Cooperative Behaviors for the Self-Regulation of Autonomous Vehicles in Space Sharing Conflicts

Mohamed Tlig, Olivier Buffet, Olivier Simonin

Cooperative Behaviors for the Self-Regulation of Autonomous Vehicles in Space Sharing Conflicts

Mohamed Tlig*†, Olivier Buffet*†, Olivier Simonin†*
Université de Lorraine† / INRIA*
Nancy, France
Email: firstname.lastname@loria.fr

***Abstract*—In real-world multi-agent systems, as in the context of the automatic transportation of goods, autonomous vehicles can face unexpected events like the failure of a vehicle, the presence of obstacles on the road, etc. Such events can generate first local congestions, and then, if they persist, global phenomena and complex traffic congestions (such as traffic jams). We want to manage space sharing conflicts at the local level, when they appear, to allow a quick (real-time) regulation, i.e., without requiring to re-plan the routes of all involved agents. Our approach relies on reactive coordination between vehicles using simple interactions between neighboring agents, using perceptions and little or no communication. We consider in particular a scenario where two queues of vehicles share a single lane, describing the model of the network as well as the agents, and proposing simple coordination rules that only involve the two vehicles at the front of each queue. We then conduct experiments that allow the analysis and the comparison of the proposed self-regulation rules.**

***Index Terms*—Multi-Agent Systems; Reactive Coordination; Space Conflict Resolution; Autonomous Vehicles; Traffic Regulation; Traffic Simulation**

I. Introduction

In real-world multi-agent systems, as in the context of the automatic transportation of goods, autonomous vehicles can face unexpected events like the failure of a vehicle, the presence of obstacles on the road, etc. Such events can generate first local congestions, and then, if they persist, global phenomena and complex traffic congestions (such as traffic jams). We aim at avoiding such undesirable emergent behaviors by exploring local rules for coordinating agents (vehicles). We want to manage conflicts at the local level, when they appear, to allow a quick (real-time) regulation, i.e., without requiring to re-plan the routes of all involved agents.

Re-planning [1] is not adapted to large multi-agent systems due to its combinatorial complexity. To avoid such a limitation, we are looking for reactive behaviors allowing to minimize delays and, if possible, to repair the plans.

Our approach relies on cooperative behaviors, based on reactive local coordination in multi-agent systems [2], [3]. Coordination is obtained from simple interactions between neighboring agents, using perceptions and little or no communication. Such assumptions allow to react to conflicts in real time. As examples of successful uses of local reactive coordination, we can mention [4] for the navigation of large sets of agents (flocking), [3] for multi-robot/flight avoidance, and [5], [6] for multi-robot navigation conflict solving.

Our work addresses the general problem of space sharing in multi autonomous vehicle/robot systems, such as the one envisaged for transporting goods (in seaports or other large platforms[1]). In such systems, vehicles receive plans, i.e., routes, to follow for transporting goods. These systems are highly sensitive to local delays/conflicts as these will impact on all the vehicles whose plans go past the local blocking. Then we consider, as a case study, a road in which a lane is suddenly blocked, e.g., by a vehicle breakdown, requiring that blocked vehicles use the other lane, initially dedicated to vehicles moving in the opposite direction. This problem generalizes the problem of sharing a common space among some agents to two infinite queues of agents.

For this purpose we investigate two approaches relying on simple coordination rules, which require only simple communications between the two vehicles at the front of the queues. We aim at ensuring the simultaneous freeing of both queues, while minimizing the delays of the vehicles. Then we conduct experiments to analyze and compare the proposed approaches.

The paper is organized as follows. Section II presents previous work related to this problem of space resource sharing among multiple agents.Section III describes a formalization of the problem and the multi-agent model, i.e., the definition of the possible actions and decision rules of the agents. Section IV proposes two decision rules that produce two different strategies. Then Section V details several experiments with deterministic and stochastic scenarios, showing the efficiency and limits of the strategies. Finally, we conclude with a discussion on these results and some promising research directions.

II. Related Work

There are two main approaches for modeling urban traffic:

Macroscopic models that consider traffic as a flow through a graph. They use analytical models based for example on fluid dynamics [7], [8]. These macroscopic models offer a high-level model, and thus do not describe individual behaviors.

Microscopic models are individual-based (or entity-oriented) models. They describe the movement of each vehicle, as well as their interactions [9]–[11]. As these models are very detailed, very complex to implement, they process a large quantity of data, which is the main restriction on their use for

[1] http://www.intrade-nwe.eu/

modeling a real network, e.g., of a city. In our case, since we want to propose local individual behaviors to solve problems in a portion of a road, we choose to use a detailed model, i.e., a microscopic model.

