Cooperative Agent Model Instantiation to Collective Robotics

Gauthier Picard

IRIT, Université Paul Sabatier,
F-31062 Toulouse Cedex, France
picard@irit.fr
http://www.irit.fr/SMAC

Abstract. The general aim of our work is to provide tools, methods and models to adaptive multi-agent systems designers. These systems consist in several interacting agents and have to optimize problem solving in a dynamic environment. In this context, the ADELFE method, which is based on a self-organizing adaptive multi-agent system model, was developed. Cooperation is used as a local criterion to self-organize the collective in order to reach functional adequacy with the environment. One key stage during the design process is to instantiate a cooperative agent model that is an extension to classical reactive models in which cooperation subsumes any other nominal behavior. A sample implementation of the agent model in the collective robotics domain – resource transportation – will illustrate a discussion on the model.

1 Introduction

Self-organization in artificial systems promises to be an appropriate solution to overcome openness, flexibility and adaptiveness requirements in dynamical environments. Adaptive Multi-Agent Systems (or *AMAS*) paradigm proposes to use cooperation notion as the engine of self-organization mechanisms in order to make the system reach functional adequacy [1]. Therefore, designers of such agent societies must focus on the parts rather than the whole global system; i.e. *a priori* equipping parts with organization capabilities rather than organizing them. Implementations of such systems have already successfully solved complex problems such as flood forecasting with STAFF in which data, originated from sensors, self-organize to reach the right prevision function [2].

In this context, ADELFE method establishes an AMAS design process [3] which aims at guiding non-specialist engineers to develop MAS from A to Z. Besides notations (UML and A-UML) and tools (OpenTool[1]), ADELFE provides a cooperative agent model which has already been described in the previous ESAW edition [4]. Figure 1 describes this model. Unlike some other multi-agent engineering approaches which try to fit agent design with nature-inspired models,

[1] OpenTool is released by TNI-Valiosys (www.tni-valiosys.com).

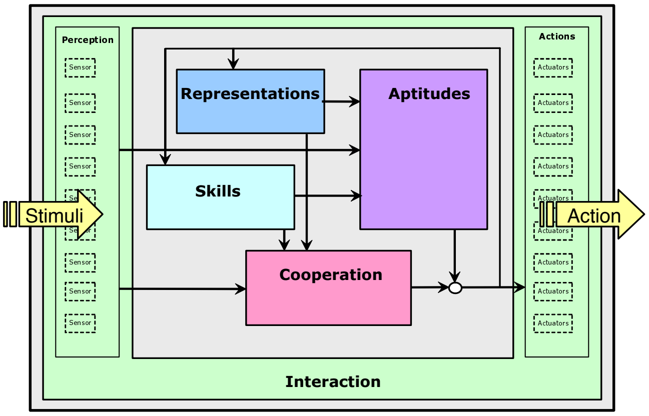


Fig. 1. The different modules of a cooperative agent and their dependencies. Aptitudes and Cooperation modules process in parallel during the decision phase to make the agent act. Cooperative behavior subsumes nominal behavior

such as ant behaviors, ADELFE lies on a theoretical notion which comes near to the social notion of *cooperation*. The challenge resides in specifying local cooperation rules that will lead agents to organizational changes and therefore to change the global function of the system.

Cooperative behavior will be defined in a proscriptive approach: "An agent is cooperative if it avoids non cooperative situations (or NCS)". The AMAS theory identifies several types of NCS, resulting from the analysis of the cooperation definition: an agent is cooperative if: (c_1) all perceived signals are understood without ambiguity *and* (c_2) the received information is useful for the agent's reasoning *and* (c_3) reasoning leads to useful actions toward other agents. Therefore, a NCS occurs when $\neg c_1 \vee \neg c_2 \vee \neg c_3$. We identify seven NCS subtypes that express these conditions: *incomprehension* (an input has no interpretation), *ambiguity* (an input has two or more interpretations), *incompetence* (the agent has no rule to process input), *unproductiveness* (the agent's reasoning do not lead to any conclusion), *concurrency* (two agents execute actions that lead to same conclusions), *conflict* (the agent's action put another agent out) and *uselessness* (the agent's action has no impact in its environment). The cooperative attitude of an agent must avoid all these NCS. Cooperative agent design focuses on NCS specification – like a kind of exception-oriented programming in which designers focus on exceptions.

