Anticipation based on constraint processing in a multi-agent context

A Version of this paper has been published in Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS), 17, pp. 339-361, 2008.

A. Doniec[1], R. Mandiau[2], S. Piechowiak[2], S. Espié[3]

April 27, 2009

Abstract

Anticipation is a general concept used and applied in various domains. Many studies in the field of artificial intelligence have investigated the capacity for anticipation. In this article, we focus on the use of anticipation in multi-agent coordination, particularly preventive anticipation which consists of anticipating undesirable future situations in order to avoid them. We propose to use constraint processing to formalize preventive anticipation in the context of multi-agent coordination. The resulting algorithm allows any action that may induce an undesirable future state to be detected upstream of any multi-agent coordination process. Our proposed method is instantiated in a road traffic simulation tool. For the specific question of simulating traffic at road junctions, our results show that taking anticipation into account allows globally realistic behaviors to be reproduced without provoking gridlock between the simulated vehicles.

Keywords: coordination, anticipation, constraints network, gridlock

1 Introduction

Coordination is one of the possible interactions that can occur in a multi-agent system. Such interaction becomes necessary when agent activities are interdependent [24] and is involved in all local processes that deal with these interdependencies in order to reach a global state or achieve a goal.

Many definitions of coordination have been introduced in the literature [18, 28, 45], each for a particular context and highlighting a specific point of view.

- Coordination can be defined as a series of meta actions, which allow agents to interact with one another. Thus, for Malone, coordination is the set of all the extra activities that must be done in a multi-agent system in order to have interaction between agents [28].
- Coordination has also been compared to a search process performed in distributed problem-solving [15, 50]. In this specific context, coordination can refer to the decomposition into sub-problems, the resolution of sub-problems, the mechanisms for transmitting intermediate results, or the diffusion of the solutions to all sub-problems, to name but a few examples.
- Coordination is also a way to solve different types of inter-agent conflicts, including resource conflicts (between agents who must share a common resource at a same time) or conflicts of interest (between agents who have divergent goals, making their actions, in extreme cases, antagonistic).
- Coordination has also been defined as *"the process by which an agent reasons about his/her local actions and the (anticipated) actions of others to try and ensure the community acts in a coherent manner"* [20]. Jennings points out that this reasoning process should not only take the current states of the system into account, but also its future states.

[1]cole des Mines de Douai, Dept IA, 941 rue Charles Bourseul, BP 838, 59508 Douai Cedex, France
email: adoniec@ensm-douai.fr
[2]LAMIH UMR CNRS 8530, University of Valenciennes et du Hainaut-Cambrésis, 59313 Valenciennes Cedex 9, France
email: {Rene.Mandiau, Sylvain.Piechowiak}@univ-valenciennes.fr
[3]INRETS, 58 Bd Lefebvre, 75732 Paris Cedex 15, France
email: espie@inrets.fr

This paper focuses on the use of anticipation in the context of multi-agent coordination. The next section presents some general information about anticipation and goes on to present a variety of studies dealing both with anticipation generally and with anticipation in a multi-agent context specifically. The third section introduces our model of preventive anticipation, which is based on constraint processing. This model provides a generic algorithm to perform preventive anticipation tasks in the context of multi-agent coordination. The last section provides a concrete example. Our model is instantiated in a road traffic simulation tool that works to simulate traffic at a road junction.

2 Anticipation in a multi-agent system

2.1 Anticipation: some generalities

The common-sense definition of anticipation usually refers to a specific cognitive capacity that allows people to predict, represent and reason about future states, based on the current situation.

The word "anticipation" has many definitions due to its use in different contexts in a wide variety of domains: for example, biology, epistemology, psychology, robotics, and computer science. Riegler has introduced several types of anticipation [35]:

- *inborn anticipation*, rooted in phylogenetic schemas (e.g., schemas resulting from species evolution);
- *emotional anticipation*, guided by individual instinct;
- *intelligent anticipation*, based on a cognitive capacity to deal with future events.

In artificial systems, anticipation is usually considered to be an intentional and reasoned act [16]. This tends to eliminate inborn and emotional anticipation from the context of multi-agent systems.

Rosen defined anticipation in artificial systems [37]: *"An anticipatory system is a system containing a predictive model of itself and/or of its environment that allows it to change current state at an instant in accord with the model predictions pertaining to a later instant"*. This definition describes anticipation as the particular capacity of a system to adapt, or fine-tune, its current state in terms of a vision of the future. Rosen also distinguished between anticipation and prediction, opining that anticipation implies more complex mechanisms than the simple ability to predict the future. Rosen's approach is anchored in the construction of a mental representation. The author postulated that anticipation results from a modeling relation that permits a switch from the real system to its predictive model.

Given the variety in the definitions of anticipation, formalizing this notion in a mathematical and/or computer model is a real challenge. In the following sub-section, we present some relevant works which have attempted to formalize anticipation.

2.2 Different levels of anticipation

In artificial intelligence, many levels of anticipation have been defined. Butz *and al* [6] have identified four classes of anticipation: implicit anticipation, payoff anticipation, sensory anticipation, and explicit anticipation. According to Rosen's previously presented definition, "implicit anticipation" is not a type of anticipation and the three last classes seem to characterize the notion of anticipation.

2.2.1 Implicit anticipation

Implicit anticipation refers to pre-programmed behaviors that allow future goals to be achieved. No predictions for the future, which could modify the current behavior, are made. Picard and Gleizes provide an example of implicit anticipation, which they call *reactive anticipation* [33]. They apply a set of rules specifying accurate behaviors for particular situations in a context of cooperation between autonomous robots; their *reactive anticipation* allows deadlock situations to be avoided.

2.2.2 Payoff anticipation

Payoff anticipation is found in systems whose decision-making process is based on predicting possible payoffs. Such systems try to evaluate the benefits expected to be accrued from each possible action, using a maximization function as a decision-making criterion. Any coordination mechanism based on maximizing payoffs can be included in this class of anticipation [8, 38].

Payoff anticipation is often employed in direct reinforcement learning. Marshal *and al* used anticipation to generate a reinforcement signal during robot learning and training procedures [30]. In their approach, errors are predicted and error

reduction techniques are learned simultaneously. Other authors (e.g. [5]) have suggested that anticipation speeds up the learning process for some types of robots.

The ACS (Anticipatory Classifier System) model applies another form of payoff anticipation. A classic classifier system is a set of rules defined as a 3-uple *(condition,action,strength)*. To this 3-uple, Stolzmann [42] added a fourth element: *effect*. This new component describes the expected changes that will be caused by applying the rule whereas *strength* combines both the payoff of the action and the quality of the anticipation. The more the classifier's expected effects are truly observed, the higher the quality of the anticipation.

2.2.3 Sensory anticipation

With sensory anticipation, the predictions have a real influence on behavior because they can modify the way the system perceives its environment. The decision-making is guided only by the current system state and the former actions of the system. Castelfranchi *and al* [7] have recently presented an architecture for cautious agents that has some anticipatory features. These agents tend to modify perceptions and behavior in relation to occurring events. The authors distinguish between positive and negative events, which are used to build transitions between four behaviors: Caution, Excitement, Curiosity, and Boredom [34].

2.2.4 Explicit anticipation

Explicit anticipation is found in systems that build an explicit representation (partial or total) of the future states. In this type of anticipation, the analysis of the predictions influences perceptions and decision-making simultaneously. For instance, Laird [23] provides a detailled explanation of how to add anticipation to a bot[1] in a video game environment. His approach is similar to those used in expert systems with prediction abilities. The bot tries to predict the behavior of players by applying its own knowledge and strategies to its observations of the environment. A set of production rules allows the bot to forecast the future until it obtains an interesting prediction. The reasoning is adapted from the forward chaining principle.

Explicit anticipation seems to us to be the most interesting class of anticipation for multi-agent coordination since it has a direct influence on decision-making. In the following sub-section, we focus on an architecture for anticipatory agents proposed by Davidsson [9], which clearly falls into this last class.

2.3 Architecture for anticipatory agents

Davidsson has proposed an architecture for implementing anticipatory agents [9], which uses a special kind of explicit anticipation, called preventive anticipation.

Preventive anticipation was first described by Rosen in 1974 [36]. Consider a system S and its model M. To make predictions, M is simulated faster than S. But, how should these predictions be used to modify the behavior of S? In the general case, answering this question is quite difficult. In the preventive case, Rosen proposes partitioning *"the state space of S (and hence of M) [...] into regions corresponding to desirable and undesirable states. As long as the trajectory in M remains in a desirable region, no action is taken by M"*. In other words, preventive anticipation is viewed as the process of trying to keep the system in a desirable state.

Davidsson's architecture for preventive anticipation has two layers: an anticipatory layer and a reactive layer. These two layers are run in parallel as two concurrent processes. The anticipatory layer uses a model of the environment as well as an "anticipator", which allow the environment to be simulated rapidly and thus generate predictions. When the anticipator detects an undesirable state, the anticipatory layer acts on the reactive layer to modify the agent behavior.

The preventive anticipation used by Davidsson is a simple kind of anticipation, which is well adapted to multi-agent coordination. For this reason, we use it in our model of anticipation.

2.4 Issues

To summarize, the concept of anticipation in a multi-agent context involves a two-phase reasoning process performed by an agent. In the first phase, the agent predicts the future state of the multi-agent system, and in the second, this agent analyzes these predictions in order to change its current behavior.

Few of the studies previously presented have tried to formalize anticipation. Only Davidsson's work has tried to introduce a framework for anticipatory agents. But some aspects remain unclear. For example, how can desirable and undesirable states be described?, and how can they be brought about using predictions? In this paper, we link anticipation

[1] a virtual opponent in first-person shooter (FPS) computer games.

and coordination by proposing the use of preventive anticipation prior to execute any coordination mechanism in order to eliminate non-relevant actions. Our formalization of anticipation is simple, combining prediction generation and prediction analysis in a single model. It uses a constraint processing approach that considers undesirable states as inconsistencies or particular domains in a constraint network. In the next section, we present our model in detail.

