Published in Lecture Notes in Computer Science, Advances in Artificial Intelligence - IBERAMIA 2004: 9th Ibero-American Conference on AI, Puebla, Mexico, November 22-25, 2004, ISBN:3-54023806-9

MARCS
Multi-Agent Railway Control System

Hugo Proença[1] and Eugénio Oliveira[2]

[1] Universidade da Beira Interior, IT - Inst. Telecom.
UBI, R. Marquês D'Ávila e Bolama, 6200-001, Covilhã, Portugal
hugomcp@di.ubi.pt
[2] FEUP/LIACC-NIADR
FEUP, Av. Dr. Roberto Frias 4200-465, Porto, Portugal
eco@fe.up.pt

Abstract. Previous research works have demonstrated that traffic control models based on the comparison between an historical archive of information and current traffic conditions tend to produce better results, usually by improving the system's proactivity behavior. Based on this assumption, we present in this paper MARCS - Multi-Agent Railway Control System, a multi-agent system for communications based trains traffic control. For this purpose we have developed a system infrastructure based on an architecture composed of two independent layers: ”Control” and ”Learning”.
”Control” layer is responsible for traffic supervision, regulation, security and fluidity, including three distinct agent types: ”Supervisor”, ”Train” and ”Station”.
The ”Learning” layer, using situations accumulated by the ”Control” layer, will infer rules that can improve traffic control processes, minimizing waiting time and stop orders sent for each train. At this moment, inferred rules seem like:
”At T_1 moment, when a train is located at $P_1 = (x1, y1)$ with destination E_1 and another one is at $P2 = (x2, y2)$ with destination E_2, a traffic conflict in L_1 after t_1 seconds” will occur.
Rules of this kind are transmitted to the control system to be taken into account whenever a similar traffic situation is to occur. In the learning process we apply an unsupervised learning algorithm (APRIORI).

1 Introduction

1.1 Motivation

Railroad traffic volume will have in the next two decades a significant increment. More people and merchandize will circulate in increasingly bigger and faster railway networks [4].

Although traffic's scheduling systems guarantee that, for the foreseen conditions, vehicles in circulation will not compete simultaneously for the same resources (do not conflict), they lack of flexibility in order to enforce security.

Once railway networks will be under dynamic conditions it is most desirable that the control system becomes more flexible knowing how to provide an answer to these new requirement in an autonomous way.

Our propose consists in a completely distributed, decentralized and adaptable architecture for railway traffic control in a communications based train control [3].

1.2 Summary

The proposed system can be decomposed in two different sub-systems: ”Control” and ”Learning”.

”Control” sub-system is responsible for traffic management and guidance in the network and includes three agent types: (”Supervisor”, ”Train” and ”Station”). Those agents must interact with the objective of providing control mechanisms to the displacement of each train in the network. In this context, security is the principal concern, trying to assure that train crashes will never occur. Having security guaranteed, it is then important to maximize systems’s overall efficiency, by minimizing conflicts between trains. In our terms, a trains ”conflict” occurs when several trains wish to cross at same time the same place. When that situation occurs, control sub-system is responsible for assigning priority, and sending ”stop” commands to trains that must wait until the resource crossing zone is free.

”Learning” sub-system is the complementary one and has the objective of analyzing system’s past accumulated situations descriptions and identify typical cases that became the origin of later conflicts. The objective is to make the learning sub-system to infer rules that anticipate train conflicts and be able to make the ”Control” system to benefit from them.

”Control” sub-system must compare all current train positions with the ones that can been identified by each rule. If any match is found, the predicted conflict must be avoided and all necessary actions must be taken in order to do it.

As we will show in section 4 this proactive behavior tends to minimize train conflicts and, under certain conditions, improve system’s efficiency.

1.3 Related Work

Probably induced by increasing traffic congestion problems, multi-agent systems applied to transportation domain, usually concern road transportation.

Typical approach tend to divide covered area by several traffic management agents, each one coping with decision responsibilities in a specific parcel. Often, authors decide to implement agents that represent every vehicle in circulation and the self physical infrastructure.

”TraMas” [7] is a system aiming at the study of multi-agent systems viability in the road transportation domain, as well as testing applicability of different models and cooperation strategies in multi-agent systems. In TraMas every cross point is represented by a traffic agent that is responsible for respective traffic control. Each one of the traffic control agents is independent enough to decide locally, but is also prepared to share information (cooperate) with other agents.