This article aims at detailing the agent design in ADELFE and showing the optimization of a problem solving concerning resources transportation, described in section 2 realized by an cooperative agents society in which the global behavior presents emergent properties. For more information about the ADELFE process usage, see [3] or the paper in the same workshop about the Mechanical Synthesis Problem [5]. In section 3, the cooperative agent model is instantiated to solve the resource transportation problem. Section 4 studies different possible cooperative behaviors that can be assigned to agents. Some experiments have been done to compare these different solutions in section 5, which leads to a discussion in section 6. Finally, section 7 concludes on perspectives.

2 Resource Transportation Problem

The resource transportation problem is a classical task in Collective Robotics [6], and was proposed as a relevant benchmark for robotic systems by [7]. Robots must transport resources (boxes) as fast a possible from a zone A to a zone B, separated by a constrained environment. In our example, these zones are linked by two corridors too narrow for robots to cross one another side by side (cf. figure 2). This environment leads to a spatial interference problem, e.g. robots must share common resources: the corridors. Once engaged in a corridor, what must a robot do when facing another robot moving in the opposite sense? Spatial interference has been tackled by [8] in the case of robots circulating in corridors and having to cross narrow passages (doors). Their solution is to solve conflicts by aggressive competition (with explicit hierarchy), similarly to eco-resolution by [9]. [10] propose to solve such problems thanks to attraction-repulsion mechanisms based on altruistic behaviors triggering – a reverse vision of the eco-resolution. In our case, we expound a viewpoint halfway between the two firsts, in which robots are neither altruistic nor individualist and cannot directly communicate any information or intention. Moreover, no planifier system will anticipate trajectories because the use of planification in multi-robot domain remains inefficient, considering the high dynamics of a robot's environment.

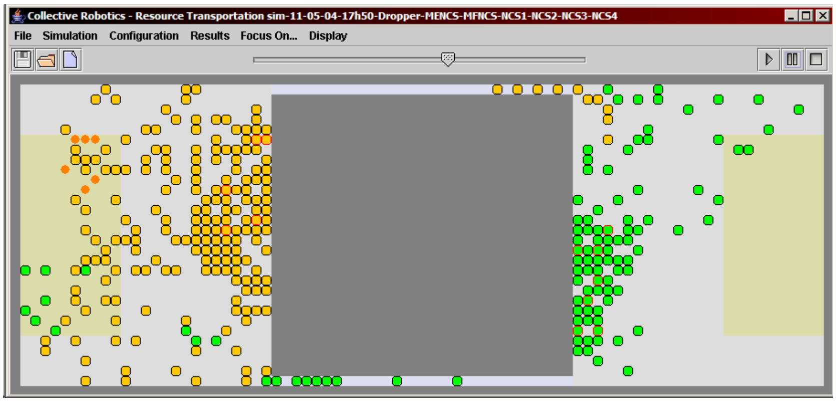

Collective Robotics - Resource Transportation sim-11-05-04-17h50-Dropper-MENCS-MFNCS-NCS1-NCS2-NCS3-NCS4
File
Simulation
Configuration
Results
Focus On...
Display

Fig. 2. The environment of the resource transportation problem is composed of: a claim room (at left), a laying room (at right) and two narrow corridors (at top and bottom). Robots pick boxes against the left wall of the claim room (claim zone) and drop them against the right wall of the laying room (laying zone)

3 Cooperative Model Instantiation

This section shows the instantiation – i.e. fulfilling each module – of the cooperative agent model in order to design robots able to realize the transportation task. This work appears in the ADELFE process in the *Design Work Definition*, and more precisely in the *Design Agents Activity* [3]. ADELFE process is an extension to the *Rational Unified Process* (RUP) and consists in four work definitions – specifically extended to agent oriented engineering – : preliminary requirements, final requirements, analysis and design. Requirements defines the environmental context of the system. Analysis identifies the agents within other object classes.

3.1 Modules Fulfilling

The *Perceptions Module* represents inputs for agents. Concerning robots, they can know positions of the two zones (claim and laying). Indeed, this paper only focuses on adaptation to a circulation problem rather than a foraging one, i.e. robots' task is not to find boxes but to transport them from a room to another. Here is a possible list of perceptions for transporter robots: position of the claim zone, position of the laying zone, a perception cone in which objects are differentiable (robot, box or wall), proximity sensors (forward, backward, left and right), a compass and the absolute spatial position. The environment is discretized as a grid whose cells represent atomic parts on which a robot, a box or a wall can be situated. The Perceptions Module also defined limit values of perceptions (e.g. 5 cells).

