Avoiding deadlock in multi-agent systems

Dominique Duhaut, Elian Carrillo, Sébastien Saint-Aimé

Avoiding deadlock in multi-agent systems

Dominique Duhaut, Elian Carrillo, Sébastien Saint-Aimé – Université de Bretagne Sud

***Abstract*—The problem of controlling the displacement of a group of robots is very difficult. On one hand, it can be solved by planning but this becomes very complex if the number of robot increases; on the other hand it can be solved by a local approach where each robot follows its own behaviour. Here we give an analysis of this last approach and present a solution to solve one of emergent possible problems: deadlock.**

I. Introduction

In this paper we address the problem of programming a team of robots. The goal here is to obtain a specific behaviour from the an entire team of robots by programming each of them individually.

Various proposals are well known for this problem. Brooks [3] proposed a behaviour based approach and later Arkin [1] proposed the motor schema based approach. Furthermore, some high level approaches are proposed in Yoshida [7] in the field of reconfigurable robots.

The particular point is that the emergent behaviour of the robot team is very difficult to predict because of the individual interaction of each robot with its environment. Due to this, the behaviour of the team is then sometimes impossible to anticipate.

In this paper we study a set of problems linked to multi-robot displacement with distributed programming.

First, we study the displacement of a group of robots moving alone and having an known target [16]. The problem is then to avoid other robots.

Secondly, we address the problem of a set of reconfigurable robots having to move over obstacles and define their reconfiguration [11].

Finally, we look for a quite similar problem but instead the modular robot has to move without disconnecting the modules part of the robot.

In each of these problems we show how we can find the same kind of problems and how we can propose a solution to them.

Manuscript received March 16, 2007. This work has been supported by the French program Robea from the CNRS.

Dominique Duhaut is with the Valoria laboratory of the University of Bretagne Sud, Campus de Tohannic; Bât. Yves Coppens, BP 573, 56017 Vannes Cedex – FRANCE (phone: +33 (0)2 97 61 72 06; fax: +33 (0)2 97 01 72 79; e-mail: dominique.duhaut@univ-ubs.fr).

Elian Carillo is with the Valoria laboratory of the University of Bretagne Sud, Campus de Tohannic; Bât. Yves Coppens, BP 573, 56017 Vannes Cedex – FRANCE (phone: +33 (0)2 97 61 72 06; fax: +33 (0)2 97 01 72 79; e-mail: elian.carillo@univ-ubs.fr).

Sébastien Saint-Aimé is with the Valoria laboratory of the University of Bretagne Sud, Campus de Tohannic; Bât. Yves Coppens, BP 573, 56017 Vannes Cedex – FRANCE (phone: +33 (0)2 97 61 72 61; fax: +33 (0)2 97 01 72 79; e-mail: sebastien.saint-aime@univ-ubs.fr).

II. Hypothesis

In this paper, we will always suppose that the robots have very small sensing capabilities. This means that the robot can only detect the environment just around him.

For instance in 2D, in an array the robot will perceive the value of the 8 positions around (6 positions in an hexagonal modelisation).

The robot can make the difference between a free space, an obstacle or another robot. The robot can detect the direction of an attractor.

In figure 1 the agent can perceive the presence of a neighbouring agent (position 2,3,4,5) and the value of the gradient of the attractor in each position

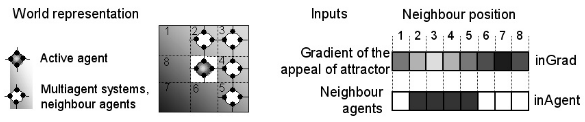

Figure 1: Perception of the environment

III. Displacement in a crowd

The problem addressed here is the displacement of robots in a crowd of robots. The problem is to study the intersection of the different paths. The hypotheses are respected and the robot can only look at the place around him and know the direction of his own goal.

