Reconciling Strategic and Tactical Decision Making in Agent-Oriented Simulation of Vehicles in Urban Traffic

Maksims Fiosins, Jelena Fiosina, Jörg P. Müller and Jana Görmer*
Clausthal University of Technology
Institute of Informatics
Julius-Albert Str. 4
D-38678 Clausthal-Zellerfeld, Germany
{maksims.fiosins,joerg.mueller,jana.goermer}@tu-clausthal.de

ABSTRACT

We consider an integrated decision making process of autonomous vehicles in agent-oriented simulation of urban traffic systems. In our approach, the planning process for a vehicle agent is separated into two stages: strategic planning and tactical planning. During the strategic planning stage the vehicle agents constructs the optimal route from source to destination; during the tactical planning stage the operative decisions such as speed regulation and lane change are considered. For strategic planning we modify the stochastic shortest path algorithm with imperfect knowledge about network conditions. For tactical planning we apply distributed multiagent reinforcement learning with other vehicles at the same edge. We present planning algorithms for both stages and demonstrate interconnections between them; an example illustrates how the proposed approach may reduce travel time of vehicle agents in urban traffic.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems;
I.6.8 [**Simulation and Modeling**]: Types of Simulation: Monte Carlo, distributed;
G.3 [**Probability and Statistics**]: Probabilistic algorithms;
I.2.6 [**Artificial Intelligence**]: Learning;
J.m [**Computer Applications**]: Transportation

Keywords

Multi-Agent Systems, Decision Making, City Traffic Control, Stochastic Shortest Path, Multi-Agent Reinforcement Learning

*Maksims Fiosins and Jana Görmer are supported by the Lower Saxony Technical University (NTH) project "Planning and Decision Making for Autonomous Actors in Traffic" (PLANETS)

SIMUTOOLS 2011, March 21-25, Barcelona, Spain

DOI 10.4108/icst.simutools.2011.245533

General Terms

Algorithms, Design, Experimentation

1. INTRODUCTION

Today, software is playing a pervasive role in complex systems with a growing tendency. A big emerging challenge is to reconcile the autonomy of software subsystems and components (agents) with the system-level requirements of efficiency, robustness and dependability. While the decentralization of control, decisions and actions among semi-autonomous computational agents creates opportunities in terms of scalability and local efficiency, it creates large challenges in terms of overall coherence. For example, the integration of car navigation with intelligent assistance functions and car-to-car communication enables a new generation of software-based driving assistants that not only assist the driver but make decisions, take actions and communicate with other vehicles and traffic control devices autonomously, i.e. without explicit commands by the driver or traffic manager.

The application of multi-agent modeling and simulation [17], [18] to traffic simulation and control problems becomes more relevant as intelligent assistant functions and car-to-X communication pave the way to a new generation of intelligent networked traffic infrastructure.

Traffic systems consist of large numbers of autonomous participants (vehicles), which intend to reach their goals (destinations) as quickly as possible and in most cases act rationally according to their own individual interests. Also traffic environments are typically regulated in a centralized manner by traffic management centers using traffic lights, traffic signs and other control elements. Such important characteristics of agents as autonomy, local views (no agent has a full global view of the system) and decentralization (agents make their decisions themselves) make the multi-agent technology very relevant for traffic system modeling.

Previous research in this area has mostly concentrated on traffic lights regulation methods, traffic lights agent architecture, coordination and decision making mechanisms. A very good overview of agent-oriented decision making for traffic lights is given by Bazzan ([5]). The UK-based Thales group used an agent-oriented approach for traffic lights coordination based on genetic algorithms (see, e.g. [12]). Multi-agent reinforcement learning (MARL) for coordination of traffic lights was applied by Bazzan, Lauer and others ([6], [11]).

In contrast, there is less research on individual driver be-

havior and architectures of "intelligent vehicle" agents: existing research is mostly focused on mesascopic models for travel demand planning [4] or adaptive cruise control for autonomous driving [13]. For intelligent vehicle agent models we refer to the works of Adler [1].

The contribution of this paper is to provide an integrated model of decision making of individual vehicles, which integrates strategic decisions with tactical planning. This is very important for simulation of realistic traffic scenarios and for better understanding how different factors influence traffic situations. To our knowledge, this is the first paper, which provides such an approach in an agent-based context.

The integrated decision making process, described in this paper, consists of two stages: strategical decision performing for choosing the optimal route, and tactical decision making for choosing the speed and lane.

For strategic planning, we modify the algorithm for stochastic shortest path calculation R-SSPPR ([3], [14]) by using Bayes posterior probabilities for historical information as well as performing resampling ([2]) to forecast unknown travel times. For tactical planning we use distributed multi-agent reinforcement learning (DEC-MARL, [8],[11]) to learn the optimal cooperative actions of agents.

The paper is organized as follows. In Section 2 we present general definitions and problem formulations for decision making process of a vehicle in urban traffic. In Section 3 we consider underlying planning algorithms: Section 3.1 describes strategic planning, Section 3.2 contains tactical planning. In Section 4 we provide first experimental results. Section 5 concludes the paper and suggests future work directions.

2. STRUCTURE OF THE PLANNING PROCESS FOR THE URBAN TRAFFIC VEHICLES

We consider a structure of decision making of a vehicle agent in an urban traffic environment. A vehicle environment is presented as a directed graph $G = (V, E)$, where nodes and edges represent intersections and streets correspondingly. Each edge $e_i \in E$ consists of $l(e_i)$ lanes and has an associated length $d(e_i)$, $n_e = |E|$, $i = 1 \dots n_e$.

Our model supposes the discrete linear time. Thus without loss of generality we suppose that the model time t takes non-negative integer values, $t \in 0, 1, 2, \dots$.