The *Actions Module* represents outputs of agents on their environment. Possible actions for transporter robots are: *rest*, *pick*, *drop*, *forward*, *backward*, *left* and *right*. Robots cannot drop boxes anywhere in the environment but only in the laying zone. They cannot communicate directly or drop land marks on the environment. In the case of social agents that are able to communicate, communication acts are specified in this module.

The *Skills Module* contains knowledge about the task the agent must perform. Skills enable robots to achieve their transportation goals. Therefore, a robot is able to calculate which objective it must achieve in terms of its current state: if it carries a box then it must go to the laying zone else it must reach the claim zone. As a function of its current goal, the Skills Module provides an action to process to achieve it. Robot's goals are: *reach claim zone* and *reach laying zone*. Moreover, robots have intrinsic physical characteristics such as their speed, the number of transportable boxes or the preference to move forward rather than backward – as ants have. Such preferences are called *reflex values*.

The *Representations Module* contains knowledge about the environment (physical or social). Representation a robot has on its environment is very limited. From its perceptions, it cannot identify a robot from another, but can know if it is carrying a box or not. It also can memorize its past absolute position, direction, goal and action.

The *Aptitudes Module* enables an agent to choose an action in terms of its perceptions, skills and representations. Concerning transporter robots, a design choice must be taken at this stage. In terms of the current goal, the Skills Module provides preferences on each action the robot may do. The Aptitudes Module chooses among these actions what will be the next action to reach the goal. Many decision functions can be considered; e.g. an arbitrary policy (the action having the highest preference is chosen) or a Monte Carlo method-based policy that is chosen for our example. Therefore, the Aptitudes Modules can be summed up in a Monte Carlo decision function on the preference vector (the list of action preferences for an agent) provided by the Skills Module. In the same manner, the *Cooperation Module* provides preference vectors in order to solve NCS described in section 4.

3.2 Action Choosing

At each time t, a robot has to choose between different actions that are proposed by the two decision modules (skills and cooperation). At time t, each action act_j of the robot r_i is evaluated. For each action, this value is calculated in terms of perceptions, representations and reflexes in the case of a nominal behavior:

$$V_{r_i}^{nomi}(act_j, t) = wp_{r_i}(act_j, t) + wm_{r_i}(act_j, t) + wr_{r_i}(act_j)$$

with:

- $V_{r_i}^{nomi}(act_j, t)$ represents the value for the action act_j at time t for the robot r_i,
- $wp_{r_i}(act_j, t)$ represents the calculated value in terms of perceptions,
- $wm_{r_i}(act_j, t)$ represents the calculated value in terms of memory,
- $wr_{r_i}(act_j, t)$ represents the calculated value in terms of reflexes.

As for aptitudes, an action preference vector is generated by the Cooperation Module: $V_{r_i}^{coop}(act_j, t)$. Once these values calculated by the two modules for each action of a robot, the vector on which the Monte Carlo drawing will process is a combination of the two vectors in which the cooperation vector subsumes the nominal vector:

$$V_{r_i}(t) = V_{r_i}^{nomi}(t) \prec V_{r_i}^{coop}(t)$$

3.3 Nominal Behavior

The nominal behavior is described with rules that modify the values in the V^{nomi} preference vector. This vector is obtained by adding values[2] from perceptions $(wp_{r_i}(act_j, t))$ and values from reflexes $(wr_{r_i}(act_j, t))$. The table 1 shows values to increase in the $wp_{r_i}(act_j, t)$ to achieve to two disjoint goals : *reach claim zone* $(\neg car)$ and *reach laying zone* (car).

[2] Memory is not necessary to process a nominal behavior.

