Infection-Based Self-Configuration in Agent Societies

Norman Salazar
IIIA, Artificial Intelligence Research Institute
Campus UAB 08193
Bellatera, Spain
norman@iiia.csic.es

Juan A. Rodriguez-Aguilar
IIIA, Artificial Intelligence Research Institute
Campus UAB 08193
Bellatera, Spain
jar@iiia.csic.es

Josep Ll. Arcos
IIIA, Artificial Intelligence Research Institute
Campus UAB 08193
Bellatera, Spain
arcos@iiia.csic.es

ABSTRACT

Norms have become a common mechanism to regulate the behavior of agents in multi-agent systems (MAS). However, establishing a stable set of norms is not trivial, particularly in dynamic environments, under changing (and unpredictable) conditions. We propose a computational model that facilitates agents in a MAS to collaboratively evolve their norms, reconfigure themselves, to adapt to changing conditions. Our approach borrows from the social contagion phenomenon to exploit the notion of *positive infection*: agents with *good behaviors* become infectious to spread their norms in the agent society. By combining infection and innovation, a mechanism allowing agents exploring new norms, our computational model helps MAS to continuously stabilize despite perturbations.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Algorithms,Experimentation

Keywords

Distributed evolutionary algorithm, norms' evolution, infection-based algorithm

1. INTRODUCTION

Norms have become a common mechanism to regulate the behavior of agents in multi-agent systems. However, establishing an adequate set of norms is not trivial. Norms exist to balance the agent's interests with respect to the society's, in such a way that each agent is able to pursue its individual goals without preventing other agents' goals. This becomes more complicated in dynamic environments where changes in the system can cause the set of norms to become unexpectedly obsolete. Therefore, methods for learning and/or evolving the existing norms are needed.

In societies, conventions result when members agree upon a specific behavior. Thus, a norm convention refers to a set of norms that has been established among the members of a society. The learning of norm conventions, usually referred to as either self-organization or emergence, has been studied from the social sciences' and multi-agent system's perspectives. One of the trends of thought in social studies is that norm conventions emerge by propagation or contagion, where social facilitation and imitation are key factors [5, 4].

From the MAS point of view, the studies in [12] and [11] show that norm emergence is possible. However, these works only analyze norm propagation, though norm innovation (the discovery of new norms) is a very important factor for the social evolution of societies. If we intend to establish norm conventions in dynamic environments, propagating norms may not be enough since it implies that at least someone in the society has the correct norms and this may not be the case. This is why a mechanism to produce new norms is also necessary. Alternatively, in [1] work has been done to optimize the norms with respect to the agent's utility function without taking into account conventions, by means of reinforcement learning. Nonetheless, reinforcement learning can also be used for social conventions as shown in [10].

Since it has been argued that norms follow an evolutionary process [2], evolutionary Algorithms (EA) have been employed to find norms or rules in agent societies. Nonetheless, they are usually applied either: (i) as a centralized process; or (ii) as an individual process for each agent. In the centralized approach, a global EA tries to find a set of norms that rules the behavior of all agents. This is achieved by maintaining multiple societies with different norm sets each, and applying the evolutionary process to them, (e.g. [6]).

This technique presents some disadvantages. First of all, it is an off-line process, i.e. the algorithm is ran for mock up systems and when the best norms are found they are hardwired to the agents. As the complexity of the domain grows, this approach becomes very slow because it requires multiple simulations at the same time. Furthermore, it is a top-down approach, whereas what we pursue is a bottom-up method (namely, agents learning norms by themselves).

An alternative approach is based on embedding an EA in each agent so that each agent employs some kind of reinforcement learning to find its appropriate behavior or policies. A study of different techniques can be found in [8]. These methodologies do not have an explicit propagation mechanism, though conventions still arise. Agents that interact frequently with each other must evolve policies that decrease conflicts among them. The problem with this approach is that the time required for all the agents to stabilize their EA may be too long, and then is also an off-line methodology.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

Therefore, observe that the approaches discussed so far are not suitable to help MAS reconfigure themselves, by adapting their norms, in highly dynamic environments, namely under changing (and unpredictable) conditions. In this paper we try to make headway in this matter by proposing a computational model that facilitates agents in a MAS to collaboratively evolve their norms, reconfigure themselves, to adapt to changing conditions.

Our approach largely borrows from the social contagion phenomenon [4]. However, we propose a different notion of infection, namely *positive infection*: agents with *good behaviors* that help to improve the social welfare become infectious to spread their norms. With agents constantly infecting each other, the MAS can reach a stable set of norms. And yet, the MAS may still lapse into stagnation. To prevent such undesired outcome, our model incorporates an *innovation* mechanism that allows agents exploring new norms though the MAS has already reached a stable state. By combining infection and innovation, our model manages to help the MAS to continuously stabilize in spite of perturbations. Our infection-based model is realized as a distributed evolutionary algorithm. We choose EA since the contagion process can be modeled as an evolutionary approach. Furthermore, we believe that selection through fitness is compatible with the idea in social sciences of *structure equality*[4], regarded as an important issue in social learning.

