Annales UMCS Informatica AI XIII, 2 (2013) 97–105
DOI: 10.2478/v10065-012-0050-y

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI
http://www.annales.umcs.lublin.pl/

Integration of multiagent crossing controllers for measure-oriented traffic control

Wiesław Trochonowicz[1,2*]

[1] *Institute of Fundamental Technological Research Polish Academy of Sciences, ul. Pawinskiego 5B; 02-106 Warszawa, Poland*
[2] *Institute of Computer Science, Maria Curie-Sklodowska University, pl. M. Curie-Sklodowskiej 5, 20-031 Lublin, Poland*

Abstract – In this paper we will present an integration of multiagent crossing controllers for measureoriented traffic control. It is the basis for the future multiagent traffic control system which uses measure oriented approach to control traffic. This integrated architecture gives us the possibility to react to the dynamic changes of traffic thanks to the data it can gather with its sensors.

1 Introduction

Making transport flow accurate is beginning to be a very big problem in our everyday life. When the traffic flow is not proper it can bring many disadvantages, not only money loss but also health problems. That is why in the age of increasing motorization it is most important to put effort into dealing with this problem as best as we can. One of such steps is the creation of the introduced enhanced architecture for the multiagent traffic control system which will be based on MATSim. Thanks to MATSim we can see the results of our work in the simulated environment. Transportation simulations such as MATSim are very important tools in making decisions concerning traffic management. That is why it is necessary to have such tool with which we can create simulations of our new designs. Such an approach was used in this paper. The presented integrated architecture needed a simulation kit to see the results it creates. For that purpose we chose MATSim.

*wieslaw.trochonowicz@gmail.com

2 Enhanced architecture

In this chapter we will show how the introduced architecture was created and which tools were used for its implementation. We will show which basic tools were used as the implementation background and also the implemented elements and their place in the architecture.

2.1 Basic tools

The basis of the created architecture is MATSim. It is a Multi-Agent Transport Simulation Toolkit created in Java language. It allows to create large scale traffic simulations with the use of agents. It is built of many modules which does not have to be combined and can be used alone. One of the most important things is the flexibility that MATSim gives. Provided modules can be extended or even replaced by our own implementations. This allows programmers to test every, even the smallest, aspect of their work. Currently, MATSim offers a toolbox for demand-modelling, agent-based mobility-simulation (traffic flow simulation), re-planning, a controller to iteratively run simulations as well as methods to analyze the output generated by the modules [**1**]. The language that this simulation toolkit is written in is Java. This allows us not only start it on almost every computer but to extend it easily to our needs.

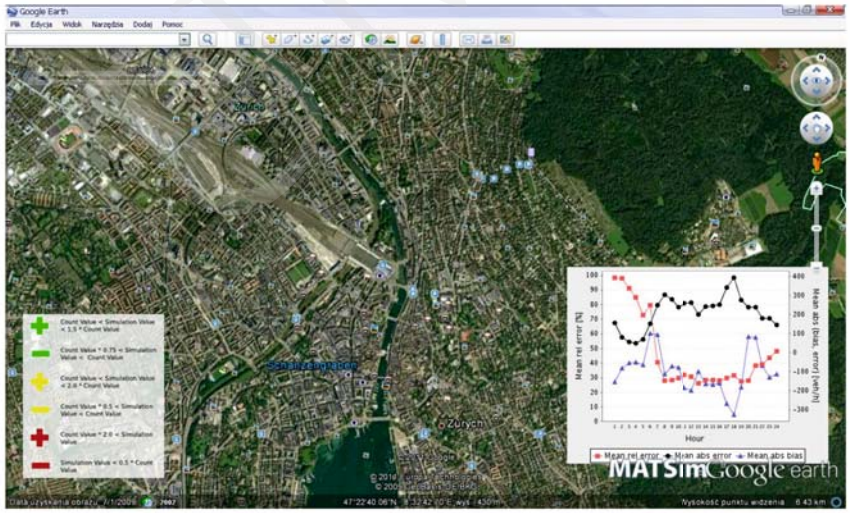

Fig. 1. MATSim simulation example using the Google earth.