3 Modelling preventive anticipation

The use of a constraint processing approach to model preventive anticipation is justified both by the simplicity of this formalism for describing inter-agent relationships, and by the abundance of algorithms in the literature for solving, simplifying or reducing constraint networks [10, 22, 46]. Our model assumes that a multi-agent system is an ordinary causal system, which presupposes that all the system states can be explained as the consequences of actions executed by system agents or the consequences of past states.

3.1 Effects of action

To define the notion of effect in our chosen context, we consider the following conditions. Given a multi-agent system made up of a set of agents $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ defined in an environment $\mathcal{E}$, $I_{\mathcal{E}}^{a_i}$ refers to the set of all information perceived by an agent $a_i \in \mathcal{A}$. $I_{\mathcal{E}}^{a_i}$ includes:

- a subset of $\mathcal{A}$: $I_{\mathcal{A}}^{a_i}$ that refers to the agents perceived in the environment, and
- a set $I_{\mathcal{P}}^{a_i} = \{p_{a_1}, p_{a_2}, ..., p_{a_j}, ...\}$ where each p_{a_j} is a set of properties describing each $a_j \in I_{\mathcal{A}}^{a_i}$.

Let us consider $\mathcal{R} = \{r(a_i, a_j)|(a_i, a_j) \in \mathcal{A}^2\}$ as a set of binary relationships existing between the agents in $\mathcal{A}$. The set of knowledge about R for an agent a_m can be defined as $\mathcal{C}_{a_m}^{+} \cup \mathcal{C}_{a_m}^{*}$; where $\mathcal{C}_{a_m}^{+} = \{r(a_x, a_y)|(x = m) \vee (y = m)\}$ and $\mathcal{C}_{a_m}^{*} = \{r^*(a_x, a_y)|x \in \mathcal{A} - \{a_m\} \wedge y \in \mathcal{A} - \{a_m\}\}$ with $r^*(a_x, a_y)$ refering to the relationships that can be deduced or calculated by a_m.

In other words, we suppose that an agent has knowledge about all the relationships in which it is involved and is able to deduce certain other relationships when there is enough information to do so. Thus, consequences of actions can be expressed as an addition or a deletion of a relationship between agents. Based on this assumption, we define *direct effects* as the consequences of an action performed by an agent a_m for the agents of $\{a_n \in \mathcal{A}|\exists r(a_n, a_m) \in \mathcal{C}_{a_m}^{+}\}$, and *indirect effects* as the consequences of an action performed by a_m for the agents of the set $\{a_n \in \mathcal{A}|\exists r(a_n, a_y) \in \mathcal{C}_{a_m}^{*}\}$.

In order to accurately predict these effects and to test whether they imply undesirable states, an agent needs a mental representation of the environment, including the known relationships between the agents and the states of these agents. Describing the mental representation of an anticipatory agent as a constraint network seems to us to be an original and relevant approach.

3.2 Mental representation as a constraint network

A constraint network is a set of variables with associated domains and a set of constraints. *"Variables are object or item that can take a variety of values. The set of possible values for a given variable is called domain. [...] Constraints are rules that impose a limitation on the values that a variable may be assigned"* [10]. We use this paradigm to introduce the mental representation of an anticipatory agent.

The mental representation M_{a_i} of an agent a_i consists of:

- a set of agents: $A = \{a_1, a_2, ..., a_n\} \subset \mathcal{A}$,
- a set $R = \{r_{a_i}(a_k, a_l)/a_k, a_l \in A\} \subset \mathcal{R}$ where each $r_{a_i}(a_k, a_l)$ is a binary relationship (perceived by a_i) between two agents a_k and $a_l \in A$, and
- a set $D = \{dom(a_1), dom(a_2), ..., dom(a_n)\}$ where each $dom(a_j)$ expresses the state of the agent a_j.

¿From such a representation, each agent is able to construct a simplified model of what it perceives in its environment. For example, temporal or spatial domains can be used to express the states of the agents perceived. The relationships used in this mental representation can be mapped as binary constraints, which can be used to restrict the different domains. The undesirable states are either specific domains, or inconsistencies. In constraint networks, an inconsistency implies the impossibility of assigning a value to each variable. Since effects of an action are considered to be additions to or deletions of the relationships in R, undesirable states can be inferred by using constraint processing techniques.

3.3 Inference of undesirable states

As mentioned above, a "direct effect" is considered to be a modification of the set R by adding or deleting one (or many) relationship(s). An *indirect effect* is the propagation of a *direct effect* in a given mental representation M. Because we use the constraint programming paradigm, this propagation is based on consistency. Since the relationships between agents are assumed to be binary, we can use arc-consistency, which, in constraint programming, ensures that for all values of $dom(a_x)$, there is a value of $dom(a_y)$ such that the constraint between a_x and a_y is satisfied. To infer undesirable states, classic algorithms like AC-3 [26] can be used and adapted for the specific application.

To infer the undesirable states, anticipatory agents must: (i) construct a mental representation of their perceived environment, (ii) compute the direct effects of the actions that must be anticipated, (iii) add these direct effects to the mental representation, and (iv) propagate these direct effects to obtain indirect effects.

3.4 Algorithm

Based on the ideas presented above, we developed a generic algorithm for preventive anticipation. To use the algorithm, the anticipatory agents must construct a mental representation and use this representation to anticipate and prevent undesirable situations [2]. In fact, the agents draw on the information about their environment to construct this representation. Three types of information are relevant: the presence of other agents in the environment, the states of these agents, and the relationships between them.

The method for determining the states of the agents present in the environment and relationships between them depends on the application. For example, in the case of situated agents, simply knowing the agent positions and directions can be sufficient to establish some relationships (e.g., obstructing, prey-predator). In other cases, it is sometimes useful to know more about the agents perceived: their goals and team or coalition memberships, for example.

To keep the algorithm generic, we suppose that the environmental information is analyzed by an external function, called *relationsOf*. This function exploits 3 parameters: a_j is an agent perceived by the anticipatory agent a_i; $I_{\mathcal{A}}^{a_i}$ is the set of all other agents perceived by a_i, and $I_{\mathcal{P}}^{a_i}$ is the set of properties describing these agents $I_{\mathcal{P}}^{a_i}$. The output of *relationsOf* is a set of binary relationships shared by a_i and certain other agents in $I_{\mathcal{A}}^{a_i}$.

The procedure *constructRepresentation* (figure 1) uses two parameters to allow an anticipatory agent a_i to build its mental representation $M_{a_i}(A, R, D)$:

- the set $I_{\mathcal{E}}^{a_i}$, and
- the initial value *initStates*, used to instantiate the domains of D.

This initial value must correspond to the set of all possible states that an agent can take on.

procedure *constructRepresentation*
in: $I_{\mathcal{E}}^{a_i}(I_{\mathcal{A}}^{a_i}, I_{\mathcal{P}}^{a_i})$, *initStates*
out: $M_{a_i}(A, R, D)$

1: **begin**
2: **for each** $a_j \in I_{\mathcal{A}}^{a_i}$ **do**
3: $dom(a_j) \leftarrow initStates$
4: $A \leftarrow A \cup a_j$
5: $D \leftarrow D \cup dom(a_j)$
6: $R \leftarrow R \cup relationOf(a_j, I_{\mathcal{A}}^{a_i}, I_{\mathcal{P}}^{a_i})$
7: **end**

Figure 1: The *constructRepresentation* procedure

The parameters of the *anticipate* procedure are the following: the list of actions LA that an agent a_i must anticipate, the perception $I_{\mathcal{E}}^{a_i}$, and a list of undesirable states LS. The list LA can include not only the actions of a_i but also the known actions of other agents in $I_{\mathcal{A}}^{a_i}$. The definition of LA also depends on the application.

The way that a_i selects the undesirable states in LS reveals two kinds of behavior. If the states in LS only concern the anticipatory agent a_i, the behavior could be described as individualist or selfish because it would reveal that a_i does not

[2] For the application described in section 5, each agent triggers this algorithm at each iteration of a simulation, if the agent enters an intersection.

care if it harms other agents. If the undesirable states concern both a_i and some of the other agents in $I_{\mathcal{A}}^{a_i}$, the behavior would be more collective.

The aim of the *anticipate* procedure (figure 3) is to delete from the list LA those actions that will very likely induce one of the undesirable states in the list LS. The first instruction of the procedure prunes the values in the M_{a_i} domains by calling a constraint propagation function. Because propagation techniques can differ depending on the nature of the domain (e.g., temporal, spatial) and the semantic associated to the relationship, an external function is used to perform the propagation in order to keep the *anticipate* procedure generic.

Once the first propagation is performed, the anticipatory agent has an idea of the possible future states if no action is executed in the multi-agent system. The second instruction then scans the list LA and determines the direct effects of each action. These effects are computed by an external function called *computeDirectEffects*. Again, the body of this function depends on the application. For example, if actions express the movement of a situated agent, the function *computeDirectEffects* will perform a trajectory calculation. The set DE of direct effects is used to update the set R. For each $r_{a_i}(a_x, a_y) \in DE$, the relationships between a_x and a_y are deleted from the set R and the new relationships from DE are added: $R \leftarrow R \cup DE$. Following the update, a new propagation corresponding to the indirect effects of the current action is performed, which allows the domains to be restricted. After this second propagation, the anticipatory agent has an idea of the possible future states if the current action is actually executed.