The paper is organized as follows. Section 2 formilizes our self-configuration problem. In section 3 we describes our proposal for an infection-based model. Section 4 presents empirical results obtained in a traffic domain. Finally, section 5 draws some conclusions and sets paths to future research.

2. THE PROBLEM. A FORMAL MODEL

Let Ag be a finite set of agents situated in some environment, S a set of environment states, Δ a finite set of actions the agents can take in the environment, and N a finite set of social norms that constrain the actions agents can take. At each time *t*, the agents in Ag can take actions, leading to a new environment state. Thus, function $\pi : S \times 2^{\Delta} \rightarrow S$ models the transition of the environment from one state to another as agents act. Finally, we consider that not all environment states are equally valuable. Hence, the need for a social welfare function $u : S \rightarrow \mathbb{R}$ that values the environment state resulting after the agents in Ag perform their actions. Putting all the elements above together, we propose the following definition of a *norm-aware multi-agent system*:

DEFINITION 1 (NORM-AWARE MULTIAGENT SYSTEM). *A norm-aware multi-agent system is represented as a six-tuple* $\langle Ag, S, \Delta, N, \pi, u \rangle$*, where* Ag *stands for a finite set of agents,* S *stands for a set of environment states,* Δ *is a set of agent actions,* N *stands for a finite set of norms,* $\pi : S \times 2^{\Delta} \rightarrow S$ *stands for the environment transition function, and* $u : S \rightarrow \mathbb{R}$ *is the social welfare function.*

Now we turn our attention to the agents within the multi-agent system. As mentioned above, agents are situated in some environment. Furthermore, agents are *social* because: (i) each agent can communicate with other agents within the multi-agent system; and (ii) each agent is aware of some social norms. Therefore, notice that agents perform *communicative* actions to exchange information and *environment* actions to change the environment state.

On the one hand, when deciding the next action to take, agents depart from their *local* perception of the environment and may take into account the social norms they are aware of. Each agent's social norms express what the agent currently believes is socially permitted. Nonetheless, agents individually decide whether to *accept*, follow, their social norms when deciding the next action to make. Whatever the case, we assume that each agent can value the result of performing actions following its social norms. In other words, agents can measure the utility of following their social norms.

On the other hand, agents must decide what to do with the information they receive from others as well as whether to share information and with whom. Regarding these communicative actions, we shall consider that agents only exchange information regarding the social norms they are aware of along with their valuations. Notice that we shall not assume that agents are connected according to any particular topology, unlike [11, 9]. Thus, as time goes by, agents may exchange information with different peers.

Finally, based on the result of the actions on the environment and the interactions with other agents, each agent must develop a *norm transition function* for changing his social norms to better act on the environment. Bundling all the agent features introduced so far, we propose the following formal definition of *norm-aware agent*.

DEFINITION 2 (NORM-AWARE AGENT). *Given a norm-aware multiagent system* $\langle Ag, S, \Delta, N, \pi, u \rangle$*, each agent* $ag_i \in Ag$ *is a norm-aware agent characterized by a tuple* $\langle N_i^t, S_i^t, \delta_i, u_i, \mathcal{I}_i^t, \mathcal{O}_i^t, \mathcal{T}_i \rangle$*, where:*

- $N_i^t \subseteq N$ *stands for the* social norms *it is aware of at time t;*
- S_i^t *is the agent's perception of the environment at time t, namely its perception of the states in* S*;*
- δ_i *stands for a decision function that allows an agent to choose the next action, in* Δ*, given its perception of the environment state* S_i^t *and the social norms it is aware of,* N_i^t*;*
- u_i *is a valuation function that stands for the individual welfare returning the value that an agent associates its state;*
- $\mathcal{I}_i^t : Ag \rightarrow 2^N \times \mathbb{R}$ *models the information received by the agent as to the valuation assigned by each agent to a given set of norms;*
- $\mathcal{O}_i^t : Ag \rightarrow 2^N \times \mathbb{R}$ *models the information the agent sends to the rest of agents conveying the valuation assigned by the agent to a given set of norms;*
- $\mathcal{T}_i : (2^N \times \mathbb{R})^n \rightarrow 2^N$ *is a norm transition function that allows an agent to change his social norms from the accepted norms an valuations of the rest of agents in the MAS. It returns as an output a new set of social norms.*

