Self-adaptive Traffic and Logistics Flow Control using Learning Agents and Ubiquitous Sensors

Stefan Bosse

University of Bremen, Dept. Computer Science, Bremen, Germany

Abstract. Traffic flow optimisation is a distributed complex problem. Traditional traffic and logistics flow control algorithms operate on a system level and address mostly switching cycle adaptation of traffic signals and lights. This work addresses traffic flow optimisation by self-adaptive micro-level control by combining Reinforcement Learning and rule-based agent models for action selection with a new hybrid agent architecture. I.e., long-range routing is performed by agents that adapt their decision making for re-routing on local environmental sensors. Agent-based modelling and simulation are used to study emergence effects on urban city traffic flows with learning agents. The approach and the proposed agent architecture can be generalised and applied to a broader range of application fields, e.g., logistics and general transport phenomena.

Keywords. Agent-based Reinforcement Learning; Traffic flow control; Logistics Transport control; Self-organising MAS; Agent-based Simulation

1. Introduction

Traffic jams and disturbance in traffic flows are ubiquitous in modern cities. Traffic is a distributed complex problem with hard predictable dynamics on global domain. Logistics is a closely related domain with overlapping dependencies. Both domains can be considered basically as a distributed optimising problem of a large-scale dynamic system including chaotic effects. Arising of jams, dead locks in traffic flows without a clearly identifiable cause, or transport delays are prominent examples. A traffic situation as well as logistics transport [1] consist of a large set of individual entities treated as agents that interact with each other and satisfying constraints (i.e., streets, traffic signs, traffic rules, dangerous situations, and so on). Individual traffic entities are controlled by a set of behaviour rules. These behaviour rules can be significantly influenced by varying parameter sets (i.e., different classes of drivers and individualism of behaviour and goals) and decision making. The individual entities are controlled by a set of sensors $\boldsymbol{S}$, representing the individual entity, local area, and global area states.

Commonly, traffic flow is controlled via traffic signals (traffic lights) and dynamic signs (e.g., speed control) based on accumulated flow data (real-time). Logistics transports obeys similar control infrastructures. There are different (basically spatial) domains considered by controllers and learner instances that have to be distinguished: Global (urban city scale), "glocal" (transition from global to local level, i.e., connected groups of streets and street areas), local (one street, part of a street, a crossroad and street junctions, crowds), and micro level (single vehicles or people). Adaptive traffic flow control on different spatial and domain levels is attractive to reduce travelling times and energy consumption (i.e., air pollution). Most work in this field focuses on traffic signal control (e.g., an overview can be found in [2]). Other aspects like individual driver decision making and path routing influencing traffic flows of is not considered. Although there are traffic simulation that consider driver behaviour, an assumption of average behaviour is made without considering real-world variations [3]. Only few work is known that incorporates crowd sensing data (one example can be found in [4]).

Machine learning can be used to improve traffic flow control and user experience on macro- and micro-scale level (local optimisation). But the re-

quired training of ML models cannot be performed in real-world environments. Simulations can overcome this limitation and can be used to (pre-)train machine learners and to investigate different traffic flow control and machine learning algorithms. But simulation relies commonly on simplified, averaged and unified behaviour models, simplified environments and situations. Agent-based modelling and simulation (ABMS) is suitable for large-scale, distributed, and complex dynamic systems with local interaction models [5].

Traffic flow control should be achieved on three levels:

1. Ensemble control by the environment (using traffic signals and signs) using common traffic control algorithms based on sensor data (collected, e.g., by street cameras);
2. Individual control by driving entities. e.g., influencing long-range routing and decision making of individuals via social media or navigation systems → *addressed in this work*;
3. Local automatic group control (i.e., car-to-car communication and control) using local interaction agents.

In [6] and [7], self-organising traffic control was applied to traffic light signal switching. In this work, there is a focus on self-organising traffic flow control on individual level (level 2) by support decision making processes of drivers (or automatic or more advanced of autonomous vehicles), particular addressing short- and long-range routing.

In previous work. the influence of behaviour model variations from an average behaviour of traffic entities (drivers, passengers) were investigated by using ABMS with digital twins derived from CWS [5]. Commonly, agent-based simulation and agent-based distributed computing (ABC) performing the traffic control are separated. The approach from [5] unites simulation and real-world data processing by an unified mobile agent-model covering ABM, ABS, and ABC, enabling the tight coupling of real and virtual (simulation) worlds in real-time.

