realistic car physical model can be connected to the system and the method can be tested on the advanced simulator. With the more realistic simulator, new

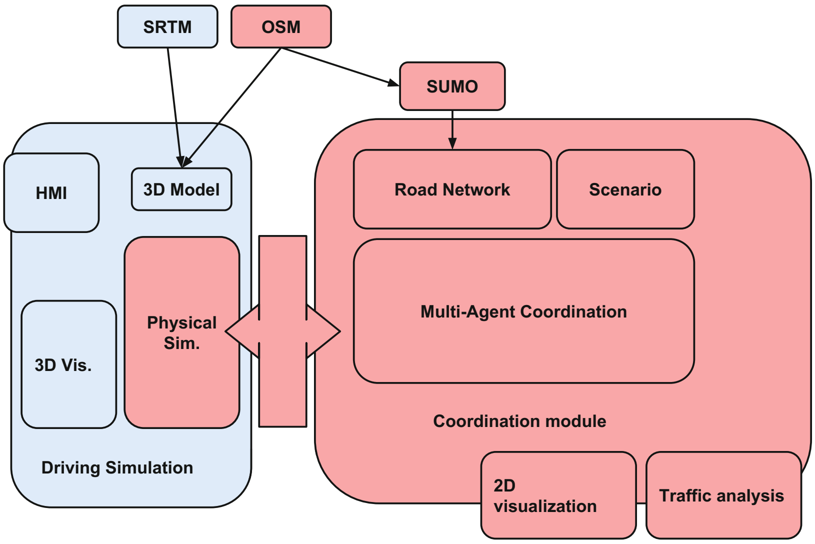


Figure 5.1: Architecture of the AgentDrive platform. The core parts are depicted in red. If the external simulator is connected, the blue modules are also used.

issues will be addressed, such as imperfect plan execution or complex physical model. This process simplifies the development of the coordination mechanism, since more important issues are developed first.

5.1.1 Integration to the AgentDrive Platform

To test the proposed algorithm from the Chapter 4, the algorithm was integrated to the AgentDrive platform specifically to the core modules from the Figure 5.1. Since the AgentDrive is an agent-based coordination platform, an agent was created. The agent is able to sense the trajectories of the higher-priority agents and the safe-maneuvers of lower-priority ones. The communication model is simplified in the sense that delays and uncertainty are not considered.

The collision-free trajectory generation logic is realized using the *Trajectory Tools* (TT) toolkit[1]. This toolkit facilitates tasks related to the trajectory planning mostly involving circular agents. It provides structures representing trajectories, implementation of the planning algorithms (A^*) and also additional visualization. Although the TT toolkit can be used without any modifications, some parts were necessary to be modified to implement the adjustments proposed in the Section 4.1. In particular, the dynamic constraints described in the Section 4.1.1 had to be implemented.

[1] <`https://github.com/mcapino/trajectorytools`>

5.2 General Description of the Scenarios

The method was simulated on several scenarios representing real-world situations. The scenarios consists of a road network (see Chapter 3) and a so called *flows* definition. Each flow represents a set of vehicles with the same route. The flow has three parameters listed below:

- The route definition in the form of a list of edges the vehicle should follow.
- Time span. The departure times of all vehicles will be distributed uniformly in this interval.
- Total number of vehicles in this flow.

Besides the road network and the flow definition, each scenario has additional parameters described in the following list:

- Maximal velocity s_{max} for every vehicle
- Maximal acceleration a_{acc} and deceleration a_{dec} factor for each vehicle
- Maximal number of vehicles in the scenario

5.3 Metrics for Evaluation

All tests are run using the simplified simulator from the core of the AgentDrive platform (see Section 5.1). This simulator assumes a perfect execution of the generated plans. Since the simulator does not consider any stochastic processes during the execution and the method is deterministic, the results are not of a statistical nature.