Table 1. Specification of the nominal behavior in terms of perceptions

Perceptions	**Effects**
$\neg car \wedge cBox$	$\nearrow wp_{r_i}(pick, t)$
$\neg car \wedge \neg cBox \wedge sBox$	$\nearrow wp_{r_i}(forward, t)$
$\neg car \wedge \neg cBox \wedge \neg sBox \wedge \neg inCZ$	$\nearrow wp_{r_i}(< CZdir >, t)$
$\neg car \wedge \neg cBox \wedge \neg sBox \wedge inCZ$	$\nearrow wp_{r_i}(backward, t)$
	$\nearrow wp_{r_i}(forward, t)$
	$\nearrow wp_{r_i}(left, t)$
	$\nearrow wp_{r_i}(right, t)$
$car \wedge cLZ$	$\nearrow wp_{r_i}(drop, t)$
$car \wedge \neg cLZ$	$\nearrow wp_{r_i}(< LZdir >, t)$

- car: r_i is carrying a box;
- $cBox$: r_i is close a box;
- $sBox$: r_i is seeing a box;
- $inCZ$: r_i is in the claim zone;
- cLZ: r_i is close to laying zone;
- cLZ: r_i is close to laying zone;
- $< CZdir >$: the move to do to go to claim zone;
- $< LZdir >$: the move to do to go to laying zone;
- $\nearrow$: increasing.

Reflex values are static and also depend on perceptions – but only on the direction of the robot. As for ants, robots may prefer moving forward then backward [11]. For example, values for $wr_{r_i}(act_j, t)$ can be :

- $wr_{r_i}(forward, t) = 50$;
- $wr_{r_i}(left, t) = 10$;
- $wr_{r_i}(right, t) = 10$;
- $wr_{r_i}(backward, t) = 0$;

Thus, even if a goal leads a robot to a wall, the robot can move by side, as ants do to forage and to avoid dead end. Nevertheless, this mechanism is not sufficient to avoid deadlocks in long narrow corridors in which robots cannot cross. The goal is more influent than reflexes. As a consequence, we need to define cooperation rules to enable all robots to achieve their tasks without deadlock.

Finally, robots do not process their nominal next action from a memory. Therefore, $\forall j, wm_{r_i}(act_j, t) = 0$.

4 Cooperative Behaviors Study

In the previous section, the different modules of a robot and its components have been detailed, except the Cooperation Module. This section aims at discussing cooperation rules to establish in order to enable the multi-robot system to be in functional adequacy with its environment.

4.1 Cooperative Unblocking

Beyond two robots acting to transport boxes in a same environment, the nominal behavior cannot be adequate. Indeed, a robot owns skills to achieve its tasks, but not to work with other robots. In this very constrained environment, spatial interference zones appear. If two robots, a first one carrying a box and moving to the laying zone and a second one moving to the claim zone to pick a box, meet in a corridor, the circulation is blocked – because they cannot drop boxes outside the laying zone. Then, it is necessary to provide cooperative behaviors to robots. Two main NCS (non cooperative situations) can be reactively solved:

A robot is blocked. A robot r_1 cannot move forward because it is in front of a wall or another robot r_2 moving in the opposite sense[3]. In this case, if it is possible, r_1 must move to its sides (left or right). This corresponds to increasing values of the cooperative action vector related to side movements: $V_{r_1}^{coop}(t, right)$ and $V_{r_1}^{coop}(t, left)$. If r_1 cannot laterally move, two other solutions are openned. If r_2 has an antagonist goal, the robot which is the most distant from its goal will move backward (increasing $V_{r_i}^{coop}(t, backward)$) to free the way for the robot which is the closest to its goal (increasing $V_{r_i}^{coop}(t, forward)$ even if it may wait). If r_2 has the same goal than r_1, except if r_1 is followed by an antagonist robot or if r_1 moves away from its goal (visibly it moves to a risky[4] region), r_1 moves backward; else r_1 moves forward and r_2 moves backward.

A robot is returning. A robot r_1 is returning[5] as a consequence of a traffic blockage. If it is possible, r_1 moves to its sides (an is no more returning). Else, r_1 moves forward until it cannot continue or if encounters another robot r_2 which is returning and is closer to its goal than r_1. Table 2 sums up the behavior in this situation. If there is a line of robot, the first returning robot is seen by the second one that will return too. Therefore, the third one will return too and so on until there is no more obstacle.

These rules correspond to resource *conflict* (corridors) or *uselessness* when a robot must move backward and away from its goal. In the case of robots, situations will not be specified as incomprehension because robots are unable to communicate directly. These rules, which are simple to express, ensure that robots cannot block each other in corridors. But, this cooperation attitude only solves problem instantly, creating returning movement and then implies time loss to transport boxes.

4.2 Cooperative Anticipation

By taking into account the previous remark, it seems possible to specify cooperation rules to anticipate blockage situations in order to make the collective

[3] If r_2 moves in another direction than the opposite direction of r_1, it is not considered as blocking because it will not block the traffic anymore.