We suppose that each vehicle is always located on some edge e_i; let $e^j(t) \in E$ be an edge, where vehicle j is located at time t. To identify this edge in the set E we introduce $\nu^j(t)$ as an index referring to the considered edge $e^j(t) = e_{\nu^j(t)}$. A relative position of the vehicle j on the edge $e^j(t)$ at time t is defined as a distance to the end of the edge $x^j(t) \in 0, \dots, d(e^j(t)) - 1$. Let $l^j(t) \in 1, \dots, l(e^j(t))$ be a lane, $v^j(t)$ be a speed of the vehicle j at time t.

We introduce some notations for traffic light: let L_i be a cycle length of the traffic light at the end of the edge e_i (the time period when the traffic light signals start to repeat), and $L_i^G(e_u)$ be the time period inside the cycle L_i, in which the vehicle goes from e_i to e_u. (green light from e_i to e_u). Let $u^j(t) \in \{0, \dots, L_i - 1\}$ be the time passed from the last traffic light cycle repeat. These definitions are illustrated in the Figure 1.

The complete state of the vehicle is given by a tuple, consisting of edge, relative position on the edge, lane, speed and

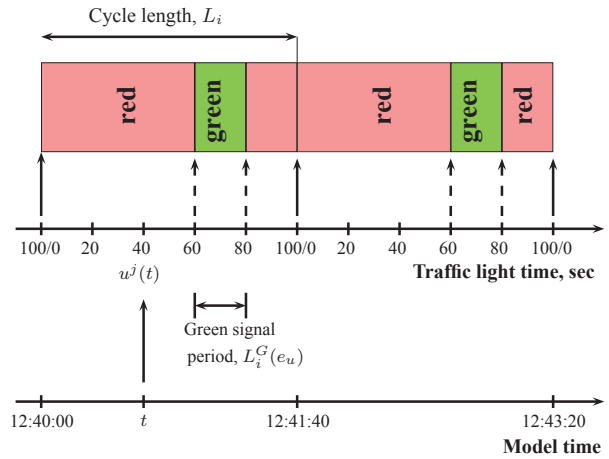

Figure 1: Traffic light related definitions

the time from the last traffic light repeat:

$$s^j(t) =< e^j(t), x^j(t), l^j(t), v^j(t), u^j(t) > . \quad (1)$$

Each vehicle agent j starts its travel at time τ_s^j from the beginning of the edge $e^j(\tau_s^j) \in E$ at the lane $l^j(\tau_s^j) \in 1, \dots, l(e^j(\tau_s^j))$ with initial speed $v^j(\tau_s^j)$. By using (1) we have the following initial state:

$$s^j(\tau_s^j) =< e^j(\tau_s^j), d(e^j(\tau_s^j)) - 1, l^j(\tau_s^j), v^j(\tau_s^j), u^j(\tau_s) >, \quad (2)$$

and ends at the end of the destination edge $e^j(\tau_d^j) \in E$ at time τ_d^j having the final position

$$s^j(\tau_d^j) =< e^j(\tau_d^j), 0, l^j(\tau_d^j), v^j(\tau_d^j), u^j(\tau_d^j) > . \quad (3)$$

The purpose of the vehicle is to reach its destination as quickly as possible, which means reaching the state (3) from the state (2) and minimizing $\tau_d^j - \tau_s^j$ the travel time from the source to the destination.

For this purpose, the planning process is separated into two stages:

- Strategic planning. This is a planning of the sequence of edges (route) from origin to destination;
- Tactical planning. This is a planning of speed and lane inside the current edge.

The structural schema of the planning process is presented in Figure 2.

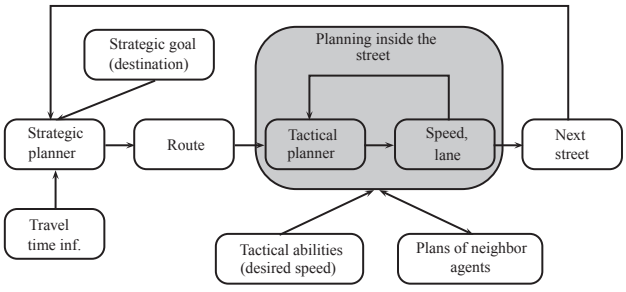

Figure 2: Structure of vehicle planning process

Let T_i^j be dependent random variables and denote the travel times of the vehicle j through the edges e_i. Vehicles plan their routes individually, based on two kinds of travel times information: available samples of previous historic trips and actual realizations of travel times, which denotes the distribution over historical samples $b^j(t)$ for the vehicle j.

Denote by $N(e_i) \subset E$ a set of successor edges of the edge e_i.

The problem of strategic planning is considered as a Stochastic Shortest Path problem with the purpose to obtain the optimal policy

$$\pi^{*j}_{str}(e^j(t), b^j(t)) \in N(e^j(t)), \quad (4)$$

which returns the next edge in the fastest path for the vehicle j after the edge $e^j(t)$. This problem is considered in Section 3.1.

During tactical planning, a vehicle plans its operative decisions together with other agents by forming groups and cooperating with group members to reach the goal. In other words vehicles on one edge plan their actions $a =< \Delta v^j, \Delta l^j > \in A$, where Δv^j is a speed change, $\Delta l^j \in \{-1, 0, 1\}$ is a lane change, A is a set of all possible actions of the vehicle. The goal of the planning process is to minimize travel time of the whole group.

The problem of tactical planning is considered as Distributed Multi-Agent Reinforcement Learning with the purpose to learn the individual state-action value function $Q^j(s^{g_i(t)}, a^j)$, which depends on the states $s^{g_i(t)}$ of the vehicles in a group $g_i(t)$ located at the edge e_i ("joint state") and an action a^j of agent j.