The distributed self-configuration problem (DSCP) is the problem of finding the norms for each agent in a norm-aware multi-agent system that maximize the social welfare and guarantee its convergence despite environmental changes or changes in the agent population, as formalized by the following definition:

DEFINITION 3 (DSCP). *Given a norm-aware multi-agent system* $\langle Ag, S, \Delta, N, \pi, u \rangle$*, the distributed self-configuration problem is that of learning the norm transition functions* $\mathcal{T}_1, \ldots, \mathcal{T}_n$ *of the agents in* Ag *that: (i) maximize the social welfare* u*; and (ii) guarantee the* continuous *convergence of the social welfare.*

3. AN INFECTION-BASED MODEL

We will start by analyzing what we require to solve the DSCP. This is done under the assumption that agents cooperate. Firstly, each agent autonomously chooses its actions based on its local observations of the environment state, and without the need for approval or guidance from a central authority. Secondly, and according to definition 3, the social welfare must be maximized. At

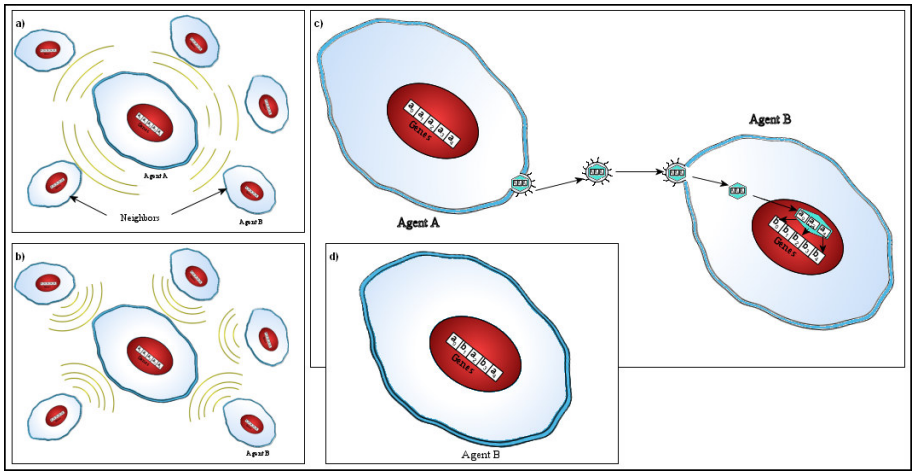

Figure 1: a) agent A probes its surrounding agents for information about their norms; b) agents respond to the probing; c) agent A selects and starts to infect agent B ; d) agent B changes its norms because of the infection

this aim, and in the spirit of the distributed nature of the problem, we must find a way of factoring the social welfare in terms of the agents' individual welfare. The factoring must grant that by maximizing the individual welfare, the social welfare is also maximized. It is important to notice that the mechanics of factoring are domain dependent, thus outside of the scope of this paper. And finally, an agent's norms should be able to adapt to expected or unexpected changes in the environment.

Real-life systems dynamically change with time, be it the number of participants, the environment properties, etc. When such changes occur the norms may become obsolete, invalidating the existing conventions and influencing the current welfare. Therefore, a mechanism to return the multi-agent system to a stable state whenever this happens is needed, namely a mechanism that guarantees *continuous stabilization*. Nonetheless, reaching a stable state is not enough, since it does not guarantee the welfare maximization. In account of this, we also require a *continuous convergence* scheme. We understand by continuous convergence the ability of re-accomplishing welfare maximization whenever the norms are required to evolve.

Social contagion [4] is a phenomenon that occurs in societies, and causes individuals to spread their behavior among them, akin to an infectious disease. We propose that by modeling social contagion into a norm-aware multi-agent system framework, stable NAMAS states can be reached, as endorsed by the work in [12]. However, likewise in societies, not any behavior is worth spreading because this may eventually lead to undesired phenomena (e.g. violent mobs).

In the context of our problem, we propose a different notion of infection, namely *positive infection*: agents with *good behaviors* that help improve the social welfare become infectious to spread their norms. A good behavior stands for a behavior that fulfills the agent's individual welfare. Since our aim is to spread the norms that are beneficial to the agent society, the stable state to be reached should have a positive effect on the social welfare, which is a nudge towards our maximization objective. Furthermore, we argue that an agent cannot infect its peer with all its norms because this would lead infected agents to pure imitation, and eventually to a prematurely stable state. Hence, infecting other agents with subsets of norms may have beneficial effects on the infected agent.

Figure 1 graphically shows the main steps of our infection-based model for the DSCP. Agent A probes its surroundings for neighboring agents. Close by agents respond to probe. Agent A then proceeds to select a neighbor (agent B) based on the responses. Agent B is then infected with some of agent A's genetic code.