[4] It is risky in the sense it may occur a lot of non cooperative situations such as conflicts.

[5] A robot is considered as returning until it has no choice of side movements.

Table 2. Example of specification of the "*a robot is returning*" uselessness NCS

Condition	***Action***
$ret \wedge freeR$	$\nearrow V_{r_i}^{coop}(t, right)$
$ret \wedge freeL$	$\nearrow V_{r_i}^{coop}(t, left)$
$ret \wedge \neg(freeL \vee freeR) \wedge ant \wedge toGoal \wedge cGoal$	$\nearrow V_{r_i}^{coop}(t, backward)$
$ret \wedge \neg(freeL \vee freeR) \wedge ant \wedge toGoal \wedge \neg cGoal$	$\nearrow V_{r_i}^{coop}(t, forward)$
$ret \wedge \neg(freeL \vee freeR) \wedge ant \wedge \neg toGoal$	$\nearrow V_{r_i}^{coop}(t, backward)$
$ret \wedge \neg(freeL \vee freeR) \wedge \neg ant$	$\nearrow V_{r_i}^{coop}(t, forward)$

With:
- ret: r_i is returning;
- $freeR$: right cell is free;
- $freeL$: left cell is free;
- ant: in front of an antinomic robot;
- $toGoal$: r_i is moving to goal;
- $cGoal$: r_i is closer to its goal than its opposite one;
- $\nearrow$: increasing.

more efficient. We call this *optimisation* cooperation rules. Previous rules enable robots to extract from blockage. A robot is in such a situation because it was crossing a zone frequented by antinomic robots. So as to prevent this situation, robots must be able to avoid such risky zones: zones from which antinomic robots come. Then, an anticipation rule can be specified:

A robot sees an antinomic robot. If a robot r_1 perceives a robot r_2 having an antinomic goal, if r_1 can move to its sides it does it else it moves forward.

Nevertheless, this reactive anticipation presents a major problem: once a robot has avoided the risky zone, no mechanism ensures that it will not go in it again, led by its goal. In order to tackle this difficulty, robots can be equipped with a memory of the risky zones (in the Representations Module). Each time t a robot r_i experiments an anticipation situation facing a robot r_j, it adds to its memory a tuple (or virtual marker) $\langle posX(r_j,t), posY(r_j,t), goal(r_i,t), w\rangle$ in which $posX(r_i,t)$ and $posY(r_i,t)$ represent the coordinates of r_j at the moment t. $goal(r_i,t)$ represents the goal r_i was achieving at time t. w represents a repulsion value. The higher the value is, the more the robot will try to avoid the zone described by the marker when it is achieving another goal than $goal(r_i,t)$. Therefore, the robot inspects all its personal markers[6] whose distance is inferior to the perception limit (to fulfill the locality principle). A marker with a weight w and situated in the direction dir at a distance d induces that $V_{r_i}^{coop}(t, dir_{opp})$ will be increased of w (dir_{opp} is the opposite direction to dir).

As the memory is limited, tuples that are added must disappear during simulation run-time. For example, the weight w can decrease of a given value δ_w (called *forgetting factor*) at each step. Once $w = 0$, the tuple is removed from the memory. This method corresponds to the use of *virtual* and *personal* pheromones. Finally, as ants do, robots can reinforce their markers: a robot

[6] Robots cannot share their memory as they cannot communicate.

moving to a position corresponding to one of its marker with another goal, re-initializes the marker. In fact, if the robot is at this position, it might be a risky zone when it tries to achieve another goal.

5 Experiments

In order to validate this approach and to compare cooperative behaviors of transporter robots, the expounded model has been implemented and simulated.

5.1 Experimental Setup

The simulation environment corresponds to two rooms (25 x 30 cells) separated by two long and narrow corridors (30 x 1 cells). 300 robots are randomly placed in the claim room. These robots can perceive at 5 cells, and can make a move of one cell at each step. If they can anticipate conflicts, their memory can contain 1500 tuples with $w = 400$ and $\delta_w = 1$.

