Contribution to the Control of a MAS's Global Behaviour: Reinforcement Learning Tools

François Klein, Christine Bourjot, Vincent Chevrier

Contribution to the Control of a MAS's Global Behaviour: Reinforcement Learning Tools

François Klein, Christine Bourjot, Vincent Chevrier

LORIA – Nancy University
Campus scientifique BP 239
54506 Vandoeuvre-lès-Nancy Cedex
{Francois.Klein, Christine.Bourjot, Vincent.Chevrier }@loria.fr

Abstract. Reactive multi-agent systems present global behaviours uneasily linked to their local dynamics. When it comes to controlling such a system, usual analytical tools are difficult to use so specific techniques have to be engineered. We propose an experimental dynamical approach to enhance the control of the global behaviour of a reactive multi-agent system. We use reinforcement learning tools to link global information of the system to control actions. We propose to use the behaviour of the system as this global information. The behaviour of the whole system is controlled thanks to actions at different levels instead of building the behaviours of the agents, so that the complexity of the approach does not directly depend on the number of agents. The controllability is evaluated in terms of rate of convergence towards a target behaviour. We compare the results obtained on a toy example with the usual approach of parameter setting.

Keywords: Control, MAS, experimental approach, emergence, global behaviour, reinforcement learning

1 Introduction

The goal of this study is to control the behaviour of a multi-agent system (MAS) or of a complex system modeled by a MAS. It takes place in the context of reactive MAS [1]. A reactive agent only owns reflex reactions to external stimuli, without a representation of its environment or itself, and has a limited memory. In such a system, interactions are essential and trigger a collective behaviour which is not directly linked to the individual behaviours.

The behaviour of the MAS is unpredictable without simulation, because of its strongly non-linear nature due to the numerous interactions that occur. However, global complex phenomena can be observed. These phenomena emerge from the local interactions between the agents despite of their limited abilities. So there are two levels of description in a MAS: the local level, where the agents evolve and their behavioural rules are set, and the global one, where a particular behaviour of the whole system can be observed.

An emergent structure or property is characterized by a phenomenon involving several agents, whose observation is done at a higher time scale than the local

evolution of the system and makes it appear stable. We define the global behaviour of the MAS as a description of its emergent phenomena. Several behaviours can be observed in a given MAS, but it is uneasy to predict which one will occur when the system is running, for any local perturbation can trigger a dramatically different one.

Our purpose is to control the global behaviour of the system, although it is uneasily linked to the local dynamics of the MAS. Namely, we wish to make the system show a target global behaviour thanks to actions correctly chosen and performed at the right time. The actions can be local modifications of the state of the system, or global quantitative changes like the modification of a parameter value. They depend on what the controller is allowed to do, particularly if the MAS models a real-world system. The goal is to give a pragmatic control method, applicable to many reactive MAS. Focusing on reactive MAS allows us to put the stress on its particular properties, without challenging the *intelligence* of the agents to achieve the control of the system.

In this contribution, an original solution of dynamical control of an agent-based model is proposed, but we keep in sight the control of a concrete distributed system modelled by a MAS.

We first expose the issues brought on by the control problem and the specificities a control solution of a MAS should meet. Then we discuss different solutions to solve this problem, especially the one of parameter setting, and their limitations. We differentiate the static and the dynamic approaches. This leads to our proposition of a dynamical method that takes more information on the system into account. We explain how we implemented it on an toy example, and we compare the control performances of our method and the parameter setting solution through different criteria.

2 Control of a Reactive Multi-Agent System

If there is only one possible global behaviour that the system always shows, there is no control problem, otherwise, we consider the different reachable behaviours as global states. So our problem is to lead the system into one particular state. Two kinds of difficulties come up to solve this problem: the ones due to the nature of the MAS, and the ones due to the control problem itself. They are discussed in this section.

Because of the complexity of the interactions in a MAS, the local and global levels are not directly linked. If there are known local rules, and even simple rules in a reactive MAS, they do not give rise to a global view upon the behaviour of the system itself. For this reason, in multi-agent field of research, it is generally admitted (cf. Wegner [2], Edmonds [3,4], DeWolf [5], Amblard [6]) that an analytical model of a MAS is infeasible. These authors recommend then to study MAS through an experimental approach.

This limits the possible ways to solve the control problem: a solution is necessarily experimental, and we cannot use powerful tools as differential equations to predict the global behaviour triggered by an action.

Hence we assume that the studied MAS is totally observable and that we can act on it at will. If it is an already deployed system, like a peer-to-peer network, we can always model it thanks to another MAS and perform our proposition on the model.

Before to handle the concrete system, the study of a MAS model is useful since it involves less experimental costs and since it allows to know if the original system can be controlled. Therefore, in the following, the discussion is restricted to the control of a model.

An experimental study implies a considerable number of simulations. In order to provide realistic tools to control the MAS in a reasonable time, we have to avoid the simulation time soaring by any means. We can consider to reduce the simulation time by reducing either the time of each simulation or the number of simulations. Many solutions have been proposed to handle this issue, we discuss them in §3.1.

Beyond these difficulties linked to the nature of the MAS, since we intent to control the MAS, three questions have to be answered: how to characterise and measure the global behaviour, what are the possible actions, and how to determine the action to perform at any time (see Fig. 1) ?