The approach in this work is composed of three paradigms to create smart traffic control: 1. Cooperating and interacting multi-agent systems; 2. Reinforcement learning (RL); 3. Self-organisation and self-adaptivity. RL (e.g., Q-learning) was aleradady applied to traffic control [8].

A model-free agent-based reinforcement learning approach is used and evaluated in this work to enable self-organising traffic flow control on micro-scale entity level [2]. Self-organising is implicitly performed by solving or rewarding constraints between physical entities and sensor feedback, e.g., distances between vehicles, spatial street constraints, and so on. Agent-based reinforcement learning is already applied in broad range of applications, including traffic control and decision making in logistics combined with simulation [1].

RL is closely related to the agent model. In [9], multi-agent systems perform distributed traffic signal control. Among distributed learning, distributed learning-agents can be deployed, each operating on a local state and optimising a sub-set set or one particular target variable. This approach requires co-ordination to optimise on global level implementing distributed co-ordination of exploration and exploitation (DCEE, introduced by Brys et al. [10]).

The main objective of this work is to find a multi-agent-based and self-organising urban traffic control architecture suitable to optimise traffic flow, i.e., increasing the average traffic flow speed, minimising or completely avoiding jams, minimising the travelling times with respect to passengers, and minimising energy for mobility. In contrast to other work ([8], [11], [9], [10], [2]) controlling traffic lights and signals only, this work will focus on the control of decision making processes of adaptive long.range routing only incorporating experience and history situations.

A novel hybrid agent architecture based on coupled reactive rule-based and learning-based action selection for long-range navigation in cities is introduced and finally evaluated with agent-based simulation.

2. Agent-based Modelling and Simulation of Traffic

This section introduces the extended agent architecture coupling the original activity and rule-based

agent model with RL and identifies state (sensor) variables required for decision making and learning.

There are two classes of agents covered by one unified agent model that is used in this work (see [5] for details): 1. **Physical behavioural agents** representing physical entities in virtual worlds (simulation) like vehicles or individual artificial humans; 2. **Computational agents** representing mobile software in real and virtual world, i.e., used for distributed data processing and digital communication, and used for implementing sensing and negotiation bots.

Both types of agents are used in the simulation, but only computational agents can migrate between the simulation world and real world environments. The computational agents are required for seamless integration of mobile crowd sensing into the simulation (optionally in real-time), discussed in the next sub-section

Hybrid Rule- and Learning-based Agent Architecture

Although RL agents itself pose a well defined architecture, in this work the RL instance is used as a co-function for the state transition and action computation, shown in Fig. 1. Now there are two output functions *action* and *RL* that select appropriate actions to be executed by a controller agent (i.e., controlling the vehicle). Although both output functions can select actions from the same set of actions *Act*, it is feasible to split the action set in two sub-sets Act_1 and Act_2 used by the two output functions, respectively. A final fusion of both action selections (that can be contrary or cross-forbidden) is performed by the *fusion* function. There are three different RL algorithms mapping state variables on actions available for the agent: 1. Temporal Difference Learning; 2. Dynamic Programming; 3. Deep Q Learning. The addition of the *RL* function extends the functional agent model of reactive state-based agents [5]:

$$\begin{aligned} percept &: Sen \times Per \rightarrow Per \\ next &: St \times Per \times Par \times R \times C \rightarrow St \\ action &: St \times Par \rightarrow Act_1 \\ rl &: r \times Per\times \rightarrow Act_2 \\ reward &: Act_2 \times Per \times Par \rightarrow r[-1,1] \\ fusion &: Act_1 \times Act_2 \rightarrow Act \end{aligned} \tag{1}$$

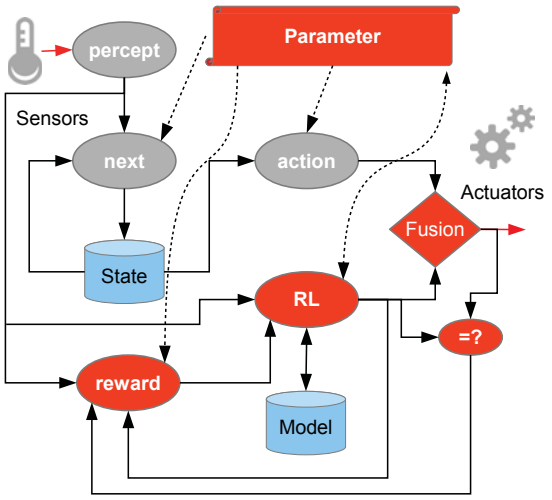

Figure 1. The proposed hybrid parameterised agent architecture combining reactive state-based action selection with RL

The parameter set can be changed at run-time by the agent itself to adapt to specific known or new situations, eventually modified by the RL instance, too. The parameters define also the superposition of rule-based and learning-based action selection (in the range from 0-100%).