With agents constantly infecting each other the continuous stabilization can be achieved. However, infection by itself can cause stagnation, halting the maximization of the social welfare. Hence, each agent must include an innovation mechanism to explore new norms even if the NAMAS is in a stable state. The combination of continuous stabilization and innovation makes the continuous convergence possible. Every time stabilization is reached, the innovation mechanism will try to break away from it. This shall only happen if another stable state with higher social welfare does exist. This will occur until there is no better stable state.

We propose a distributed *evolutionary algorithm*. We choose EA to emerge norms for two reasons: i) the evolution of the contagion process can be modeled as an evolutionary approach in such a way that innovation becomes a natural part of the process; and ii) the *take over* effect of the selection mechanism can guide the NAMAS population toward norm conventions. Since our algorithm borrows from *genetic algorithms* (GA), we recall the basic concepts of GA in the subsequent subsections while we progressively, elaborate our infection-based model.

3.1 Individuals and Agents

Individuals are the basis of GA and other evolutionary algorithms. They take inspiration on the genotype of biological organisms, whereas agents in a multi-agent system are akin to organisms in an environment. Therefore, we believe both ideas are complementary and can be fused into a single entity. This new entity, hereafter referred to as *evolutionary agent*, represents an agent that has genes encoding some of the agents' behavior (rules, norms, policies, etc). The exact encoding of the norms varies depending on the features of the problem.

Thus, an evolutionary agent *contains* an individual. As to the DSCP, an evolutionary agent is a norm-aware agent whose norms are encoded in a gene vector. In classic GA an individual represents a possible configuration of a *whole* system or problem. The fitness function is a measure of its potential as a solution for the system. However, with individuals representing norms within agents, a fitness function must measure an agent's performance instead of the system's. Not surprisingly, a norm-aware agent shall employ its

individual welfare as fitness function to measure its performance. Hence, with the individuals distributed and the performance measured agent-wise, the selection and the genetic operators cannot be applied as a global process.

At this aim, an evolutionary agent includes local selection, crossover and mutation functions so that each agent can decide by itself (with sexual operators, the partner(s) can also affect the decision), who to select, when and how to recombine and mutate.

3.2 Selection

In GAs (and other EA), the selection operator dictates the direction of the search process. It is a *centralized* operator that selects, out of a population of individuals, the best (fittest) individuals to which subsequently apply genetic operators. Thus, its application makes the fitness of the individuals impact on the evolution process [3]. The selection mechanism is also closely related to the genotypic diversity of the individuals in the genetic search, i.e. the number of different gene vectors in the population. Since this is one of the reasons why we chose an approach based on GA, it follows that selection is central to solve our problem.

Regarding a NAMAS, we must endow each agent with a selection mechanism that allows it to select one of its peer agents (figure 1.c), out of the agents it is capable of communicating with (figure 1.a), based on the received information (figure 1.b). We demand the agent selection mechanism to satisfy the following properties: (1) it must rank agents according to their fitness values (survival of the fittest); (2) it must select the best agents more frequently;(3) a weak gene vector should not overtake an already established one in the population.

1: $N \leftarrow neighborhood(ag)$
2: **foreach** $ag_i \in N$ **do**
3: $\quad \Upsilon(ag_i) \leftarrow \Phi(g) - \phi_{ag_i}$
4: **end for**
5: $\Upsilon(ag) \leftarrow 0$
6: $Sorted \leftarrow$ sort $\{ag\} \cup N$ order by Υ descending
7: **if** $(indexof(ag) \in Sorted > |Sorted| - k_{selection})$ **then**
8: $\quad$ mark myself *for-replacement*
9: **end if**
10: **if** $(\exists\, ag_i \in N \mid \Upsilon(ag_i) < 0)$ **then**
11: $\quad$ **foreach** $ag_i \in N$ **do**
12: $\quad\quad \Upsilon(ag_i) \leftarrow \Upsilon(ag_i) + ||min\{\Upsilon(ag_i)|ag_i \in N\}||$
13: $\quad$ **end for**
14: **end if**
15: $roulette \leftarrow$ distribution roulette from N using Υ
16: $ag' \leftarrow select$ from $roulette$

Algorithm 1: *Selection* operator of agent ag.

We propose to adapt a *roulette wheel* [3], one of the most common selection methods, to satisfy properties 1-3 listed above. Algorithm 1 outlines such adaptation. Given an agent ag, the mechanism selects another agent ag' from its current neighborhood. First, the neighbors' fitness are recomputed in terms of ag's fitness Φ (lines 2-4): subtracting from ag's fitness value ($\Phi(g)$, where g stands for ag's genes) its neighbors' (ϕ_{ag_i} is the fitness value received from agent ag_i) when minimizing (reversed when maximizing). The new values are stored into array Υ. If ag's fitness is one of the lower ones (according to the $k_{selection}$ parameter) then it marks itself for replacement (lines 7-8). This occurs because ag realizes that some neighbor(s) it has probed are better and it is worth being replaced (totally infected) by them. When neighboring agents with negative recalculated fitness exist (line 10), the fitness of all neighbors will be scaled (lines 11-13). Thereafter, the mechanism creates a distribution roulette using the fitness values in Υ [7](line 15). Finally, a neighbor agent can be selected for infection using the roulette (line 16).