The evaluated properties of each testing scenario are listed below:

- Number of collisions
- Relative average speed for each flow with respect to the maximal speed s_{max}
- Relative planning time compared to the overall time of the simulation
- Simulation time

5.4 Results

The results of the individual test scenarios follows. Common parameters for all tests are listed in the Table 5.1.

Parameter	**Value**
Maximal speed s_{max} $[m/s]$	5
Maximal acceleration a_{acc} $[m/s^2]$	2
Maximal deceleration a_{dec} $[m/s^2]$	-2
Maximal number of vehicles in the scenario	50

Table 5.1: Common parameters for all tests.

5.4.1 Test 1

First scenario consists of a simple two-lane highway merging into one lane as depicted in the Figure 5.2. The first part of the test contains a single flow: 100 vehicles in the bottom-most lane. This scenario is compared with the next part of the test[2]. In the next part the vehicles are divided into two flows of 50 vehicles, each flow departing from different lane. The results obtained from these scenarios are in Table 5.2.

Figure 5.2: Scenario of the first test. Two lanes are merging into a single lane and the vehicles are able to change the lanes freely before the merge.

There were no collisions in both scenarios. The relative speeds in the lower lane are almost the same, which shows that the merging is done efficiently without slowing down the traffic in the lower lane. The vehicles in the upper lane have slightly lower average speed than the ones in the lower lane. The reason for this is that those vehicles in the upper lane have lower priority than the ones in the other lane. The lower-priority vehicles have to slow down and let the higher-priority vehicles pass, while the vehicles with higher priority can maintain almost maximal velocity. Figure 5.3 shows the priorities of the individual agents.

[2]Screencast of the scenario: <`http://agents.fel.cvut.cz/agentdrive/vid/adpp/adpp_100_agents_merge.mp4`>

Parameter	**One lane**	**Two lanes**	
		Upper lane	Lower lane
Number of collisions	0	0	
Relative average speed	94.0 %	93.7 %	91.7 %
Relative planning time	1.54 %	1.50 %	
Simulation time [s]	454	409	

Table 5.2: Comparison of the two sub-scenarios on the merging highway.

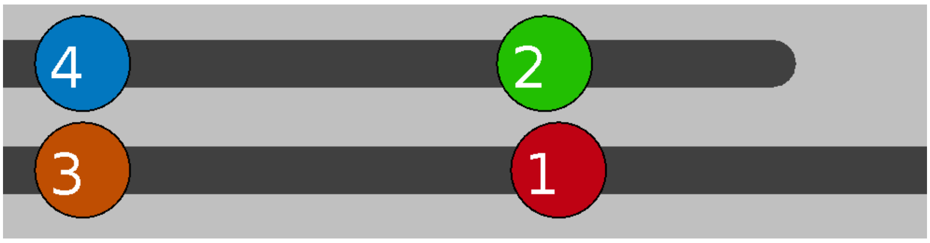


Figure 5.3: Priorities of the agents in the two-lane scenario. Lower id corresponds to a higher priority. The pattern remains the same for all agents during the test.

Finally Figure 5.4 shows travel times for each vehicle during the simulation. It can be seen that the travel times for the upper lane are slightly longer than the times for the lower lane which was explained above.