The fusion function is basically a lazy constraint solver that checks the two actions provided by the rule-based action function and the predicted action from the RL function. The constraint solver checks for contradiction and invalid actions. That means it refuses invalid actions with respect to the current agent state. For instance, changing the direction is

currently not possible (spatial constraints) or re-routing exceeds a specific frequency. The selected action output of the *fusion* is feed back to the *reward* function (comparing the fusioned and the predicted action).

3. Coupling Multi-Agent Systems and Reinforcement Learning for Traffic control

The self-organised traffic control (in sense of optimisation) MAS consists of the following agent classes: 1. *Vehicle controller* agents performing basic rule-based car control and short-range navigation; 2. *Communication* agents (bridging vehicles and TSC entities); 3. *Navigation* agents (coupled to vehicles) performing adaptive and optimised long-range navigation using Reinforcement learning (RL); 4. *Traffic* controller agents (rather simple in this work)

RL can be applied to traffic signal control (RL-TSC) on global or local environmental domain level and/or to entity control on individual level, e.g., vehicle control. Hierarchical and domain-based learning was proposed by Abdoos et al. [11]. An RL instance is associated to a learner agent that can be coupled to other agents, like traffic control or driver agents. The RL agent outputs a recommendation for actions to be considered by a controller agent, e.g., speed limitation, traffic light switching, or route planning and decision making. There are centralised Single Agent RL (SARL) and decentralised Multi Agent RL (MARL) approaches. This work covers a hierarchical MARL approach.

RL requires sensor input **S** and a feedback via utility and conflict functions defining the reward function $u(\mathbf{S})$. An RL model outputs an action a from a set of possible actions A executed by the controller agent. The sensor system for traffic lights is usually based on stationary vehicle detectors, like inductive loops or cameras using vision algorithms to identify vehicle flows. Most traffic control algorithms consider only actual traffic situations without considering predictions of future variations, flows, and context changes. The sampling of vehicle data is often inaccurate, e.g., the speed of vehicles, introducing sources of error. Other variables influencing traffic like crowd flows, events (emergencies, work closings, shopping), and time-specific crowd flow variations are commonly not considered in traffic flow control. Sensory input from social media and other urban sensors can be considered by ML, too.

The sensor input $\boldsymbol{S}$ is union of three sets:

1. Internal sensors (GPS, speed, direction)

2. Environmental sensors (distance)

3. Global sensors (traffic, jam, crowds)

$$S(\vec{x}, t, state) = \mathrm{S}_{\mathrm{internal}} \cap S_{\mathrm{external}} \cap S_{\mathrm{global}} \quad (2)$$

On macro-level, the main issue in RL-TSC is the determination of the state of the environment from sensors that is represented to the RL agent. Typical state variables (on a macro-scale level) are:

```
StateVariables-Macro = {
  ql: Average queue length,
  qt: Average waiting time,
  fr: Average flow rate,
  va: Average vehicle speed
      (normalised to speed limit),
  te: Average travelling time
      efficiency,
  pe: Average travelling distance
      efficiency,
  st: Signal change times and delays,
  en: Energy (electrical energy or
      fuel consumption),
  ut: User satisfaction (overall utility)
}
```

With respect to the micro-scale level, another set of state variables are introduced to provide sensors S of variables capable to detect traffic situations from the view of point of single vehicles:

```
StateVariables-Micro = {
  v0:  Normalised average speed,
  ds0: Distances to s={front, back,
       left, right}
       neighbour vehicles,
  de, Δde: Distance to destination and
       delta change (progress),
  td:  Direction to destination
       (0-360 degree),
  r0:  Direction of vehicle
       (0-360 degree),
```

```
  qt0: Queuing time,
  dd:  Allowed driving and turning
       directions,
  P:   Set of possible paths from
       current position to destination
}
```

These state variables are primarily used to adapt the vehicle driving control based on self-organisation of local ensembles and to recognise jam situations (present and future situations). For each state variable x_0 there is a desired value x_1. Fig. 2 shows the principle MAS configuration, the sensors of the agents, and their communication paths. Vehicle agents (consisting of a coupled controller and learner agent pair) can sense their neighbourhood using their own sensors, the sensors from neighbouring vehicles, and the sensors of nearby traffic light signals stations.