3.3 Infection and Innovation

Once an agent has selected a peer agent, it is ready to engage in an infection process aimed at improving both agents' genes. Intuitively, an infection occurs by having an infectious agent injecting its genes into the selected agent. The infected agent shall employ the genes from the contagious agent to change its own genes. This is akin to a *retrovirus infection*. Once again we shall take inspiration on GA.

In GAs, the crossover, is a sexual operator that works under the assumption that by combining useful segments of parent individuals, the resulting new individuals are improved with respect to them. The classic crossover, called *single-cut crossover*, randomly selects a cut point in the parents' gene sequences and then exchanges the genes between the parents, resulting in two new individuals. Consider now a parent as a contagious agent and a child as the infected one. Unfortunately, a straightforward application of this crossover is not possible, because: (i) whereas in GAs the parent individuals are discarded, in our NAMAS setting we cannot discard agents; (ii) whereas in GAs non-selected individuals are lost for the next generation, agents uninfected must remain in the NAMAS; and (iii) whereas in GAs multiple children can be produced by the same individual on the same iteration, we cannot create new agents on a whim.

Therefore, we propose a mechanism that shall distinguish between two cases: (a) two-way infection, from the two agents, two new gene sequences are created to replace their current ones. With the replacement happening only if the agent has not participated in an infection during the current iteration; and (b) a one-way infection, where an agent marked for replacement during the selection replaces all its genes with the exact copy of the other agent's genes. Notice that each agent seeks to infect another agent and in turn can also be infected, though only once (per iteration).

Finally, to implement an innovation operator that helps agents escape local optima and explore new norms, we resort to the mutation operator employed by GAs. Although different operators have been proposed in the literature [3, 7], we use a basic one, consisting in the mutation (change) of genes with some probability.

3.4 Infection-Based Algorithm

With the components of our model defined, algorithm 2 stands for our distributed evolutionary infection-based algorithm.

Algorithm 2 considers that agents act over the environment a time step in the *incubation time* set (e.g. $T_{incubation} = \{100, 200, \ldots\}$) is reached. Then all agents locally start their evolutionary process. Thus, the algorithm repeatedly interleaves agents' actions and evolution. We introduce the notion of incubation time (line 3) because the effects of changes to an agent's genes may not be reflected immediately. Likewise virus, infections take some time to take effect. Thus agents require time to assess the influence of the contagion by acting over the environment during a period of time. Algorithm 2 runs agent-wise, hence agents can join in and leave the NAMAS at will.

Let us analyze the core flow of each agent (lines 5-8). Firstly, each agent, ag, evaluates itself using its individual welfare, i.e. its fitness function (Φ). Secondly, ag runs the *selection* operator with parameter $k_{selection}$ (algorithm 1) to select a peer, ag', to infect . Thirdly, with probability $p_{infection}$, ag tries to infect ag' (*infection*). Finally, the *mutation* operator changes some of the agent's genes with probability $p_{mutation}$. Although the core of each agent

may resemble a GA, recall that the *selection* and *infection* operators work by having agents exchanging genetic information on a distributed manner.

```
1: repeat
2:     shuffle the agents in the multi-agent system
3:     if (t ∈ T_incubation) then
4:         foreach ag ∈ NAMAS do
5:             ag.evaluate()
6:             ag' ← ag.selection()
7:             ag.infection(ag', p_infection)
8:             ag.mutation(p_mutation)
9:         end for
10:    end if
11: until NAMAS stops
```

Algorithm 2: Infection-based Algorithm.

In terms of social theory, our approach combines the concepts of imitation and innovation. The selection operation guides the search by selecting *whom to imitate* while the infection operator determines *how to imitate*. In our case an agent imitates by copying (parts of) other agents' norms to subsequently combine them with its own in hopes of getting better ones. Through selection and infection the NAMAS is deemed to reach norm conventions (a state where most of the agents have common norms). Even though this is a desired outcome it may be the case that the established convention(s) do(es) not maximize the social welfare, i.e. social stagnation is reached. To prevent this outcome we introduce the mutation operator as a purely innovative mechanism for creating new norms.