TraMas system includes three layers (Cooperative, Decision and Control).

[8] proposes a peculiar road traffic multi-agent system. Main objective consists in vehicles movement coordination inside a delimited area. Having a GPS-like localization method, every vehicle is represented by an agent, and each network parcel is managed by a control agent. Each control agent is responsible for analyzing specific traffic volume and construct the system essential’s component: ”co-field”. The co-field is a 3D representation of the environment, having two principal characteristics:

– Areas with high traffic density are represented as higher altitude (mountains).
– Areas with low traffic density are represented as lower altitude (depressions).

Vehicle agents are responsible for route selection through minimization travel cost. As expected, it is cheaper to travel in descendent directions. This fact induces vehicles to avoid areas with high traffic density, potentially where they could spent more time in result of traffic conflicts.

Proposed in 1994, dMARS [9] is a multi-agent system where two agent types interact: ”Intersection” and ”Street”. They establish cooperation mechanisms with neighbors in order to produce an emergent system behavior. Every agent gets as input the traffic volume in represented area, being responsible for maintain that information and provide it to neighbors agents that request it.

[12] propose a multi-agent system for trains traffic coordination with one peculiar characteristic: natural language interface. When a traffic agent fells a high uncertain degree about what the correct action is, it starts user interaction section. User will analyze current traffic situation and communicate correspondent action in an oral form. This information must be achieved by the agents, and applied in next similar situation.

Several multi-agent systems [11] [10] and models [13] have been proposed, each of them with specific characteristics but sharing with the ones referred above: area (and responsibilities) sharing and inclusion of cooperation mechanisms enabling to go from a local to global perspective.

2 MARCS Architecture

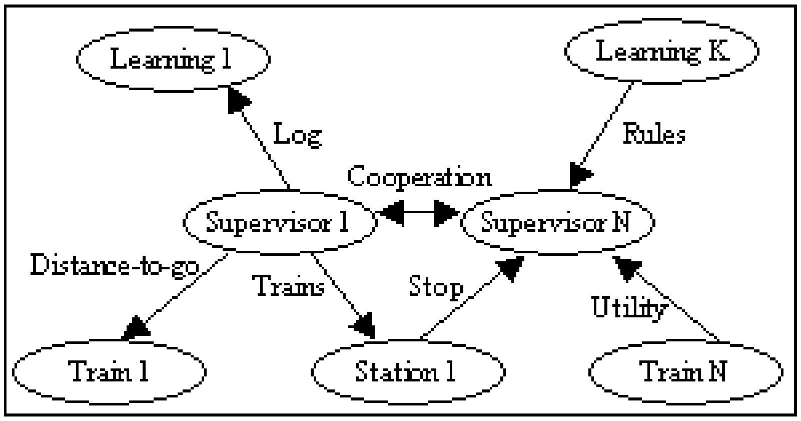

Fig. 1. MARCS architecture

Figure 1 gives a global perspective of MARCS architecture, existing agents in both sub-systems (”Control” and ”Learning”), and interaction mechanisms between them.

Following what has been said before, the figure shows the interaction between four distinct agent types:

– $Supervisor$. These agents must control, guide and guarantee security in traffic network for each specific area. Each area is delimited by latitude and longitude coordinates. $Supervisor$ agents are the only ones that simultaneously belong to control and learning sub-systems.
– $Train$. This agent type exclusively belongs to control sub-system. Train agents represent correspondent interests and are responsible for train velocity control, depending on free-distances (”distance-to-go”) assigned by $Supervisor$ agents.
– $Station$. Represents the interests of a railway station, and also belongs exclusively to control sub-system. Station agents objectives consist in administrate platforms, giving orders for trains arrivals and departures and providing useful users (passengers in train stations) information.
– $Learning$. This agent type belong to ”Learning” sub-system. Learning agents task consists in asking for control agent registry log, analyze it, to identify possible existent meaningful patterns to infer possible rules that can optimize traffic fluidity.

2.1 Control Sub-System

Control sub-system main objective consists of providing secure and efficient routing for all trains while preventing crash situations and maximizing traffic fluidity.

This is accomplished through $Supervisor$, $Train$ and $Station$ agents information exchange consisting of both data and plans.

2.2 Learning Sub-System

As it was reported above , MARCS learning sub-system includes two different agent types: $Learning$ and $Supervisor$, being this one also part of control sub-system. Performance and efficiency were primary factors analyzed at design time, inducing these architecture based on two parallel sub-systems.