5.2 Reaction Versus Anticipation

The figure 3 shows a comparison between the results of the previously presented cooperative behaviors. The unblocking behavior-equipped robots obtain a linear efficiency with no blockage, unlike nominal behavior-equipped individualist robots. By adding blockage anticipation, the collective becomes more efficient (at least 30% more boxes are transported). This corresponds to an optimisation of the unblocking behavior. According to the AMAS paradigm, we can experimentally observe that the local resorption of NCS leads to the collective functional

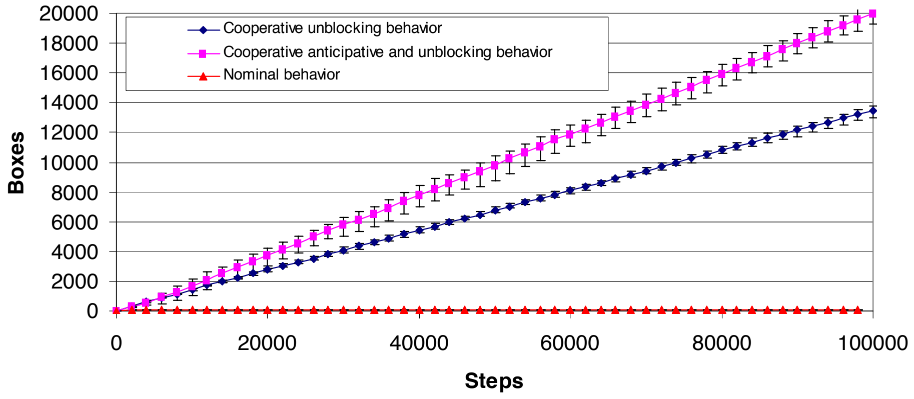
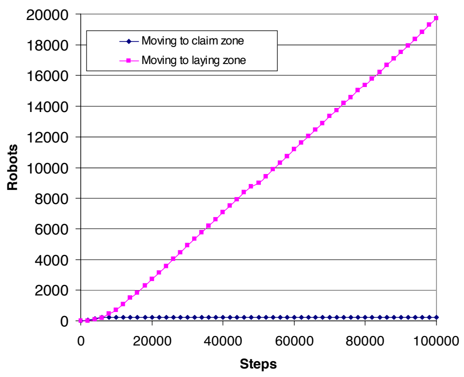


Fig. 3. Number of transported boxes for 15 simulations (300 robots, 2 corridors, 5-ranged perception), corresponding to the nominal behavior (individualist) and the two cooperative ones: the cooperative unblocking behavior (see section 4.1) and the cooperative anticipation behavior (see section 4.2)



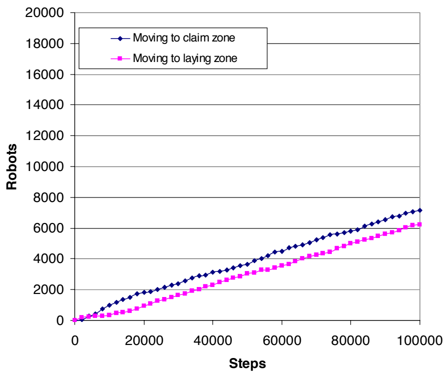


Fig. 4. Number of incoming robots for a corridor and for the two cooperative behaviors: unblocking behavior (right) and anticipation unblocking behavior (left)

adequacy. Finally, the more the NCS are taken into account, the higher the performances are.

5.3 Emergence of Corridor Dedication

The figure 4 presents the corridor-going for the two cooperative behaviors. In the case of anticipation behavior, we can observe the emergence of a sense of traffic. Robots dedicate corridors to particular goals. We can assign the emergent property to this phenomenon because robots do not have any notion of corridor – unlike some previous work [12]. Thus, just thanks to local data, robots established a coherent traffic behavior that leads to an optimization of the output.