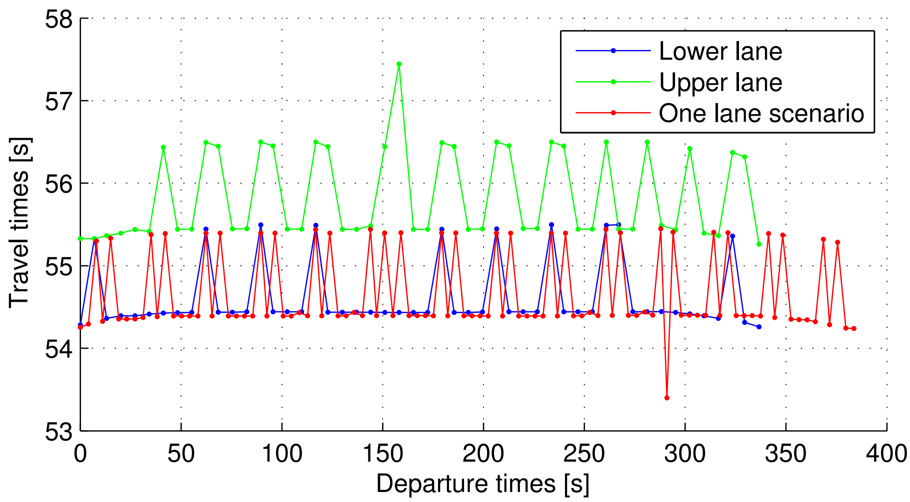


Figure 5.4: Plot illustrating the travel times for each vehicle. The blue and green lines are from the two-lane scenario, whereas the red line is the scenario with single lane. Note that the points are connected with lines only for more readability.

5.4.2 Test 2

The scenario represents a typical situation at a T-shaped junction[3]. The road network is depicted in the Figure 5.5. There are 6 flows defined:

1. 70 vehicles from the left to the right
2. 20 vehicles from the left to the bottom
3. 70 vehicles from the right to the left
4. 20 vehicles from the right to the bottom
5. 20 vehicles from the bottom to the right
6. 20 vehicles from the bottom to the left

This scenario represents a main road with dense traffic, with occasional vehicles either merging to the main road from the bottom road or with some vehicles turning from the main road.

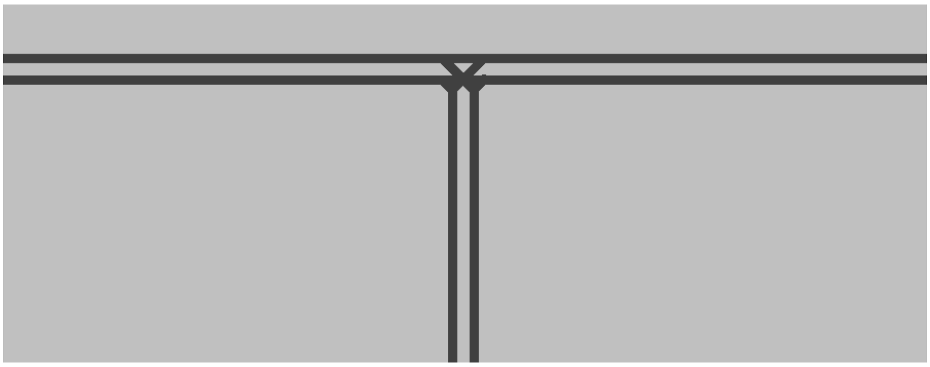


Figure 5.5: Scenario with a T-shaped junction. The horizontal road represents a main road with denser traffic.

The results of the simulation are in Table 5.3. For easier interpretation of the results the flows were merged into 3 groups by the similar nature of the individual flows:

- Straight flows (1 and 3)
- Flows turning from the main road (2 and 4)
- Flows merging from the vertical road to the main (5 and 6)

Parameter	**Flow groups**		
	Straight	Turn	Merge
Number of collisions	0		
Relative average speed	90.9 %	91.8 %	89.8 %
Relative planning time	2.7 %		
Simulation time [s]	426		

Table 5.3: Results of the Test 2

Again the algorithm was able to plan the trajectory for all agents without any collisions. The relative speeds for all flows are around 90 %, which shows a good throughput of the junction.