Usually control systems have rigid time requirements making them adequate for ”real time”. On other hand, learning processes (specially data mining ones) tend to consume much computational resources and spend too much time until useful results become available. For our application, it was crucial that control processes priority was preserved, as well as guaranteeing that each real time traffic situation could be effectively analyzed, with no interference of any other task or objective that an agent possibly could have.

Another important factor is system modularity, which could also facilitate overall implementation. Once learning process tasks are easily distinguished from control ones, it becomes more intuitive the implementation by means of different computational entities.

Based on these requirements, $Learning$ agents are learning sub-system essential components. Their unique objective consists in asking registry activity to $Supervisor$ agents, concatenate and analyze it and infer rules that potentially increment system´s efficiency.

At creation time, each $Learning$ agent receives a list of $Supervisor$ addresses and becomes responsible for periodically asking for their activities record.

This consists in a ”log” file maintained by each agent. It contains all received and sent messages and most relevant actions taken. For example:

```
1065189498 SEND     tell:sender supervisor1:receiver train1:content Distance 290.474074
1065189501 RECEIVE  tell:sender simulator1:receiver supervisor1:content Location train1 112.2 21.3
1065189501 RECEIVE  tell:sender simulator1:receiver supervisor1:content Location train2 213.2 -125.9
1065189501 PROCESS  Reserve vertex 12 Train train1
1065189501 SEND     tell:sender supervisor1:receiver train1:content Distance 230.882501
1065189501 SEND     tell:sender supervisor1:receiver train2:content Distance 75.127011
1065189503 RECEIVE  tell:sender simulator1:receiver supervisor1:content Location train1 117.2 55.1
1065189503 RECEIVE  tell:sender simulator1:receiver supervisor1:content Location train2 158.4 1.4
1065189503 PROCESS  Conflict Vertex 13 Trains 2 train1 train2
```

After complete log file content´s transmission, $Supervisor$ work in learning process´s compass is complete, being all following tasks performed by $Learning$ agents, like related in the next sections.

3 Learning Process

Proposed learning process consists in the analysis of potential conflict situations (Section 2.1) that can occur and identification of train positions that originated those conflicts. If similar traffic conditions will repeat, control system must anticipate that conflict and take necessary actions to prevent and avoid it.

Figure 2 shows learning process state diagram. It consists of a preliminary phase of data pre-processing, followed by the algorithm execution, result analysis and knowledge acquisition. In a final phase, new knowledge is sent back to control sub-system, hoping that it will contribute for traffic fluidity and system effectiveness, by avoiding traffic conflicts.

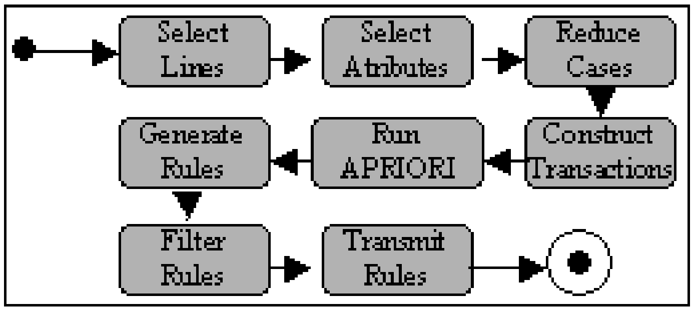

Fig. 2. Learning Process (State Diagram).

3.1 Data Pre-Processing

An agent activity record consists in a text file containing all exchanged messages plus relevant actions taken.

Every line has format:

<time> <type> <description>

Where $type$ can be one of ”send”, ”receive”, ”process” or ”exception” respectively for sent and received messages, actions taken or exceptions handled. $description$ contains message or action additional description.

In the pre-processing phase there are four stages [5]: line selection, attribute selection, instance reduction and transaction construction.

In the end, data is formatted in an adequate way to be dealt with by next learning phase.

Transaction Construction The next step consists in building transactions needed for algorithm execution. In the APRIORI [2] context terms, a transaction consists of an items set grouped by any criteria. For this purpose, we will group items relative to historic train locations that later originate a traffic conflict. A conflict identification between two trains (C_a and C_b) in location L_a and moment T_t, implies the selection of lines relative to absolute position of C_a and C_b in past $(t - i)$ moments, $i = 1..n$, being all grouped in the same transaction.