Used programs:

(1) MATSim 0.3.0
(2) Eclipse Helios Service Release 2
(3) Java JDK 6
(4) Senozon via - Visualization and Analysis tool

2.2 Implemented architecture

Created architecture has got many elements that make it work properly. First elements are the xml files in which we have the structure of the simulated scenario. These elements are loaded when the simulation is started and are turned into their implemented representations. These are the needed (and additional) elements:

(1) config.xml - the configuration file for the simulation
(2) network.xml - the network file is the representation of the simulated network on which the simulation takes place
(3) lanes.xml - this special file represents all lanes that are on the links that the network consists of
(4) signal_groups.xml - in this file we have got signal groups which group signal lights
(5) signal_systems.xml - consists of signal systems and adequate signal lights representations
(6) amber_times.xml (not needed for proper work of the system) - this file contains static amber lights times for signal lights.

Next we have the extended side of the program shown in Fig. 2.

Fig. 2. Schema of the implemented extended architecture.

Combination of these elements gives us the possibility to control crossings in the simulation created by MATSim. A simple example of one such controlled element is shown in Fig. 5. This figure is a snapshot of the simulation visualized in Senozon via. Of course it does not have to be one crossing that we can control. There can be almost an infinite number of them. Default the signal system controller, which controls traffic lights, is defined in the file signal_control.xml (of course the name of the file can be different from the one used in this paper, it depends on the programmer). Definition

of one such controller is shown in Fig. 3. Extended architecture does not need to define the agent crossing controller in a special way. Each controller is turned into an agent and no special instructions in the definition files are needed. Fig. 3 shows a signal system controller which we turn into an agent crossing controller. So the basic approach is changed. This signal system controllers are no longer just switches which just switch on some traffic lights but have evolved into agent crossing controllers which of course, still maintain their basic duty of changing the traffic lights.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<signalControl xsi:schemaLocation="http://www.matsim.org/files/dtd
http://www.matsim.org/files/dtd/signalControl_v2.0.xsd"
xmlns="http://www.matsim.org/files/dtd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <signalSystem refId="11">
    <signalSystemController>
      <controllerIdentifier>DefaultPlanbasedSignalSystemController</controllerIdentifier>
      <signalPlan id="11">
        <cycleTime sec="120"/>
        <offset sec="0"/>
        <signalGroupSettings refId="0111">
          <onset sec="0"/>
          <dropping sec="55"/>
        </signalGroupSettings>
        <signalGroupSettings refId="1011">
          <onset sec="0"/>
          <dropping sec="55"/>
        </signalGroupSettings>
        <signalGroupSettings refId="1211">
          <onset sec="0"/>
          <dropping sec="55"/>
        </signalGroupSettings>
        <signalGroupSettings refId="2111">
          <onset sec="0"/>
          <dropping sec="55"/>
        </signalGroupSettings>
      </signalPlan>
    </signalSystemController>
  </signalSystem>
</signalControl>
```

Fig. 3. Definition of signal system controller.

We can easily extend its range of work space by adding new traffic lights which the controller will control. This can be achieved by adding proper definitions of signal lights in the signal_system.xml file and then consider them in files signal_control.xml and signal_groups.xml. Controlled traffic lights does not have to be even connected to the same crossing that we control. One controller can control many signal lights scattered across the network but in the basic approach we assume that all traffic lights that the controller controls are connected to one crossing.

The start of the extended architecture in implementation takes place with the creation of the simulation shown in Fig. 4.

The most important part is the point where we set the extended signal controller listener factory for the simulation controller. This allows us to add our own signal controller listener implemented in class WTSignalControllerListener. In this class we set our signal manager and sensor manager. Setting signal manager requires to call our WTSignalModeFactory which allows us to create WTController. This is the most important class and the heart of the system. Creating these elements is just half of the success. Now having this controller running still does not change anything. These controllers are still blind and operate only on default level. What we need is the interaction between the controller, his traffic lights and the actual traffic. This is achieved by another element of this architecture. The sensor manger is responsible for managing all sensors on created links and lanes. By creating an extended controller automatically there are set sensors on lanes that are connected to it. Thanks to this move the created controller can get data about the density of the traffic from the sensors that are on lanes connected to the crossing it controls. After creation of all these elements, the architecture is complete. It can now not only send control data to elements it controls but also get data from them.

