Modelling and optimizing city traffic using an agent platform

Robert Pniewski[1], Katsiaryna Stankevich[1] Maria Ganzha[1], and Marcin Paprzycki[2]

[1] Warsaw University of Technology, Warsaw, Poland
[2] Systems Research Institute Polish Academy of Sciences, Warsaw, Poland
r.pniewski@student.mini.pw.edu.pl,ostankevichea@gmail.com,
m.ganzha@mini.pw.edu.pl, marcin.paprzycki@ibspan.waw.pl

Abstract. Increasing number of urban residents is systematically increasing the demand for smart traffic management. This contribution presents an intelligent solutions for dealing with common traffic issues, such as unexpected roadblocks, accidents, or public transport failures. To this effect a city traffic simulation tool, based on software agents, which incorporates traffic management strategies is proposed and experimented with. It is experimentally shown that the proposed approach can decrease the travel time.

Keywords: Smart city, Software agents, Traffic management, Traffic simulation

1 Introduction

The concept of a smart city aims, among others, at the development of ubiquitous transport ecosystems, in which participants move to their destinations with minimal disruptions. Here, minimal disruption may mean that the public transport schedules are appropriately synchronized to minimize transport time between most popular destinations. It can also mean that in case of disruptions the system can provide alternative ways of reaching the destination in such a way that the total "time loss" will be minimized. In this context, Raphael Gindrat points out (see, [4]) that more than 50% of the world's population lives in metropolitan areas, which directly contributes to the increasing need for intelligent traffic management.

In our previous work (see, [5] for all details), an agent-based city traffic simulator has been introduced. The developed system involved two main capabilities: (1) intelligent traffic light management, based on vehicle-light communication, and (2) smart bus transport enhancements, based on traveller-to-bus-stop communication. Initial experiments indicated that the proposed approach may be able to reduce the travel time, encouraging further exploratory research.

The aim of this contribution is to discuss how the initial simulator has been extended, to manage commuting obstacles, such as road accidents, public transportation failures, or unexpected roadblocks. Specifically, the following scenarios have been considered. (1) Let us assume that cars could communicate with an "obstacle management entity" (OME). The OME may inform the vehicles about road accidents, and route obstructions, so that they can react accordingly. This, in turn, may prevent unnecessary increase of travel time. (2) Assume that travellers and buses can communicate with a "public transport management entity" (PTME). In the case of a bus failure, its passengers may be provided (by the PTME) with information about best solutions to reach their travel destination. Again, both message-based communication, and strategic decision making, are the core concepts of the proposed agent-based system.

Before proceeding further let us note that, due to the space restrictions, and the fact that no further pertinent literature has been found, we have decided to direct interested readers to the State-of-the-Art summary presented in [5].

The remaining parts of this paper are organized as follows. In Section 2 we discuss the design of the extended version of traffic simulator. Next, in Section 3 the way that the simulator was implemented is briefly outlined. Finally, in Section 4, description of experiments has been presented.

2 Agent-based simulator design

The main assumptions of the traffic simulator, outlined in [5], remained unchanged. However, roles of selected agents have been adjusted, and few new agents have been introduced. Note that, in the system, all modelled entities are represented as agents. One should keep this in mind to understand that, for instance, *Bus* agent is an agent that represents a bus moving on the strets of the city.

- *SmartCity*: single agent responsible for overseeing transport movement. Its role, was considerably reduced, in comparison with the original simulator (it remains in the new system mainly for technical reasons – as an entity that simplifies the implementation of the simulator).
- *TroubleManager*: single agent that, during the simulation, monitors traffic jams, construction sites, and accidents. Using information from *Car* agents, *TroubleManager* marks "trouble points" and broadcasts them to travellers.
- *LightManager*: agents managing traffic lights (one per crossroad). The role of the original *LightManager* was extended by the capability of detecting traffic jams. Information about such situations is forwarded to the *TroubleManager*.
- *StationManager*: bus stop agent, responsible for communication between busses and travellers, to allow making the departure times more flexible.
- *Bike*: configurable number of agents that facilitate change of mode of communication. They represent persons travelling by bikes and inform the *SmartCity* when the destination is reached. they communicate with *LightManager*s, providing specific crossroads with ETA of reaching them.
- *Car*: configurable number of agents. Representing situation when drivers notice an obstacle, during the simulation *Car*s "generate" trouble spots, (construction/accident) and inform *TroubleManager* about them (type and location). *Car*s adjust routes, based on information about obstacles.

