A CONFLICT-DIRECTED COORDINATION MECHANISM FOR TRAFFIC LIGHT CONTROL IN A SIMULATED ENVIRONMENT

José L. Aguirre
División Académica de Informática y Sistemas, Universidad Juárez Autónoma de Tabasco, Tabasco, México

Jesús H. Domínguez, Ramón Brena
Center for Intelligent Computing and Robotics, Tecnológico de Monterrey, Campus Monterrey, Monterrey, México

Keywords: Multi-agent coordination schemes, Conflict resolution, Simulation, Traffic light control.

Abstract: Car traffic control is a big issue nowadays, because of increasing time spent in traffic jams. A common mechanism that allows control of car flow is the use of traffic lights. Beyond static traffic lights' time assignments, we think it is possible to make traffic lights adjust the green time based on the current traffic conditions and coordinating by themselves. This paper presents a multi-agent based coordination mechanism followed by traffic lights in a simulated traffic intersection aiming to reduce the average car waiting time, compared to a traditional mechanism of static green time assignation. We call the mechanism "conflict-directed" since it is based on a conflict resolution strategy. The proposed mechanism has been tested with different cases on one and two independent intersections in a simulated environment. An evaluation of the conflict-directed mechanism performance is given.

1 INTRODUCTION

Car traffic control is a big issue in cities nowadays, because of increasing time spent in traffic jams. A common mechanism that allows control of car flow in big cities is the use of traffic lights and of centralized mechanisms where each traffic light on an intersection is assigned a constant green time. Beyond static traffic lights' time assignments, we think it is possible to make traffic lights coordinate themselves through decentralized coordination schemes, where the green time of the traffic lights is assigned based on the present conditions of traffic. With intelligent green time assignments, it is reasonable to think that the cars' waiting time could be reduced.

Simulation can be a useful tool when exploring coordination mechanisms applied to traffic lights because it allows us to experiment with different settings and analyze the effects of them in the variables of interest. In last years several car traffic simulators have been proposed to show how better results can be achieved with respect to traffic flow variables, like cars' route time and cars' waiting time (van den Bosch et al., 2003), (Balmer et al., 2004).

With this in mind, this paper presents a coordination mechanism for multi-agent (MA) systems followed by traffic lights in a simulated traffic intersection, aiming to reduce the average car waiting time in traffic intersections; then it is compared against the traditional mechanism of static green time assignation. We call the mechanism "conflict-directed" (CD) since it is based on a conflict resolution strategy. An implementation of the mechanism for conducting some experiments has been done using a MA based simulation environment.

The structure of the document is as follows: Section 2 presents a description of the proposed mechanism. Section 3 shows the conducted experiments along with the obtained results. Section 4 presents some related works. Finally, in section 5, conclusions and future work are given.

2 THE CD MECHANISM

Figure 1 shows some important concepts used in traffic light control that are needed to understand the CD mechanism (Bazzan, 2005). The *death time* is the number of time steps between a change of green light state at two consecutive traffic lights in the intersec-

Aguirre J., Domínguez J. and Brena R. (2009).
A CONFLICT-DIRECTED COORDINATION MECHANISM FOR TRAFFIC LIGHT CONTROL IN A SIMULATED ENVIRONMENT.
In *Proceedings of the International Conference on Agents and Artificial Intelligence*, pages 483-488
DOI: 10.5220/0001658104830488

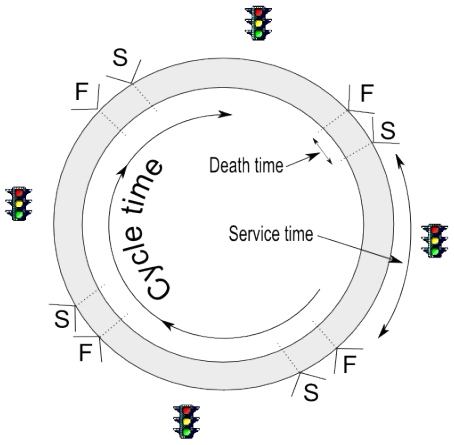


Figure 1: A graphical rendition of some traffic light control concepts. Time flows as indicated by the big curved arrow at the center of the gray ring.

tion. Every traffic light in the intersection is in its red light state during the death time. The *service time* is the total number of time steps where a traffic light is in its green light state. The *cycle time* is the number of time steps needed for a complete cycle around each service time of each traffic light in the intersection, counting also the total death time; the cycle time is represented in the figure as the grey ring. The *start time*, represented by the "S" letter, is the time step in the cycle time where a traffic light will change to green light state. The *finish time*, represented by the "F" letter, is the time step in the cycle time where a traffic light will change to red light state.