Let i_{conf} be a ”Conflict” at time t_{conf} and C=$\{C_1, C_2, \ldots, C_n\}$ the set of trains involved. Define $\alpha \in N$, as the analysis retrospective limit. For each i_j line relative to C_j train location at t_j time: if ($t_j < t_{conf}$), ($t_j >= (t_{conf}$ - α)) and $C_j \in C$ then add i_j to same transaction as i_{conf}.

As an example, see the set of items displayed below:

ID	Time	Type	Description	Destination
1	t_1	Location	Train C_1 (x_3, y_3)	Destination D_1
2	t_2	Location	Train C_2 (x_2, y_2)	Destination D_1
3	t_2	Location	Train C_1 (x_1, y_1)	Destination D_1
4	t_3	Conflict L_1	Trains C_1, C_2	
5	t_8	Location	Train C_3 (x_2, y_2)	Destination D_1
6	t_8	Location	Train C_4 (x_1, y_1)	Destination D_1
7	t_9	Conflict L_1	Trains C_3, C_4	

This set originates two transactions (T_1 and T_2):

Transaction	Items
T_1	1, 2, 3, 4
T_2	5, 6, 7

3.2 Algorithm for Association Rules

APRIORI allows the identification of association rules from large data sets grouped through transactions. Just as described in [2], let $\mathcal{I}$=$\{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let $\mathcal{D}$ be a set of transactions, where each transaction is an items set $\{i_j\}$, j=1..k, such that $i_j \in \mathcal{I}$. An *association rule* is an implication of form X $\Rightarrow$ Y, where X $\subset \mathcal{I}$, Y $\subset \mathcal{I}$ and X $\cap$ Y = $\emptyset$. The rule X $\Rightarrow$ Y holds in the transaction set $\mathcal{D}$ with *confidence* c if c% of transactions in $\mathcal{D}$ that contain X also contain Y. This rule has support s if s% of transactions in $\mathcal{D}$ contain X $\cup$ Y.

For a transactions set $\mathcal{D}$, the problem of mining association rules consists in generate all association rules with support and confidence higher than a specific threshold.

The logic of this algorithm is based on the observation that if any given set of attributes S has support lower than the threshold, any superset of S will have lower support and consequently any effort to calculate the support for such supersets will be wasted. For example, if we know that $\{$A,B$\}$ is not supported it follows that $\{$A,B,C$\}$ or $\{$A,B,D$\}$ will also not be supported [6].

3.3 Rule Generation

After the execution of APRIORI [2], we have identified a frequent sets group $\mathcal{F}$, such that $\mathcal{F}$=$\{F_1$, F_2,…,$F_n\}$, being each F_i an items set. Let F_i=$\{i_1$, i_2, …, $i_n\}$ be a frequent set with dimension n. Adding a constraint to force that consequent only have one item, we build a group of n association rules R, with R=$\{\forall$ i $\in F_i$: (F_i - i) $\Rightarrow$ i$\}$.

For our propose, we consider relevant association rules those with consequent type equal to ”Conflict”, meaning that we are interested in identifying those items (states) that usually occur together with a ”Conflict” item.

For every frequent set F_i with dimension n, we can identify an association rule set R_i of dimension m (m < n), such that:

$$r \in R : \{i_{r1}, i_{r2}, ..., i_{rk}\} \Rightarrow i_{conf}, Type(i_{conf}) = "Conflict"$$

Like referred above, filtering only two distinct items type (”Location” and ”Conflict”), all identified rules during the learning process have the form: ”**IF** Train C_1 is at (x_1,y_1) with destination D_1 **AND** Train C_2 is at (x_2,y_2) with destination D_2 **AND** … **THEN** Conflict in L_1, Trains (C_1, C_2, …), Time t.

Rule Selection After rules generation process, it is important to select the most relevant ones and communicate them to the control system´s agents. Above described process allows the identification of a large association rules set, most of them irrelevant. For instance, an inferred rule that will foresee a conflict between trains in the next second is not of great utility. On the other hand, if two rules r_1 and r_2 foresee conflicts for the same trains at the same place in, respectively, t_1 and t_2 seconds ($t_1 < t_2$) with the same support and confidence, then r_1 is irrelevant, because there is another rule that anticipates the same situation at a former stage.