- *Traveller*: configurable number of agents, representing travellers.The new feature is dealing with bus crashes. Here, the *Traveller* can choose from two options: wait for the next bus, or travel by a bike. The decision is based on the length of the remaining journey and bike availability.
- *Bus*: number of *Bus* agents depends on the number of bus lines passing through the chosen simulation area (simulations used information about actual bus lines in Warsaw). After a bus crash, all travellers leave the bus.
- *BusManager*: delivers information about bus departure times, (e.g by responding to *Traveller*s requests). Moreover, it is informed (by *Bus* agents) about bus crashes. This info is used to update the bus schedule.

Overall, the simulator was extended by the following features: trouble places (locations which cause traffic disruption, e.g. accidents or construction sites), traffic jam management, and bus crashes. Therefore, to facilitate needed functions, new communication between concerned entities was introduced. The agent knowledge map is shown in Figure 1. It overlaps with the information exchange map, and represents the interactions described above. Connected agents send messages both ways.

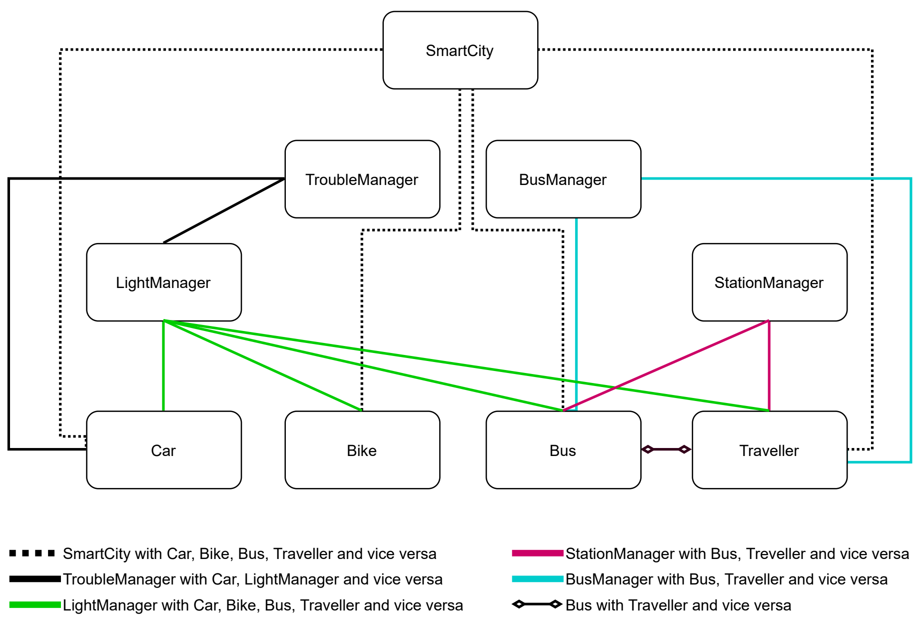

Fig. 1. Agent knowledge & information exchange map

Let us now describe the key communication patterns (represented in Figure 1) in more detail. As noted, *Car*s are responsible for detection of construction/accident locations. When an obstacle is spotted, *Car* agent sends an INFORM message to the *TroubleManager*, with information about the position and type of obstruction. The *TroubleManager* checks if the problem is already registered. If not, then *TroubleManager* saves the received data (updates the internal representation of the traffic situation in the city). The traffic-update-broadcast, in the form of a PROPOSE message, is sent to all road users, allowing change of the route, to evade the construction/accident location.

When a traffic jam occurs, the *LightManager* gathers the necessary information: position, length of the jam (measured by the number of cars), and sends an INFORM message to the *TroubleManager*. The *TroubleManager* processes this information and broadcasts a PROPOSE message to all *Car*s. Hence, *Car*s can make decisions about their route. Next, when the traffic jam disappears, the *LightManager* sends an INFORM message to the *TroubleManager*, reporting that the traffic jam is over. *TroubleManager* broadcasts a PROPOSE message to all *Car*s, allowing them to, possibly, adjust travel routes, again.