Also, the different types of agents handled by the mechanism must be introduced. A *Car Agent* represents cars; it can stop, decelerate, accelerate and turn direction based on what it detects in its forward movement direction under a "vision" limit. A *Light Agent* represents traffic lights; its internal states are green and red light colors. A *LightSet Agent* represents a group of Light Agents; the LightSet is in charge of changing the state of its aggregated Light Agents at the same time. Each LightSet has a *start* time and a *finish* time which determine when the LightSet will be on its green state. A *LightController Agent* represents a group of LightSet Agents; it synchronizes the LightSet Agents, determining which LightSet should reach the green state. Finally, a *Source Agent* spawns cars with a predefined probability λ into the position where the Source agent is placed.

2.1 The Mechanism

We will call a *conflict situation* during distribution of cycle time the case where some LightSet Agent is requesting non-available resources. In such a case, if the requested service time were assigned, the summation of every service time of every LightSet in the intersection would be greater than the cycle time. In *conflict situations*, we will say that a service time assignation *has conflict* or *is conflictive*.

The mechanism is based on the idea of transforming a conflictive assignation towards the *default assignation*, that has no conflict because the entire cycle time is evenly distributed among the LightSets. Of course, the idea is to solve the conflict before reaching the default assignation. The mechanism, which is shown in figure 2 as a state diagram, is followed by each LightSet Agent on the intersection. Each LightSet Agent can be in any of the states of that figure.

The general logic of the mechanism is the following: Every LightSet starts in the *normal* state. Each LightSet is measuring a variable called *utilization rate* that measures how crowded are its lanes. If a LightSet measures that the utilization rate grows above some critical utilization rate and there is not an active negotiation group in the intersection, then it starts a negotiation group and sends an *Invitation* message to other LightSet Agents in the intersection. The LightSet will be called the *owner* of the group. The message has the purpose to request from other LightSet Agents their *required service time*.

After sending the *Invitation* message, the owner goes to state *receivingConfirmations* where he will wait for all the *Confirmation* messages from other LightSet Agents. After receiving the *Invitation* message, the other LightSet Agents, which must be at *normal* state because there exists an active negotiation group, will register to the negotiation group with role *negotiator* and they will send back a *Confirmation* message containing their required service time. Every negotiator goes to *responding* state after receiving the *Invitation* message. When all *Confirmation* messages are received, the owner will go to *decidingConflict* state where he will verify if there is conflict in the assignation formed with the required service time of each negotiator (plus the required service time of the owner).

If there is no conflict, the owner will distribute any not-requested service time among the LightSets, in order to guarantee that the entire cycle time is being used. The distribution is done following this heuristic: "Give bigger parts of the not-requested service time to those LightSets who require more service time". The owner will prepare a *FinalAllocation* message

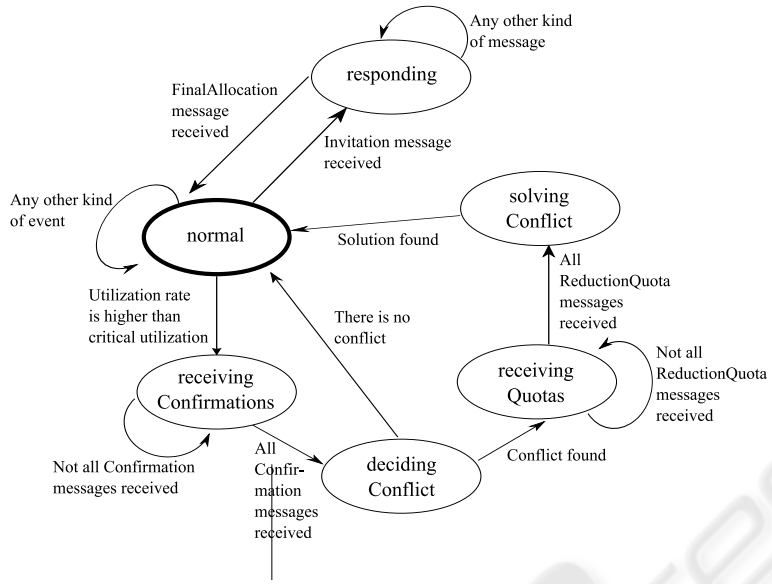


Figure 2: State diagram for a LightSet Agent working under the conflict-directed mechanism. The initial state is *normal*.

that will be sent to every negotiator in the negotiation group. The message contains the final assignation. The message informs every negotiator to go from *responding* state to *normal* state. The owner will also inform the LightController about the final assignation. The LightController will change the service times accordingly and he will close the negotiation group. The owner will return to *normal* state.