```
Controler controler = new Controler(configFile);
controler.setSignalsControllerListenerFactory(new
WTSignalControlerListenerFactory()); controler.setOverwriteFiles(true);
controler.run();
```

Fig. 4. Place of the beginning of extended architecture.

The element shown in Fig. 5 represents one crossing with connected links (roads) and lanes that are placed on them. The controller of the crossing is an agent. For now this agent works independently of other agents. It makes adjustments to the traffic flow only locally.

2.3 Agent schema approach

How we can see this architecture is decentralized. One agent does not depend on another. It works alone and makes decisions depending on the state of its crossing. Creating a decentralized infrastructure of agent crossings which can work autonomous is a very important point of future development. Each agent can work independently of other agents making its own decisions. Of course a crossing agent is not hermetically closed for the world. In this basic approach that is presented we show how agents work independently of each other making it a base for future work.

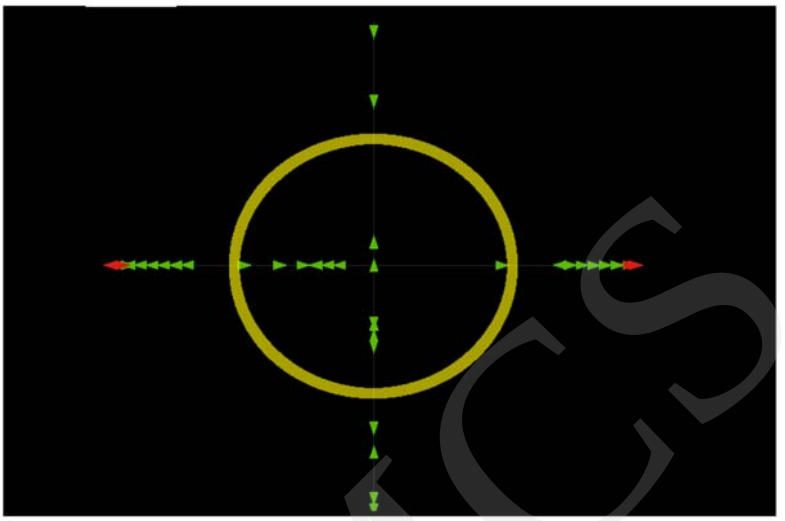

Fig. 5. Controlled element.

3 Possibilities

These are the possibilities that are given by the changes that were made:

(1) replacing original controller by an agent controller
(2) getting information of traffic flow
(3) manipulation of traffic light times
(4) information exchange between connected agents

3.1 Replaced controller

Basically the default controller was just like a switch which changed lights according to the static schema which was written for it in the proper xml file. What we wanted to be achieved was a creation of an agent which could dynamically react to the information that it gathers and of course, interact with other agents. One agent controlling the crossing, shown in Fig. 5, can change the flow of the traffic by adjusting proper times of signal lights it controls. Thus it can manipulate traffic flow. As already mentioned, it adjusts flow of the traffic only locally. But this is just the basic approach. Each agent is equipped with interaction possibilities with agents that are connected to him. This feature is created but it can be the subject for future work and therefore it is not used for simulation made for this paper.

3.2 Getting information of traffic flow

For created agents that control their crossings, it is crucial to know what happens around them. In this basic approach agents know what happens on their crossings.

One of the goals was to get information about the density of the traffic on lanes that are connected to the controlled crossing. This was achieved by creating sensor elements and putting them on proper lanes. Those sensors can gather information about traffic density on their lanes. When the agent wants to know what traffic is on his crossing he makes a call to the sensor manager and gathers the information about the traffic from proper sensors.

3.3 Manipulation of traffic light times

After getting the information from the sensors about the traffic, the agent controller makes proper changes to the times of the signal lights. The algorithm of time choosing is not the subject of this paper and therefore it will not be considered. Changes to the times of proper lights is the main tool of controlling traffic in this case. It allows the controller to make some adjustments and thanks to them make traffic flow more accurate.