As a result, the optimal policy $\pi^{*j}_{tact}(s^{g_i(t)}) \in A$, which gives an action for j-th agent being at the joint state $s^{g_i(t)}$, which ensures the optimal travel time for the whole group. This problem is considered in Section 3.2.

The integrated policy of the vehicle j consists of strategic and tactical policy

$$\pi^{*j} =< \pi^{*j}_{str}, \pi^{*j}_{tact} > . \quad (5)$$

3. PLANNING FOR THE VEHICLE AGENT

3.1 Strategic Planning

In this section we present the method for strategic planning of a vehicle agent. We modify the algorithm R-SSPPR ([3],[14]) for calculation of the Stochastic Shortest Path with imperfect information.

The idea of the R-SSPPR algorithm is to pre-calculate the shortest paths for all possible realizations of edge travel times, which are taken from historical realizations. We modify this algorithm by allowing arbitrary values of travel times for the case, when all possible realizations of edge travel times are unknown.

The idea of our algorithm is that the actual values of travel times may differ from historical realizations, and this difference has a normal distribution. The actual realization of edge travel time corrects the probabilities of the historical samples, obtaining posterior probabilities given actual travel times. For each edge, we pre-calculate an optimal route for every possible probability distribution of historical samples (it is clear that for practical realization we perform a discretization of probabilities by taking them with some small step, say ϵ).

The strategic planning process is illustrated in Figure 3.

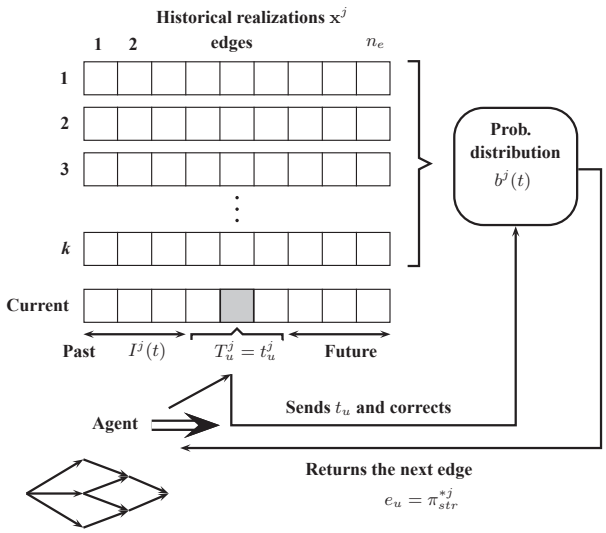

Figure 3: Structure of vehicle planning process

We recall that the strategic planning problem for vehicle j is to construct a path from the beginning of the edge $e_s^j \in E$ to the end of the edge $e_d^j \in E$ with minimal travel time. Agents solve their strategic planning tasks individually.

On each edge, the agent j makes its decision about the next edge in its route. We call this decision rule a strategic policy π^j_{str}. The arguments of the strategic policy are current edge $e^j(t)$ as well as current probability distribution $b^j(t)$ over historical samples. So $\pi^j_{str}(e^j(t), b^j(t)) \in N(e^j(t))$ defines the next edge after $e^j(t)$ in the j-th agent's path. Next we explain how the probability distribution $b^j(t)$ is formed and corrected.

Let $T^j = \{T_1^j, T_2^j, \ldots, T_{n_e}^j\}$ be a set of travel times, which are random variables, for the agent j trough all edges of the graph. Note that it is more convenient to define for each agent its own travel time, because they travel at the same edge in different conditions (time, speed, type of vehicle, etc.). We also assume, that for the given agent the information in historical samples is also selected for the same conditions of driving (time, day, season, etc.).

We suppose that elements of T^j are dependent, because traffic conditions on one node have an influence on the conditions of another node. We assume that travel times T^j have the following, independent on the model time joint distribution function

$$F_T(x_1, x_2, \ldots, x_{n_e}) = P\{T_i^j \le x_1, T_2^j \le x_2, \ldots, T_{n_e}^j \le x_{n_e}\}.$$

However, the general form of the distribution $F_T(\cdot)$ is unknown, but only samples of historical realizations are available. We suppose that a sample of travel time realizations $\mathbf{x}^j = \{\mathbf{x}_1^j, \mathbf{x}_2^j, \ldots, \mathbf{x}_k^j\}$ is available to agent j, where $\mathbf{x}_q^j = \{x_{q,u}^j\}$, is a set, containing values of travel times for all edges of the q-th historical realization, $q = 1, 2, \ldots, k$, $u = 1, 2, \ldots, n_e$.

A (prior) distribution $b^j(\tau_s^j) = \{p_i^j(\tau_s^j)\}$ of historical samples is available to the agent j at the beginning of its trip, where $p_q^j(\tau_s^j)$ is a probability that the actual travel times t_u^j will be approximately equal to the q-th historical realiza-

tion. Let us define the event A_q^j: "the sample $\mathbf{x}_q$ contains approximate travel times for the agent j". Then

$$p_q^j(t) = P\{A_q^j\}.$$

An approximate equality of historical sample q and actual travel time t_u^j for edge e_u means that the difference $\delta_u^j = |t_u^v - x_{q,u}|$ is normally distributed with zero expectation and variance $\sigma_{q,u}^{2\,j}$: $\delta_u^j \sim N(0, \sigma_{q,u}^{2\,j})$, $q = 1, \ldots, k$, $u = 1, \ldots, n_e$. We suppose that a sample of variances $\Omega^j = \{\Omega_1^j, \Omega_2^j, \ldots, \Omega_k^j\}$ is available to j-th agent, where $\Omega_i^j = \{\sigma_{q,u}^{2\,j}\}$ is a set, containing variances of differences between actual travel times and historical realizations $x_{q,u}$.