If there is conflict, the owner will prepare a *ConflictAllocation* message, requesting from the negotiators a reduction quota. The reduction quota is the quantity that needs to be subtracted from the requested required time of the negotiator such that his service time becomes the default service time, if the required service time is greater or equal to the default time (the default time is the time that results from the default assignation). The owner will go to *receivingQuotas* state where he will wait for the reduction quotas of the negotiators. The negotiators will respond with a *ReductionQuota* message containing their reduction quotas. After receiving all *ReductionQuota* messages, the owner will go to *solvingConflict* state where he will solve the conflict. The way the reduction quotas are computed guaranties that more rounds of reduction quota requests are not needed in order to solve the conflict. The owner will solve the conflict by taking away units from the originally requested service times following the heuristic: "take away more service time from those LightSets who require less service time". After solving the conflict, the owner will prepare a *FinalAllocation* message that will be sent to every negotiator in the negotiation group. The message contains the final assignation. The message informs every negotiator to go from *responding* state to *normal* state. The owner will also inform the LightController about the final assignation. The LightController will change the service times accordingly and he will close the negotiation group. The owner will return to *normal* state.

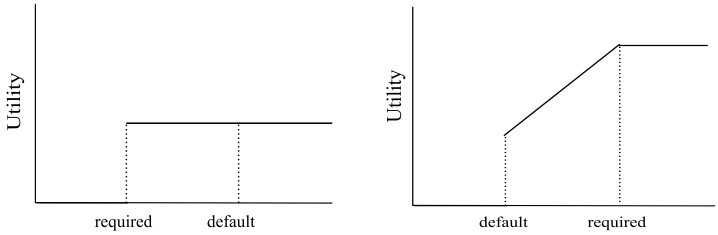
2.2 Obtaining a Solution Set

Now let us describe in detail how feasible and "good" proposals are calculated by LightSet agents.

An *allocation* is a n-tuple $(t_1, t_2, \ldots, t_n)$ where each $t_i \in \mathbf{N}$ $(1 \leq i \leq n)$ is the service time allocated to LightSet Agent i, and n is the number of LightSet Agents in the intersection. Let A be the set of all valid allocations, that is, the set of all allocations such that the entire resource (i.e. the cycle time) is used, each LightSet Agent does not have in common any portion of the resource and each LightSet Agent has at least the minimum service time. Let us assume that each LightSet following the conflict-directed mechanism has an utility function over allocations in A. Figure 3 shows the form of the proposed utility function.

The idea behind the utility function of figure 3(a) is that *the utility of a LightSet Agent does not increase once he has obtained his required service time*. A LightSet Agent will remain indifferent if it receives more resources than needed; the number of served cars will not increase. Moreover, the LightSet Agent will not accept an allocation that gives him less than his required service time because he has already sacrificed the default service time.

For cases like figure 3(b), the LightSet Agent will



(a) Case: $t_r^i \leq t_d$ (b) Case: $t_r^i > t_d$

Figure 3: Utility function that is implicitly defined on a LightSet. t_r^i is the required time of LightSet i and t_d is the default time which is common knowledge to every LightSet.

not accept an allocation that gives him less than the default service time (the unacceptable allocation will have an utility of 0) because the LightSet Agent can ensure himself the default service time when he cannot obtain his required service time. In order to simplify the analysis, we assume that utility grows linearly between the default and required service times.

Under the assumption of the utility function just described, we show in (Sánchez, 2007) that every allocation that the conflict-directed mechanism arrives to, maximizes the sum of the utilities that the allocation gives to every LightSet. Thus, the conflict-directed mechanism is emulating a negotiation mechanism that maximizes some social welfare criterion, in this case the sum of individual utilities of each of the agents. Let us define as U the set of allocations in set A that maximizes the summation of the utility of every LightSet Agent in the intersection, where the utility function is given as depicted in figure 3.

Let us now establish what could be considered a good solution. We start by defining a concept of distance over every allocation in set A with respect to a *perfect allocation*. Figure 4 shows a *resource allocation plot*. The X-axis represents the required service time, the Y-axis the allocated service time. The *perfect allocation* is represented by the straight line with slope one: the required time is equal to the allocated time. *Each* allocation corresponds to a set of n points at the plot For example, figure 4 shows a case of an intersection with four LightSet Agents and two different allocations.