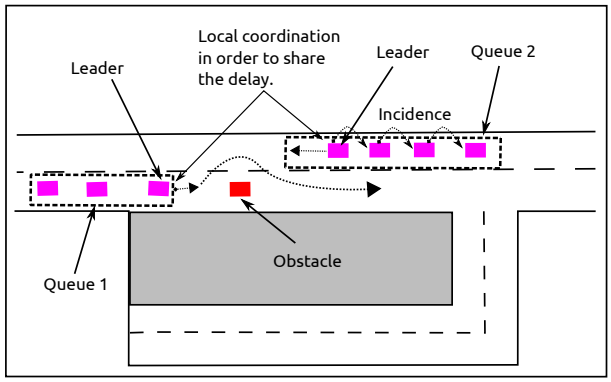

Fig. 1. Two flows of vehicles blocked by an obstacle

Our interest is focused on controlling autonomous vehicles which transport goods from a source to a destination. As shown in Figure 1, our problem is a space sharing problem. Let us assume that we have a two-lane road, the traffic being interrupted by an obstacle at $t = 0$ on one of the lanes (e.g., due to a vehicle breakdown). This results in a space sharing problem between two queues of vehicles, which is equivalent to managing a crossroads intersection, but without traffic lights.

This situation is traditionally studied in operations research and queueing theory. To our knowledge, there is no work proposing vehicle behaviors to deal with such conflicts, but various approaches have been proposed to model and analyze traffic flow interrupted by incidents. In 2002, Hidas proposed a microscopic traffic network simulator with a multi-agent system, and presented lane changing models (unforced, forced, and cooperative) to avoid accidents [12]. His results indicate that only forced and cooperative behaviors reproduce realistic flow-speed relationships in congested situations. Baykal-Gürsoy et al. in 2009 presented a queueing model to describe the traffic flow on a road link that is subject to a roadway incident [13]. For some cases, they present analytical results and compare them to simulation results.

A problem very similar to ours was treated by Tanner [14] in 1953. It is the only paper we know which is interested in the same setting. He defined a mathematical model to estimate delays that occur when two opposing flows (queues) of vehicles try to pass simultaneously through a single lane. All vehicles in this model have the same constant speed and their starting and stopping times are negligible. However, contrary to Tanner, our objective is not to estimate delays that occur in such conflict problems but to find an efficient approach to reduce delays.

III. Problem Formalization

In this work, we discretize space and time at an appropriate level to simplify the microscopic model. We use a discrete time step (1 second) and all vehicles have the same constant speed when moving. Space is thus discretized with the unit length l of displacement in $1s$.

A. Network Model

The network is modeled here by a set of discrete (directed) arcs of size $n \cdot l$. These arcs are connected together by nodes. Each flow of vehicles in the network follows a particular path, i.e., a sequence of arcs. The traffic is considered as a set of vehicle flows.

Our particular network is modeled by a set of arcs as shown on Fig. 2. Here, two flows pass through the network. The first one traverses the arcs A_1, A_2, A_3 and the second one B_1, B_2, B_3. On a particular road –composed by A_2 (for vehicles from source A) and B_2 (for vehicles from source B)– vehicles travel in both directions. This is the *conflict* edge, which must be shared by both flows.

We divide each arc into cells of size l which determine the position of agents as shown in Fig. 2. Each cell can be occupied only by one vehicle at a time.

Here we consider two flows of vehicles that will fill in the queues of the network in case of obstacle as shown in Fig. 1. The first vehicle in each waiting queue (waiting before the conflict edge) is referred to as its leader. Negotiations about crossing will take plIl a ajout qu'ils vont aider rparer les dommages et qu'ils vont fournir de nouveaux quipements lace between the two leaders.

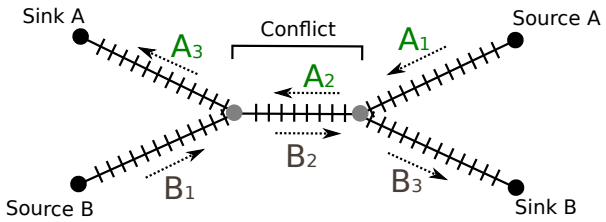

Fig. 2. Representation of the network at hand

B. Agent Model

The purpose of this section is to define the agents (the vehicles) and their interactions inside the network.