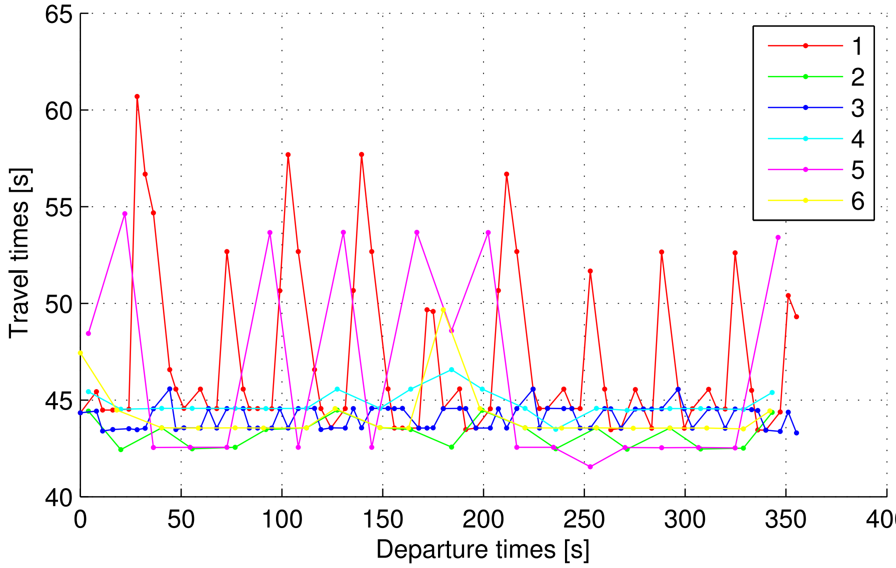


Figure 5.6: Plot of the travel times for each vehicle. The vehicles are grouped in their respective flows. Note that the points are connected with lines only for more readability.

Figure 5.6 shows travel times for all vehicles during the simulation grouped by their respective flows. It can be seen that the travel times do not depend on the traffic rules, i.e. the vehicles on the main road do not have shorter travel time as they are not given way by the vehicles on the vertical road. This is caused by the assignment of the priorities for the individual agents. When the agent is created, he is put to the end of the road and he is given an id which is at the same time his priority.

[3] Screencast of the scenario: <`http://agents.fel.cvut.cz/agentdrive/vid/adpp/adpp_200_agents_T-junction.mp4`>

Lower id means higher priority. The id's are given to the agents in ascending order without respect to the flows. This leads to a situations similar to the one depicted in the Figure 5.7. The agent on the vertical road has lower id (higher priority) because he was created before the agents 5 and 6 on the horizontal road. So they have to let the agent 4 pass, although they are on the main road.

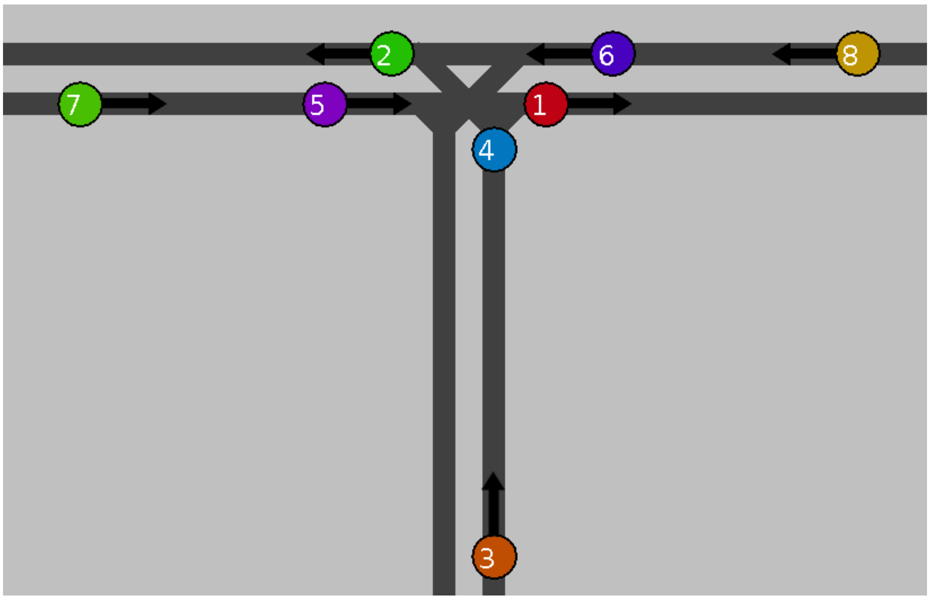


Figure 5.7: Situation after the start of the simulation. Lower number means higher priority. The vehicle number 4 has higher priority than vehicles 6 and 5, so they must slow down and let the vehicle 4 pass.