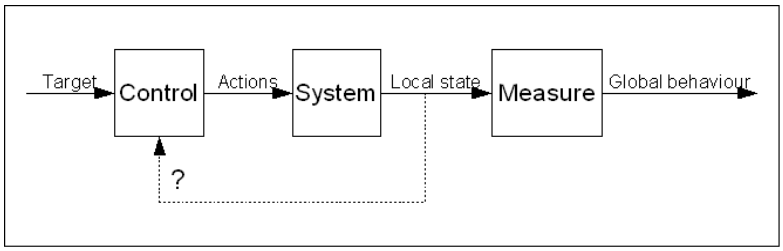

Fig. 1: Principle of control of a MAS. A control method chooses the actions to reach the target, and can take into account features of the system.

Considering the high number of simulations, the study has to be automated – and in particular the evaluation of the influence of each tested action within an experimental study. We do not want a human observer to check each simulation to identify the global behaviour. Thus we have to engineer an automatic measure to detect the global behaviour of the MAS from its local state at any time, at least to verify if the target behaviour is reached.

The actions are instant modifications of the system that have an influence on its behaviour. They can be modifications of the environment – by changing its attributes like its size or by introducing local perturbations like adding obstacles – or modifications of the agents – by modifying their number, adding luring agents or changing their characteristics: size, local behavioural rules, etc. The details of the actions depend on the system itself. Still, the actions that are made available to the control system must be chosen.

Once the behaviour and the actions have been defined, the question of the control itself remains: an action to perform has to be chosen in order to reach the target behaviour. The controller must decide which features of the system to take into account in that choice, and engineer a method to learn the good action to perform given these features.

3 Approaches of Control in Multi-Agent Systems

We expose in this section different approaches found in other works to the problems of controlling a MAS or guaranteeing a global behaviour. We discuss the proposed answers to the difficulties raised in the former section. We can split these approaches into two classes: static and dynamic ones. Static solutions do not take the evolution of the system into consideration while they control it (no loop in Fig. 1), whereas dynamic solutions perform an on-line control depending on the current state of the system.

3.1 Static Solutions

One way to make the system converge to the target is to set its parameters so as to optimise a convergence criterion. The principle of parameter setting is so to find optimal constant values for the controlled parameters of the system, typically with the use of a metaheuristic performed on the parameter space [7, 8]. We have to emphasise that the only possible actions when using parameter setting are the modification of parameter values, which restricts our definition of controlling a MAS.

A metaheuristic [9] is an algorithm that rules the exploration of a space by testing values of the space. The rules determine the values to choose and try to focus on the relevant areas in spite of a part of random decision, striving to balance the exploration of the space and the exploitation of promising results. So the parameter setting is essentially simulation-based and respects the experimental necessity specified in §2.1.

Parameter setting in the domain of MAS is not necessarily used to control the global behaviour, but any feature of the system (for instance [10]), so the question of characterising the global behaviour is not handled by this approach.

However, much work has been done for the question of reducing the simulation time: [10] proposes to use *equation-free* tools to partially predict the result of a simulation and then speed up each simulation, [11] proposes to reduce the actions space by limiting its dimensionality in a *divide and conquer* way, and many works [12-16] reduce the number of simulations thanks to a principle called *dynamical design of experiments* [6]. The principle consists in exploring the actions space in a non-homogeneous way, focusing on its relevant parts, by contrasting the exploration of the space and the exploitation of the best results found. The distinction can be made with regard to the space areas (this is the point of metaheuristic approaches [12, 13] and of some other works in MAS domain [14, 15]), but also to the quality of the estimation of the results in each point of the space [16].

Eventually, the main limitation of this approach is its static, off-line nature. It gives an optimised solution, but when the parameter values are set, there is no way to decide how to change them, whether the system reaches the target or not. If the MAS undergoes perturbations, that is if the control frame is different from the learning frame, the accurately optimised solution is not relevant anymore.

3.2 Dynamical Approaches

A dynamical control involves the use of information on the system to choose on-line the action to perform.

A dynamical approach is proposed in [17] and [18]. A morphological description of the global behaviour is proposed to characterise it. A control solution is given which takes into account the current state to correct it. Still, the solution of control is purely heuristic: the controller has to explain the system what action to perform in each state, but no method is proposed to determine the actions in a general case, especially if the link between the actions and the global behaviour is hard to determine.

The same remark can be applied to the AMAS theory [19] and the works based on it [20]: this method asks a human to determine the non-cooperative situations and decides what is to correct in order to make the MAS show a wanted behaviour. There is no notion of optimality and no automatism of the process.

Another dynamical approach, more usual, consists in considering the system as a decentralised Markov decision process (DEC-MDP) [21] and to apply reinforcement learning tools. But in that case, the only possible actions are at the agent level: each agent chooses what local action to perform. Thus actions as adding an obstacle or changing a global parameter are not handled. Furthermore, the complexity of a DEC-MDP problem soars with the number of agents and actions [22].

4 Proposition of a Dynamical Solution

We propose an on-line method to control a MAS using reinforcement learning. First the actions to perform are learned thanks to an experimental approach, then the best actions are chosen during the control phase. It must be noticed that if the MAS models another deployed system, only the learning phase is applied on the first one, and the resulting knowledge can be used to control the "real" system.

The drawback of the dynamicity is that it involves an observation of the controlled system to choose an action. Therefore, the observability of the controlled system must be discussed: if it is observable enough to compute the states described below then the proposition can be used, otherwise an approximation of the states must be computed.

The actions decision is dynamically taken, depending on features of the system: a distinction is made between different states of the MAS, letting us learn what to do in each of them to reach the desired behaviour. In this section, we explain which states to choose and then how to determine the actions to perform, thanks to reinforcement learning tools.