An agent takes sensory inputs from its environment and performs actions that affect it as outputs. We are interested in reactive agents, acting locally in real time. Each such agent uses only local perceptions coming from its own sensors.

In an agent's model, we distinguish the "action model" and the "decision rules" as shown in Fig. 3. The "action model" describes the actions which can be performed. Each action can be executed only under certain precise preconditions. After the execution of each action, an effect on the environment is expected. The main problem for an agent is to choose an action in order to best satisfy its objectives. The "decision rules" in our case are the reactive behaviors and coordination rules of the agents. They should, here, allow to (possibly) avoid or solve conflicts by triggering appropriate actions.

Each agent moving on the network has three internal variables: T_{goal}, the date beyond which the agent is considered to be late; Arc, which indicates in which arc the agent is; and Abs, its position on the current arc in the network, which is incremented as it progresses.

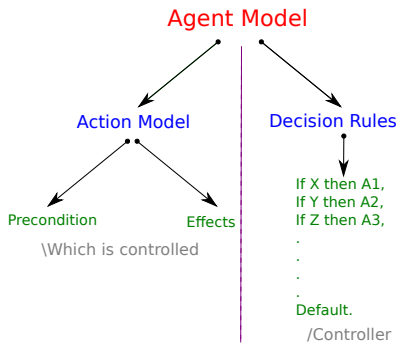

Fig. 3. Action-decision model

Action Formalism and Model: There is no shared representation formalism in the field of reactive multi-agent systems. In order to describe environment states and transformations, we choose a representation inspired from STRIPS (Stanford Research Institute Problem Solver) like Ferber in [2]. STRIPS was proposed by Fikes and Nilsson to address planning problems in Artificial Intelligence (AI) [15]. This is a good choice making a compromise between expressiveness –its ability to describe many problems– and simplicity –to ease the development of efficient algorithms.

In this formalism, a state of the environment is described as a set of clauses composed of literals. Their conjunction asserts the validity of this state. An operator, which characterizes actions, is composed of:

- a precondition, which is a set of predicates that should be true for the operator to be authorized,
- post-conditions, which are a set of predicates that will be true and a set of predicates that will be false after the execution of the operator.

Each operator is described under the following form (t and $t+1$ being the current and next time steps):

$$\begin{aligned}&\langle name : Action()..., \\ &\ pre\text{-}condition : A(t), B(t)..., \\ &\ post\text{-}condition : C(t+1), D(t+1)...\rangle.\end{aligned}$$

In our case, the action model relies on 3 operators, (which makes use of multi-valued variables): $DoNothing$, $Forward$, and $ChangeArc$.

$DoNothing$ consists in waiting for one time step.

$Forward$ describes the displacement within arcs:

$$\begin{aligned}\langle name : \quad & Forward(), \\ pre : \quad & \neg Last(Abs, Arc), Free(Abs+1, Arc), \\ post : \quad & Free_{t+1}(Abs_t, Arc_t), \\ & Abs_{t+1} = Abs_t + 1, \\ & \neg Free_{t+1}(Abs_t + 1, Arc_t)\rangle,\end{aligned}$$

where:

- $Free(Abs, Arc)$ is true iff the position Abs of the arc Arc is empty;
- $Last(Abs, Arc)$ is true iff Abs of the agent is the last position of the arc Arc.

In this action, the agent must verify that the next position of its arc is free before moving.

$ChangeArc$ describes how an agent moves from one arc to next:

$$\begin{aligned}\langle name : \quad & ChangeArc(), \\ pre : \quad & Last(Abs, Arc), Free(1, NextArc), \\ post : \quad & Free_{t+1}(Abs_t, Arc_t), \\ & Arc_{t+1} = NextArc_t, \\ & Abs_{t+1} = 1, \\ & \neg Free_{t+1}(1, NextArc_t)\rangle,\end{aligned}$$

where $NextArc$ indicates the following arc to the agent. Here, if the agent wants to move on an arc, it must verify that it is in the last position of its arc, and that the first position of the next arc is free.