5.4.3 Priority Assignment Dependency Test

This test shows the impact of the different setting of the priorities for the individual agents. The scenario is similar to the one in the Test 1, but more parallel lanes are added. Figure 5.8 shows the road network of the scenario and the initial positions of the agents. The test is divided into two parts. In the first part the priorities are set in descending order i.e. lower agent's id means higher priority. In the second part the order is reversed.

The results of the test are in Table 5.4. In this test there are no flows, instead the routes of the individual agents are measured.

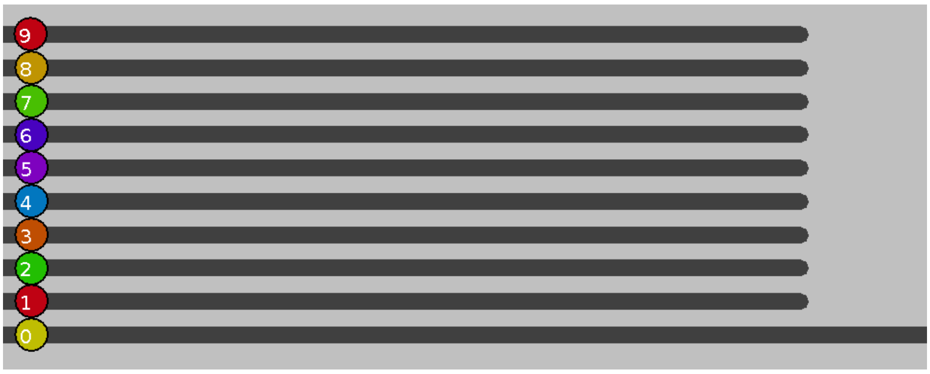
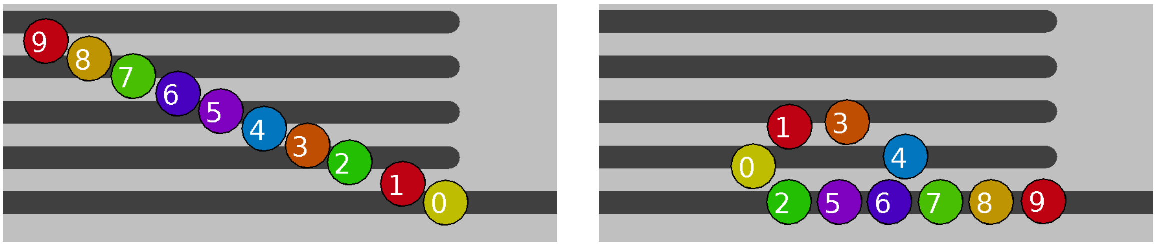


Figure 5.8: Scenario for the test of the priority assignment. The numbers are the id's of the individual agents. In the first part of the test, the agent 0 has the highest priority, in the seconds it's the agent 9.

Parameter	**Priority order**	
	Descending	Ascending
Number of collisions	0	
Relative planning time	0.89 %	0.89 %
Simulation time [s]	39	45
Average relative speed		
Agent with priority 9	92.2 %	90.1 %
Agent with priority 8	90.2 %	86.7 %
Agent with priority 7	87.4 %	83.3 %
Agent with priority 6	85.4 %	83.3 %
Agent with priority 5	83.3 %	76.8 %
Agent with priority 4	81.4 %	71.6 %
Agent with priority 3	79.5 %	69.6 %
Agent with priority 2	77.3 %	67.2 %
Agent with priority 1	75.5 %	65.0 %
Agent with priority 0	74.0 %	61.1 %

Table 5.4: Results of the test comparing the impact of different priority assignments. With the ascending order of the priorities, the average speeds are lower for all agents.