3.4 Information exchange between connected agents

This feature is created in the architecture but was not used to determine the behaviour of agents controlling crossings. This is mainly because a way of using information from the neighbour agent must be taken under consideration. This interface was created for future work which will be focused on describing how different information is used by different agents which control their crossings. Such a connection is shown in Fig. 6 where an example of crossing connections is given. Each crossing is controlled by a proper agent and crossing connections in network are adequate to those established by the agents with each other.

4 Future work using enhanced architecture

As mentioned this architecture is a base for future work which will grow around it. The problems that will be taken under consideration:

(1) algorithm of traffic lights times manipulation
(2) use of information exchanged between different agents
(3) types of information gathered by agents

4.1 Algorithm of traffic lights times manipulation

This is a crucial element which must be taken under consideration. There are many questions which must be answered here. The main question is how the gathered information alters the duration of green, red, yellow light times turned on. What effect do times chosen on one lane have on times on other lanes. This is just a basic question which has to be dealt with when approaching an algorithm of traffic lights manipulation. This problem is more complex and needs focusing on each of its parts.

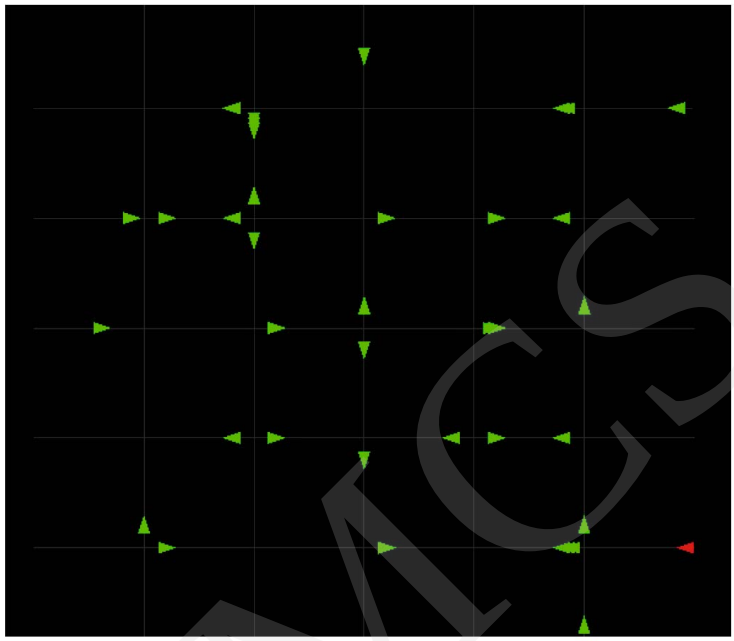

Fig. 6. Agent crossing controllers connected together, represented as crossings.

4.2 Use of information exchanged between different agents

One agent can only create local improvement of the traffic flow. This is not enough. One of the goals was to allow agents to communicate with each other. This was achieved but it is just a communication with no effects on the environment. The future work will focus on the use of globally gathered information and what kind of impact it has on the traffic flow. A way to use this information has to be created. It is not so important how we gather this information as how we use it, or rather how the system consisting of crossing controller agents will use this information.

4.3 Types of information gathered by agents

In the basic model the only information gathered by the controller agents is the number of vehicles located on lanes. This information is only the first information that is taken under consideration. But this is not the only information that could be gathered by the agents controlling the crossings. Many other types of sensors could be created and could gather other types of information. Types of information that will be gathered will be the subject of future papers. Gathering everything that we can gather is not necessary and it would be only a waste of time and processing power of the system. Of all of the information that the controller agents can gather there have to be chosen those which can in reality bring performance in the meaning of better traffic flow.

References

[1] MATSIM www page. MultiAgent Transport SIMulation. http://matsim.org/, accessed 2011. URL www.matsim.org.