The previous paragraphs have described the prediction phase of our anticipation model. With the last instruction, the *anticipate* procedure moves on to prediction analysis. During this analysis phase, the anticipatory agent searches for undesirable states. The *searchForUndesirableStates* procedure (figure 2) verifies whether each domain of the set D as been included in the list LS. If an undesirable state is detected, the current action is deleted from the list LA.

function *searchForUndesirableStates*
in: $M_{a_i}(A, R, D)$,UndesiredStates LS
out: boolean

```
1: begin
2:  for each dom(a_x) ∈ D do
3:      if dom(a_x) ∈ LS then
4:          return(true)
5:  return(false)
6: end
```

Figure 2: The *searchForUndesirableStates* procedure

procedure *anticipate*
in: Actions LA,UndesiredStates LS,$M_{a_i}(A, R, D)$, $I_{\mathcal{E}}^{a_i}$
out: Actions LA

```
1: begin
2: propagate(M_{a_i})
3: M' ← M_{a_i}
4: for each action ∈ LA do
5:     DE ← computeDirectEffects(action,I_E^{a_i})
6:     update(R,DE)
7:     propagate(M_{a_i})
8:     if searchForUndesirableStates(M_{a_i},LS) then
9:         LA ← LA − {action}
10:    M_{a_i} ← M'
11: end
```

Figure 3: The *anticipate* procedure

4 Discussion

4.1 Accessibility of the environment

The efficiency of our algorithm depends on the quality of the prediction. A prediction is correct only if the environment has certain properties. Since our approach is based on the *a priori* evaluation of the effects of actions, we assume that implementing an anticipatory agent in a multi-agent system requires that the environment be deterministic[3]. The environment must also be fully observable[4], or at least "partially observable". By "partially observable", we mean that the agents' perceptions can be local, but the information describing their environment has to be perfect and without noise. These properties insure that the anticipatory agents will be able to establish a set of good predictions about the system's future states.

4.2 Local vs complete perception of the environment

In designing his anticipatory architecture, Davidsson distinguished between two types of anticipation: linear anticipation and tree structure anticipation [9]. When the environment is deterministic and the world model is complete, it is always possible to predict exactly what will happen, which means that anticipatory agents will always calculate only one possible state for each point in time. This prediction of n successive states is linear and results from a sequence of predictions (figure 4).

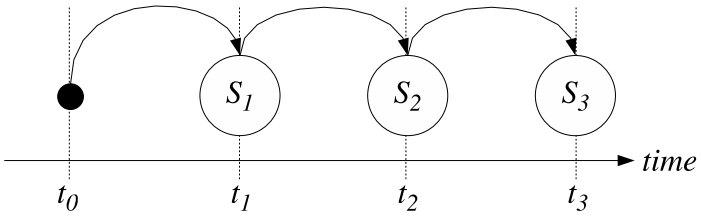

Figure 4: Linear anticipation [9]

However, most of the time, anticipatory agents have only a local perception of their environment; consequently, their model of the world is partial. This lack of information leads to the consideration of several possible states for each point in time. This prediction of future states requires the construction of a tree structure (figure 5).

Davidsson notes that linear anticipation *"makes strong assumptions concerning the quality of the world model and the behaviour of the environment"* but claims that *"there exist non-trivial applications where it is possible to have a sufficiently good world model [...] so that linear anticipation can be used successfully"*. However, even if we assume that the environment includes all the necessary good-quality properties, anticipation in the context of multi-agent coordination is usually not linear, but rather has a tree structure (combination of the action executed by different agents). Still, preventive anticipation can improve action selection by deleting inappropriate actions as leading to undesirable states, assuming, of course, that the anticipation procedure is activated before the actions are selected. Given that most of the time, executing different actions implies different possible future states, anticipation usually, of necessity, has a tree structure.

In our algorithm, the examination of each action on the list LA is a node on the first level of a tree structure anticipation. To maintain an acceptable degree of complexity over time, we chose to limit the tree branches to one level. For example, once an anticipatory agent has predicted a state S_t, it no longer tries to search the next states that could be reached from S_t. Since this first level must be evaluated over a finite time interval, the predictions are also limited by an "anticipation length". In the general case, this length can be taken into account during the *computeDirectEffects* procedure. When the domains are temporal, the length is explicitly defined by the *initStates* parameters (i.e, the finite set of values used to initialize the domains in the *constructRepresentation* procedure - figure 1).

4.3 Potential limitations

Having a partially observable and deterministic environment may seem to some to be a strong assumption. However, this assumption is acceptable for many multi-agent applications, especially for simulation applications which already require

[3]According to Russell and Norvig [39], an environment is deterministic "if the next state of the environment is completely determined by the current state and the actions executed by the agents.

[4]Russell and Norvig define "fully observable" as the capacity to perceive the complete state of the environment at each point in time [39].

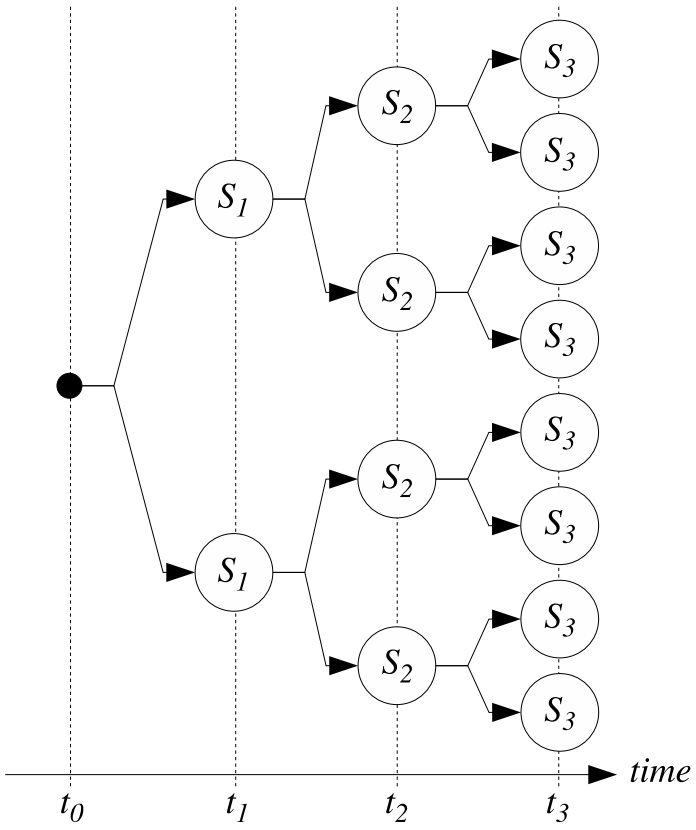

Figure 5: Tree structure anticipation [9]

many simplifications about environment in order to keep computing time low. Our approach reaches its limits when agents must evolve in a physical environment, as is the case in robotics. In this case, the perception of environment usually includes measurement errors and imprecisions. A possible solution could be to describe errors and imprecisions stochastically, for example using a stochastic constraint network [4, 21, 48].

4.4 Complexity

The complexity of our algorithm depends on:

- the number of actions: $|LA|$,
- the number of undesirable states that need to be checked: $|LE|$,
- the number of possible states expressed in the domain: $d = \max_{a_j \in \mathcal{A}}(|dom(a_j)|)$, and
- the complexity of external function: *propagate*.

In the literature, constraint propagation algorithms have a complexity between $O(ed^2)$ [31] and $O(ed^3)$ [26], with d being the maximum size of the domains and e being the total number of constraints in the network. In the worst case, the complexity of our *anticipate* procedure is within $O(|LA|(|R|d^3+|LE|))$. In practice, this complexity does not always imply prohibitive amounts of computing time. Indeed, for most applications, the number of possible states and the number of actions remain relatively low.

5 Application

Our approach was applied in the context of a road traffic simulation [11, 12, 13]. The algorithm proposed in section 3.4 was fully implemented in a behavioral traffic simulation tool based on a multi-agent system.

5.1 Road traffic simulation

We think that it is difficult to describe the individual behavior of each human driver. Thus, we decided to simulate artificial agents in order to highlight a collective behavior that is close to real applications. More generally, our study aimed to provide a *globally realistic* traffic simulation, with the secondary objective of characterizing vehicle avoidance techniques. Clearly, collisions and gridlock halt the simulation, which distorts the traffic studies.

A car driver is a typical example of an anticipatory agent [47]. Good drivers have to react to what happens in the present, but also need to make decisions by anticipating what may happen in the future. Drivers need to be aware of potential future states; to increase this awareness, they observe the curve of road, the traffic signals, the traffic conditions, the behavior of other drivers and so on. So, we chose to evaluate the relevance of our approach by testing it in a road traffic context.

The road traffic simulation in this study required realistically reproducing the movements of vehicles on a limited-capacity road network. To evaluate our approach, we wanted to focus on the specific situation of traffic at a road junction. Depending on the degree of realism desired, simulating traffic at road junctions can be a hard problem. For instance, given an intersection with no traffic signals, accurately simulating behaviors at the microscopic level, while also maintaining realistic traffic conditions (e.g., traffic flow, average speed) at the macroscopic level, is quite complex.

In response to this complexity, most simulation tools employ a centralized scheduler to coordinate the traffic flow at each branch of the intersection. But this drastically simplifies the initial problem since the scheduler usually allows only those vehicles whose trajectories are not in conflict to enter the intersection. Still, with a good configuration from this centralized source, it is usually possible to obtain fairly accurate general traffic conditions. Unfortunately, the behavior of individual simulated drivers is not at all realistic. Because of the obvious limitations of such simulation tools in terms of reproducing the actual behaviors of real drivers, much research has been done to try to consider road traffic simulation from a multi-agent point of view [8, 32, 43].