4.1 States Description

In order to choose a states description, the level of these states must be decided first. The local states can easily be defined by describing the exact disposition of the elements of the system – typically the position, speed and internal state of each agent

and each object in the environment. The global states represent the behaviour of the whole system.

Since this paper shows a first step in a larger study, we choose the simplest states description to show the utility of the method, that is, global states. Only the behaviour of the MAS is considered in the actions decision, as represented in Fig. 2, which can be compared to Fig. 1. Indeed, there are few possible behaviours compared to the number of sharp local states, so a global states space is easier to explore exhaustively.

The utility of each action in each state will be estimated thanks to many experiments for this couple {state, action}. The decrease of the number of states leads to a lower complexity of an algorithm based on this exploration.

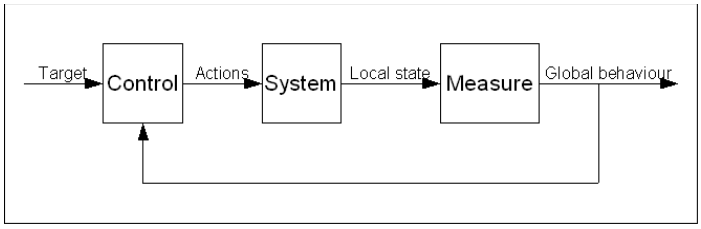

Fig. 2: Dynamical control using global information. We propose to consider the global behaviour in the action decision.

Furthermore, the dynamics of the MAS is observed at the global level, so the choice of states describing the global stable structures of the system is likely to give a good representation of this dynamics. That justifies the use of global states.

4.2 Short Presentation of Reinforcement Learning

All the following refers to [23]. A Markov decision process (MDP) is defined by a quadruple <S, A, T, R> where S is a set of states, A a set of actions, T a transition function that gives the probability to reach a state $s' \in S$ from a state $s \in S$ when $a \in A$ is performed (such as $T(s,a,s') = Pr\{s_{t+1}=s' \mid s_t=s, a_t=a\}$ at any time t), and R a reward function of each transition (such as $R(s,a,s') = E\{r_{t+1} \mid s_t=s, a_t=a, s_{t+1}=s'\}$ where E denotes the expected value).

The transition function T verifies the Markov property that establishes that the probability to reach a given state for a given action only depends on the current state, not on previous states of the system.

The goal of reinforcement learning (RL) is to find a near-optimal policy that defines which action to perform in each state so as to maximize the return (a function that gives the expected reward, parameterised by a value γ, called the discount rate). The policy can be stochastic when it gives probabilities over the actions to perform in each state.

The base of RL is dynamic programming: in order to evaluate the return of an action *a* performed in a state *S*, we try this action, which sends the system in a new state *S'*, and we use the current evaluation of the policy on *S'* to modify the evaluation of *a* in state *S*. Very efficient algorithms exist when T and R are known.

Otherwise, the functions T and R are experimentally estimated, that is the goal of Monte Carlo methods for instance. The combination of dynamic programming and Monte Carlo methods leads to temporal-difference (TD) learning which allows estimating T and R while the optimal policy is learned.

There are two classical algorithms of TD learning: Sarsa and Q-learning [23]. The difference is that Sarsa is an on-policy algorithm which evaluates the policy used to explore the action space in each state, while Q-learning is off-policy and learns an optimal policy different from the one used to explore the action space. Typically, the Q-learning can be used to find a deterministic policy and Sarsa a stochastic one.

4.3 Application to MAS Control

We formulate the hypothesis that our global state model verifies the Markov property, so as to apply these tools on it, even if we know this is just an approximation. Since the evolution of the system is unpredictable, the past global states are insignificant. We presume that in two different situations identified as the same state, the evolution is likely to be the same whatever action is performed.

The probability to reach a state *S'* by doing the action *a* in the state *S* is denoted *P(S, a, S')*. It can be approximated by the proportion of local states in the global behaviour *S* that stabilise in the global behaviour *S'* when an action *a* is performed (cf Fig. 3). This is our transition function. We define the rewards as follow: when the MAS reach the target global behaviour, the MDP receives a positive reward (namely 1), and 0 otherwise. So we have a MDP equivalent to our control problem.

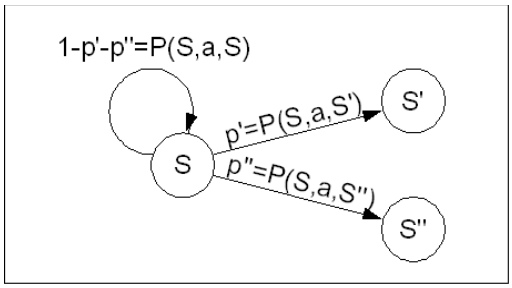

Fig. 3: Transition function. Transitions from a state *S*, for an action *a*, in a 3-states graph.

We would rather learn a stochastic policy to always allow an exit from a global state, since there is a risk that a deemed good action in a global state S keeps the system in S in particular situations. Although the Markov property is a correct approximation, it is likely to be wrong, and the aggregation of local states in a same global behaviour could be a too big approximation.

If we analyse the solution of reinforcement learning with regard to the specificities of MAS control problem, we notice that it is an effective experimental solution and a first step to the reduction of simulation time. RL implements a dynamical design of experiment on the estimation of the results triggered by the actions. In a given state, an action *a* is tested according to the exploration policy which takes into account the

estimated reward associated to *a*. Therefore, the better an action is, the higher the probability to test it is. In other words, the actions that seem the most promising have the best estimation, and we do not lose time with the estimation of irrelevant actions.