In case of a crash, the *Bus* sends an INFORM messages to all concerned *Traveller*s, *Station*s, and the *BusManager*. This message includes all key data: crash coordinates, bus line number, and accident time. When the *BusManager* receives this information, it updates the schedule. *Traveller*s react according to their strategy: switch to the bicycle, or wait for the next bus. When *StationManager*s get the message, they send INFORM messages to upcoming *Traveller*s, and those already waiting at bus-stops, so that they may adjust travel plans. Since the *BusManager* agent was created, communication pattern, between *Traveller*s and *StationManager*, has been modified accordingly.

As far as communication *Traveller* ↔ *StationManager, BusManager* is concerned, the most suitable bus line is obtained by *Traveller*s using the REQUEST message sent to *BusManager*, specifying the destination. Based on the content of the response, *Traveller* sends an INFORM message to the pertinent *StationManager*, containing the ETA, and the desired bus line. When the *Traveller* arrives at the bus stop, REQUEST-WHEN message is sent to the *StationManager*, with the request to enter the bus straightaway. When the *Bus* arrives, the *StationManager* sends a REQUEST message, asking *Traveller* to enter the bus. En route, at each station, *Bus* informs concerned *Traveller*s about their arrival.

The *Bike* agent inherited the communication with the *LightManager* from the *Car*. The remaining aspects of communication remained untouched.

3 Simulator implementation

The new release of the simulator has been implemented. Here, the backend did not change (other than updates and bug fixes). It is implemented as a Maven project, chosen for convenient dependency management. The code is written in Java. Moreover, JADE framework is used in order to implement agents and agent communication. However, the frontend is now implemented as a web page. It is written in JavaScript, and is using React framework. This dramatically increased the responsiveness of the system, allowing for larger simulations. Communication between frontend and backend is handled via WebSockets.

Moreover, a closer look had to be taken at the simulator design. The system had been heavily dependent on the quality of the Internet connection, due to frequent calls to external APIs [5]. The current simulator, is no longer so sensitive to the connection issues, due to the introduction of a new, complex caching functionality. Specifically, the data retrieved from the external APIs is cached for future reuse. Hence, the simulator can generate cars and other objects significantly faster, allowing captured traffic to be denser. It can be claimed that, deployment of a system based on the principles outlined above, could be possible in a real city, due to the increased tool reliability.

4 Experimental validation

Series of simulations have been performed, with an increasing number of cars 5, 10, ... , 40, while maintaining the same size of the simulation area. For instance – 5 cars means, that the journey time has been measured for the last (5th) car, which joined the simulation. The following questions have been posed: (1) Does the proposed traffic jam response improve cars' travel times? (2) Does the agent communication help in case of obstacles? and (3) Does the implemented communication improve travellers' commuting experience in case of bus failures?

4.1 Traffic jam strategy

To answer the first question, the average time of car to travel from point A to point B with, and without, smart traffic jam management was measured. (1) Without the strategy, cars remain on the jammed route. When the traffic jam management is applied, affected cars may be able to change their route accordingly. The main factor of the strategy is the anticipated journey time on the jammed route vs. the predicted trip duration through the route, which evades the jam. Based on this, the traffic jam management system decides whether the car shall choose the alternative route. The comparison is depicted in Figure 2.

It is obvious that the traffic jam strategy vastly improves the cars' journey times. In the largest experiment, an improvement of ≈**63%** was observed.

4.2 Traffic obstacle management strategy

To answer the second question, the average trip time has been measured for cars travelling from point A to point B with, and without, obstacle management. In "normal conditions", if the cars' routes involve an obstacle, cars do not change routes, until they stumble upon the trouble point (generated over a fixed period). On the other hand, if the traffic obstacle management is active, affected cars adjust their routes as soon as they are informed about it by the system (even if it is located "far away"). The results are presented in Figure 3.

It is clearly visible that the trouble point strategy helps in case of road construction sites and unexpected accidents. Again in the largest experiment, an enhancement of ≈**34%** could be observed. Moreover, it was establisged that the average gain was ≈**21%**.