Later we update the distribution $b^j(t)$ based on knowledge of the actual travel time t_u^j on the edge e_u, where the probability of difference with historical realizations are calculated according to the normal distribution.

During its trip, the vehicle receives new information about the realization of travel time. The vehicle j at time t can split the set of edges E into two disjoint subsets: the subset $E_{kn}^j(t) \subset E$ of the edges, which travel times are known and the subset $E_{rnd}^j(t) \subset E$ of the edges, which travel times are unknown $E_{kn}^j(t) \cup E_{rnd}^j(t) = E, E_{kn}^j(t) \cap E_{rnd}^j(t) = \emptyset$.

Let $I^j(t)$ be the information, available to the vehicle j, which consists of known travel times till time t; it is a set of events, that random variables T_u^j take their fixed values t_u^j correspondingly

$$I^j(t) = \bigcup_{e_u \in E_{kn}^j(t)} \{T_u^j = t_u^j\}. \tag{6}$$

Let us suppose that the travel time at the edge $e_u \in E$ becomes known to the vehicle at time τ. The vehicle now updates the sets $E_{kn}^j(\tau)$, $E_{rnd}^j(\tau)$ as well as $I_j(\tau)$:

$$\begin{array}{l} E_{kn}^j(\tau) = E_{kn}^j(\tau') \cup \{e_u\}, \\ E_{rnd}^j(\tau) = E_{rnd}^j(\tau') \setminus \{e_u\}, \\ I^j(\tau) = I^j(\tau') \cup \{T_u^j = t_u^j\}, \end{array} \tag{7}$$

where τ' is the time instant of previous update.

The information $I^j(t)$ defines the posterior probabilities $b^j(t)$. Given old posterior probability distribution $b^j(\tau')$ and actual travel time t_u^j it is possible to calculate the posterior distribution $b^j(\tau)$ using Bayes' formula:

$$p_q^j(\tau) = P\{A_q^j | I^j(\tau'), t_u^j\} = \frac{p_q^j(\tau') P\{t_u^j | A_q^j, I^j(\tau')\}}{P\{t_u^j | I^j(\tau')\}} \tag{8}$$

for all $q = 1, 2, \ldots, k$.

In this formula, $p_q^j(\tau') = P\{A_q^j | I^j(\tau')\}$ is the probability that actual travel time will be approximately equal to the historical sample $\mathbf{x}_q^j$ without knowledge of t_u^j. The probability $P\{t_u^j | A_q^j, I^j(\tau')\}$ is a probability to have an actual travel time t_u^j in the case if travel times are approximately equal to q-th historical sample. This probability can be calculated using normal distribution function

$$P\{t_u^j | A_q^j, I^j(\tau')\} = \Phi\left(\frac{|t_u^j - x_{q,u}^j|}{\sigma_{q,u}^{2\,j}}\right), \tag{9}$$

where $\Phi(x)$ is a distribution function of the standard normal distribution with zero mean and 1 variance: $N(0, 1)$.

Denominator of (8) may be simply calculated, ensuring that the sum of all posterior probabilities is equal to 1.

Strategic planning process consists of two stages: pre-planning and routing. During strategic pre-planning stage, we pre-calculate the optimal routes for each edge and possible distribution of historical samples.

For pre-planning stage, it is convenient to represent the problem in a form which enables dynamic programming in stochastic case.

It is known ([7]) that the recurrent equation for the dynamic programming in stochastic case looks like

$$V(s) = \min_{a \in \Gamma(s)} \{F(s, a) + E_G[V(s')|s, a]]\}, \tag{10}$$

where s is the state at the current step, s' is the state at the next step, $\Gamma(s)$ is a set of possible actions from the state s, G is a distribution of all possible next states given current state s and action a, $F(s, a)$ is a value, which depends on current state and action and which sum should be minimized, $V(s)$ is an expected value, if one starts from the state s (value function).

In our case the state consists of the current edge e_i and the distribution $b^j(t)$ over historical samples; the set of possible actions $\Gamma(s)$ consists of the next edges $N(e_i)$; the value $F(e_i)$ is a travel time trough the edge e_i; and the distribution $G(e_i, b^j(t), b')$ is defined as a probability to get the distribution b' on the next step, if current edge is e_i and current distribution is $b^j(t)$. Thus we denote by $V_{\pi_{str}}^j(e_i, b^j(t))$ an expected travel time of the vehicle j from the beginning of the edge e_i to the destination edge e_d^j under the decision rule π_{str}.

The following recurrent equation is true for $V_\pi^j(e_i, b^j(t))$:

$$V_{\pi_{str}}^j(e_i, b^j(t)) = \begin{cases} \sum\limits_v p_v^j(t) x_{v,i}^j, & \text{if } e_i = e_j^d, \\ \sum\limits_v p_v^j(t) x_{v,i}^j + \\ + \sum\limits_{b'} G(e_i, b^j(t), b') \cdot \\ \cdot V_{\pi_{str}}^j(\pi_{str}(e_i, b^j(t)), b') \\ \text{otherwise}, \end{cases} \tag{11}$$

where the sum is taken over all possible distributions of historical samples on the next edge.