For the rules selection process we have defined a lower threshold l_i ($l_i > 0$) and consider every rule r_i=(i_1, i_2,…,i_n) $\Rightarrow i_{conf}$, with Time(i_{conf}) $< l_i$ irrelevant. In a second phase we compare every remaining rule with each other, to analyze if exists a better rule for same situation.

Consider r_i a rule that anticipate a conflict at L_i place in about t_i seconds. Also let (i_{l1}, …, i_{ln}) be locations of most ancient states attached to n trains involved in L_i conflict. If exists a rule r_j that also anticipate a conflict at L_i in t_j seconds ($t_j < t_i$) with (j_{l1}, …, j_{ln}) as respective locations for trains and there is only a single possible path between respective j_{lx} and i_{lx} places then r_i is irrelevant.

3.4 Rule Transmission

After the rule selection process, a relevant set of rules has been identified and we can proceed for last stage, its communication to control sub-system agents. This process concerns about trains localization, $Supervisor$ agents identification, those with responsibilities for traffic management at the specific place, and rules transmission.

Having a rule r_i, such that r_i=($i_1, \ldots, i_n$) $\Rightarrow i_{conf}$, we analyze parameter ”Position” of every item i_j, j=1,..n, and determine who’s $Supervisor$ is responsible for those coordinates. Hereinafter, we proceed for rules transmission in KQML coded messages like the one exemplified in section 4.

This message informs $Supervisor_1$ that, if two trains have positions of respectively (861.0, -4.9) and (714.2,-263.1) and travel with direction $Station_1$ and $Station_2$, it will occur a traffic conflict in about 61 seconds, at Vertex 0, being Vertex 0 the internal representation of a specific railway cross point (switch point).

3.5 Control System Repercussions

On analyzing present train positions, $Supervisor$ agents make the comparison with every received rule and, finding a match, proceed to ”conflict avoid” process.

At this moment, this process consist in asking evolved trains to increase or decrease their usual velocity by α% during t seconds, α,t > 0, conforming with the conflict time foresaw.

Traveling during a time interval at superior or inferior velocity then desired, can be enough for trains to arrive at predicted point at different moments avoiding the conflict and improving system’s performance.

4 Experimental Results

Evaluation of a traffic control system could be done under multiple perspectives, perhaps with subjective components about the selection of most relevant characteristics.

”Security” should always be on top of priorities. It is crucial to assure that the system provide sufficient security mechanisms to avoid train collisions. ”Capacity” and ”Efficiency” could also be system evaluation parameters.

In our traffic simulator system, we implemented five evaluation parameters:

Crashes Number of train collisions.
Average Velocity Average velocity trains.
STOP Number of stop orders sent for trains.
Time Proportion at Desired Velocity Every train has a optimal velocity, according to its physic characteristics. This parameter represents the time proportion ([0,1]) that trains traveled at a velocity near the optimal one.
Simulation Time Total time spent until all trains arrive at final stations.

Experiments on several simulation scenarios like displayed on figure 3, allow us to conclude that rules inferred by the learning system have improved system performance by reducing ”Stop” and ”Simulation Time” parameters, and increase ”Average Velocity” and ”Time Proportion at Desired Velocity” without compromising security:

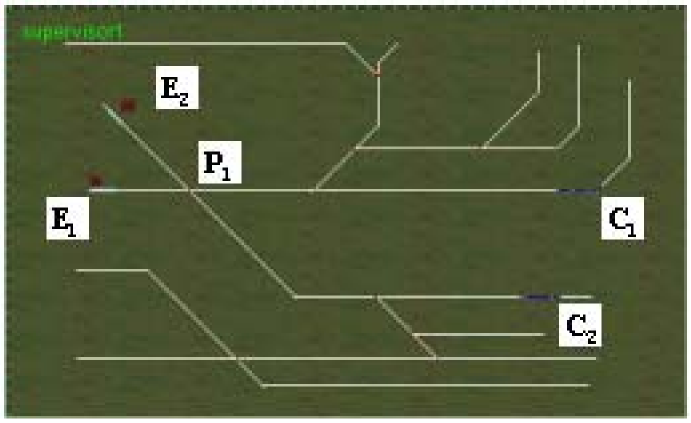

Fig. 3. Traffic simulation process

Parameter	**Before Learning Process**	**After Learning Process**
Crashes (Total)	0	0
Stop (Total)	1	0
Average Velocity (Km/h)	61	66
Time Proportion at Des. Vel. [0,1]	0.366	0.282
Simulation Time (mm:ss)	1.54	1.48

In the scenario displayed in figure 3 learning sub-system has identified the rule:

```
(tell:sender Aprender1
     :receiver Supervisor1
     :content (RULE
        Confidence 1.0
        Time -67
        TotalPremisses 2
            Premisse Local 861.0 -4.9 Destination Station2
            Premisse Local 768.0 -260.0 Destination Station1
        Consequent
            Conflict Vertex 0)
)
```

Rule displayed above informs ”Supervisor” that if two trains with position (861.0, -4.9) and (768.0, -260.0) move respectively to ”$Station_1$” and ”$Station_2$” it will occur one traffic conflict at vertex 0 (P_1).