The tests showed large influence of the different priority assignment on the outcome of the simulation. The relative speeds are in all cases higher in the first part of the test, where the order of the agents is more natural, as the agent with highest priority is closer to the merging lane. The agents sort themselves before the merge just by slowing down and then proceeding into the merging lane. In the second part with the priorities reversed, the merging becomes more difficult as the agent with the highest priority has come all the way from the top to the merging lane and the other agents must clear the way for him. Figure 5.9 compares both situations just before the merging.



(a) Agent with lower id has higher priority. This order is more natural and leads to higher average speeds for all agents.

(b) Agent with lower id has lower priority. This order leads to lower average speeds for all agents.

Figure 5.9: Comparison of different priority assignments.

5.4.4 Planning Time Test

The test shows the scalability of the algorithm for increasing number of controlled agents. The scenario for this test is a cross-shaped junction as depicted in the Figure 5.10. There are 100 vehicles travelling from each end of the road to the junction. At the junction 50 vehicles continue forward, 25 turn left and 25 turn right. This can be easily modelled by 12 different flows with total of 400 vehicles.

To measure the impact of the number of agents to the planning time, the scenario is simulated several times using different number of agents each time. This means that at each time the scenario is simulated there is only limited number of vehicles present. The test results are in visualized in a plot in Figure 5.11.

The results show strong dependency of the relative planning time on the number of agents. This is an expected result as with increasing number of agents in the scenario, the number conflicts between them also rises. With the presence of conflict between two agents, the A^* algorithm has to search large portion of the planning space, which can be a time-demanding task.

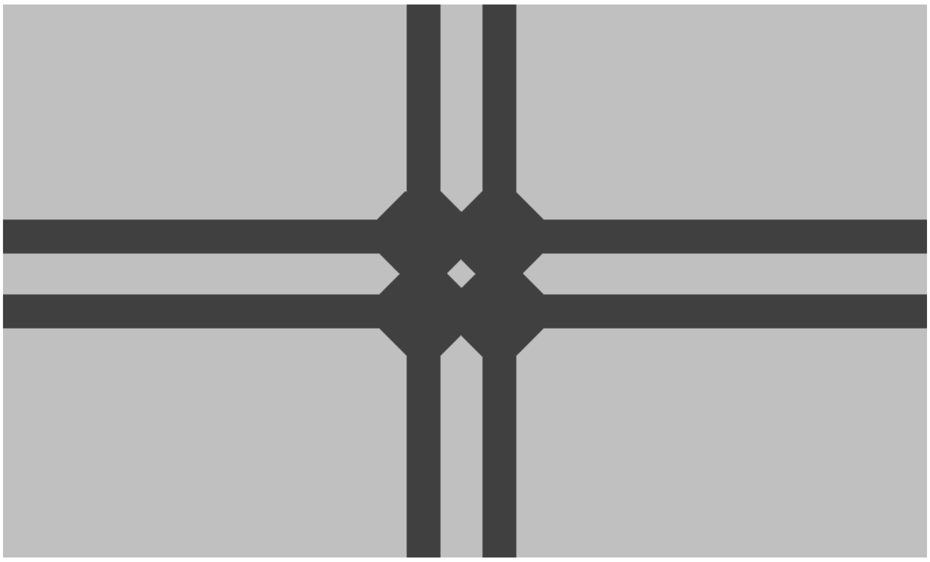


Figure 5.10: Scenario for the planning time comparison test.