We wish to underline that in such a representation, an action is not necessarily a quantitative modification of a parameter value like in parameter setting, but can also be any qualitative change of the system, like adding an obstacle.

5 Implementation on a Pedestrians Model

We expose in this section an application of our proposition on a toy example that models pedestrians. The goal of this part is to illustrate how our approach can be applied and to demonstrate its feasibility. The three steps of our method are developed. The first step gives a solution to the global behaviour characterisation issue (for this system only), in the second step we choose the action means, and in the last step we present how to use RL tools to learn a control policy.

5.1 Application Example

Our application example is a MAS that roughly models pedestrians walking in a circular corridor. Realism of the model is not relevant here, we just need a system complete enough to apply the proposition. Agents are leaded by a sum of forces (like boids [24]), which come from their own goals and from the repulsion with other agents and with the walls. This is a reactive multi-agent system, with many parameters to study, already seen in other works [25].

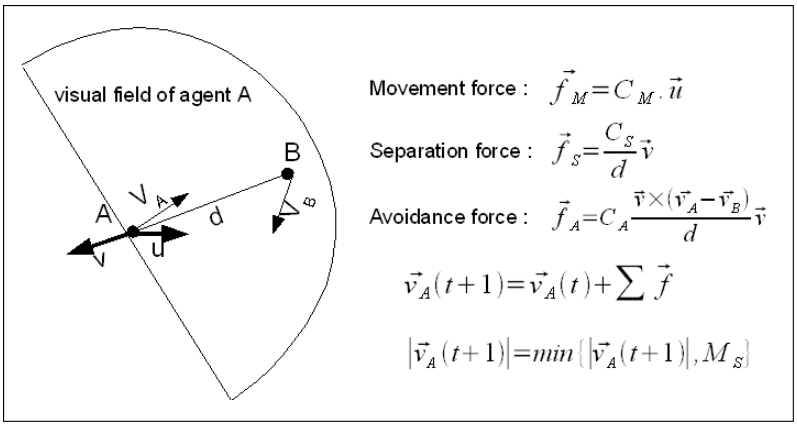

Fig. 4: Forces in the pedestrians system. Computing of the new speed of an agent A, that has a single agent B in its visual field. We assimilate the sum of forces to the acceleration because the mass of the agent can be considered in the coefficients Cm, Cs and Ca without adding a new parameter.

Specifically, agents have a dimension (they cannot overlap) and a visual field directed by their current speed. Each agent has a predilection direction in the corridor and undergoes a force in that direction proportional to a coefficient of movement *Cm*. When an agent has an obstacle (other agent or wall of the corridor) in his visual field, it undergoes an opposite force proportional to a coefficient of separation *Cs*. If the agent in the visual field goes in the opposite direction, a third force occurs, proportional to a coefficient of avoidance *Ca*. The sum of these forces modifies the agent speed according to its inertia. The speed norm is bounded by a maximum speed *Ms* (Fig 4). Finally, the agent moves as far as possible in the direction of its new speed: until it reaches an obstacle or the speed norm.

When simulated, the system shows up emergent groups of agents: lines of same direction agents and blocks of opposite agents (see Figure 5).

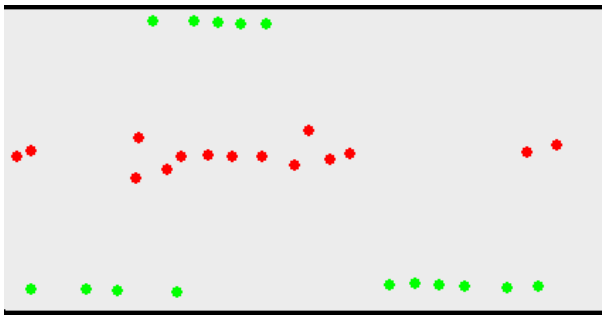

Fig. 5: Emergent structures. Three lines of agents emerging in the pedestrians system. Red (dark) agents go to the right and green (light-colored) agents to the left.

5.2 Global Behaviour Characterisation

The lines and blocks of agents are emergent structures of the system. Their arrangements define a global behaviour. But the description of the arrangements which characterise the global behaviour must be decided.

The global behaviour must be specified with regard to the goal of the controller. Even for such a simple system, several descriptions can be chosen for the global behaviour. If the controller's goal is to avoid blocks in the system, the global behaviour may be the presence or absence of blocks as structures involving agents, or the mean speed of each agent on a given time.

For our study, we consider that the target is to obtain a certain number of lines and a certain number of blocks: the behaviour is the number of each kind of group (lines and blocks), for instance one block and two lines.

Now, we have to find an automatic characterisation to determine the behaviour. Whatever the description, the point is to identify the emergent structures. In our case, this is a clustering problem, with an unknown number of clusters. We tried two solutions, the first using classical clustering tools and the second based on a decentralised clustering.

Many algorithms exist [26], which give a repartition of entities into clusters for a given number of clusters to find and a distance defined between these entities. For the

pedestrians, the distance can be a combination of the difference between the positions and between the speeds of two agents. The difficulty is to determine the number of clusters, so we implemented a solution proposed in [27]: we used a hierarchical clustering algorithm to create clustering solutions for any number of clusters, and we compared them with regard to the dispersion inside each cluster and between the clusters to find the optimal number of clusters. The solution gives good results but lacks generalisation if the system is slightly modified: the distance could have to be redefined to match the new emergent structures.