A traffic agent uses the following state variables:

```
StateVariables-Traffic  = {
  ql: Queuing length,
  qT: Queuing time,
  va: Average vehicle speed,
  fr: Flow rate,
}
```

There are three different RL algorithms mapping state variables on actions available for the agent: 1. Temporal Difference Learning; Dynamic Programming; 3. Deep Q Learning. Only the latter is suitable for continuous state spaces!

The reward function returning a value in the range [-(a+b+c),(a+b+c)] (a,b,c are weight factors) of the vehicle agent uses the superposition of the ratio of actual and average speed, actual and average queuing time, and the progress to reach the destination:

$$u = a\frac{v_0-\bar{v}_0}{max(v_0,\bar{v}_0)} + b\frac{qt_0-\overline{qt}_0}{max(qt_0,\overline{qt}_0)} + c\frac{sd-\overline{sd}}{max(sd,\overline{sd})} \quad (3)$$

Vehicle agents can communicate with neighbouring nodes to get group (vehicle ensemble) sensor data (e.g., queue length). Additionally, vehicle agents nearby a traffic light signal can communicate with the traffic controller agent to get switching information (remaining time of green/red phase, switching times, ..) and traffic flow sensor data.

The learned RL models are portable and mobile, so they can be exchanged between a set of agents selecting the best trained models from a set of models or by permuting models to improve prediction. Since RL is an incremental learning method, the pre-trained models are improved during runtime (application and learning) with new feedback (reward).

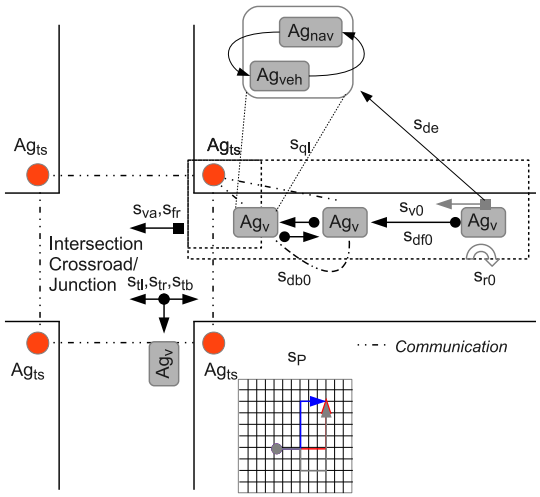

Figure 2. Agents, their sensors (s_x with respect to the sensor variable x), and agent communication paths (Ag_{ts}: Traffic sign agent, Ag_v: Vehicle agent)

Behaviour of Vehicle Agent

The vehicle agent is responsible to implement decision tree and rule-based automatic driving to satisfy the following constraints: 1. Driving on the right side of a street in the currently selected direction (North, South, West, East); 2. The vehicle speed v may not be higher than the speed limit on the current road: $v<v_{max}$; 3. The distance to the next vehicle ahead may not be lower than a speed-dependent distance limit: $df < df_{min}$; 4. There may no collision (two vehicle may not occupy the same place): $(x_a,y_a) \neq (x_b,y_b)$. Any violation of these constraints results in the execution of an action of the set of actions A:

```
Act = [
  Moving one step
    left, right,
    backward,
    or ahead: |Δ|=1,
  Increasing or decreasing the
    vehicle speed,
  Follow short-range Δ displacement
    vector (minΔ),
  Stopping movement
]
```

A rule-based decision tree is used to select an action a of the set of actions A.