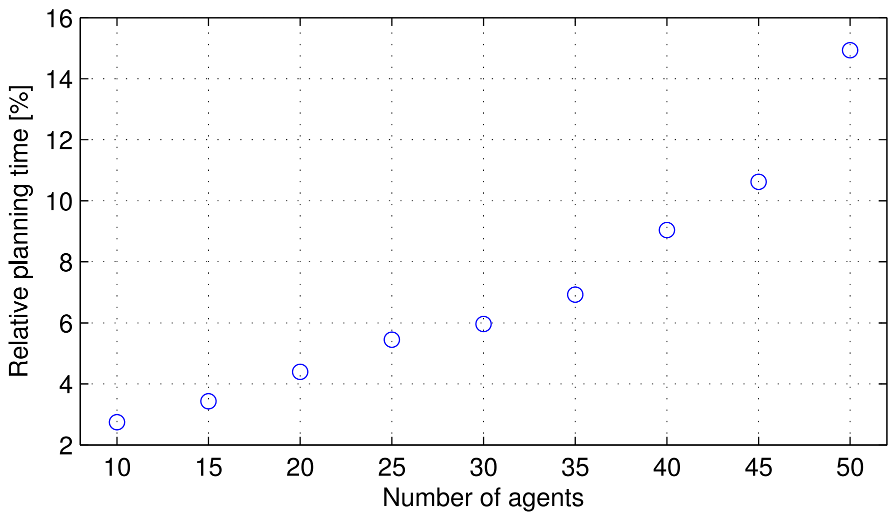


Figure 5.11: Results of the planning time comparison test. The dependency of the overall planning time on the number of controlled agents can be clearly seen.

5.4.5 Planning Horizon Test

This test measures the dependency of the length of the planning horizon on the overall planning time. The scenario for this test is the same as in Test 2 – T-shaped junction. The flows remained also the same. The maximum number of vehicles in simulation is set to 30. The planning horizon is set to several values starting from 5 seconds to 19 seconds. The results of the test are depicted in the Figure 5.12.

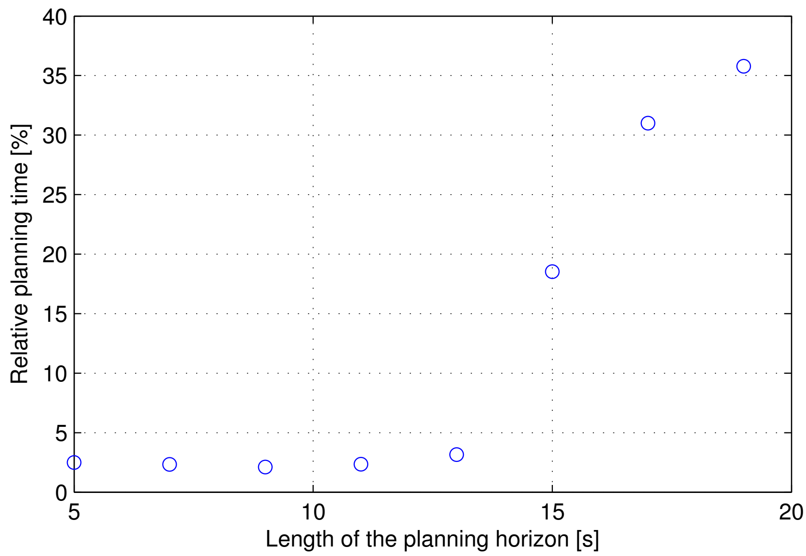


Figure 5.12: Results of the Planning Horizon Test. With longer planning horizon, the algorithm has to search larger portion of the planning space. With the planning horizon longer than 20 seconds, the simulator ran out resources.

The test clearly shows the dependency of the overall planning time on the length of the planning horizon. With the planning horizon set to 15 seconds, the relative planning time rises almost to 20 %. With the values higher than 20 seconds, the simulator ran out of resources, while searching for the collision-free trajectory. Also with values less than 13 seconds, the relative planning time stays around 2 %. This is caused by the fact, that with shorter planning horizon, the collision-free trajectory is planned in a shorter time, but there is also higher number of re-plans. So the overall planning time stays the same.

Chapter 6

Conclusion

The ADPP algorithm was applied as a coordination mechanism for the autonomous vehicles in the road traffic domain. Several modifications of the original ADPP algorithm were proposed to satisfy the requirements and specifics of this domain. The modified algorithm was evaluated in simulation to obtain experimental results.