We define the *vertical distance* from any point with coordinates (t_r^i, t_i) to the straight line $f(t_r^i)$ representing the perfect allocation as: $|f(t_r^i) - t_i| = |t_r^i - t_i|$ (the straight line has function: $f(t_r^i) = t_r^i$ for any t_r^i). Thus, the vertical distance is the amount of service time necessary for an arbitrary service time t_i allocated to LightSet i to become the required service time of LightSet i, either from above or below because of the absolute value involved.

We also show in (Sánchez, 2007) that whenever d is minimized under the restriction that the minimal

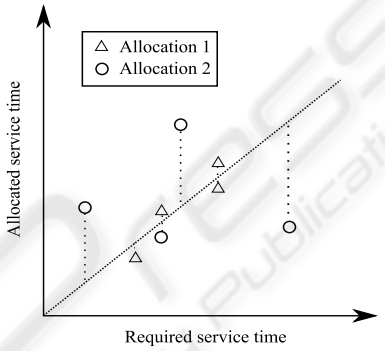


Figure 4: Example of a resource allocation plot. Two allocations are plotted against the perfect allocation. Vertical distances to the perfect allocation of each point of both allocations are shown.

allocation must be an element of set A, two heuristics "pop out" when going towards the minimal distance allocation: *Give bigger parts of service time to those LightSet agents whose required service time is bigger* and *Give the same quantity of service time to LightSet agents with the same required service time*. Under the assumption that the cycle time is much bigger than the number of LightSet Agents in the intersection, these heuristics are indeed implemented by the conflict-directed mechanism. We call G the set of allocations in set A which fulfill both heuristics defined earlier.

If the *solution set* of the conflict-directed mechanism is defined as the set of all and only those allocations that the mechanism arrives to, then, by the preceding discussion, the solution set *is contained* in $U \cap G$.

3 EXPERIMENTAL EVALUATION

For testing the mechanism, experiments were conducted using a multi-agent based simulator. The simulated world is a $n \times m$ grid of cells where cars inhabit one cell at a time. It can represent any number of traffic intersections with various lanes in every di-

Table 1: Summary of experimental results for cars' average waiting time in one intersection without arrival rate measurement.

	1°	2°
N	CD	T-14
N,W	CD	T-16
N,W,E	CD	T-12
N,W,E,S	CD,T	CD,T

rection (North, East, South and West). Time is measured by simulation steps. The entire simulation is synchronous and it is built using the synchronous engine of the MadKit Multiagent platform(Madkit,).

In the experimentation we compare the average waiting times achieved by the CD mechanism and the traditional mechanism. The CD mechanism was tested in one and two intersections with and without measurement of the arrival rate. The variables "number of sensors that count the number of incoming cars per lane" and "time each sensor waits to measure the arrival rate" also known as "time window" were selected for making the experiments. For each variable, we tested the mechanism with three different values corresponding to low, medium and high categories. Also, the CD mechanism was tested when some directions in the intersection are congested. Directions were congested incrementally: North, West, East and South in that order. In total, 76 experiments were executed.

For space reasons we only present some of the results for one intersection; similar results were obtained for two intersections. Results are summarized in tables 1 and 2. The tables show the mechanism that achieved the lowest level of car average waiting time on each experiment. The column on the left represents the congested directions. The columns marked as 1° and 2° represent the mechanism that achieved the first place and the second place for the same experiment. Symbols CD and T represent the CD and the Traditional mechanisms respectively. The numbers to the right of the symbols in the column 2° indicate how much that mechanism loses with respect to the mechanism in column 1°. When the mechanisms obtain similar results, the names are separated by commas in the same cell. Table 2 only shows the results for experimentation with one "time window" variable (7 simulation steps for arrival rate measurement) and two values for the number of sensors.

3.1 Discussion of Results

We can observe that in general the CD mechanism achieved better results than the traditional mechanism. The results are even better whenever it was allowed to see the true car arrival rate as it is the case in table 1. For the cases where the arrival rate is measured (table 2), the measurement process produces a constantly changing value in the car arrival rate. The case of four congested directions is a special one, as there is no unused service time to distribute if every lane at every direction is congested. Under the assumption that no direction is more important than other, then the best thing to do is to equally distribute the cycle time among the LightSet Agents. The results also show that the CD mechanism tends to improve as the number of sensors increases, as we would expect due to an improvement in the car arrival rate measurement. By the contrary, the "time window" variable did not show a clear pattern.

Table 2: Summary of experimental results for cars' average waiting time in one intersection with seven simulation steps for arrival rate measurement.