C. Optimization Criteria

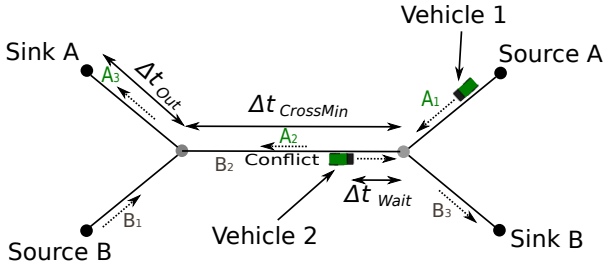

Fig. 4. A particular scenario and the durations that add up to estimate $Vehicle$ 1's traversal time

To estimate the delay upon arrival, we must calculate the time remaining for the vehicle to exit the network. Consider a generic scenario where we have two vehicles in the network, $Vehicle$ 1 and $Vehicle$ 2, as shown on Fig. 4. $Vehicle$ 1 wants to enter the conflict edge. However, it must first wait for $Vehicle$ 2 to pass, implying an initial waiting time. When $Vehicle$ 2 leaves the conflict edge, it enters its last arc in the network (B_3), and it is $Vehicle$ 1's turn to pass (on A_2). Finally, to exit the network, $Vehicle$ 1 must go through its last arc (A_3).

In this case, the date at which $Vehicle\ 1$ reaches its goal is computed as follows:

$$T_{estimated} = T_{real} + \Delta t_{Wait} + \Delta t_{CrossMin} + \Delta t_{Out}, \quad (1)$$

where T_{real} is the current date, Δt_{Wait} is the time required to free/empty the conflict edge of vehicles in the other direction, $\Delta t_{CrossMin}$ is the time required to cross the conflict edge, and Δt_{Out} is the time required for the vehicle to cross the last arc in the network and to leave it.

More generally, we call delay of a vehicle the time lost with respect to the original plan given by the system user. We define the delay D as

$$D = \max\left(0, T_{estimated} - T_{goal}\right). \quad (2)$$

Now, consider N vehicles $v_1, v_2, v_3, ..., v_N$ in the network and, for each vehicle v_i, its delay D_i. As the arrival of vehicles is stochastic, we have to optimize an expected criterion. We can express our objective to minimize the delay in various ways as, e.g., with the three following formulas:

$$f_{Sum}(\pi) = \min_{\pi} E\left[\sum_{i=1}^{N} \frac{D_i}{N}\right], \quad (3)$$

$$f_{Max}(\pi) = \min_{\pi} E\left[\max_{i\in\{1..N\}} (D_i)\right], \quad (4)$$

$$f_{Sum^2}(\pi) = \min_{\pi} E\left[\sqrt{\sum_{i=1}^{N} \frac{D_i^2}{N}}\right]. \quad (5)$$

The first formula –a linear criterion– minimizes the average delay over all vehicles, but some vehicles may incur very long delays. The second formula seeks to minimize the worst delay over all vehicles, but may lead to a very bad average delay. That is why we introduce the third formula –a quadratic form–, which is a compromise between equations (3) and (4) using the Root Mean Square of the delay. In all these cases we attempt to have a global behavior that allows sharing delay between agents.

IV. Proposed Coordination Behaviors

We propose two strategies relying on reactive coordination rules executed by the vehicles at the front of the waiting queues.

A. Alternating

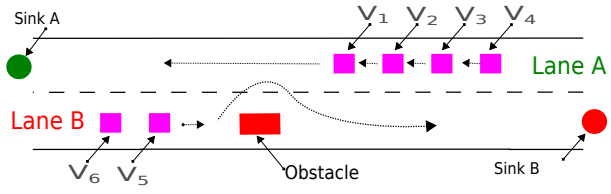

Fig. 5. First approach : Alternating vehicles

The first behavior is inspired from the civic behavior of drivers when they have to share a one lane road. In case of conflict, vehicles pass alternately, i.e., one at a time, from each side of the $conflict$ edge, as in Fig. 5, with four cars (V_1, V_2, V_3 and V_4) from lane A and two cars (V_5 and V_6) from lane B. The resulting passing order is (from left to right) $V_5\ V_1\ V_6\ V_2\ V_3\ V_4$ or $V_1\ V_5\ V_2\ V_6\ V_3\ V_4$, depending on who goes first between V_1 and V_5.

Alternating is a simple process that does not require high level communications since the order is automatic (regardless of the delays). Only the perception of vehicles on the conflict edge and at its entrance is required. Nevertheless, we must treat the particular case of the simultaneous arrival of a vehicle on both sides of the conflict edge when this edge does not contain any vehicle. In this situation, each vehicle transmits a release signal after a (very short) random delay. As soon as a vehicle receives such a signal, and if it does not emit at the same time, it sets out on the road. If both transmit simultaneously, they restart this process.