Thus we propose a simple and partially decentralised method which can be generalised to some other systems. Here is the principle: we ask each agent A to remember which other agents are *often* in its visual field (*often* is to be defined, we took 70% of the 100 last steps of simulation), and we consider that these agents belong to the same cluster as A does. In a graphic representation, if the agents are vertices connected by an edge when they belong to the same cluster, the clusters are the strongly connected components of the graph. This allows us to determine the clusters in the system and gave empirically as good results as the former solution: when a global behaviour is unambiguously observed, the measure identifies the same behaviour.

5.3 Control Actions

Control actions can be distinguished between environmental and behavioural actions. The first ones consist in modifying the environment, like its size or its structure (obstacles), in order to make the agents that interact with these changes react. Even if they are centralised, these actions do not counteract the distribution of the system. The second kind of actions are modifications of the behaviour of an agent or a group of agents. In our study, we restricted the possible actions to these modifications of individual behaviours.

Since we wish to compare in a first time our proposition to parameter setting solution, we have to define a problem solvable by the latter: the only actions we take into consideration are modifications of parameter values (an action is the decision of the value of each controllable parameter). The parameters we choose to use are the coefficients Cm, Cs and Ca, and the maximum speed Ms (instead of, for instance, the number of agents or the environment size).

The modification of the value of an individual parameter is equivalent to change the behaviour of the agent. In that way, the proposition is close to the notion of adjustable autonomy [28], except that each agent has only one level of autonomy: it applies a policy chosen by the control system. The agent is not aware of the behaviour of the whole system, especially the target behaviour.

5.4 Model of the Dynamics of the MAS

Given the states and the control actions discussed before, the dynamics of the MAS could be represented by the Figure 6. But this is just a formal model, which shows the hypothesis that the Markov property stands at this level. The only useful information

to control the MAS is to know which action to perform in each state. This is the goal of the RL algorithm presented in the next section.

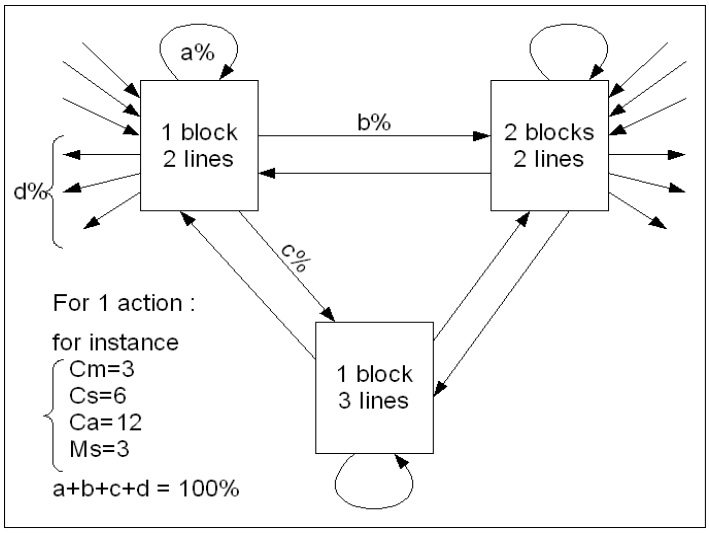

Fig. 6: Representation of the model of the global dynamics. The states are the global behaviours. There is a different graph for each possible action. The probability to escape from a state to another by performing an action is equal to the proportion of observed transitions between the two same states when this action is performed.

5.5 Implementation and Algorithmic Choices

We summarise here the decisions we made in the implementation of our control solution. In a MDP view, the transition function of our system is unknown, so we chose to use a TD-learning algorithm to evaluate the influence of the actions and achieve the control of the system.

We saw in §4.3 why to choose a stochastic policy that could avoid the MAS to be in an absorbing state different from the target by repeating a wrong action. Many stochastic policies can be imagined, from the simple ε-greedy to Boltzmann policy for the most classical ones [23]. They differ on the choice of under-optimal actions, depending or not on how these actions are close to the optimal one. As our intent is just to achieve a better control than parameter setting allows (cf §6.2), not to create an optimal control model, we limit this study to ε-greedy policies.

Since the chosen policy is a stochastic one, we use an on-policy algorithm, namely the Sarsa algorithm, to learn this policy. Finally, to compute the return of an action in a given state, we use the classical value $\gamma=0,9$ for the discount rate.

The sarsa algorithm only stores the expected reward value for each action in each states Q(s,a), without explicitly computing the model. The complexity of this algorithm mainly depends on the number of values Q(s,a) to estimate. Since it gives a better estimation of the best actions, the number of actions is less relevant than the number of states for the complexity.

We give below the algorithms used to learn the policy and to control the MAS, without the details of the sarsa algorithm. Two limits are used in order to avoid to indefinitely wait for a simulation to stop: a maximum number of actions denoted k, and a maximum number of steps of simulation before a new state is identified.

A particular state s0 is added, where no behaviour is observable. In a simulation, if the number of steps is too high before a stabilised behaviour is identified, we consider that s0 is reached. The initial situation of a simulation is a random distribution of the agents in the environment, and the initial state is s0.