ArchiSim [17], a behavioral traffic simulation tool developped by INRETS, is one solution to the limitations described above. ArchiSim considers traffic as an emergent phenomenon resulting from the actions and interactions of the various road situation actors (e.g., car drivers, pedestrians, road operators). In other words, the traffic conditions at the macroscopic level are the result of the local behaviors of road users simulated at the microscopic level. It uses a multi-agent computing model, in which each simulated driver is an autonomous software agent that evolves in a virtual environment, interacting with the other simulated agents to reach its goals in accordance to its current skills and situation. At each step of the simulation, an agent receives a set of information describing the surrounding environment. Based on this information, the agents make their own decisions, resulting in the longitudinal and lateral acceleration needed in the next time step.

5.2 Traffic simulation at road junctions: a multi-agent coordination issue

In ArchiSim, dealing with road junctions is treated as a multi-agent coordination problem. In fact, when crossing an intersection, real drivers must solve their conflicts with the drivers of other vehicles. In a simulation, this conflict-solving task can be expressed as a problem of coordination: all agents reaching an intersection have to coordinate their actions in order to avoid a collision. In real life, this inter-driver coordination is facilitated by traffic signals (e.g., stop signs and traffic lights) and regulated by the rules of the Highway Code. However, these rules are not always respected by real drivers. To make the task even more complicated to analyze, according to psychology research [2, 40, 44] the driving task of crossing an intersection differs from one country to another. In southern Europe and in Asia, this task is highly competitive, especially for Latin drivers. In northern Europe, the task is less competitive and more cooperative.

The coordination mechanism used in ArchiSim tries to be flexible in order to take the above elements into account and to reproduce the diversity of driver behaviors. This mechanism models the conflicts that take place in the intersection by breaking the complex interactions between agents down into elementary situations involving only two agents.

An actual driver perceives a complex intersection as a succession of elementary T-type or X-type intersections. A roundabout or example, can be seen as a succession of T-type intersections, while a 4-corner stop is a simple example of an X-type intersection. The interactions between drivers in elementary intersections are regulated by the priority relationships defined in the Highway Code. Such priority relationships can be expressed using the predicate *prio*. In a X-type intersection, four interactions are possible and can be illustrated by the following conjunctions:

1. $\neg prio(x, y) \wedge \neg prio(y, x)$: no conflict exists between the agents x and y,
2. $prio(x, y) \wedge \neg prio(y, x)$: y comes from a minor road, or comes upon a road sign, and consequently has no priority over x,
3. $\neg prio(x, y) \wedge prio(y, x)$: x comes from a minor road, or comes upon a road sign, and consequently has no priority over y,

4. $prio(x, y) \wedge prio(y, x)$: x and y both have the priority.

The coordination mechanism has the following dynamics. An agent approaching an intersection searches for all vehicles with which it is potentially in conflict and assesses the priority it has in terms of each of those vehicles. Each priority relationship is used as a local rule that indicates whether the agent must speed up or slow down . When multiple priority relationships are involved in the current situation, the agent chooses the behavior that will lead to the lower speed [29]. At this point in the ArchiSim development, the coordination algorithm only manages the longitudinal acceleration, meaning that the agent can choose between two actions: *Go* or *Stop*. *Go* and *Stop* are extrapolations of the real objective which is to move the agents (forward or stop them). It should be noted that each movement or braking action complies with the kinematic model for the movement of the agent. More precisely, the vehicles do not stop merely because the *Stop* action was selected; they perform a braking procedure according to their physical constraints for a certain cycle. This means a certain number of simulation cycles are necessary in order to stop (provided, obviously, that this decision is not questioned in the subsequent iterations by any new information that has been received).

Each agent decision depends on how the agent interprets the priority relationships. To take into account the fact that a real driver may violate the Highway Code in some traffic situations, the agents in the simulation are allowed to alter the priority relationships established in the Highway Code, in effect replacing them by the informal priority relationships corresponding to specific practices frequently observed in the real life [12]:

- speed-related priority: drivers who have priority at an intersection (from point of view of the Highway Code) tend to give way that priority if they observe another vehicle approaching the intersection at a high speed,
- impatience-related priority: drivers who have been at a stop for a long time tend to consider that they have priority over other vehicles, even when the Highway Code would say otherwise.

Allowing the simulator to apply these types of priority relationships widely improves the similarity of the simulated driver behaviors. In [12], we demonstrated that ArchiSim was able to simulate realistic traffic in a non-signaled intersection located in southern Italy. Observed on the microscopic level, the behaviors seem realistic, and certain traffic conditions, such as traffic flow, differ only by 6% compared to the traffic data really measured at the existing road junction.

Unfortunately, such informal practices can sometimes lead to gridlock inside the intersection. Indeed, although the non-respect of the official rules encourages the use of the most central section of the intersection (which is more or less realistic and can be observed at many intersections), it also increases the risk of gridlock. In figure 6, a series of double left-turns[5] have provoked gridlock in the center of the intersection, which is clearly an undesirable state for all agents blocked in the intersection. To control such undesirable effects in the simulation, we implemented our preventive anticipation model in ArchiSim.

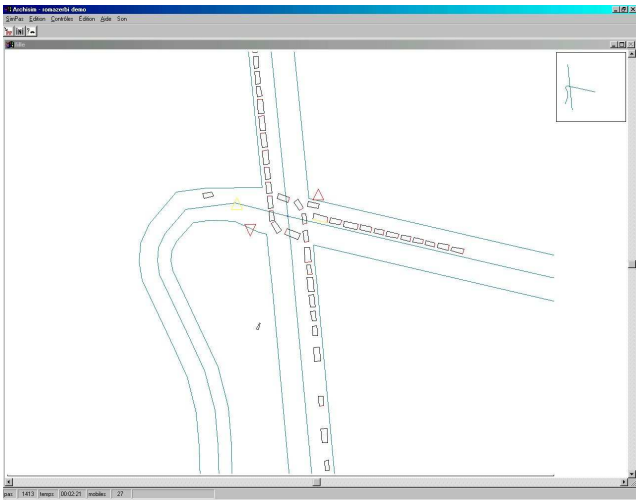

Figure 6: Gridlock in an intersection during a traffic simulation

[5]Please remember that, in most European countries, drivers drive on the left-hand side of the road.

5.3 Anticipation to avoid gridlock during simulation

5.3.1 Use of preventive anticipation in ArchiSim

To use our model of anticipation in road traffic simulation, we had to define the tree components of the mental representation M_{a_i} of an anticipatory agent a_i. The set A can be built with the agents present in the intersection and perceived by a_i.

The domain associated with each agent in A is temporal and expresses the next time step of the simulation. At the current time step t, if a value x is not present in the domain of an observed vehicle, this vehicle will be blocked at $t + x$. For instance:

- $dom(a_z) = [1, +\infty[$ is equivalent to: "a_z can act and move during the interval $t + 1$ to $+\infty$".
- $dom(a_z) = [1, 4] \cup [8, 10]$ is equivalent to: "a_z will be blocked between $t + 5$ and $t + 7$.

The set R is composed of (i) priority relationships (*prio* predicates) which has been previously introduced and (ii) three types of "blocking" relationships:

- $bph_{a_i}(a_x, a_y)$ meaning "a_x is physically blocked by a_y from the point of view of agent a_i",
- $bpha_{a_i}(a_x, a_y)$ meaning "a_i observes that a_x will be physically blocked by a_y", and
- $bpr_{a_i}(a_x, a_y)$ meaning "a_y has priority over a_x from the point of view of the agent a_i".

5.3.2 Example

To explain how the proposed algorithm works in a road traffic context, consider an agent a_x located at the entrance to an intersection (figure 7). Three vehicles (a_t, a_s, a_z) are stopped, blocked by a_y. The direction of each vehicle is as follows:

- a_s comes from the north and wants to turn left,
- a_t comes from the east and wants to turn left,
- a_z comes from the east and wants to turn left,
- a_y comes from the west and wants to turn left, and
- a_x comes from the south and wants to go straight.

The agent a_x computes the following representation[6] at the instant t:

$$\begin{aligned} A &= \{a_x, a_t, a_s, a_z, a_y\} \\ D &= \{dom(a_x) = [1, +\infty[\ ,\ dom(a_t) = [1, +\infty[\ , \\ &\quad\ dom(a_s) = [1, +\infty[\ , dom(a_z) = [1, +\infty[\ , \\ &\quad\ dom(a_y) = [1, +\infty[\ \} \\ R &= \{bpha(a_x, a_t), bph(a_t, a_z), bpr(a_t, a_s), \\ &\quad bph(a_s, a_z), bph(a_z, a_y), bpha(a_y, a_t), \\ &\quad prio(a_x, a_y) \wedge \neg prio(a_y, a_x)\} \end{aligned}$$

Applying our anticipation algorithm yields the following reasoning:

1. propagation: The domain of the five agents can be reduced. For instance, since a_z is blocked by a_y, it is possible to deduce that a_z will be stopped till a_y has passed the conflict point. This time span can be obtained through kinematic calculation[7]. The time span is obtained in a continuous domain and is expressed in seconds. Given the length of a simulation time step (a parameter of the simulation), this time is converted into a number of time steps so that it can be used to bound the agent domains. For instance, if this time span is equal to 2 time steps, the values 1 and 2 can be deleted from the a_z domain: $dom(a_z) = [3, +\infty[$. The same reasoning can be used to simplify the a_s and a_t domains: $dom(a_s) = [5, +\infty[$ and $dom(a_t) = [7, +\infty[$. The simplification of $dom(a_t)$ does not

[6] The suffixes a_i have been omitted for the relations of R.

[7] Classic kinematic formulas are used to compute the time span. If the vehicle is moving, the time is $t = \frac{(\sqrt{v^2+2ad}-v)}{a}$.