Algorithm 1 gives the essential part of an agent's behavior by focusing on the rules for changing arc. It considers only the decision to be taken by an agent located at the entrance of the conflict edge. The first two tests correspond to (i) if the edge is already occupied by a vehicle in the opposite direction, then it waits, or (ii) if there is no agent on the opposite entrance, then it can move forward. The third test is the nominal case. It concerns a vehicle that moves forward after a vehicle coming from the opposite direction has left the conflict edge. The special case remains of two agents on the two entrances of the conflict edge, when there was no agent traversing it (line 8). In this case the perception of the release signal can solve the situation, otherwise each one emits the signal after a random time. The last case (line 14) is when the agent has to wait because the conflict edge is already occupied and an agent is waiting at the opposite entrance.

Algorithm 1: Passage rules for alternating

```
1  input : agent at an entrance of the conflict edge
2  if there is currently an agent on the conflict edge
     in opposite dir then
3      DoNothing()
4  else if no agent on the opposite entry then
5      ChangeArc()
6  else if there was previously an agent on the conflict edge
     in opposite dir then
7      ChangeArc()
8  else if there was no previous agent on the conflict edge
     then
9      break-tie()
10     if winner then
11         ChangeArc()
12     else
13         DoNothing()
14 else
15     DoNothing()
```

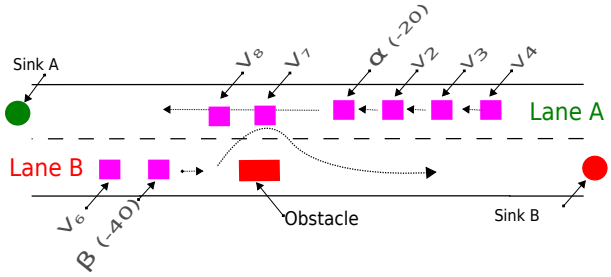

Fig. 6. Second approach : Highest delay first strategy

B. Local Greedy Optimization (LGO)

The second behavior that we propose tries to optimize the transition by promoting vehicles that are more delayed than others. Delay comparisons are done using the communication between the two agents which want to cross the conflict edge simultaneously. Let us denote α and β the leading vehicles of the queues A and B respectively (assumed nonempty). For decisions to be local, as in the first approach, only the two leaders of the waiting queues can communicate together.

For example, consider Fig. 6, where we noted in parentheses the delay of the two leaders assuming each of them goes first (e.g., vehicle α has 20 seconds of delay). Intuitively, they will go in this order: β then α. But if, as in Fig. 6, there are vehicles in the middle like V_7 and V_8, or the ordering of delays is complicated, choosing the order of passage is not trivial. If α passes first, β will wait an extra time $\epsilon_1 = 40s + \alpha\ crossing\ time$. Else if β passes first, α will wait an extra time $\epsilon_2 = 20s + V_7\ crossing\ time + \beta\ crossing\ time$.

To make this decision, we use the optimization criteria presented in Sec. III-C. Having chosen to consider only these two vehicles, we restrict the evaluation of the selected criterion to them, and only have to compare two orderings: (1) α before β ($\alpha \rightarrow \beta$), and (2) β before α ($\beta \rightarrow \alpha$).

1) Each vehicle first calculates its two possible delays: $D^v_{\alpha\rightarrow\beta}$ and $D^v_{\beta\rightarrow\alpha}$, where v is α or β, then transmits them to the other vehicle.
2) Each agent compares, based on its own estimates and those received, the two possible passing orders using the optimization criterion at hand. For example, if the criterion used is Formula 4, the passing order will be α then β if $f^{\alpha\rightarrow\beta}_{Max} = (D^{\beta}_{\alpha\rightarrow\beta} + D^{\alpha}_{\alpha\rightarrow\beta})$ is greater than $f^{\beta\rightarrow\alpha}_{Max} = (D^{\beta}_{\beta\rightarrow\alpha} + D^{\alpha}_{\beta\rightarrow\alpha})$, else β then α in the opposite case.

Algorithm 2 gives the essential behavior of vehicle β where $f^{\beta\rightarrow\alpha}_*$ (respectively $f^{\alpha\rightarrow\beta}_*$) is the value of one of the 3 criteria if β passes before α (resp. if α passes before β).

Algorithm 2: Passage rules for LGO of the vehicle β

```
if (agent waiting on the other side) then
    Send/receive delays
    if (f*^{β→α} > f*^{α→β}) then
        if
        (agent on the conflict edge in opposite direction)
        then
            wait for release of conflict edge
        ChangeArc()
    else DoNothing()
else
    if (agent on the conflict edge in opposite direction)
    then
        DoNothing()
    else ChangeArc()
```

V. Experimental Results

A. Simulation

We developed a prototype simulator on the JADE[2] platform (Java Agent Development Framework), which offers a Java middle-ware to develop agent-based applications. In our work, the actions and decisions are the same for all agents.