Learning algorithm

```
∀ state, ∀ action
  Q(state, action) ← 0
Repeat (nbSimulations)
  MAS.initialise() // random positions for instance
  S1 ← MAS.getCurrentState()
  // try to reach the target in less than k actions
  nbActions ← 0
  While (nbActions<k & S1 ≠ target)
    // choice of an action
    action ← ε-greedy(Q(S1, a))
    MAS.apply(action)
    nbActions++
    nbSteps ← 0
    // let the MAS stabilise in less than a maximum number of steps
    repeat
      MAS.simulate()
      nbSteps++
    until (MAS.isCurrentStateIdentified()
           OR nbSteps = limit) // the current state is then s0
    // update Q-values with sarsa
    S2 ← MAS.getCurrentState()
    sarsa(Q(S1,action), S2)
    S1 ← S2
```

Control algorithm

```
MAS.initialise()
// try to reach the target
Repeat forever
  currentState ← MAS.getCurrentState()
  // choice of an action
  action ← ε-greedy(Q(currentState, a))
  MAS.apply(action)
  nbSteps ← 0
  // search for a stable state
  repeat
    MAS.simulate()
    nbSteps++
  until (MAS.isCurrentStateIdentified()
         OR nbSteps = limit)
```

6 Experimental Comparison of Control Solutions

The goal of this section is to evaluate and compare the performances of three different control solutions: a parameter setting based method, our reinforcement learning proposition and a reference control method based on random actions. We first present two scenarios corresponding to the control problems on which the methods are applied. Then we present the performance measures used to evaluate them. The control methods and their implementation are clarified in a third sub-section. Finally we expose the experimental results and compare them.

6.1 Scenarios

Two control problems – or scenarios – have been tested, each one involving a target behaviour and possible actions. In the first problem, the target behaviour to reach is one block and two lines of agents. We control the agents thanks to two parameters: the coefficients of movement Cm and separation Cs (§5.1). These two parameters can take five values each, so that we have only 25 possible actions.

In the second problem, the target behaviour presents two lines and no block, and the controlled parameters are Cm, Cs, and the maximum speed of the agents Ms. The last parameter can take 5 values too, bringing the number of different actions to 125.

The parameters are discrete because it is simpler to solve in a first study, and it is enough to prove the utility of reinforcement learning tools. RL algorithms exist in continuous spaces (of states and especially of actions [29]) and could be used if necessary for continuous parameters.

6.2 Performance Measures

A control method is evaluated with regard to the adequacy between the target and the behaviour of the system obtained using it. The evaluation is statistical and involves two aspects: the ability to reach the reference and the difficulty or the time necessary to reach it. The first aspect can be approximated by the proportion π of simulations that reach the reference when the method is used to control the system, and the second by the average number ν of actions needed before to reach the reference.

A simulation begins by setting the MAS in a random local state. The controller then decides what action to realise, and the MAS is let run with the specified parameters until a global behaviour is identified by the measure. This step is repeated until a stop criterion: either the target behaviour is reached or a maximum number of steps *k* has been performed and we consider that the MAS will never reach the target.

We took k=50, what can be experimentally justified by plotting the repartition of the simulations by their number of steps. For instance, the figure 7 represents the plotting of 300 simulations following the policy computed with the RL method on the first problem. We can see that a great majority of simulations that reach the reference do it after the first few actions. Hence there is no use to consider too many simulation steps after the 30^{th}: k=50 seems a good balance between a good approximation and the limitation of the simulation time.

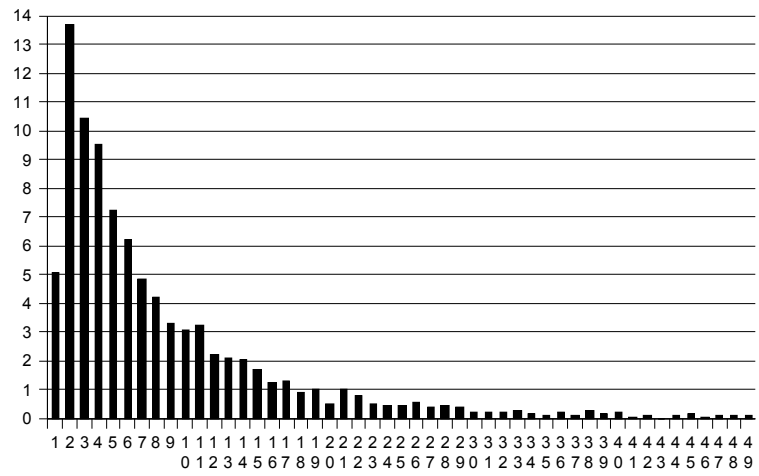

Fig. 7: Number of steps. Repartition in % of the simulations among the number of steps necessary before to reach the target: for instance, 3% of the simulations reached the reference after 10 steps.

A total of *n* simulations are realised, so as the proportion of convergence ⊓ is approximated by the number *m* of simulations that reached the reference divided by n. The greater n, the smaller the approximation and we give the radius of the statistical confidence interval at 90%: ⊓=80%±5 means that there are 90% chances that ⊓ is in [75, 85].

In order to estimate the average number of actions ν we only consider the number of steps of the m simulations that actually reached the reference, otherwise the simulations artificially stopped after k steps would bias the estimation.

6.3 Evaluated Control Methods

We compare three different methods to choose the actions to perform: a reference random choice, a parameter setting based method and our RL proposition.

The last one involves a learning phase where the Sarsa algorithm is used, and a control phase where the resulting policy is evaluated. In the learning algorithm, simulations are performed as in the control phase. We limited these simulations to k steps in order to learn the policy in the same conditions than its evaluation. The following results are given for 3000 learning simulations and n=300 evaluation simulations.

The parameter setting solution presents the two same phases of learning and control, but with its static nature the simulations are of only one step long. We learn the best parameter setting action by computing 500 simulations for each action. In order to get performance measures comparable to those of our proposition we slightly improve the evaluation simulations. Instead of using a static policy we alternate the optimal found action and a random action: first we apply the optimal parameter values recommended by the parameter setting, then we set random values, and so on, until

the target is reached or k actions have been performed. Hence we can compute estimations of Π and ν comparable to those found with the proposed method.