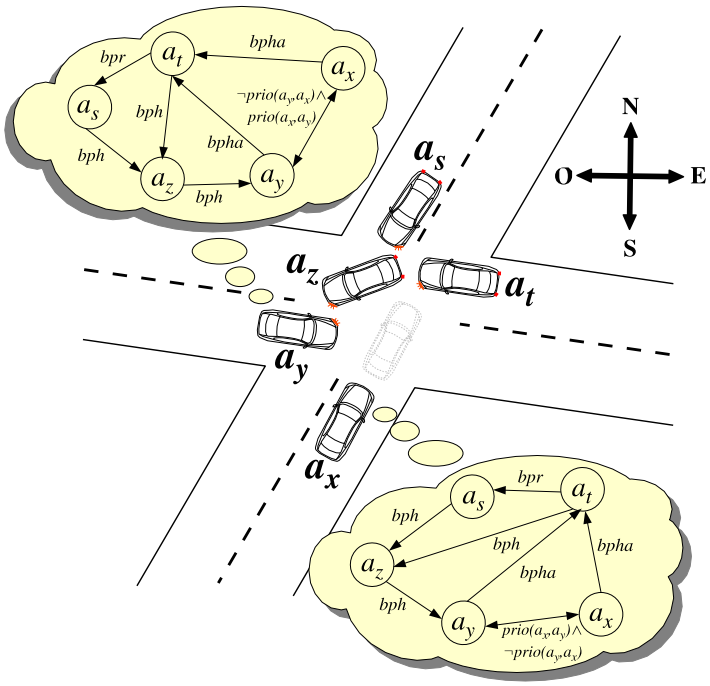

Figure 7: Mental representations of two agents in a crossroad

prevent $dom(a_y)$ from being reduced since a relationship between a_t and a_y exists: $bpha(a_y, a_t)$. Finally, the set D is simplified and is now equal to:

$$\begin{aligned} D \quad = \quad & \{dom(a_x) = [11, +\infty[, dom(a_t) = [7, +\infty[, \\ & dom(a_s) = [5, +\infty[, dom(a_z) = [3, +\infty[, \\ & dom(a_y) = [1, 5] \cup [9, +\infty[\ \} \end{aligned}$$

2. effects computation of actions of a_x: Since the coordination algorithm only deals with longitudinal acceleration, a_x has a choice between two possible actions: *Stop* or *Go*. If a_x chooses the action *Go*, its position on the road changes. Consequently, its distance from a_t is reduced, and the blocking relationship between a_x and a_t takes effect. This is expressed in the mental representation by replacing the relationship $bpha(a_z, a_t)$ by $bph(a_z, a_t)$. In addition, the priority relationship $prio(a_x, a_y) \wedge \neg prio(a_y, a_x)$ also becomes a blocking relationship: $bph(a_y, a_x)$. The set R is thus updated as follows:

$$\begin{aligned} R \quad = \quad & \{bph(a_x, a_t), bph(a_t, a_z), bpr(a_t, a_s), \\ & bph(a_s, a_z), bph(a_z, a_y), bpha(a_y, a_t), \\ & bph(a_y, a_x)\} \end{aligned}$$

3. propagation: Adding the relationship $bph(a_y, a_x)$ allows the a_y domain to be simplified once again. However, the simplification of $dom(a_y)$ implies the simplification of all domains, which can then be reduced to an empty interval. This reduction is coherent since none of the vehicles can now move:

$$\begin{aligned} D \quad = \quad & \{dom(a_x) = \varnothing, dom(a_t) = \varnothing, dom(a_s) = \varnothing, \\ & dom(a_z) = \varnothing, dom(a_y) = \varnothing\} \end{aligned}$$

4. undesirable states search: From the point of view of a_x, having an empty domain can be considered as an undesirable state. To avoid this undesirable state, the agent a_x can eliminate the action *Go* from its list of possible actions, giving this agent no other option but to stop. However, the anticipation algorithm applied by a_y does not have to change its list of actions. Invoking the impatience-related priority, it considers that it has waited a sufficiently long time and decides to accelerate.

5.4 Evaluation

To evaluate the anticipatory abilities of the simulated drivers, we used different scenarios that placed the agents in traffic situations that would be produce gridlock: for example, simulated traffic at non-signaled intersections with high traffic densities and series of double left-turns.

5.4.1 Avoiding gridlock during simulation

Consider a congested traffic situation at a X-type intersection. Because it is non-signaled intersection, drivers must yield to the traffic coming from the right. Fourteen simulated vehicles are at the road junction. Vehicle number 13 is approaching the intersection, and the vehicle 8 is stopped inside the intersection. Figures 8 and 9 illustrates the two possible results: without anticipation and with anticipation.

In figure 8, the agents have no anticipatory abilities. At time step $t = 1$, only vehicles 13 and 8 are still able to move; all other agents inside the intersection are blocked. In this situation, the coordination algorithm leads vehicle 13 to choose to *Go* and vehicle 8 to choose to *Stop*. These decisions produce total gridlock in the intersection by $t = 270$.

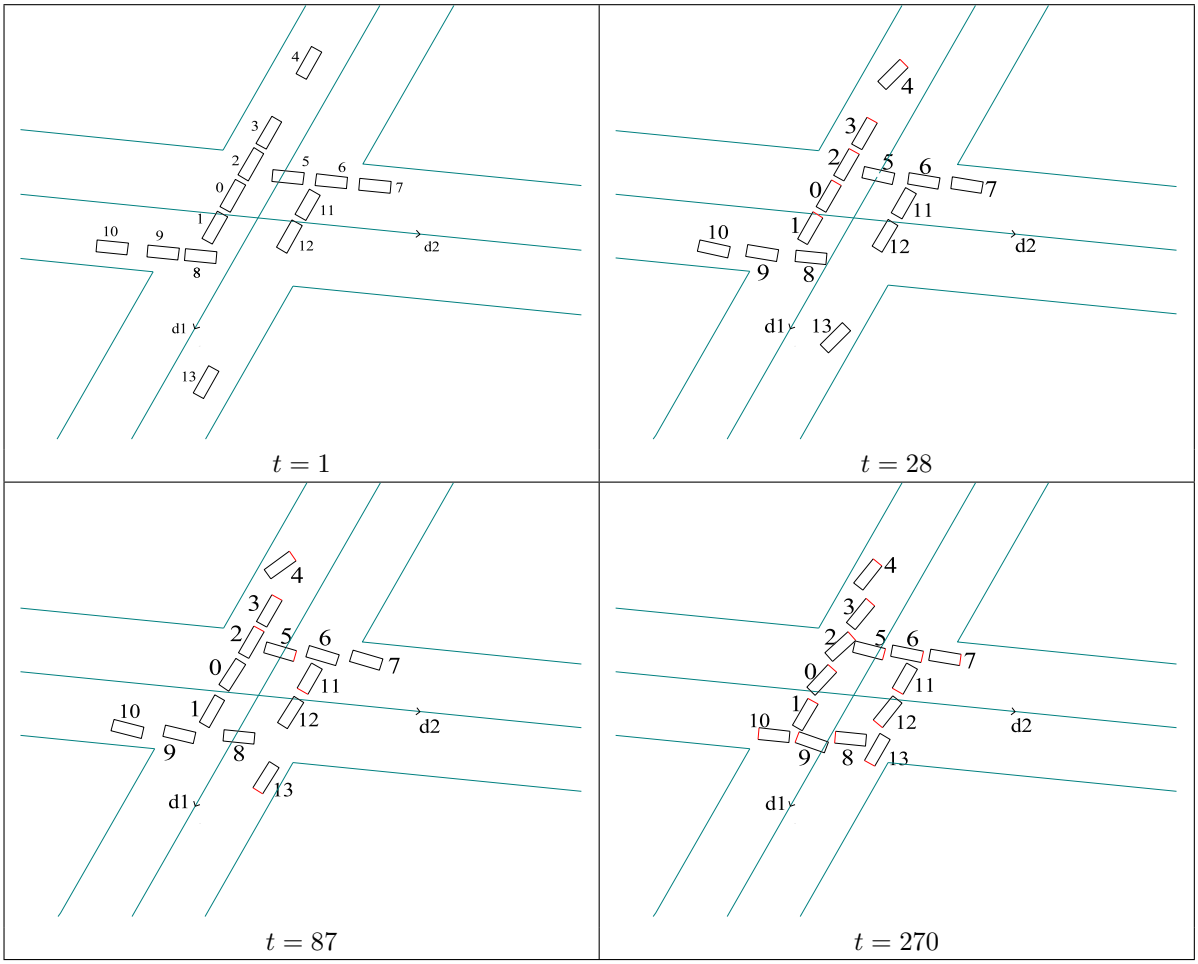

Figure 8: Sequences of the simulation without anticipation in ArchiSim

Figure 9 resents the same scenario as figure 8, but in figure 9, the agent 13 has access to the anticipatory algorithm. At time step $t = 1$, the agent 13 builds a mental representation of the situation. It takes into account the agents located inside the intersection and adds the following relationships: $bpha_{13}(13, 12) \wedge bph_{13}(12, 11) \wedge bph_{13}(11, 6) \wedge ... \wedge bph_{13}(0, 1) \wedge bph_{13}(1, 8)$. Then, it considers the vehicles that are still outside the intersection and adds the relationships, $bpr_{13}(10, 0)$, $bpr_{13}(10, 2)$ and so on. The agent 13 finishes building its representation by associating a domain to each of the agents observed in the intersection. Before calling the *anticipate* procedure, all domains are initialized to $[1, T]$, where T corresponds to the length of anticipation introduced in section 4.2 (in our experiments, $T = 15$).

In the next step, the algorithm calculates the effects of the action *Go* for the agent 13. Given the agent's current position, speed and acceleration, the algorithm computes the agent's future position and deduces the new topological

and spatial relationships with the other agents. These topological relationships are mapped as blocking and priority relationships, which are then added to the network, and a new propagation is performed. The agent 13 can now search for undesirable states. In doing so, it deduces that its domain has become $dom(13) = \varnothing$, which will put it into infinite gridlock. In response to this deduction, it deletes the action *Go* from the list of possible actions and decides to *Stop*.