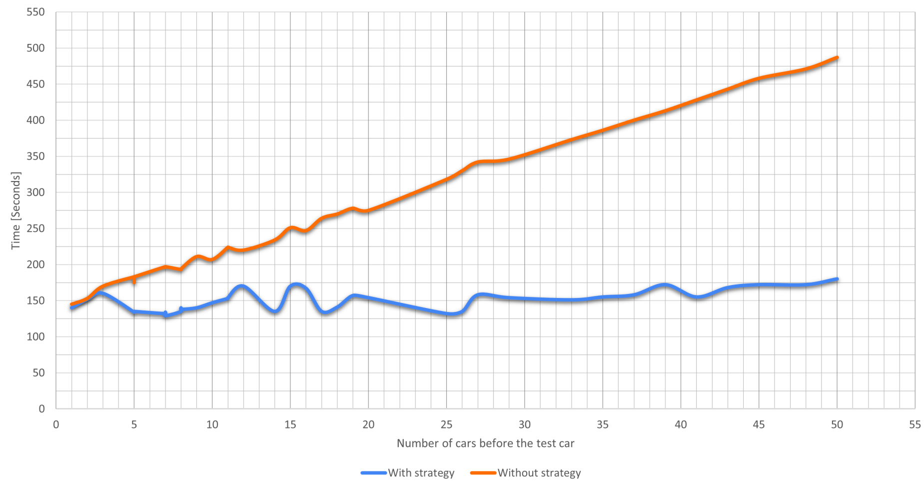

Fig. 2. Traffic jam strategy results

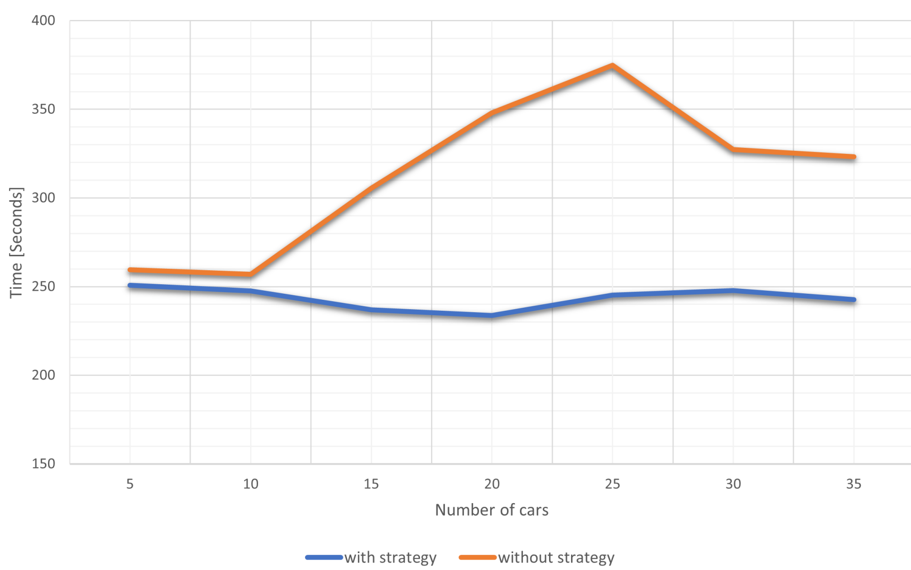

Fig. 3. Trouble point strategy results

4.3 Transport mode switching strategy

For the last question, the average trip time was measured for those travelling with, and without, bus failure management. The test scenario involved a bus, which crashed, letting the passengers disembark at the failure point. Normally, the traveller, upon leaving the damaged bus, goes to the nearest station and awaits the next bus. When the bus failure management was applied, travellers, upon bus failure, could choose to continue their journey either by the next bus or by a nearby city bike. The decision was made on the basis of bike availability and resulting travel time (to establish if switching to a bike makes sense or not). The results are shown in Figure 4.