The tests proved the method to be a reliable coordination mechanism as there were no collisions detected in all testing scenarios. The tests also showed that the optimality of the solution depends on the priority assignment of the agents. The assignment of the priorities was simplified as the optimality of the solution is not the prime focus of this thesis. The method is able to find a solution with any priority assignment. However providing a better system of assigning the priorities can be a topic of further development.

Also the planning time increased significantly with increasing number of controlled agents. The last test also showed the need for the planning horizon. Without the planning horizon the planning space became so large, that the algorithm was not able to find the collision-free, because the system ran out of memory. With higher values of the planning horizon, the planning time was too long, which is undesirable for the road traffic domain, where the trajectory has to be planned within hundreds of milliseconds. This showed, that finding the collision-free trajectory is very computation-time demanding task even with the various proposed modifications. Speeding up the planning phase can be also a topic for further development.

6.1 Future Work

As the tests demonstrated in the Chapter 5, the method depends strongly on the assignment of the priorities for the individual agents. In the current state of the implementation, the priorities are assigned statically based on the time of the departure of the agent. However this does not correspond well to the demands of the road traffic domain. In this domain, the priorities should correspond to the current situation, instead of being statically set. The situation on the road can change dramatically during time and the priorities should reflect this change. Consider two vehicles driving, one behind the other, on a straight highway. In this situation it is obvious, that the vehicle behind should have lower priority than the vehicle in front. However the situation changes, if the vehicle behind overtakes the one in front, the situation is almost the same, but the two vehicles have swapped placed. If the priorities were assigned statically, the vehicle behind would still have higher priority, which is not desirable. In future we would like to extend the existing method with a system of dynamic priority assignment that could solve these issues.

Bibliography

[Erdmann and Lozano-prez, 1987] Erdmann, M. and Lozano-prez, T. (1987). On multiple moving objects. *Algorithmica*, 2:1419–1424.

[Ferber, 1999] Ferber, J. (1999). *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading.

[Lalish et al., 2008] Lalish, E., Morgansen, K., and Tsukamaki, T. (2008). Decentralized reactive collision avoidance for multiple unicycle-type vehicles. In *American Control Conference, 2008*, pages 5055–5061.

[Schaefer, 2014] Schaefer, M. (2014). *Collision Avoidance of Highway Traffic*. PhD thesis, České vysoké učení technické v Praze. Vypočetní a informační centrum.

[Schaefer and Vokřínek, 2015] Schaefer, M. and Vokřínek, J. (2015). Agentdrive: Towards an agent-based coordination of intelligent cars. In *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection*.

[Van Den Berg et al., 2011] Van Den Berg, J., Guy, S. J., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer.

[Čáp et al., 2014] Čáp, M., Novák, P., Kleiner, A., and Selecký, M. (2014). Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Transactions on Automation Science and Engineering (in review)*.

[Velagapudi et al., 2010] Velagapudi, P., Sycara, K., and Scerri, P. (2010). Decentralized prioritized planning in large multirobot teams. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4603–4609.

[Ye et al., 2008] Ye, F., Adams, M., and Roy, S. (2008). V2v wireless communication protocol for rear-end collision avoidance on highways. In *Communications Workshops, 2008. ICC Workshops '08. IEEE International Conference on*, pages 375–379.

Appendix A

Source code

The source code of the AgentDrive project with the implemented method is on the enclosed CD. The source codes related to this thesis are in the following directories:

- `highway/src/main/java/cz/agents/highway/agent`
- `highway/src/main/java/cz/agents/highway/agent/adpp`
- `highway/src/main/java/cz/agents/highway/environment/planning`

Appendix B

Video

The videos presenting the implemented methods in simulation are part of the enclosed CD.

- `videos/simulation1.mp4`
- `videos/simulation2.mp4`