Behaviour of Navigation Agent

The navigation agent is responsible for long-range navigation to optimise the following target variables: 1. Distance to destination; 2. Average Velocity; 3. Travelling time. It uses the previously introduced RL mapping sensor input on actions passed to the vehicle agent. The navigation agents gets sensor input from the vehicle agent.

```
Act = [
  Change direction to N/S/W/E,
  Keep direction (forced),
  Change vehicle speed,
  Change destination,
  Escape blocking situations
]
```

4. Simulation Study and Evaluation

The experiments were performed with the SEJAM2 simulation framework (detailed description in [5]). The simulation world consists of an artificial street map with 14 long streets (36 street segments) arranged in a two-dimensional grid (horizontal and vertical orthogonally crossing streets), 49 crossroads and junctions, and 144 traffic light signals. For the sake of simplicity the world is discretised by a mesh grid (100×100 cells).

The simulation was carried out with 200 vehicle-navigation agent groups with a static parameter set. The vehicle agent represents a physical vehicle and performs rule-based short-range navigation. The coupled navigation agent performs rule- and learning-based long-range navigation.

For a first evaluation of the new micro-level traffic control approach, fixed green-red cycles (50%-50% duty cycle) and mutual exclusive switching of perpendicular crossings of streets are assumed.

Each navigation agent has a randomly chosen start and end point on the street map. After a vehicle reaches its destination it restarts driving to its original starting point and vice versa.

A simplified RL function using Q-learning (DQN agent) with a neural network was chosen with a sub-set of vehicle state variables and a sub-set of vehicle control actions changing the direction of the vehicle by long-range navigation:

$$\begin{aligned} &rl(s_l, s_r, s_f, s_b, de, \Delta de, td, r_0, v_0) : \\ &(s_l, s_r, s_f, s_b, de, \Delta de, td, r_0, v_0) \rightarrow \\ &\{North, South, West, East, turn, speed+, speed-\} \end{aligned} \tag{4}$$

The s_{dir} sensors (vectors) provide spatially resolved information of vehicles and streets in current left (*l*), right (*r*), front (*f*), and back (*) *direction of a vehicle (deriving *df*, *db*, *dl*, *dr*, *dd* state variables). The reward function *us*(*S*) consider the current time scaling *ts*, path efficiency *pe*, distance progress Δde, and the suitability of the vehicle direction.

The iterative learner of each instance used a discount factor of γ=0.9, an ε-greedy policy value of 0.2, and a learning rate of α=0.05.

Some preliminary simulation results showing the progress and improvement of traffic in Fig. 3 with continuous training runs of navigation agents. The learning leads to an an increasing influence on the route planning (short-range and long-range). The average vehicle speed was 1/5 world grid steps / simulation step. About 2000 training events (rewarding actions) in 400k simulation steps were applied to each learning instance.

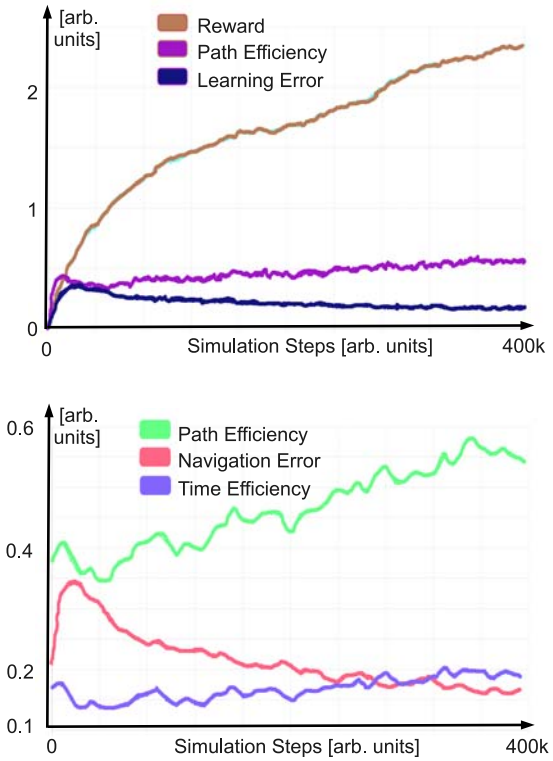

Figure 3. (Left) Progress of learning of all vehicle instances (Right) Improvement of utility on global scale with respect to the path and travelling time efficiency

There is a significant improvement observed of the utility on global scale with respect to the path and travelling time efficiency (defined by the ratio of the shortest to the routed path length and travelling times) by about 50%. The reward (accumulation of all 200 learning instances) show a monotonous increasing (starting initially with negative values not shown). The overall learning error (returned by the learner instances in arb. units) decreases monotonous, too. The short-range navigation error (suggesting not allowed directions) decreases significantly during learning progress. No traffic jams or deadlocks were observed with RL navigation in contrast to earlier work in [5] using pure rule-based and non-adaptive routing.