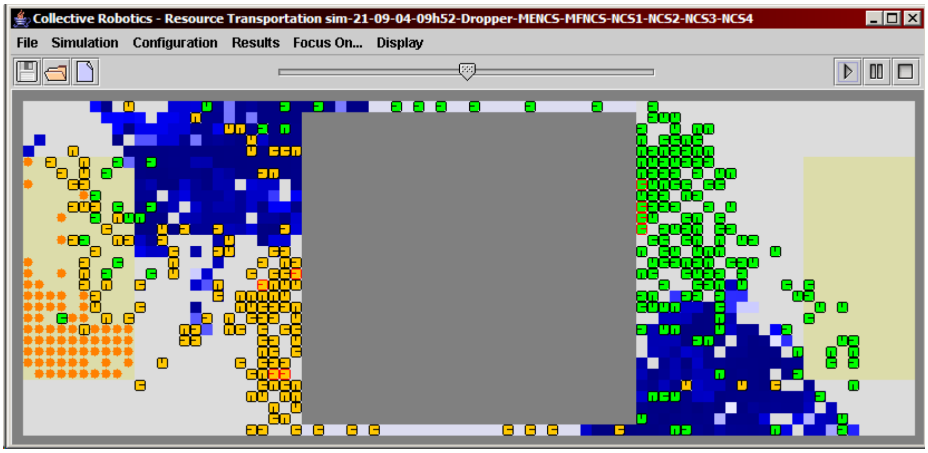

Collective Robotics - Resource Transportation sim-21-09-04-09h52-Dropper-MENCS-MFNCS-NCS1-NCS2-NCS3-NCS4
File
Simulation
Configuration
Results
Focus On...
Display

Fig. 5. Global markers positioning (sum of all individual memories)

Moreover, the sense of traffic varies from a simulation to another with only little initial variations of some robots.

In fact, markers are positioned only at one corridor entry for one direction as figure 5 shows. This figure shows the sum of all the markers for all the robots. This data is not known from robots. It is only calculated for monitoring purpose.

According to the distinction made by [13], the functionality of system is *weakly* emergent because robots do not have any global knowledge about the number of boxes they have transported and it does not motivate them to transport more boxes.

5.4 Adaptation and Robustness

Some simulations has been launched in dynamic environments with random corridor closures. These simulations show the collective always sets another corridor dedication unless the corridor closure frequency is to high in comparison with the forgetting factor (δ_w).

6 Discussion

By regarding the previous results, instantiating the agent model proposed by ADELFE has several advantages. Firstly, unlike ant-inspired algorithms [7], robots do not mark their environment with pheromones but memorize virtual and personal markers. Secondly, contrary to competition-based [8] or altruistic [10] approaches, robots do not need direct communication to alarm, inform close robots or exchange requests and intentions. Thirdly, cooperative behavior encoding is insensitive to the number of robots, to the topography and to the dimensions of the environment. Fourthly, no global feedback is needed to lead the system to functional adequacy, which prevents the system to reach local extrema. Finally, the incremental method proposed by ADELFE to define non cooperative situations – necessary ones and optimization ones – opens up a new way toward a living design methodology within which behaviors are assigned to robots (or agents) as design and development progress in terms of designers' requirements (this activity is called *fast prototyping* in the ADELFE process). Of course, this will need to develop a simulation/design platform, as adequate to the cooperative agent model as possible.

Nevertheless, some choices have been taken concerning the affectation of values that can drastically modify the global behavior. By now, ADELFE does not provide any guidance to appropriately instantiate these values; designers must do it by themselves. For instance, the initial weight for markers and the forgetting factor have been adjusted to the time robots spend to cross the entire environment. This might be completely different for a more complex environment with more or less corridors which can dynamically open or close. Some simulations has been done with such environments, and the affected values seem correct unless corridors are too near or frequence of closure is too fast. These values also may be learned during run-time, which is one of our perspectives.

Moreover, from the resource transportation problem, we focused on particular NCS: conflict and uselessness. If robots were equipped with high-level communication capabilities (to exchange data as markers in order to share their experiences), incomprehension and ambiguity may raise. In this case, ADELFE proposes to analyze interaction protocols between agents and identify such situations.

Lastly, the application we chose and the solution we considered are typical examples of “flat” systems within which no *a priori* hierarchy is defined. Self-organization leads the society to an adequate functioning without having specified a static organization; that is due to the homogeneity of the collective. On the contrary, if the collective is heterogeneous (different speeds, different functions, complementary or not), notions of hierarchy and/or priority are relevant. So as to ensure extensionality[7] and irreductibility[8] properties of emergent systems [14], predefining an organization is prohibited; in this case, the function of the system is intentionally defined, and therefore is not adaptive. Consequently, organization is an emergent phenomenon of relations between agents and is not a predefined schema.

7 Conclusion

In this paper, we presented an instantiation of the cooperative agent model proposed by ADELFE in the domain of Collective Robotics. This application helps us highlighting the possibility to iteratively and compositionally design agents' behaviors. Considering the ignorance of the global task and the environment, the self-organizing collective reaches an emergent coherent behavior, which is then more robust to environmental risks (such as traffic jams). Our simulation application tackles a simple problem with a simple environment.