Finally, we evaluate a reference random method that needs no learning phase since it chooses a random action at each step of the evaluation simulations. It is used to verify that both the other methods get better performances than if there was no learning at all.

6.4 Experimental Results and Discussion

In the proposed method, we use a stochastic, ε-greedy policy, but the value of ε is to determine so as to optimise the performances. In this study, the optimisation is secondary, our goal is essentially to make sure that a dynamical control is applicable. Thus we do not try to find the best value of ε, but we give the results found for different values of ε.

Table 1. Evaluation of the control performances of the three methods (with different values of ε for our proposition) on two problems. The performance measures are the proportion of target reaching simulations Π, the radius of its confidence interval at 90%, and the average number of actions ν needed before reaching the target.

Method		**First problem**			**Second problem**		
		Π (%)	radius	ν	**Π (%)**	radius	ν
Random method		**68,7**	4,4	15	**23**	3,1	15,5
Parameter setting		**89**	3	11,8	**48**	3,7	7,1
Proposed method (RL)	ε = 10%	**89,6**	2,9	10,6	**66,6**	3,5	11,4
	ε = 20%	**91,3**	2,7	7,4			
	ε = 30%	**93,2**	2,4	7,8	**60,8**	3,6	12,8
	ε = 40%	**94,2**	2,2	8,2			
	ε = 50%	**93,3**	2,4	9,7			

All the control performances of the three methods of each scenario are summarised in Table 1. In the first problem, our proposition improves the reference performances, with a raise of 25 points for Π (69% to 94%), and twice less necessary actions (8 instead of 15). We observe an optimum close to ε = 40% whereas the classical value used for ε is 10%. Our solution is just slightly better than the parameter setting based method.

In the second problem, the improvement of the proportion Π is even greater, from 23% to more than 60%, and the parameter setting method only reaches the target in 48% of the situations. Still, this method gives the best improvement for the number of actions needed. We see that the parameter setting makes it possible to reach quickly the target in simple situations, but our method has the target reached in more situations. Here, a less stochastic policy (with the classical value of ε = 10%) gives better performances.

The second problem has more possible actions, hence possibly more actions which lead the system away from the target. We can see here that a random policy gives

performances dramatically lower than in the first problem. Thus a policy that chooses "good" actions triggers an improvement even better than it would do on a simpler problem. The difference between parameter setting and our proposition is explained by the fact that no action is "good" in every situations (states), but different best actions can be found for each state.

Finally, the proposed method proves to give a pragmatic and usable way to control MAS. An optimisation of the method can be done depending on the studied system and scenario, for instance by setting an optimal value for the parameter ε.

A difficulty has been ignored in the paper: the states of the MDP and the actions have been chosen *a priori*, but the main improvement of the proposition remains in this choice and its experimental evaluation. An experimental approach imposes to repeat this step with different choices and to compare the results. For this reason, in particular, the chosen discretisation of the actions is to question just as the parameters and the use of the global behaviour itself.

7 Conclusion

In this paper we propose a first step in an approach to create a pragmatic and experimental control method for a multi-agent system, especially a reactive one, and to evaluate the control performances. The method involves three main steps: the characterisation of the global behaviour and its automatic measurement, the selection of the possible control actions, and the determination of a policy that indicates the actions to perform in order to reach a target.

The originality of our proposition is to control the system on-line, by considering its current state, so that if it undergoes perturbations we can counteract them. Furthermore, the complexity of the approach does not directly depend on the number of agents since the states description is global and involves all the agents, and since the actions can be described at a global level – instead of building the behaviours of each agent. So the complexity does not suffer the scalability of the MAS.

The proposition can be pragmatically applied thanks to the limitation of the number of necessary simulations, by focusing on the relevant actions with RL tools and by reducing the size of the exploration space with the use of global information instead of local one. We show that the use of global information and actions improve the controllability – in terms of rate of convergence to the target – compared to classical parameter setting solution, and allows to build a dynamic control method.

As it is presented, the proposition can be applied to a simulated MAS. The use of the computed model to control of a concrete, distributed system is feasible under two conditions: the control actions can be applied to this system, and the observation is sufficient to measure the global states.

The actions chosen in our application system are just an example of what can be done. Tuning identical parameters of all the agents can seem quite centralised, but the actions space can be changed without challenging the proposition. For instance, environmental actions can be considered, or the model computed in this paper can be applied only to a part of the agents. In this last case, a further study is needed to check the robustness of the model if a part of the agents is not controlled.

Another interesting further development would be to estimate global states thanks to local information when the controlled system is not fully observable. Anyway the automatic measure is already an estimation of the behaviour, and uses few global knowledge. But a study is necessary to know how the control behaves when the states used during the control are less reliable than the states used to learn the policy.

We also think that the states description could be optimised and that local or semi-local information available in a distributed system could be enough to control the system in a decentralised way.

Eventually, as a future technical improvement of the approach we could optimise the learning phase and the tools used. For instance other policies (Boltzmann) and algorithms (Sarsa(λ), [23]) could improve the control performances and the learning speed. A constraint of the RL algorithms used is that they learn a policy for a single target and they do not allow to change the target without learning a new policy from the beginning. Other learning methods could be researched and applied to avoid this.