We reproduce the network as shown in Figures 2 and 4:

- the speed of each vehicle is 10 meters per second (36km/h), thus $l = 10$;
- the length of each arc is 300 meters $(30 \cdot l)$;
- at each entrance of the network, we have installed a source that generates vehicles.

Each source injects vehicles following a Bernoulli process with a parameter λ such that:

$$\lambda = \frac{1}{T}, \tag{6}$$

where T is the average time, in seconds, between two consecutive vehicles. This process is equivalent to flipping a coin at each time step.

In all our simulations, we verified that we do not meet the pathological case where a single queue passes, at the expense of the other queue (which is blocked). In the remainder of this paper, we will call Alt the Alternating strategy, and Sum, Max and Sum^2 three variants of the LGO strategy corresponding to the criteria presented in Formulae 3, 4, and 5.

B. Release of the Two Lanes

The particularity of this scenario is that we start with the Alternating strategy, then, after 50 vehicles have been injected in the network, either we continue with the same strategy ($Alt - Alt$) or we choose the second strategy (LGO) with one of the 3 criteria. After the injection of 100 vehicles, we stop injections and wait for the network to empty. We used high injection frequencies—thus, dense traffic—with parameter $T = 10$s for the Bernoulli process of each queue.

Fig. 7 shows the simulation results of the Alternating strategy and the 3 variants of the LGO strategy. The curves plotted in Fig. 7 are averages over 100 simulations. The X axis represents the time in seconds, and the Y axis represents the number of vehicles in the network at time t.

We notice that the $Alt - Alt$ strategy is not good and takes a lot of time –on average, 600 seconds– before releasing the

[2]http://JADE.tilab.com

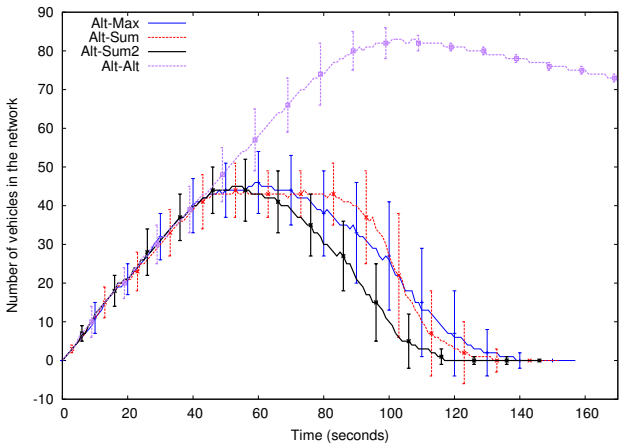

Fig. 7. Observation of the release of 100 vehicles

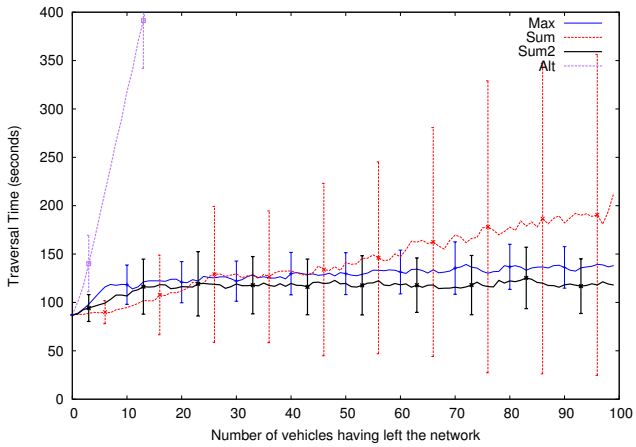

Fig. 8. Comparison of averages and standard deviations of traversal time (T=10–10)

network. We also note that, when switching to any version of the LGO strategy, the number of vehicles for strategies $Alt - Sum^2$, $Alt - Max$, and $Alt - Sum$ progressively decreases until both paths are finally empty. The fastest evacuation is given by the curve $Alt - Sum^2$. We observe the existence of two stages for the $Alt - Sum$ criterion. Upon the outbreak of the strategy, the curve makes a plateau, followed by a steeper slope than on any other curve. This is due to the Sum criterion avoiding to switch queues (as we will see in the next sub-section). When there are injections, while one of the queues is running, the other saturates (generations are then forbidden, and therefore the Bernoulli process is not respected). This saturation is a way to limit the increase of the number of vehicles (hence the plateau), and further delays the moment when the total of 100 injected vehicles is reached. Once the injections have stopped, $Alt-Sum$ releases its queues faster by avoiding the wasted time associated to switching queues. We see that $Alt - Max$ and $Alt - Sum^2$ have a slightly higher maximum number of vehicles than $Alt - Sum$, but no such plateau.