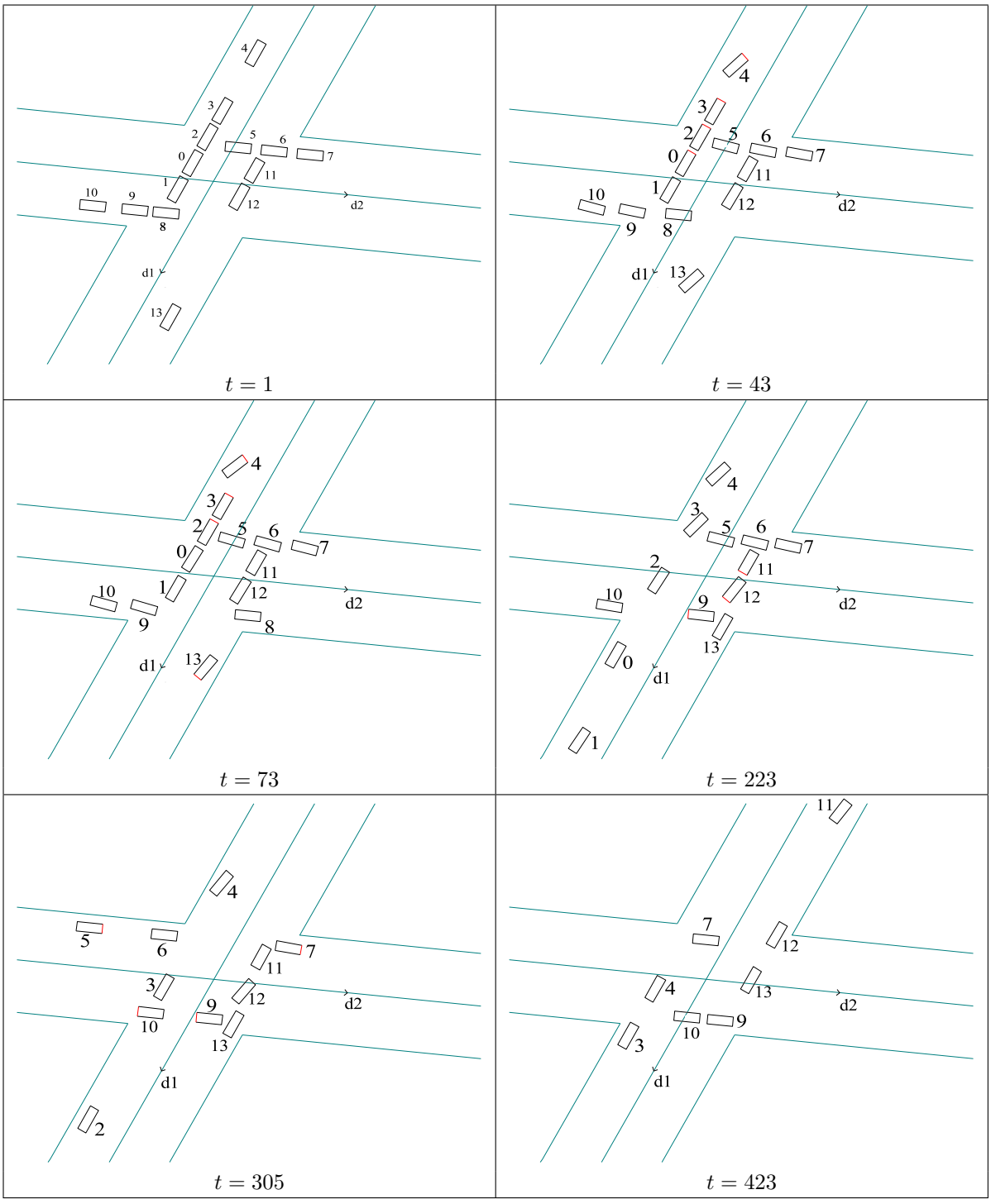

Figure 9: Sequences of the simulation with anticipation in ArchiSim

Figures 10 and 11, which show the speed and acceleration curves for vehicle 13 (with and without the anticipation ability), illustrate the dynamism of the situation. Between steps 0 and 25, the variation of the speed of vehicle 13 is the same both with and without anticipation: 13 breaks in order to avoid bumping into vehicle 12. Between steps 25 and 50, agent 13 applies the coordination algorithm directly and chooses to move forward: it continues to decelerate but more

gradually. At the same time, the anticipatory algorithm detects that the action *Go* will create total gridlock. In response to this, vehicle 13 chooses to break hard. During this deceleration period, vehicle 8 is likely to decide that it has priority and move before 13, which progressively allows the intersection to be cleared: from step $t = 320$, agent 13 can accelerate until it reaches a desirable speed. Without anticipation, vehicle 13 blocks the intersection at step $t = 270$.