References

1. Ferber, J.: Multi-Agent System: An Introduction to Distributed Artificial Intelligence. Harlow: Addison Wesley Longman 1999
2. Wegner, P.: Why interaction is more powerful than algorithms. In: Communications of the ACM, Volume 40, pp. 80 – 91, New York (1997)
3. Edmonds, B.: Using the Experimental Method to Produce Reliable Self-Organised Systems. In Engineering Self Organising Sytems: Methodologies and Applications, Springer, 2004
4. Edmonds, B, Bryson, J.: The Insufficiency of Formal Design Methods - the necessity of an experimental approach for the understanding and control of complex MAS. In: Proceedings of the 3rd International Joint AAMAS'04, New York, ACM Press, pp 938-945 (2004)
5. De Wolf, T, Holvoet, T.: Towards a Methodology for Engineering Self-Organising Emergent Systems. In: Proceedings of SOAS 2005, Glasgow, Scotland, UK
6. Amblard, F.: Comprendre le fonctionnement de simulations sociales individus-centrées. Thèse de doctorat en Informatique, Université Clermont II (2003)
7. Sauter, J.A., Parunak, H.V.D., Brueckner, S., Matthews, R.: Tuning Synthetic Pheromones Withe Evolutionary Computing. In: Genetic and Evolutionary Computation Conference Workshop Program (GECCO 2001), San Fransisco, CA, 2001.
8. Sierra, C., Sabater, J., Agusti, J., Garcia, P.: Evolutionary Computation in MAS Design. In: Proceedings ECAI, pp188-192, 2002.
9. Dréo, J., Petrowski, A., Taillard, E., Siarry, P.: *Metaheuristics for Hard Optimization Methods and Case Studies*, Springer, 2006
10. De Wolf, T, Samaey, G, and Holvoet, T.: *Engineering Self-Organising Emergent Systems with Simulation-based Scientific Analysis*, Proceedings of the Third International Workshop on Engineering Self-Organising Applications (Brueckner, S. and Di Marzo Serugendo, G. and Hales, D. and Zambonelli, F., eds.), pp. 146-160, Utrecht, The Netherlands, 2005
11. Fehler, M., Klügl, F., Puppe, F.: Approaches for resolving the dilemma between model structure refinement and parameter calibration in agent-based simulations. AAMAS 2006:120-122
12. Calvez, B., Hutzler, G.: Automatic tuning of agent-based models using genetic algorithms. In: Proceedings of the 6th International Workshop MABS'05. Volume 3891 of lnai., 2005

13. Narzisi, G., Mysore, V., Bud Mishra, B.: Multi-objective evolutionary optimization of agent-based models: An application to emergency response planning. In Kovalerchuk, B., ed.: The IASTED International Conference on Computational Intelligence (CI 2006).
14. Klein, F., Bourjot, C., Chevrier, V., Approche expérimentale pour la compréhension des systèmes multi-agents réactifs, JFSMA06, Annecy (2006)
15. Calvez, B., Hutzler, G.: Ant Colony Systems and the Calibration of Multi-Agent Simulations: a New Approach. In: MA4CS'07 Satellite Workshop of ECCS 2007
16. Sven Brueckner, H. Van Dyke Parunak: Resource-aware exploration of the emergent dynamics of simulated systems. AAMAS 2003: 781-788
17. Campagne, JC, Cardon, A, Collomb, E, Nishida, T.: Using morphology to analyse and control a Multi-Agent system, an example. In: STAIRS ECAI'2004, August 2004
18. Campagne, JC, Cardon, A, Collomb, E, Nishida, T.: Massive Multi-Agent System Control, FAABS III 2004, IEEE Workshop on Formal Approaches on Agents-based Systems, LNCS 3228, NASA Goddard Space Center, Greenbelt MA, USA, April 2004.
19. Bernon, C., Camps, V., Gleizes, M.-P., Picard, G.: *Engineering Adaptive Multi-Agent Systems: the ADELFE Methodology*. In B. Henderson-Sellers and P. Giorgini (Eds.), Agent-Oriented Methodologies. Idea Group Pub, June 2005, pp.172-202
20. Bernon, C., Gleizes, M.-P., Picard, G.: Enhancing Self-Organising Emergent Systems Design with Simulation. In the 7th International Workshop on Egineering Societies in the Agent World 2006. Springer-Verlag, LNCS 4457, 284-299, 2007.
21. Thomas, V., Bourjot, C., Chevrier, V.: Interac-DEC-MDP: Towards the use of interactions in DEC-MDP. In Third International Joint Conference on Autonomous Agents and Multi-Agent Systems - AAMAS'04. (New York, USA). 2004. pp.1450-1451.
22. Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S.: The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research* 27(4):819–840. 2002
23. Sutton, R., Barto, A.: Reinforcement Learning: an introduction. MIT Press, Cambridge, MA (1998)
24. Craig Reynolds' boids: http://www.red3d.com/cwr/index.html
25. Lacroix, B., Mathieu, P., Picault, S.: Time and Space Management in Crowd Simulations. Proceedings of the European Simulation and Modelling Conference (ESM'06), pp. 315-320, Toulouse, France, 2006.
26. Jain, A.K., Murty, M.N., Flynn, P.J.: Data Clustering: A Review. ACM Computer Survey 31(3): 264-323 (1999)
27. Handl, J., Knowles, J.: Multiobjective clustering with automatic determination of the number of clusters. In: Technical Report TR-COMPSYSBIO-2004-02. UMIST, Manchester (2004)
28. Scerri, P., Pynadath, D.V., Tambe, M.: Towards Adjustable Autonomy for the Real World. J. Artif. Intell. Res. (JAIR) 17: 171-228 (2002)
29. Van Hasselt, H., Wiering, M.: Reinforcement Learning in Continuous Action Spaces. In: Proceedings of IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL), Honolulu, HI, USA, pp. 272-279 (2007)