In order to find the optimal strategic policy $\pi_{str}^{*j}(e_i, b^j(t))$, we need to minimize (11) over all possible next edges:

$$V^{*j}(e_i, b^j(t)) = \begin{cases} \sum\limits_v p_v^j(t) x_{v,i}^j, & \text{if } e_i = e_j^d, \\ \sum\limits_v p_v^j(t) x_{v,i}^j + \\ + \min\limits_{e_u \in N(e_i)} \sum\limits_{b'} G(e_i, b^j(t), b') \cdot \\ \cdot V^{*j}(e_u, b') & \text{otherwise.} \end{cases} \tag{12}$$

The optimal policy corresponds to the edge, chosen for $V^{*j}(e_i, b^j(t))$

$$\pi_{str}^{*j}(e_i, b^j(t)) = \begin{cases} \text{none} & \text{if } e_i = e_j^d, \\ \arg\min\limits_{e_u \in N(e_i)} \sum\limits_{b'} G(e_i, b^j(t), b') \cdot \\ \cdot V^{*j}(e_u, b'^j(t')) & \text{otherwise.} \end{cases} \tag{13}$$

The implementation of (12) supposes calculation of the optimal route for each pair of edge and each possible distribution vector. In practice, we perform a discretization of distributions and generate a set of distribution vectors $\mathbf{B}$ with step ϵ on every component, where ϵ is taken sufficiently small.

During the routing phase, the closest vector $\tilde{b}^j(t) \in \mathbf{B}$ to the actual probability distribution $b^j(t)$ is selected

$$\tilde{b}^j(t) = \arg\min_{b \in \mathbf{B}} [D(b^j(t)||b)], \tag{14}$$

where the divergence between distributions D(P||Q) may be calculated using Kullback-Leibler divergence [9]

$$D(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}. \tag{15}$$

However, there is some difficulty in calculation of the possible distributions $G(e_i, b^j(t), b^j(t'))$ for the next edges. For this purpose, we need to consider all possible travel times of the edge e_i, which are not necessary equal to one of values in historical realizations.

In order to avoid this difficulty, we use the resampling of current edge travel times ([2]). We repeat r times the following resampling procedure. For the fixed distribution b we extract one value of travel times $x_{v,i}$ and add a random value $n_i^{*\eta}$, distributed with an expectation 0 and variance $\sigma^{2\,j}_{v,i}$, obtaining an η-th resampling realization of the travel time $t_i^{*\eta}$:

$$t_i^{*\eta} = x_{v,i} + n_i^{*\eta}. \tag{16}$$

The value $t_i^{*\eta}$ (16) allows us to calculate the posterior distribution $b^{j*\eta}(t')$ using (8). We find the closest distribution $\tilde{b}^{j*\eta}(t')$ using (14) and (15) from available distributions.

Then we take each edge $e_u \in N(e_i)$ and calculate $V(e_u, \tilde{b}^{j*\eta}(t'))$. Their average over all r resampling realizations gives an expected value function for the next edge e_u

$$\tilde{V}^*(e_u) = 1/r \sum_{\eta=1}^{r} \left[t_i^{*\eta} + V(e_u, \tilde{b}^{j*\eta}(t')) \right]. \tag{17}$$

Note that a number of resamples r can be taken sufficiently big, it is bounded by the computational resources only. As we will see later, complexity of the algorithm linearly depends on r.

Then (17) is minimized over all next edges $e_u \in N(e_i)$, obtaining the optimal value function $V^{*j}(e_i, b^j(t))$ (12).

Now we describe the mentioned procedure in the form of algorithms. First, Algorithm 1 presents the resampling procedure.

Algorithm 1 Function RESAMPLE

function RESAMPLE($e_i, e_u, b, r, X, \Omega$)
  for all $\nu \in \{1, \dots, r\}$ **do**
    $v \leftarrow random(b)$ ▷ selects one historical sample
    $n_i^{*\eta} \leftarrow Normal(0, \sigma^2_{v,i})$ ▷ random difference
    $t_i^{*\eta} \leftarrow x_{v,i} + n_i^{*\eta}$ ▷ calculates the travel time
    ▷ calculates the posterior distribution
    $b'^{*\eta} \leftarrow UpdateDistr(b, t_i^{*\eta})$
    ▷ takes closest available distribution
    $\tilde{b}^{*\eta} \leftarrow FindClosest(b', B)$
  end for
  ▷ returns an average over all generated distributions
  return $1/r \sum_{\eta=1}^{r} [t_i^{*\eta} + V^{*j}(e_u, \tilde{b}^{*\eta})]$
end function

Now we present a pre-planning stage of strategic planning in Algorithm 2.

Algorithm 2 Pre-planning stage of the strategic planning

$e_i \leftarrow e_d^j$
$B \leftarrow GenerateDistr(\epsilon)$ ▷ generates all distribution vectors
while $prev(e_i) \neq \emptyset$ **do**
  for all $b \in B$ **do** ▷ for each distribution vector
    for all $e_u \in N(e_i)$ **do** ▷ for each neighbor edge
      ▷ calculates an avegare reward
      $\tilde{V}^*(e_u) \leftarrow RESAMPLE(e_i, e_u, b, r, X, \Omega)$
    end for
    ▷ minimizes it by neighbor edges
    $V^{*j}(e_i, b) \leftarrow \min_{e_u \in N(e_i)} \tilde{V}^*(e_u)$
    $\pi^{*j}(e_i, b) \leftarrow \arg\min_{e_u \in N(e_i)} \tilde{V}^*(e_u)$
  end for
  $e_i \leftarrow \text{prev}(e_i)$
end while

This algorithm uses the function $prev(e_i) \in E, e_i \in E$. This function corresponds to ordering of edges $\{e_{(1)}, e_{(2)}, \dots, e_{(n_e)}\}$ that for any pair $(e_{(i)}, e_{(u)}), i < j$ the edge $e_{(u)}$ does not start at the edge, where $e_{(i)}$ starts. For this purpose, the graph G should not contain loops. So the vehicle uses only a part of roads in its route calculation process such that the resulting graph does not contain loops. The loops add more complexity to the algorithm, however in our future research we plan to construct an algorithm taking loops into account.