C. Regulation of a Continuous Traffic

In the second scenario, we do not stop the injections of vehicles as in the previous simulations, but record the traversal time of the first 100 vehicles leaving the network.

Fig. 8 gives the average traversal time of each strategy for injections with $T = 10s$. The Y axis gives the traversal time in seconds, and the X axis gives the number of vehicles having left the network. The plotted curves are averages over 100 simulations.

The Alt strategy is the worst one again. The Sum criterion is significantly worse than Sum^2 and Max. It does not favor any lane switches (as we will see below) and accumulates vehicles on one of the sides. We observe that the best criteria are Sum^2 and Max with the lowest averages, noting that the minimum traversal time is 87 seconds. Max, besides caring about the worst delay, reduces the standard deviation, while Sum^2 works more on reducing the average of the traversal time between vehicles.

TABLE I
SUMMARY OF AVERAGES AND STANDARD DEVIATIONS

$Average$	Alt	Sum	Max	Sum^2
$10-10$	1260± 86	142±94	127±22	115±28
$30-30$	782±173	104±28	105±19	102±19

Table I gives a summary of the measured traversal times with standard deviations for each strategy and for injections of vehicle flow $T = 10$s and $T = 30$s. With less frequent injections we find that Sum^2 is the best criterion and that the 3 variants of the LGO strategy have close averages and standard deviations.

To better understand the LGO strategy, Fig. 9 presents the result of a simulation chosen randomly for each criterion with injections on average every $T = 10$s. Each curve represents the sequence of 100 vehicles in their output order, with the date of injections on the X axis, and the traversal time on the Y axis.

The first thing that questions us is that the Sum criterion takes a lot of time before switching queues, unlike Max that switches very often. According to the figures, the Sum^2 criterion appears as a compromise between the two others. Switching queue often wastes a lot of time, but not switching queues leads to accumulating delays of waiting vehicles. Overall, all the measurements show that an approach focused on local coordination rules proves to be efficient to regulate traffic around the conflicts generated by the space sharing problem. We demonstrate that a multi-agent approach, based on the exchange of information between the top vehicles from each queue, allows to implement an efficient regulation resulting from global delay optimization criteria.

VI. DISCUSSION

The problem treated in this paper is to solve space sharing conflicts in a multi-agent system. Only the two vehicles at the front of each waiting queue communicate together in order to know which one goes first. The advantage of our approach is

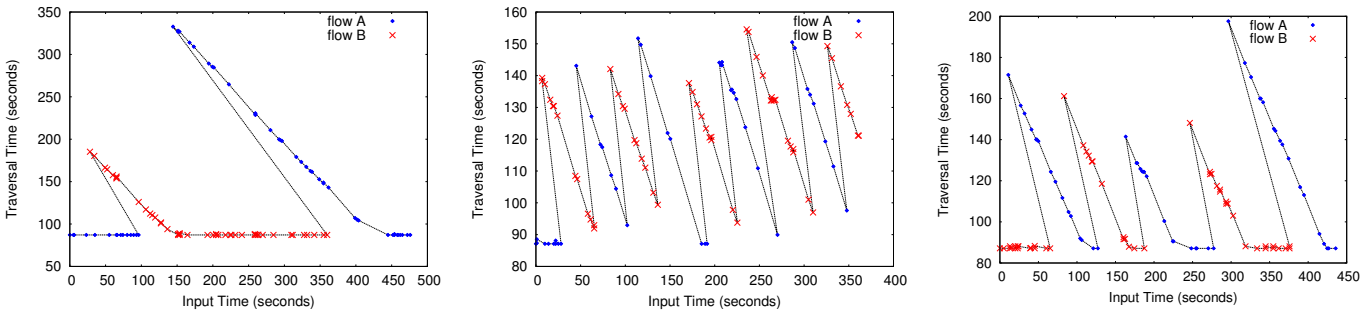

Fig. 9. Observation of one simulation of each criterion, Input time and Traversal time of vehicles (T=10–10)