5. Conclusion

In contrast to common traffic management controlling traffic lights and signals only, this work addressed traffic flow optimisation on micro-level by adapting decision making processes, primarily re-routing and vehicle speed control. Vehicles were represented by vehicle agents provided with an extended set of sensors and performing automatic driving (short-range navigation). A navigation agent performing long-range routing is controlled by a novel agent architecture with a composition of the classical reactive state-based agent model and rule-based action selection with RL and reward functions.

Preliminary results from an agent-based simulation of an artificial urban area show that the deployment of micro-level vehicle control just by individual decision making and re-routing based on local environmental sensors can improve the traffic flow on global level significantly in terms of path length and travelling times compared with shortest path navigation (without RL).

The proposed hybrid agent architecture can be deployed in a broad range of self-organised path optimisation problems with interacting entities, e.g., logistics and transport problems. Local and ubiquitous as well as global sensor input is required to perform RL. Training requires a high number of learning iterations (RL feedback updates) not suitable to be performed in real worlds. Instead large-scale agent-based simulation is used to pre-train the agent finally adapting in real worlds.

Further investigations have to be carried out to evaluate the global emergence and stability. The decision making of vehicle agents relies on rules and a black-box function learned from only a few state variables. One highly interesting aspect to be considered and evaluated is the possibility to exchange already learned models with other navigation agents introducing multi-agent co-operation and im-

proved optimisation (model permutation).

6. References

1. D. Tamagawa, E. Taniguchi, and T. Yamada, *Evaluating city logistics measures using a multi-agent model*, in Procedia Social and Behavioral Sciences, The Sixth International Conference on City Logistics, 2010.
2. T. LeoMcCluskey, A. Kotsialos, J. P. Müller, R. Schumann, O. Rana, and F. Klügel, Eds., *Autonomic Road Transport Support Systems*. Springer, 2012
3. P. G. Balaji, X. German, and D. Srinivasan, *Urban traffic signal control using reinforcement learning agents*, IET Intell. Transp. Syst., vol. 4, no. 3, 2010
4. S. A. Fayazi and A. Vahidi, *Crowdsourcing Phase and Timing of Pre-Timed Traffic Signals in Presence of Queues: Algorithms and Backend System Architecture*, IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 3, 2016.
5. S. Bosse, U. Engel, *Real-time Human-in-the-loop Simulation with Mobile Agents, Chat Bots, and Crowd Sensing for Smart Cities*, Sensors (MDPI), 2019, doi: 10.3390/s19204356
6. M. Ferreira, R. Fernandes, H. Conceição, W. Viriyasitavat, and O. K. Tonguz, *Self-Organized Traffic Control*, in VANET'10, September 24, 2010, Chicago, Illinois, USA, 2010
7. B. Płaczek, *A self-organizing system for urban traffic control based on predictive interval microscopic model*, Engineering Applications of Artificial Intelligence, vol. 34, 2014
8. M. Abdoos, N. Mozayani, and A. L. C. Bazzan, *Traffic Light Control in Non-stationary Environments based on Multi Agent Q-learning*, in 2011 14th International IEEE Conference on Intelligent Transportation Systems Washington, DC, USA. October 5-7, 2011
9. I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, *Reinforcement learning-based multi-agent system for network traffic signal control*, IET Intelligent Transport Systems, vol. 4, no. 2, pp. 128–135, 2010
10. T. Brysa, Tong T. Pham, and M. E. Taylor, *Distributed Learning and Multi-Objectivity in Traffic Light Control*, Connection Science, 2014
11. M. Abdoos, NasserMozayani, and A. L. C. Bazzan, *Hierarchical Control of Traffic Signals unisg Q-learning with Tile Coding*, Appl Intell, 2013
12. J. Wan, J. Li, Z. Shao, A. V. Vasilakos, M. Imran, K. Zhou, *Mobile Crowd Sensing for Traffic Prediction in Internet of Vehicles*, Sensors, vol. 16, 2016.
13. M. Rabe and F. Dross, *A reinforcement learning approach for a decision support system for logistics networks*, in Proceedings of the 2015 Winter Simulation Conference, 2015.