Now the function $prev$ is simply defined as $prev(e_{(i)}) = e_{(i-1)}$ for $i > 1$.

Now let us analyze the complexity of the Algorithm 2. We can easily see three loops here: outer loop over all edges, second loop over the elements of B (possible distributions) and inner loop over all possible neighbor edges, which also is bounded by number of edges. It is easy to see that the number of vectors in B is bounded by $1/\epsilon^{n^e}$. Inside these three loops there is a call of the resampling procedure, which performs r steps, $UpdateDistr$ and $FindClosest$ procedures require k steps (the number of historical realizations). So the complexity of the proposed procedure is

$$T(n_e, k, r, \epsilon) = O\left(\frac{n_e^2 r}{\epsilon^{n_e} k}\right). \tag{18}$$

We conclude that this algorithm is polynomial by r and ϵ, but exponential by n_e and k. This does not allow to apply it for very large networks; in our future research we will work on reducing of the complexity of this algorithm.

3.2 Tactical Planning

According to strategic plan, a vehicle enters some edge together with other vehicles. Its tactical plan allows sharing an edge with other vehicles by choosing appropriate speed and lane changes to pass through the edge as quickly as possible.

We suppose that vehicles which are located on the same edge can freely exchange their information such as states and are fully collaborative, thus namely they follow the joint policy, which is optimal for the whole group.

As it was introduced previously, a state of the vehicle $s^j(t)$ is described by its edge, relative position at the edge, lane, speed and traffic light time. We denote by S a set of all

possible states of an individual vehicle. Possible vehicle actions consist of pairs $a =< \Delta v, \Delta l >\in A$, which correspond to speed and lane change. So state change is described as

$$s^j(t+1) = \begin{cases} < e^j(t), x^j(t) - v^j(t), l^j(t) + \Delta l, \\ v^j(t) + \Delta v, u^j(t+1) >, \\ \text{if } x^j(t) - v^j(t) > 0, \\ < \pi^{*j}_{str}(e^j(t), \cdot), x^j(t) - v^j(t) + d(e^j(t)), \\ l^j(t) + \Delta l, v^j(t) + \Delta v, u^j(t+1) > \\ \text{otherwise.} \end{cases} \quad (19)$$

where the condition $x^j(t) - v^j(t) > 0$ checks if the edge does not end at current step (here $v^j(t)$ is a distance, which a vehicle passes on one time step).

Note that the second situation is only possible if the traffic light signal for the desired direction is green, so $u^j(t) \in L^G_i(\pi^{*j}_{str}(e^j(t), \cdot))$.

We assume that for each state $s^j(t) \in S$ a corresponding reward $r(s^j)$ is available. We further assume that the reward structure is fully additive:

$$r(s^j(t)) = r^x(x^j(t)) + r^l(l^j(t)) + \\ + r^v(v^j(t)) + r^u(u^j(t)). \quad (20)$$

The position part $r^x(\cdot)$ has smaller values at the beginning of the edge and bigger values at the end; the lane part $r^l(\cdot)$ has bigger values for the lane, which has a turn to the next edge in the vehicle route; the speed part $r^v(\cdot)$ has larger values for larger speeds; the traffic light part $r^u(\cdot)$ has a big negative value for $u^j(t) \notin L^G_i(e_u)$ and small $x^j(t)$; for all other states it is equal to zero; as a result, vehicles will regulate their speed to go to fit the green signal phase of the traffic light.

In order to construct agent optimal tactical policy, we use the reinforcement learning (RL) ([15]). This is a computational approach to learn from interaction with the environment, if supervision is not available. There are several types of RL such as dynamic-programming based RL (for models with perfect information), Monte Carlo Learning (MC) (if the reward is available at the end of the episode only) and Temporal Difference Learning (TD) (if reward is available after each simulation step).

RL methods may be divided to on-policy methods (the most popular is SARSA) and off-policy methods (the most popular is Q-learning). The difference is that on-policy methods work with one policy and the policy update is performed after some time; off-policy methods perform the optimization on each learning step, so do not work with a fixed policy. In our case we use Q-learning approach, because the state space is large and the separate loop over all states for policy optimization may be computationally expensive. Instead, we perform at each learning step maximization on all actions.

We introduce a state-action value function $Q^j(s^j, a^j)$, which represents an average reward for the j-th agent, which starts from the state $s^j \in S$ and performs the action $a^j \in A$. Then for the state-action pair (s^j, a^j) and the next state s'^j, there exists the following recurrent equation:

$$Q^j(s^j, a^j) = Q^j(s^j, a^j) + \\ + \alpha \left[r(s'^j) + \gamma \max_{a'^j} Q^j(s'^j, a'^j) - Q^j(s^j, a^j) \right], \quad (21)$$

where α is learning parameter, which represents an influence of difference of two neighbor action-value functions and γ is learning parameter, which represents an influence of the next state-action value function.

Tactical policy $\pi^j_{tact}(s^j) \in A$ defines an agent action in state s^j. The optimal tactical policy $\pi^{*j}_{tact}(s^j)$ guarantees the maximal expected reward; it may be calculated by maximizing $Q^j(s^j, a^j)$ with respect to all possible actions $a^j \in A$:

$$\pi^{*j}_{tact}(s^j) = \text{argmax}_{a^j \in A} Q^j(s^j, a^j). \quad (22)$$

The single-agent learning procedure is presented in Algorithm 3.