Concerning the cooperative agent model, extending the ADELFE method to development and automatic code generation based on the MDA (*Model Driven Architecture*) paradigm seem to be promising perspective [15]. Actually, by formally specifying the model and by defining transformation rules, proceeding from design models to development instantiations becomes conceivable.

References

1. Capera, D., Georgé, J., Gleizes, M.P., Glize, P.: The AMAS theory for complex problem solving based on self-organizing cooperative agents. In: 1st International Workshop on Theory and Practice of Open Computational Systems (TAPOCS) at IEEE 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2003), IEEE (2003) 383–388

[7] This means the function of the system is defined by relations between input and output, but not by an algorithm.

[8] Churchland defines the emergence in terms of the irreductibility of properties assigned to a high-level theory associated to components in a lower-level theory.

2. Georgé, J.P., Gleizes, M.P., Glize, P., Régis, C.: Real-time simulation for flood forecast: an adaptive multi-agent system staff. In Kazakov, D., Kudenko, D., Alonso, E., eds.: Proceedings of the AISB'03 symposium on Adaptive Agents and Multi-Agent Systems(AAMAS'03), University of Wales, Aberystwyth (2003)
3. Picard, G., Gleizes, M.P.: The ADELFE Methodology – Designing Adaptive Cooperative Multi-Agent Systems. In Bergenti, F., Gleizes, M.P., Zambonelli, F., eds.: Methodologies and Software Engineering for Agent Systems, Kluwer (2004)
4. Bernon, C., Camps, V., Gleizes, M.P., Picard, G.: Designing Agents' Behaviours within the Framework of ADELFE Methodology. In: Fourth International Workshop on Engineering Societies in the Agents World (ESAW'03), Imperial College London, UK, 29-31 October. Volume 3071 of Lecture Notes in Artificial Intelligence., Springer-Verlag (2004)
5. Picard, G., Capera, D., Gleizes, M.P., Glize, P.: A Sample Application of ADELFE Focusing on Analysis and Design : The Mechanism Design Problem. In: Fifth International Workshop on Engineering Societies in the Agents World (ESAW'04), 20-22 October 2004, Toulouse, France. (2004)
6. Vaughan, R., Støy, K., Sukhatme, G., Matarić, M.: Blazing a trail: Insect-inspired resource transportation by a robotic team. In: Proceedings of 5th International Symposium on Distributed Robotic Systems. (2000)
7. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press (1999)
8. Vaughan, R., Støy, K., Sukhatme, G., Matarić, M.: Go ahead make my day: Robot conflict resolution by aggressive competition. In: Proceedings of the 6th International Conference on Simulation of Adaptive Behaviour. (2000)
9. Ferber, J.: Multi-Agent System: An Introduction to Distributed Artificial Intelligence. Harlow: Addison Wesley Longman (1999)
10. Lucidarme, P., Simonin, O., Liéègeois, A.: Implementation and Evaluation of a Satisfaction/Altruism Based Architecture for Multi-Robot Systems. In: Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA, IEEE (2002) 1007–1012
11. Topin, X., Fourcassié, V., Gleizes, M.P., Theraulaz, G., Regis, C., Glize, P.: Theories and experiments on emergent behaviour: From natural to artificial systems and back. In: Proceedings of the 3rd European Conference on Cognitive Science (ECCS'99), Certosa di Pontignano, SI, Italy (1999)
12. Picard, G., Gleizes, M.P.: An Agent Architecture to Design Self-Organizing Collectives: Principles and Application. In Kazakov, D., Kudenko, D., Alonso, E., eds.: AISB'02 Symposium on Adaptive Multi-Agent Systems (AAMASII). Volume 2636 of LNAI., Univerity of London, UK, Springer-Verlag (2002) 141–158
13. Müller, J.P.: Emergence of collective behaviour: simulation and social engineering. In: Fourth International Workshop on Engineering Societies in the Agents World (ESAW'03), Imperial College London, UK, 29-31 October. (2004)
14. Ali, S., Zimmer, R., Elstob, C.: The question concerning emergence : Implication for Artificiality. In Dubois, D., ed.: Computing Anticipatory Systems : CASYS'97 - First International Conference. (1997)
15. Soley, R., the OMG Staff Strategy Group: Model driven architecture. White paper Draft 3.2, OMG (2002)