the complete decentralization of the model. Especially, there is no centralized mechanism that manages the intersection (e.g., to receive the delays, to organize an auction...) which is different from the multi-agent approaches to control intersections [16]. Actually, if we consider an exhaustive approach (centralized), and if n and m are the number of vehicles in each queue, the number of vehicle orderings to consider – and thus the algorithm complexity– is the number of ways to interleave the vehicles from both queues (without changing the order within each queue)

$$\binom{n+m}{n} = \frac{(n+m)!}{n!\ m!},$$

which have to be considered at each time step t in order to know the best crossing order. The worst case for a fixed number of vehicles is when $m = n$, whose asymptotic behavior can be derived from Stirling's approximation as:

$$\binom{2n}{n} \sim \frac{4^n}{\sqrt{\pi n}} \text{ as } n \to \infty.$$

The complexity of our approach is significantly lower. It consists in the number of messages sent, i.e., at most two messages in each negotiation.

VII. Conclusion

In this article we addressed the resolution of space sharing conflicts between queues of vehicles, or more generally between mobile agents (e.g., robots). For this, we explored multi-agent approaches based on reactive coordination behaviors. We first proposed an approach using only local perceptions (alternating), and then one integrating communications between vehicles at the top of the queues. The experimental study has shown the ability to regulate conflicts (congestions) of these behaviors, generated in different traffic scenarios. Congestion phenomena, which are undesirable emergent phenomena, are treated here locally, thus independently of any external planning system, and in real time. The introduction of simple communications of delays significantly improves on the Alternating strategy commonly used by drivers.

We plan to continue this study by generalizing the approaches to any number of queues, but also by proposing to take into account delays of more vehicles present in the queues to further improve traffic management (searching how many vehicles to consider so as to best trade off between complexity and quality).

Acknowledgement

This work was partially supported by the InTraDE European project (http://www.intrade-nwe.eu).

References

[1] B. Nebel and J. Koehler, "Plan reuse versus plan generation: A theoretical and empirical analysis," *Artificial Intelligence*, vol. 76, pp. 427–454, 1995.

[2] J. Ferber, *Multi-Agent Systems. An introduction to Distributed Artificial Intelligence.* John Wiley and Sons Inc., New York, 1999.

[3] K. Zeghal, "A comparison of different approaches based on force fields for coordination among multiple mobiles," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1998.

[4] C. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *Computer Graphics*, vol. 21, no. 4, pp. 25–24, 1987.

[5] O. Simonin and J. Ferber, "Modeling self satisfaction and altruism to handle action selection and reactive cooperation," in *6th Int. Conf. on the Simulation of Adaptative Behavior (FROM ANIMALS TO ANIMATS 6).* in proceedings Supplement SAB 2000, 2000.

[6] P. Lucidarme, O. Simonin, and A. Liegeois, "Implementation and evaluation of a satisfaction/altruism based architecture for multi-robot systems," in *Proc. of ICRA*, 2002.

[7] M. J. Lighthill and G. B. Whitham, "On kinematic waves. 2. a theory of traffic flow on long crowded roads," *Proc. of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 229, no. 1178, pp. 317–345, 1955.

[8] B. Greenshields, "A study of traffic capacity," in *Proc. of the Highway Research Board*, vol. 14, 1935, pp. 448–477.

[9] P. Simon and K. Nagel, "Simplified cellular automaton model for city traffic," *Phys. Rev. E*, vol. 58, no. 2, pp. 1286–1295, Aug 1998.

[10] I. Kosonen and M. Pursula, "A simulation tool for traffic signal control planning," in *3rd Int. Conf. on Road Traffic Control.*, 1990.

[11] D. Meignan, O. Simonin, and A. Koukam, "Simulation and evaluation of urban bus-networks using a multiagent approach," *International Journal Simulation Modelling Practice and Theory*, vol. 15, no. 6, pp. 659–671, 2007.

[12] P. Hidas, "Modelling lane changing and merging in microscopic traffic simulation," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 5-6, pp. 351–371, 2002.

[13] M. Baykal-Gürsoy, W. Xiao, and K. Ozbay, "Modeling traffic flow interrupted by incidents," *European Journal of Operational Research*, vol. 195, no. 1, pp. 127–138, 2009.

[14] J. C. Tanner, "A problem of interference between two queues," *Biometrika*, vol. 40, no. 1/2, pp. 58–69, 1953.

[15] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[16] K. Dresner and P. Stone, "Multiagent traffic management: An improved intersection control mechanism," in *Proc. of AAMAS*, 2005.