Algorithm 3 Single-agent tactical learning algorithm

1: initialize $s \leftarrow < x, l, v, u >$
2: **while** $x > 0$ **do**
3: $a \leftarrow SEL\_A(s, \epsilon)$
4: take action a
5: observe next state s'
6: observe reward $r(s')$
7: $Q^j(s,a) \leftarrow Q^j(s,a) + \alpha[r(s') + \\ + \gamma \max_{a'} Q^j(s', a') - Q^j(s,a)]$
8: $\pi^{*j}_{tact}(s) = \text{argmax}_a Q^j(s,a)$
9: $s \leftarrow s'$
10: **end while**

The function SEL_A in line 3 of this algorithm selects an action depending on the state s. This can be done using ϵ-greedy method

$$SEL\_A(s, \epsilon) = \begin{cases} \pi^{*j}_{tact}(s) & \text{with prob. } 1 - \epsilon + \epsilon/|A| \\ \text{other } a \in A & \text{with prob. } \epsilon/|A|, \end{cases}$$

where ϵ is sufficiently small. This ensures that the agent will try all actions, not only the first which seems to be good.

Now let us consider a case of several agents, where we apply the Distributed Multi-Agent Reinforcement Learning algorithm ([8], [11]) for solving the cooperative task of multiple agents.

Let $g_i(t)$ be a set of agents (group), which are located at edge $e_i \in E$ at time t. Let $s^{g_i(t)} \in S^{|g_i(t)|}$ be a joint state, which includes states of all agents in the group g_i, where S^a is a times cross product of the set S with itself.

A local state-action value function $\tilde{Q}^j(s^{g_i(t)}, a^j)$ depends on the action of the j-th agent and the joint group state $s^{g_i(t)}, j \in g_i(t)$. By analog with (21) it represents an average reward, which the agent j receives, performing the action a^j in the group state $s^{g_i(t)}$.

The function $\tilde{Q}^j(s^{gr_i(t)}, a^j)$ is updated in the following manner, which ensures maximum of joint-action Q-functions [11]:

$$\tilde{Q}'^j(s^{g_i(t)}, a^j) = \max \Big\{ \tilde{Q}^j(s^{gr_i(t)}, a^j), \\ r^j(s'^{g_i(t)}) + \gamma \max_{a'^j} \tilde{Q}^j(s'^{gr_i(t)}, a'^j) \Big\}, \quad (23)$$

where maximization is done with respect to all possible local actions a'^j. The optimal policy $\tilde{\pi}^{*j}_{tact}(s^{g_i(t)})$ is being updated only if function $\tilde{Q}^j(s^{g_i(t)}, a^j)$ was changed:

$$\tilde{\pi}^{*j}_{tact}(s^{g_i(t)}) = \begin{cases} \arg\max_{a^j} \tilde{Q}'^j(s^{g_i(t)}, a^j), \\ \text{if } \max_{a^j} \tilde{Q}'^j(s^{g_i(t)}, a^j) > \\ \max_{a^j} \tilde{Q}^j(s^{g_i(t)}, a^j), \\ \tilde{\pi}^{*j}_{tact}(s^{g_i(t)}) \quad \text{otherwise.} \end{cases} \quad (24)$$

A multi-agent learning procedure is given in the Algorithm 4.

Algorithm 4 Multi-agent tactical learning algorithm at the edge $e_i \in E$

while not end of the simulation **do**
  for all $j \in g_i(t)$ **do**
  ▷ for all agents on the edge execute actions and observe individual next states
    $a^j \leftarrow SEL\_A(s^{g_i(t)}, \epsilon)$
    take action a^j
    observe next state s'^j
  end for
  ▷ collect group joint state
  $s'^{g_i(t)} \leftarrow \{s'^j\},\ j \in g_i(t)$
  for all $j \in gr_i(t)$ **do**
    observe reward $r^j(s'^{g_i(t)})$
    ▷ learn individual state-action functions
    $\tilde{Q}'^j(s^{g_i(t)}, a^j) = \max\{\tilde{Q}^j(s^{g_i(t)}, a^j), r^j(s'^{g_i(t)}) +$
    $+\gamma \max_{a'^j} \tilde{Q}^j(s'^{g_i(t)}, a'^j)\}$
    ▷ and update policy if necessary
    if $\max_{a^j} \tilde{Q}'^j(s^{g_i(t)}, a^j) > \max_{a^j} \tilde{Q}^j(s^{g_i(t)}, a^j)$ **then**
      $\tilde{\pi}^{*j}_{tact}(s^{g_i(t)}) \leftarrow \arg\max_{a^j} \tilde{Q}'^j(s^{g_i(t)}, a^j)$
    end if
    $Q(s, a_i) \leftarrow Q'(s, a_i)$
  end for
end while

4. EXPERIMENTS AND RESULTS

We simulate a traffic network in Hanover, Germany, by using AimSun, a specialized simulation software for traffic applications. The road network is shown in the Fig. 4.

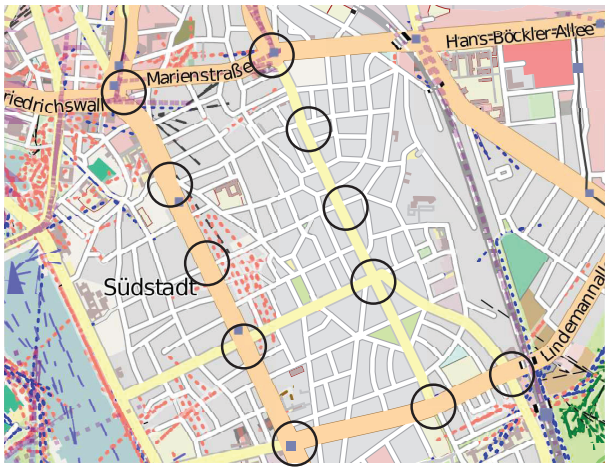

Figure 4: Road network of the south part of Hanover city

All intersections are regulated by traffic lights with fixed control plans, known to vehicles. We use the realistic traffic flows, collected in given region of Hanover in morning rush hours. There are traffic flows in all directions; we are interested in the flow $1 \rightarrow 11$; these vehicles use the graph, shown on the Fig. 5, for their decisions.