Repeating simulation process, we observe that ”$Supervisor_1$” demand ”C_2” to reduce his optimal velocity, being this action enough to avoid the conflict at P_1.

5 Conclusions and Work in Progress

Experimental results allow us to conclude that, in specific cases, MARCS performance has been improved by applying learning system´s inferred rules.

Extending the learning process perspective, it is our intention to apply it to actions performed by the system in result of conflict anticipations. By now, learning system has the ability to anticipate conflicts, but cannot select the best action to avoid it, and cannot also anticipate whether actions performed to avoid a specific conflict will induce other future conflicts herder to be resolved.

Our work is currently focused on the analysis of the effects of actions performed to avoid conflicts, and determine those which are the optimal ones.

For this purpose, we plan to improve ”line selection” phase, passing to select ”Action” instances too. These elements specify actions taken by $Supervisor$ agents with the aim of avoiding a conflict.

Having this, we expect to infer new knowledge represented in the following example:

”Having a train C_1 located at P_1=(x_1, y_1) with destination D_1 and another C_2 located in P_2=(x_2, y_2) with destination D_2 it will occur a conflict in L_1 in t_1 seconds. **The best way to avoid this conflict is ask C_1 to decrease 10% his average velocity. Train C_2 must not accelerate because it will conflict with another one (C_3) in L_2 about t_2 seconds later ($t_2 > t_1$)**”.

Conflicts transitivity is other factor that we also plan to analyze, grouping conflicts that apply to common trains. We want to derive optimal actions to avoid groups of conflicts and not just isolated ones.

References

1. A.G. Hobeika, C. F. Kim: Traffic flow prediction systems based on upstream traffic. Vehicle navigation and information systems IEEE conference (1994).
2. R. Agrawal, A. Strikant: Fast algorithms for mining association rules. VLDB (1994).
3. Transportation Systems Design: Communications Based Train Control. http://www.tsd.org/cbtc (2003).
4. Toby Moncaster: More Room On The Tracks. The future of railway signalling. http://www.essex.ac.uk/ese/siddiquiaward/ (2002)
5. Pavel Brazdil: Data Mining. Master Course in Artificial Inteligence and Computation. Universidade do Porto (2002).
6. Frans Coenen: The Apriori algorithm. Department of Computer Science. University of Liverpool. http://www.csc.liv.ac.uk/ frans/Notes/KDD/AssocRuleMine/apriori.html (2001)
7. Eugénio Oliveira and José M. Fernandes: TraMas - Traffic Control Through Behaviour Based Multi-Agent System. http://www.ieeta.pt/ jfernan/tramas/ (1999).
8. Marco Mamei and Michael Mahan: Engineering Mobility in Large Multi-Agent Systems. SELMAS 2002 (110–122) (2003)
9. Tihomir Gabri and Emma Norling and Gil Tidhar and Liz Sonenberg and Nicholas Howden: Multi-agent Design of a Traffic-Flow Control System. (1994)
10. Vikram Manikonda and Renato Levy and Goutam Satapathy and David Lovell and Peter Chang and Anna Teittinen: Autonomous Agents for Traffic Simulation and Control. Transportation Research Record 1774 (1–10) (2003)
11. Danko A. Roozemond and Jan L.H. Rogier: Agent controlled traffic lights. ESIT(2000)
12. Alexander Huber and Bernd Ludwig: A Natural Language Multi-Agent System for Controlling Model Trains. http://www-wv.informatik.uni-erlangen.de/ bdludwig/pubs/huber_ludwig_camready.pdf (2002)
13. M. Antoniotti and A. Göllü: SHIFT and SMART-AHS: A Language for Hybrid System Engineering, Modeling and Simulation. USENIX Conference on Domain Specific Languages, Santa Barbara, CA (1997)