	1 sensor		6 sensors	
	1°	2°	1°	2°
N	CD	T-6	CD	T-8
N,W	CD	T-4	CD	T-8
N,W,E	CD	T-5	CD	T-5
N,W,E,S	CD,T	CD,T	CD,T	CD,T

4 RELATED WORK

In (Gershenson, 2005) three self-organizing mechanisms for traffic light control are proposed. The mechanisms are "aware" of changes in their environment, and therefore are able to adapt to new situations. Besides some differences in the modeling approach, they focus on self-organization that produces *automatic* coordination seen as an emergent behavior at different intersections. In (Penner et al., 2002) it is proposed a swarm-based simulation environment where cars communicate through a pheromone model in order to avoid collision. They propose a traffic light optimization system where the strategies followed by the traffic lights evolve in order to minimize the average waiting time of all cars in the traffic system. The main difference with our work is that they optimize in several intersections at once while this paper was focused on optimization in one intersection.

In (Dresner and Stone, 2004) and (Dresner and Stone, 2005) a multi-agent traffic management system is described. The system implements a reservation mechanism by which cars request and receive time slots from the intersection during which the vehicles may cross the intersection. The main differ-

ence with our paper is that they assume that they work with autonomous vehicles, thus they do not necessarily need to use traffic lights in an intersection. Also, they do not use waiting time as the metric of interest, but average delay and maximum delay. In (France and Ghorbani, 2003) a multi-agent system is presented for optimizing urban traffic not only in one intersection but in several. They use a hierarchical organization of agents, ranging from agents that control the light patterns in one intersection to agents that coordinate groups of intersections. The difference with this paper is that they model the cars as a quantity trying to minimize the traffic density from a global point of view. In our work, we are centered on optimization of *independent* intersections trying to minimize the car's average waiting time.

5 CONCLUSIONS

This paper presented a coordination mechanism for multi-agent systems followed by traffic lights in a simulated traffic intersection. The mechanism is based on a conflict resolution strategy as it identifies conflictive situations where the green time demanded by traffic lights in an intersection is greater than the cycle time of that intersection. The mechanism tries to solve the conflict distributing the cycle time among the traffic lights on the intersection. The mechanism has been tested in a multi-agent based simulator, and results show that it was able to reduce average car waiting time in cases involving one, two and three congested directions, at one and two independent intersections. Under some assumptions we described, the CD mechanism is emulating a negotiation process that maximizes the summation of utilities.

Several improvements could be suggested to the mechanism. Other methods, for example neural networks, could be proposed for improving the car arrival rate measurement process. Also, more experiments can be conducted increasing the number of independent intersections, introducing diagonal lanes for having more than four different directions, etc. Finally, coordination methods using the results provided by the mechanism could be proposed for controlling several intersections.

ACKNOWLEDGEMENTS

This work has been done while Jose Luis Aguirre was full time professor at the Tecnologico de Monterrey, and was supported by CAT145 Research Chair at the same institution.

REFERENCES

Balmer, M., Cetin, N., Nagel, K., and Raney, B. K. (2004). Towards truly agent-based traffic and mobility simulations. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, pages 60 – 67, New York, NY, USA.

Bazzan, A. L. C. (2005). A distributed approach for coordination of traffic signal agents. In *Autonomous Agents and Multi-Agent Systems*, volume 10, pages 131–164.

Dresner, K. M. and Stone, P. (2004). Multiagent traffic management: A reservation-based intersection control mechanism. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, pages 530 – 537, New York, NY, USA.

Dresner, K. M. and Stone, P. (2005). Multiagent traffic management: An improved intersection control mechanism. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*, pages 471 – 477, Utrecht, The Netherlands.

France, J. and Ghorbani, A. A. (2003). A multiagent system for optimizing urban traffic. In *Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology, IAT 2003*, pages 411 – 414.

Gershenson, C. (2005). Self-organizing traffic lights. *Complex systems*, 16:29–40.

Madkit. The MadKit project. Web site. http://www.madkit.org.

Penner, J., Hoar, R., and Jacob, C. (2002). Swarm-based traffic simulation with evolutionary traffic light adaptation. In *Proceedings of Applied Simulation and Modelling (AMS 2002)*.

Sánchez, J. H. D. (2007). Two multiagent traffic light coordination mechanisms for reducing average car waiting time in a traffic intersection. Master's thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey - Campus Monterrey.

van den Bosch, A. T., Menken, M. R., van Breukelen, M., and van Katwijk, R. T. (2003). A test bed for multiagent systems and road traffic management. In *Proceedings of the 15th Belgian-Netherlands Conference on Artificial Intelligence (BNAIC'03)*, pages 43 – 50, Nijmegen, The Netherlands.