In our model, we divide each street to cells of 4 m length. The possible speeds of vehicles are: {0,5,...,50}km/h. One simulation step corresponds to 1/2 sec.

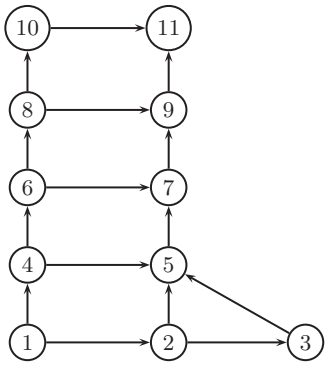

Figure 5: Graph representation of the considered in Figure 4 road network

Table 1: Travel times for the route $1 \rightarrow 11$ (sec.) depending on the flows ratio

	Flows ratio				
Planning usage	**0.5**	**0.7**	**0.9**	**1.0**	**1.2**
Without planning	241.1	256.2	401.4	567.4	934.4
With strategic planning	231.2	242.4	378.1	528.2	856.9
With tactical planning	231.9	246.2	385.0	543.7	870.8
With strategic and tactical planning	225.7	236.1	367.7	518.6	818.5

Experimental results are summarized in Table 1. We calculate travel times depending on the ratio to the flows in Hanover in morning hours.

We conclude that the application of integrated planning allows reducing the travel time of vehicles to about 10%; this is more than SP or TP separately.

5. CONCLUSIONS

Planning for the vehicle agents in city traffic is very important task, because it allows vehicles to use an existing information for decentralized planning and implies more effective infrastructure usage.

In this paper, we proposed an integrated planning process for vehicle agents, which includes both strategic and tactical planning.

For strategic planning, we showed how to apply existing information for effective solving of routing problems under imperfect information. The proposed algorithm allows working with posterior probabilities of historical samples and uses resampling of future travel times.

For tactical planning we used a modification of Distributed Multi-Agent Reinforcement Learning (DEC-MARL), which allows vehicles to collaborate inside a road segment in order to traverse it in the most quick way. The model reflects the growing experience of vehicles in tactical decisions as well as in cooperation processes. We used the resampling procedure in order to reduce working time of the algorithm.

First experiments show that the demonstrated approach allows reducing the travel time of vehicles. The integrated nature of planning process allows improving strategic and

tactical decisions of vehicles.

In the future we will work on an integration of the approach with centralized regulations from traffic management centers, as well as more dynamic agents group formation for flexible cooperation in strategic and tactical planning. Another important direction of our research aims at the application of computational statistics (e.g. resampling) for reduction of algorithms complexity. We are going to avoid as possible the exponential complexity of the algorithm and take possible loops in the routing graph into account.

Acknowledgments

We thank our partners Jan Fabian Ehmke, Daniel Schmidt (Braunschweig Technical University), Huges Tchouankem, Henrik Schumacher (Hanover Technical University) for their ideas which contributed to the paper.

6. REFERENCES

[1] J. L. Adler, G. Satapathy, V. Manikonda, B. Bowles, and V. J. Blue. A multi-agent approach to cooperative traffic management and route guidance. *Transportation Research Part B: Methodological*, 39(4):297 – 318, 2005.

[2] A. Andronov, H. Fioshina, and M. Fioshin. Statistical estimation for a failure model with damage accumulation in a case of small samples. *Journal of Stat. Planning and Inference*, 139:1685–1692, 2009.

[3] M. Baglietto, G. Battistelli, F. Vitali, and R. Zoppoli. Shortest path problems on stochastic graphs: a neuro dynamic programming approach. In *Proceedings of 42nd IEEE Conference on Decision and Control*, volume 6, pages 6187–6193, 2003.

[4] M. Balmer, N. Cetin, K. Nagel, and B. Raney. Towards truly agent-based traffic and mobility simulations. In *Proceedings of 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 60–67, 2004.

[5] A. L. Bazzan. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multi-Agent Systems*, 18:342–375, June 2009.

[6] A. L. Bazzan, D. de Oliveira, and B. C. da Silva. Learning in groups of traffic signals. *Engineering Appl. of Artificial Intelligence*, 23(4):560 – 568, 2010.

[7] R. E. Bellman. *Dynamic programming*. Princeton University Press, 1957.

[8] L. Busoniu, R. Babuska, and R. de Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 38:156 – 172, 2008.

[9] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[10] L. Kuyer, S. Whiteson, B. Bakker, and N. Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 656–671. Springer-Verlag, 2008.

[11] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of Seventeenth International Conference on Machine Learning (ICML-00)*, pages 535 – 542. Stanford, 2000.

[12] T. Masterton and D. Topiwala. Multi-agent traffic light optimisation and coordination. In *Thales Research and Technology*, volume 2, 2008.

[13] J. E. Naranjo, M. A. Sotelo, C. Gonzalez, R. Garcia, and T. de Pedro. Using fuzzy logic in automated vehicle control. *IEEE Intelligent Sys.*, 22:36–45, 2007.

[14] G. H. Polychronopoulos and J. N. Tsitsiklis. Stochastic shortest path problems with recourse. *Networks*, 27:133–143, 1996.

[15] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, 1998.

[16] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, 1998.

[17] G. Weiß, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1999.

[18] M. J. Wooldridge. *Reasoning about Rational Agents*. The MIT Press, Cambridge, Massachusetts, 2000.