Finally, it is time to analyze: (i) whether the agents considered in algorithm 2 satisfy our theoretical notion of norm-aware agent; and (ii) the way our model tackles the DSCP. On the one hand, the communicative actions of a norm-aware agent are realized by exchanging genetic information with the neighbors. Furthermore norm transition function $\mathcal{T}_i$, results of combining operators selection, infection, and mutation. Thus, the DCSP is solved by letting evolutionary agents autonomously learn their norm transition functions. In next section we empirically illustrate how indeed our is valuable to tackle the DSCP in a particular scenario.

4. EMPIRICAL RESULTS

The DSCP that we study and experiment with is based on the "rules of the road" example, formerly presented in [10]. In particular, we consider an environment with multiple roads intersecting at multiple points, and populated by a high number of cars that move at their discretion along the roads. We chose this case because the cars' autonomy poses a problem: they may crash when two cars occupy the same space at the very same time. Crashes particularly occur around junctions. When two cars from neighboring roads arrive at the same time to a junction, they crash whenever none of them stops. Ideally, one of them should stop, while the other continues. However, both of them might stop, leading to problems for the upcoming cars, i.e. bottlenecks or more crashes. To summarize, our objective is for each car to learn the norms it must follow so that the number of crashes and bottlenecks is minimized. Notice though that we do not tackle the same problem as [10].

Our environment is represented as a torus-grid where each road can go from north to south or west to east. It is formed by a sequence of contiguous cells in the same direction. A *junction* is the area where two roads from different directions intersect. The central junction cell allows cars to change directions. Moreover, roads do not have dead ends (thanks to the torus-grid).

Time Interval	% Pert.	Time to Stability		Crashes		Waiting Time	
		IQM	Mdn	IQM	Mdn	IQM	Mdn
$[0, 20000]$	0	2087.5	2000	8.88	9	43.92	43
$[20001, 30000]$	20	50	50	10.48	10	54.32	51
$[30001, 40000]$	40	366	350	12.72	12	57.84	57
$[40001, 50000]$	60	756	700	11.52	12	49.76	46
$[50001, 60000]$	80	1397.6	1350	9.48	9	48.12	48

Table 1: Results for 50 simulations

*Note: IQM = InterQuartile Mean; Mdn = Median

Each car, a_i, corresponds to an evolutionary agent with two possible actions: *go* or *wait*. The environment states are represented by the cars' positions, the crashes, and the bottlenecks. Each car has a local (partial) view of the environment state. The norms take the form *if precondition then action*. Since crashes occur more frequently at junctions, the goal of the norms is to regulate the behavior of cars at junctions. Hence, when a car arrives at a junction at the same time that another car, it must learn whether to keep going or stop depending on the direction (norms N0 and N1 below). Moreover, cars need a norm to avoid crashes caused by sudden stops from other cars on the same road (norm N2 below).The parameterized norms are represented by the following templates:

N0$_i$ IF (car IN *junction*) AND (car-direction = *north-to-south*) AND (other-car AT *right*) THEN X_i^0

N1$_i$ IF (car IN *junction*) AND (car-direction = *west-to-east*) AND (other-car AT *left*) THEN X_i^1

N2$_i$ IF (car NOT IN *junction*) AND (other-car AT *immediate-front*) THEN X_i^2

where X_i^0, X_i^1, X_i^2 are variables over Δ.

Notice agents learn their norms by finding values for X_i^0,X_i^1,X_i^2. Thus, this instance of the DSCP consists of learning for each agent in the NAMAS, the norm transition function, $\mathcal{T}_i$, that leads to the values of X_{i_Δ} that maximize the social-welfare, u.

In our scenario the social-welfare increases when the number of crashes and waiting time diminishes (crashes are the priority). Therefore, by measuring the performance of each car in function of these terms we have that $u = \sum u_i$. The utility function that measures the above mentioned individual performance is given by:

$$u_i = \frac{1}{1 + \gamma_0 \cdot crashes + \gamma_1 \cdot waiting} \quad (1)$$

where γ_0 and γ_1 are penalty factors and must be positive numbers.

Finally, each agent can request to its neighboring agents their current performance, for selection purposes, as well as their norms, for infection purposes.

4.1 Experiments & Results

We designed our experiments to verify three hypotheses: whether our computational model 1) allows NAMAS to reach stable sets of norms; 2) requires all operators for the norms to converge; and 3) allows NAMAS to continuously stabilize and converge in spite of perturbations.