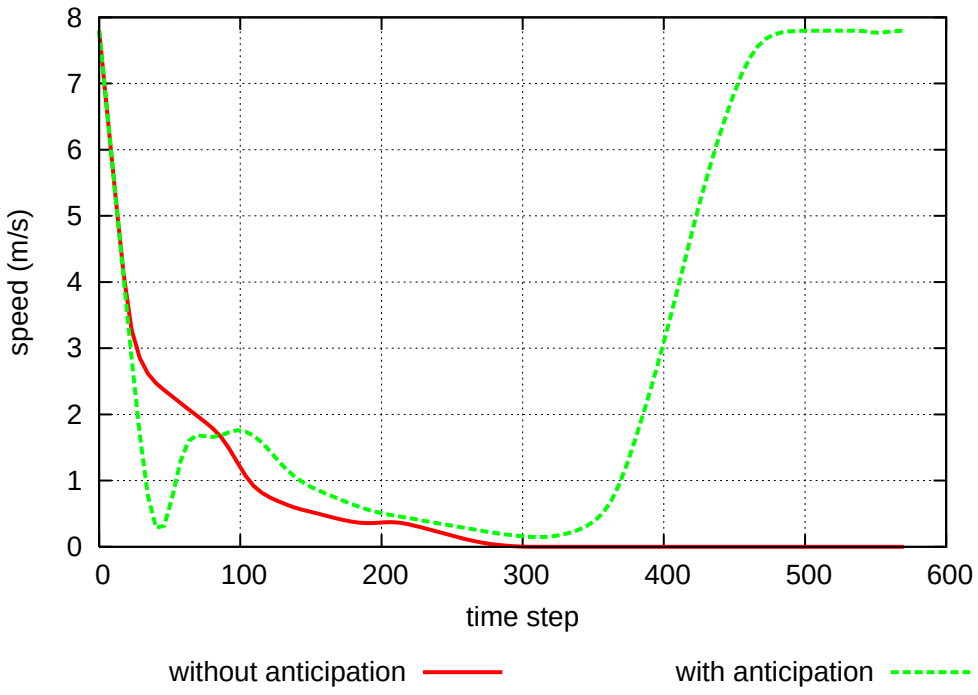

Figure 10: Speed curve of agent 13

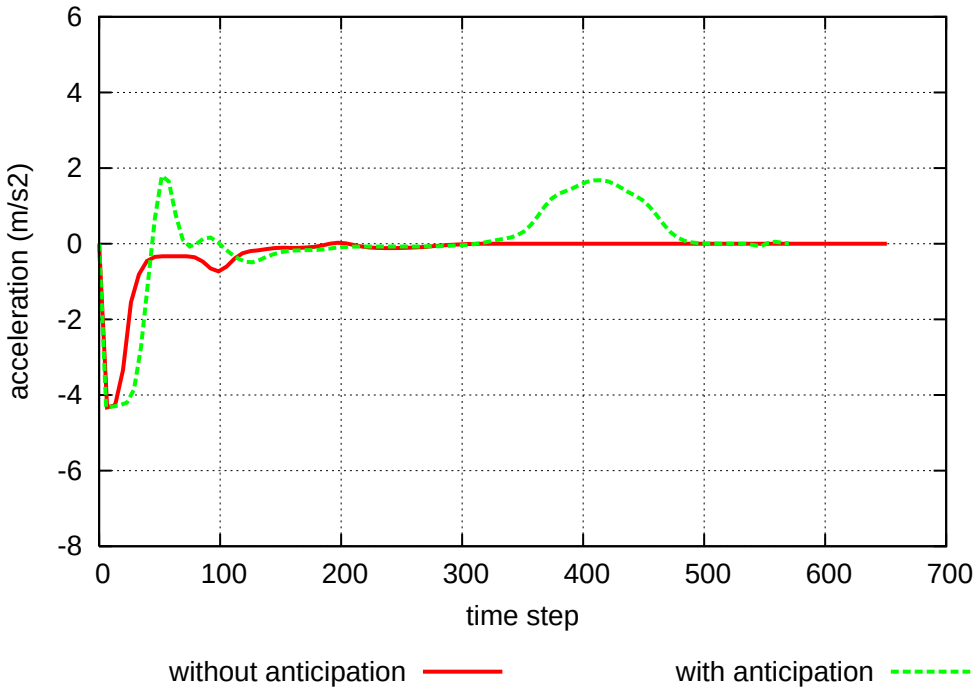

Figure 11: Acceleration curve of agent 13

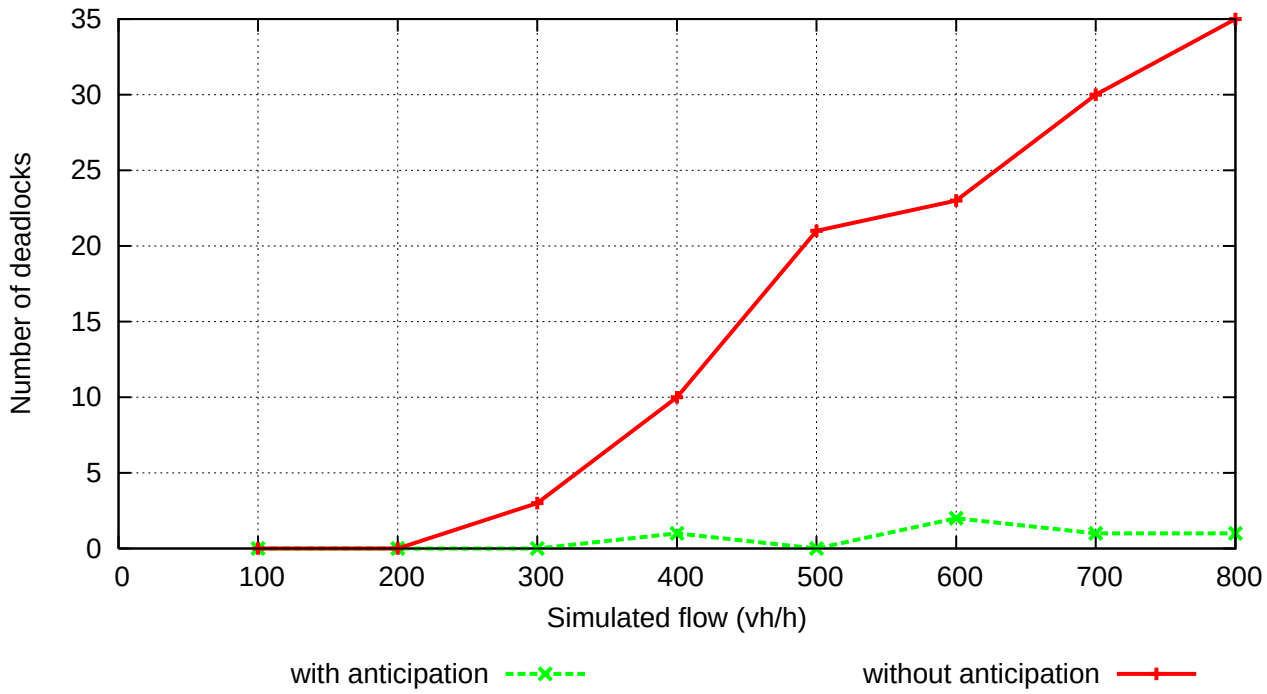

Figure 12: Variation of the number of gridlocks in terms of the traffic flow

5.4.2 Evaluation of gridlock reduction

For the same intersection, we simulated several traffic flow variations. For each simulation, the resulting gridlock situations were counted via a supervision process that is part of the simulator but totally independent of agent decision-making. This supervisor checks at each time step to see whether or not a gridlock appears. If gridlock is detected, the simulation is suspended, the intersection is completely cleared, and then the simulation proceeds.

Figure 12 shows how anticipation reduces the number of gridlocks during the simulation of a non-signaled intersection. For a one-hour simulation using non-anticipatory agents, the number of gridlocks varied from 0 to 35 depending on the traffic flow. When the agents had anticipatory capabilities, the number of gridlocks remained near zero almost all the time for simulated flows of up to 800 vehicles/hour per axis.

5.4.3 Additional Results

For the same intersection, we simulated several traffic flow variations and studied the time that the artificial agents take to "decide" what action to perform. Figure 13 shows that the decision-making time used by each agent increases according to the traffic density. For low traffic densities (less than 300 vehicles/hour), the necessary execution time for all the agents is very reasonable: less than 0.1 seconds. For high traffic densities (800 vehicles/hour), the time needed to execute the proposed model was an average of 0.35 seconds. We can thus assert that this time is relatively low, depending on the importance of the traffic density for the proposed simulations.

We would also like to point out that validating a traffic simulation is a complex problem. As has been noted by Lieberman and Rathi [25], traffic engineers usually lack the current data that would allow them to compare their simulations with real data. For the specific case of intersections, this lack of data is due to the difficulty of collecting/obtaining data from the institutes responsible for road infrastructure. For example, to compare traffic flows, it is necessary to record not only the number of vehicles per second on each axis, but also the directional percentage inside the intersection. This second element is difficult to automate since it requires tracking each vehicle that enters the intersection until it exits. To improve the accuracy of our results, we conducted an evaluation procedure, in which the traffic in a real intersection was simulated and the results were compared with the traffic data actually measured. The chosen statistical criteria is the RSD (Root Standard Deviation), which evaluates the difference between simulated and real flows. To *obtain a globally realistic behavior*, this criteria must be less than 10% (standard value for the traffic, given the traffic literature). The intersection used in this evaluation is located in the Italian city of Reggio Calabria (Southern Italia) [14]. This intersection is the junction between a main road running north to south and a secondary trunk road running east to west. The values obtained are considered to be of good quality.

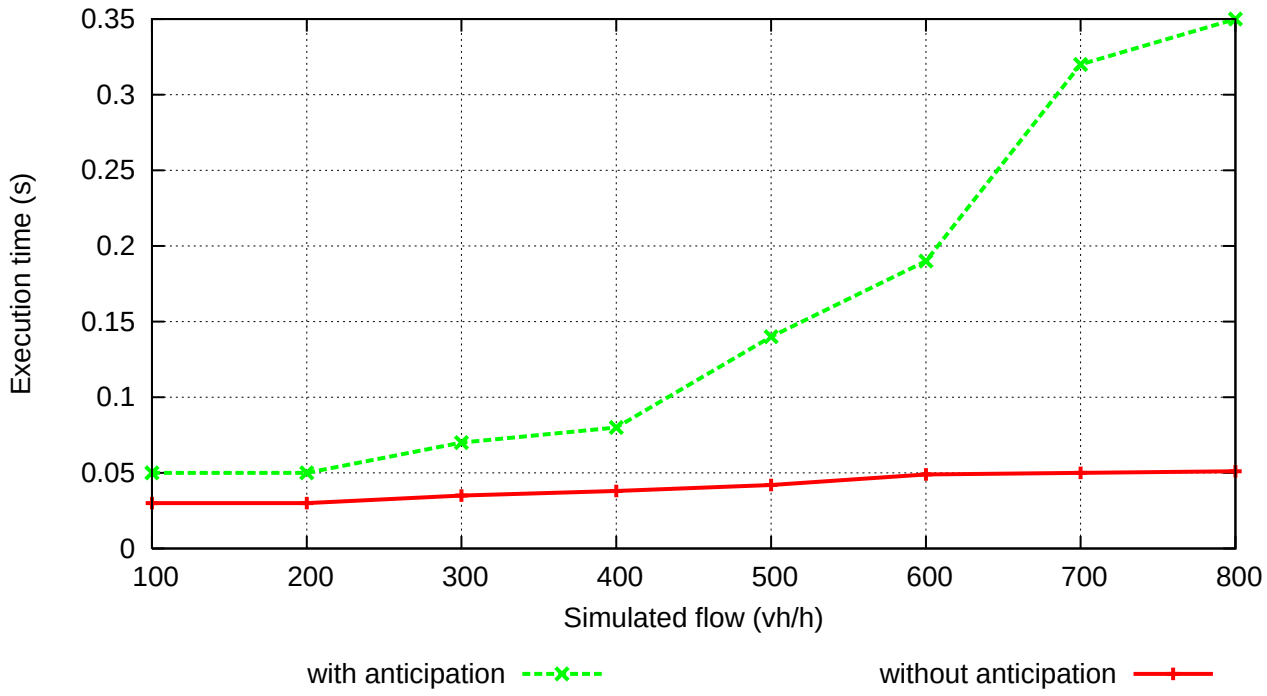

Figure 13: Temporal performance between behaviours with/without anticipation

6 Future research

Our method employs a constraint network based on agent mental representations. In this version of the method, we assume that all system agents anticipate individually. However, in some cases, a set of agents can have a common objective to anticipate certain undesirable states: for instance, in the formation of coalitions [1, 41]. In this case, anticipation takes on a collective dimension, in which the agents reason together in order to detect undesirable states. In future research, we plan to investigate this collective aspect by considering an extension of the constraint networks: distributed constraint satisfaction problems.

6.1 Towards a collective anticipation model

An extension of constraint networks have been introduced to formalize and resolve naturally distributed problems [50]. Such problems generally deal with a set of data, shared out among many sites, whose centralization is often impossible. A disCSP (X, D, C, A) is an extension of the triplet (X, D, C), where A is a finite set of agents $\{A_1, A_2, ...A_p\}$ in which each $A_k (1 \leq k \leq n)$ has a subset of X. Many algorithms in the literature could be used to achieve our proposed extension, provided that agents have some ability to exchange messages [15, 50]. We plan to investigate such an approach for the case of Automated Guided Vehicles [49], in which communication is easily handled.

Complementary research proposed by Boman et al. [3] supports our studies about on the modelling of the collective anticipation. Their works aims at describing a multi-agent architecture based on the anticipation of the detection of collective actions that are or not acceptable by the group. One of the essential differences with our current proposal is their consideration of the effect of uncertainty and risk on the actions of the various agents. This approach could be useful for decisions-making, given the large number of actions and constraints in our anticipation mechanism.

6.2 Generalization to n-ary relations

The anticipation model we have proposed in this article uses binary relations between agents. These relations are considered as constraints, allowing each agent to construct a constraint network for use as a mental representation. Binary constraint networks are a particular case of constraint satisfaction problems (CSP) in which each constraint can be mapped as an edge of a graph.

In some applications, it is interesting to deal with n-ary relations instead of binary relations. In literature, many studies have dealt with such relations. Most algorithms dedicated to constraint networks have been generalized for non-binary

constraints [27]. Some studies have also proposed a binarization approach that encapsulates non-binary constraints into an additional variable. All this research would seem to be easily applicable to our anticipation model. In an application like an Automated Highway System, our approach should be relevant to the study of the vehicle platoon [19].