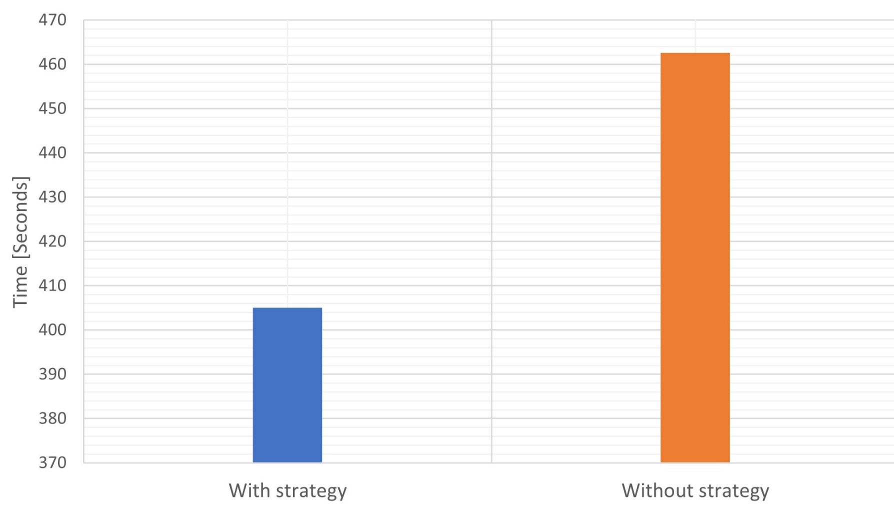

Fig. 4. Transport change strategy results

As can be seen, the transport mode change strategy has improved the average travel time by up to ≈**12.5%**.

Apart from the described, newly-introduced traffic management strategies, the system has also been improved in terms of its core functionalities, namely in the field of smart traffic lights. The average journey time has been measured for cars travelling from point A to point B with, and without, smart traffic light management. Results of experiments for the old simulator design can be found in [5], while the results for experiments performed with the current, improved simulation tool, have been depicted in Figure 5.

Overall, cars' average journey times' improvement has risen up to ≈**48%** in the largest case, **as opposed to** the ≈ **6%** enhancement achievable in the previous version of the simulator.

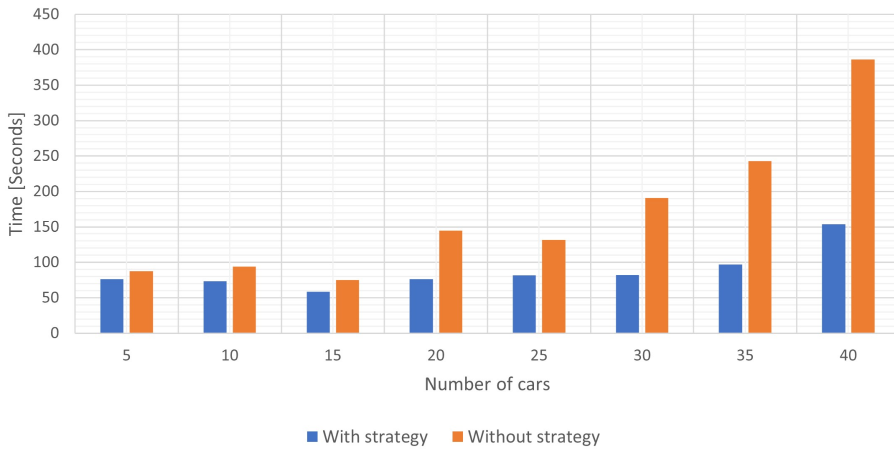

Fig. 5. Traffic light strategy results achieved with current simulator iteration

5 Concluding remarks

This work concerned modelling and optimisation of traffic in a smart city. Strategies, responding to traffic congestion, accidents and public transport failures have been proposed and experimented with. It was observed that the proposed optimisation strategies result in travel time improvements reaching 60% for the largest simulations.

References

1. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. 147, 195?197 (1981). doi:10.1016/0022-2836(81)90087-5
2. May, P., Ehrlich, H.-C., Steinke, T.: ZIB structure prediction pipeline: composing a complex biological workflow through web services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148?1158. Springer, Heidelberg (2006). doi:10.1007/11823285_121
3. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
4. R. Gildrat: In a world of autonomous vehicles, this is why we'll need more public transport than ever. In: July 2019 BigThink https://bigthink.com/technology-innovation/autonomous-vehicles
5. Pniewski R., Sellin D., Stankevich K., Ganzha M., Paprzycki M. (2021) Applying Software Agents to Make City Traffic Management Smarter. In: Goyal D., Gupta A.K., Piuri V., Ganzha M., Paprzycki M. (eds) Proceedings of the Second International Conference on Information Management and Machine Intelligence. Lecture Notes in Networks and Systems, vol 166. Springer, Singapore https://doi.org/10.1007/978-981-15-9689-6_72