Each *experiment* consists of 50 discrete event simulations, each one up to 60000 ticks. A simulation runs a randomly created environment consisting of a 50 x 50 torus-grid, 21 roads, 48 junctions and a population of 100 car agents with some initial norms. At each time step, each agent can perform an action based on its current norms. Every 50 ticks (incubation time in alg. 2) the agents'

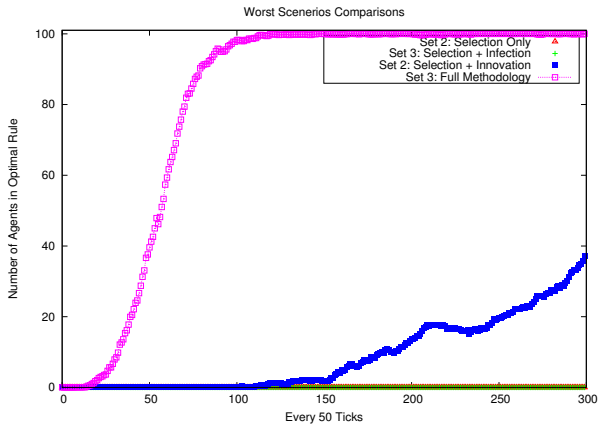

Figure 2: Worst case for each of the operators combinations.

performances are measured and the evolutionary process (alg. 2) starts within each agent. All changes to the genes become effective after all agents completed their evolutionary process.

We set parameters as follows: i) for algorithm 2 $p_{infection} = 0.90$ and $p_{mutation} = 0.0003$ in all agents; and ii) utility function's $\gamma_0 = 50$ and $\gamma_1 = 5$ to put more pressure on reducing the crashes, while preventing traffic jams.

For this particular problem we know beforehand that 4 attractors exist: two optimal (*go-wait-go* & *wait-go-go*) and two sub-optimal ((*go-wait-wait* & *wait-go-wait*)).

Hypothesis 1. We consider that a NAMAS reaches a stable state when at least 80% of its agents settle to the same norms so that no other norms can upstage them. Thus, we measured from the simulations the time (ticks) to reach stability and the quality of norms (average crashes and waiting time per incubation time). The first row of Table 1 shows the aggregated measures for an experiment that ran for 20000 ticks. We observe that the NAMAS readily reaches stability ($\sim$2000 ticks) and the quality of norms correspond to norms in the optimal attractors.

Hypothesis 2. Now we run experiments that set up cars differently by combining a choice of operators with a choice of initial norms. We draw the initial norms out of the sets in table 2: 1) all possible norms; 2) attractor-free norms; and 3) norms whose combination does not create attractors. We draw the operators from the following combinations: (a) *selection*, (innately infects the worst neighboring agent); (b) *selection + infection*, (has the proper infection operator); (c) *selection + mutation*; (d) *all operators*. Selection appears in all cases because it is a fundamental part of the model.

We measured the number of agents that converge to an optimal attractor per tick (the most dominant attractor per simulation). Figure 2 shows the settings that lead to the worst results for each combination of operators. We observe that: (i) *all operators* helps reach stability around 3500 ticks (close to the results above); (ii) *selection + mutation* deceivingly seems to converge to around 40 agents (this is an artifact of the aggregation because sometimes the NAMAS reaches a stable state (> 80 agents) whereas it does quite poorly (< 10 agents) others); (iii) as to *selection+infection*, not a single agent can obtain the desired norms because it is impossible to create an optimal attractor through gene recombination; (iv) *selection*'s worst case, occurs whenever there is not an (sub-)optimal attractor in the population to take over the rest.

Initializing the norms from set 1 works the best for all the combinations and the performance difference between the four of them is minimal. Nevertheless, notice that for these cases a certain number of optimal attractors already exist in the populations. In more complex problems with bigger search spaces, the chances of a situation like this arising is potentially very low. Thus, our results confirm that using all the operators is the most reliable option for open NAMAS for which we cannot know the norms agents depart with.

Set 1	(go-go-go),(go-go-wait),**(go-wait-go)**,**(go-wait-wait)** **(wait-go-go)**,**(wait-go-wait)**,(wait-wait-go), (wait-wait-wait)
Set 2	(go-go-go),(go-go-wait),(wait-wait-go),(wait-wait-wait)
Set 3	(go-go-go),(go-go-wait) OR (wait-wait-go),(wait-wait-wait)

Table 2: Initial norms sets

Hypothesis 3. Now we introduce perturbations during the simulations by randomly changing the actions of each norm for a number of agents. We list the perturbations we employed in our simulations in the second column of Table 1. For instance, when reaching 20001 ticks, 20% of the agents had their norms changed. We measured the time to stability and the quality of norms as shown in rows 2-5 in Table 1. In all simulations of the experiment, the NAMAS converged to one of the optimal attractors. Figure 3 shows the results of a particular simulation. We observe that the *wait-go-go* attractor becomes stable early on. After each one of the first three perturbation, we observe a downwards spike. Nevertheless the NAMAS rapidly reaches the stable state afterwards while maintaining the same attractor. However, the last perturbation leads to an interesting phenomenon: a transition between attractors occur, but once again the NAMAS reaches a stable state. From this results we conclude that our approach is robust against perturbations.