7 Conclusion

This article addresses the issue of modelling anticipation in a multi-agent coordination context. Our contribution to the scientific discussion is the proposed formalization of preventive anticipation that allows agents to reason about the effects of their actions and thus avoid undesirable states. Our anticipation principle is based on a constraint processing approach which considers effects of actions as constraints in a mental representation of the agent's world. Based on this model, a generic algorithm was proposed and evaluated for a traffic simulation problem. The results of this algorithm show that integrating anticipation allows behaviors to be reproduced without creating gridlock between the simulated vehicles.

Acknowledgments

We would like to thank Professor Dominico Gattuso, for allowing us to exploit his measured data. The authors thank also the anonymous reviewers for their numerous constructive remarks.

The present research work has been supported by the European Community, the "Délégation Rgionale la Recherche et la Technologie", the "Ministère de l'Education Nationale, de la Recherche et de la Technologie", the "Nord-Pas de Calais" region, the "Centre National de la Recherche Scientifique", and the French National Institute for Transport and Safety Research (INRETS). We gratefully acknowledge the support of these institutions.

References

[1] S. Aknine, S. Pinson, and M. F. Shakun. New coalition formation methods for multi-agent coordination. *Journal of Autonomous Agents and Multi-Agent Systems*, 2000.

[2] G. M. Björklund and L. Åberg. Driver behaviour in intersections: Formal and informal traffic rules. *Transportation Research Part F*, 8:239–253, 2005.

[3] M. Boman, P. Davidsson, J. Kummeneje, and H. Verhagen. An Anticipatory Multi-Agent Architecture for Socially Acceptable Action. In *The 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, IOS Press, 2000.

[4] L. Bordeaux and H. Samulowitz. On the Stochastic Constraint Satisfaction Framework. In *Proceedings of the 2007 ACM symposium on Applied computing*, Seoul, Korea, 2007.

[5] D. S. Blank, J. M. Lewis, and J. B. Marshall. The multiple roles of anticipation in developmental robotics. In *AAAI 2005 Fall Symposium: From Reactive to Anticipatory Cognitive Embodied Systems*, pages 8–15, Arlington, USA, 2005.

[6] M. V. Butz, O. Sigaud, and P. Gérard. *Anticipatory Behavior in Adaptive Learning Systems: Foundations, Theories, and Systems*, chapter Internal Models and Anticipation in Adaptive Learning Systems. Springer, 2003.

[7] C. Castelfranchi, R. Falcone, and M. Piunti. Agents with anticipatory behaviors: To be cautious in a risky environment. In *Proceedings of European Conference of Articicial Intelligence 2006 (ECAI06)*, 2006.

[8] A. Champion, S. Espié, R. Mandiau, and C. Kolski. A game-based, multi-agent coordination mechanism - application to road traffic and driving simulations. In *Summer Computer Simulation Conference*, pages 644–649, Canada, 2003.

[9] P. Davidsson. *Anticipatory Behavior in Adaptive Learning Systems*, chapter: A Framework for Preventive State Anticipation. Springer, 2003.

[10] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[11] A. Doniec, S. Espié, R. Mandiau, and S. Piechowiak. Multi-agent coordination and anticipation model to design a road traffic simulation tool. In *EUMAS'06, Proceedings of the fourth European Workshop on Multi-Agent Systems*, Lisbon, 2006.

[12] A. Doniec, R. Mandiau, S. Espié and S. Piechowiak. Non-normative behaviour in multi-agent system: some experiments in traffic simulation. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT) 2006 Main Conference Proceedings*, pages 30–36, Hong Kong, China, 2006.

[13] A. Doniec, R. Mandiau, S. Espié, and S. Piechowiak. Dealing with multi-agent coordination by anticipation: Application to the traffic simulation at junctions. In *EUMAS'05, Proceedings of the Third European Workshop on Multi-Agent Systems*, 2005.

[14] A. Doniec, R. Mandiau, S. Piechowiak and S. Espié. Controlling non-normative behaviors by anticipation for autonomous agent. *Web Intelligence and Agent Systems: An International Journal*, 6:1–14, 2008.

[15] A. Doniec, S. Piechowiak, and R. Mandiau. A DisCSP solving algorithm based on sessions. In *Recent Advances in Artificial Intelligence: Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference (FLAIRS'05)*, pages 666–670, Clearwater, Floride, USA, 2005.

[16] B. Ekdahl. Anticipatory systems as linguistic systems. In *Proceedings of CASYS'99: Computing Anticipatory Systems*, pages 678–689, Lige, Belgique, 1999.

[17] S. Espié. Archisim, multi-actor parallel architecture for traffic simulation. In *Proceedings of the Second World Congress on Intelligent Transport Systems*, Yokohama, Japan, 1995.

[18] L. Gasser. *Distributed Artificial intelligence: theory and praxis*, chapter DAI approaches to Coordination. Kluwer, 1992.

[19] S. Hallé and B. Chaib-draa. Collaborative driving system using teamwork for platoon formations. In *Proceedings of the AAMAS-04 Workshop on Agents in Traffic and Transportation*, pages 35–46, New York, USA, 2004.

[20] N. R. Jennings. Coordination techniques for distributed artificial intelligence. In *Foundations of Distributed Artificial Intelligence*, pages 187–210. Wiley, 1996.

[21] V. Kirillov and V. Honavar. Simple Stochastic Temporal Constraint Networks. *Technical Report TR95-16*, Department of Computer Science, Iowa State University, 1995.

[22] V. Kumar. Algorithms for constraint-satisfaction problems: A survey. *Ai Magazine*, 1992.

[23] J. Laird. It kwows what you're going to do: Adding anticipation to a quakebot. In *Proceeding of Autonomous Agent 2001*, Montreal, Canada, 2001.

[24] V. R. Lesser. Reflections on the nature of multi-agent coordination and its implications for an agent architecture. *Autonomous Agents and Multi-Agent Systems*, 1:89–111, 1998.

[25] E. Lieberman and A. Rathi. *Traffic flow theory.* Oak Ridge National Laboratory, Chapter Traffic simulation, 1997.

[26] A. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[27] A. Mackworth. On reading sketch maps. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'77)*, pages 598–606, Cambridge MA, 1977.

[28] T. Malone. What is coordination theory. In *National Science Foundation Coordination Theory Workshop*. 1998.

[29] R. Mandiau, A. Champion, J-M. Auberlet, S. Espié and C. Kolski. Behaviour based on decision matrices for a coordination between agents in urban traffic simulation. *Applied Intelligence*, 28:121–138, 2008.

[30] J. Marshall, D. Blank, and L. Meeden. An emergent framework for selfmotivation in developmental robotics. In *Proceedings of the 3rd International Conference on Development and Learning*, pages 104–111, La Jolla, USA, 2004.

[31] R. Mohr and T. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28:225–233, 1986.

[32] P. Paruchuri, A. R. Pullalarevu, and K. Karlapalem. Multi agent simulation of unorganized traffic. In *Proceeding of The Internationnal Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, Bologne, Italie, 2002.

[33] G. Picard and M.-P. Gleizes. Cooperative self-organization to design robust and adaptive collectives. In *Proceedings of the Third European Workshop on Multi-Agent Systems (EUMAS'05)*, Bruxelles, 2005.

[34] M. Piunti, C. Castelfranchi, and R. Falcone. Surprise as shortcut for anticipation: clustering mental states in reasoning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI07)*, 2007.

[35] A. Riegler. The role of anticipation in cognition. In *Computing Anticipatory Systems. Proceedings of the American Intitute of Physics 573*, pages 534–541, 2001.

[36] R. Rosen. Planning, management, policies and strategies: Four fuzzy concepts. *International Journal of General Systems*, 1974.

[37] R. Rosen. *Anticipatory Systems - Philosophical, Mathematical and Methodological Foundations*. Pergamon Press, 1985.

[38] J. S. Rosenschein, M. Ginsberg, and M. R. Genesereth. Cooperation without communication. *Proceedings of the Fifth National Conference on Artificial Intelligence, AAAI'86*, 1986.

[39] S. Russell and P. Norvig. *Artificial intelligence: A Modern Approach*. Prentice-Hall, 1995.

[40] F. Saad. In-depth analysis of interactions between drivers and the road environment: contribution of on-board observations and subsequent verbal report. In *Proceedings of the 4th Workshop of ICTCT*, University of Lund, 1992.

[41] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Articial Intelligence*, (101):165–200, 1998.

[42] W. Stolzmann. Anticipatory classifier systems. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, San Francisco, USA, 1998.

[43] R. Sukthankar, J. Hancock, S. Baluja, D. Pomerleau, and C. Thorpe. Adaptive intelligent vehicle modules for tactical driving. In *Proceedings of AAAI-1996 Workshop on Intelligent Adaptive Agents*, 1996.

[44] H. Summala. Towards understanding driving behaviour and safety efforts. In *Proceedings of the International Workshop on Modelling Driver Behaviour in Automotive Environments*, pages 105–111, Ispra, Italie, 2005.

[45] K. P. Sycara, s. Roth, N. Sadeh, and M. Fox. Distributed constrained heuristic search. In *IEEE Transactions on Systems, Man and Cybernetics*, pages 1446–1461, 1991.

[46] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London and San Diego, 1993.

[47] L. H. Tsoukalas. Neurofuzzy anticipatory systems: A new approach to intelligent control. *International Journal on Artificial Intelligence Tools*, 6(3):365–395, 1997.

[48] T. Walsh. Stochastic Constraint Programming. *Proceedings of the 15th European Conference on Artificial Intelligence, ECAI'2002*, Lyon, France, 2002.

[49] D. Weyns, K. Schelfthout, T. Holvoet, and T. Lefever. Decentralized control of e'gv transportation systems. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05) Industrial Applications*, pages 67–74, 2005.

[50] M. Yokoo. *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-agent Systems*. Springer Verlag, 2001.