5. CONCLUSIONS AND FUTURE WORK

In this paper we tackle the distributed self-configuration problem, namely the problem of finding the norms for each agent in a norm-aware multi-agent system that maximize the social welfare and guarantee its convergence despite environmental or agent population changes.

We have proposed a computational model that aids agents in a NAMAS to collaboratively evolve their norms, reconfigure themselves, to adapt to changing conditions. Our approach largely borrows from the social contagion phenomenon. However, we propose it as a positive notion: agents with *good behaviors* that help to improve the social welfare become infectious to spread their norms. Our infection-based model is computationally realized as a distributed evolutionary algorithm.

As a proof of concept we have selected a case study over which we formulated a DSCP. We showed that our infection-based model allowed the NAMAS to reach stable sets of norms, and to continuously stabilize in spite of heavy perturbations. To justify the need of the different operators, we showed that by removing some of them there are cases where no stabilization can be achieved. And yet, our approach shares drawbacks with other evolutionary algorithms: (i) the sensitivity to evolutionary parameters; and (ii) we cannot guarantee that the solution to which the system converges is always the best. We plan to tackle the first problem by resorting to the self-adaptation techniques common in EA.

In this paper we have studied how to create a global epidemic in a domain where agents can move around the whole environment. Inspired in the work of [9] we intend to carry out a similar study on unconnected or loosely connected populations, i.e. scale-free networks, small world networks, etc.

Finally, notice that cooperation among agents is an important component of our selection and infection mechanisms. We find interesting to explore how our model applies to scenarios where cooperation can be no longer assumed.

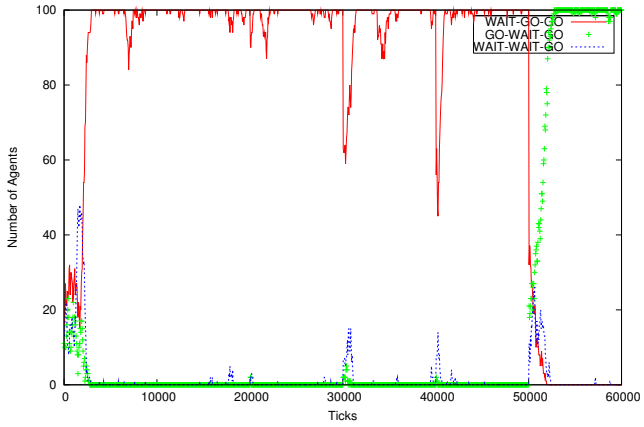

Figure 3: Transition from norm attractor to norm attractor.

6. ACKNOWLEDGMENTS

The authors thank CONACyT for the scholarship of the first author. This work was funded by IEA (TIN2006-15662-C02-01), OK (IST-4-027253-STP), eREP(EC-FP6-CIT5-28575) and Agreement Technologies (CONSOLIDER CSD2007-0022, INGENIO 2010).

7. REFERENCES

[1] S. Abdallah and V. Lesser. Multiagent reinforcement learning and self-organization in a network of agents. In *AAMAS 2007*, pages 172–179, 2007.

[2] R. Axelrod. An evolutionary approach to norms. *The American Political Science Review*, 80(4):1095–1111, 1986.

[3] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, 1996.

[4] R. Burt. Social contagion and innovation: Cohesion versus structural equivalence. *American J. of Sociology*, 92:1287–1335, 1987.

[5] R. Conte and M. Paolucci. Intelligent social learning. *Artificial Society and Social Simulation*, 4(1):1–23, 2001.

[6] I.-K. Jeong and J.-J. Lee. Evolving multi-agents using a self-organizing genetic algorithm. *Applied Mathematics and Computation*, 88:293–303, 1997.

[7] Z. Michalewicz. *Genetic Algorithms+Data Structures=Evolution Programs*. Springer Verlag, 3rd edition, 1996.

[8] D. Moriarty, A. Schultz, and J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Artificial Intelligence Research*, 11(1-1):241–276, 1999.

[9] J. Pujol, J. Delgado, R. Sangüesa, and A. Flache. The role of clustering on the emergence of efficient social conventions. In *IJCAI 2005*, pages 965–970, 2005.

[10] S. Sen and S. Airiau. Emergence of norms through social learning. In *IJCAI 2007*, pages 1507–1512, 2007.

[11] Y. Shoham and M. Tennenholtz. On the emergence of social conventions: Modeling, analysis, and simulations. *Artificial Intelligence*, 94(1-2):139–166, 1997.

[12] A. Walker and M. Wooldridge. Understanding the emergence of conventions in multi- agent systems. In *ICMAS 1995*, pages 384–389, 1995.