The behaviour of an agent is given by the following pseudo-code translated as if the agent was someone with his own goal:

Pseudo-code of the Behaviour:

```
loop
    if I have reached my goal
        then if someone asks me to move
            then I search for a place P where to go; (E1)
                if P exists
                    then if P is free
                        then I move in P
                        else I ask one in P to move(E2)
                    else I do not move
                else I do not move
            else I search for a place Q where to go;(E3)
            if Q is free
            then I move in Q
            else I ask the one in Q to move (E4)
    endloop
```

It is clear that E1 and E3 are not the same. E1 is performed when a movement is asked of someone who is satisfied in the place where he is. E3 is performed when the agent wants to reach a goal and finds someone in his path. So even if it is the same action (asking someone to move) E1 and E3 can be seen with two different intensities, as can be imagined using potential field technique [15].

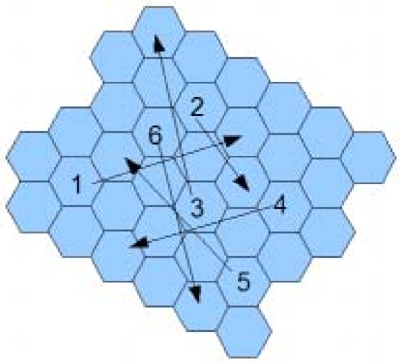

Figure 2: Displacement in a crowd

A. Deadlock 1

In this first case we can see that some deadlock can appear [14]. For instance, figure 3 is an example of the potential deadlock of all the robots having a destination that crosses the path of the other. No one will change its path to let the other move because there is no priority between robots.

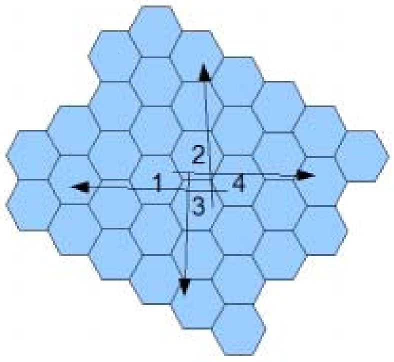

Figure 3: Deadlock 1

In this example we can see that without any information about the global situation the deadlock will not be solved.

B. Deadlock 2

In this other example, deadlock can also appear but for other reasons. In this case only one robot wants to move (n°1) all the other are satisfied in their place. N°1 will ask the one in his line to move according to the previous behaviour. Then it can appear that a chain of asked displacement can be constructed (the ring on figure 4).

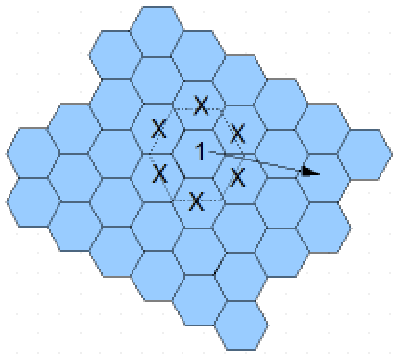

Figure 4: Deadlock 2

C. Solution to avoid deadlock : random move ?

One solution to break the deadlock is usually to include some random moves in the behaviour of the robot. The basic idea is to add the following code:

```
if I have not moved in a long time and I still want to go somewhere
then move in any free position
```

This solution is quite classical and already used in potential field approaches. But this cannot guaranty that the deadlock will be completely avoided. Due to the fact that the system is still "running" because there is always one robot moving, we can expect that (and it is usually the case) the system will finally auto-organize and find a solution.

In fact, we can verify that some robots when trapped in a local minima cannot get out without a simple move in a free position. It seems that they make a move in a random direction then come back in the previous place and then begin an oscillation cycle. To avoid this, it is possible to increase the size of the memory to remember the n-th last displacement to move back far away before moving randomly. In this case, we can escape from a local minima. In the case of collective robotics, this technique does not seem realistic because the environment is dynamic and moving back in the previous displacement does not guaranty reaching an environment in which the situation would be identical to the one crossed in the past.

This approach supposes that we include a memory of the recent past because the robot must remember the last time he made a move. When we program the robot application, the question is then to decide what "a long time" is . It depends on the dynamic of the application, and is not necessarily easy to adjust.

D. *Solution to avoid deadlock: hierarchy ?*

A second way to solve the problem of deadlock is to define a fixed rule of priority that can be shared between all the robots.

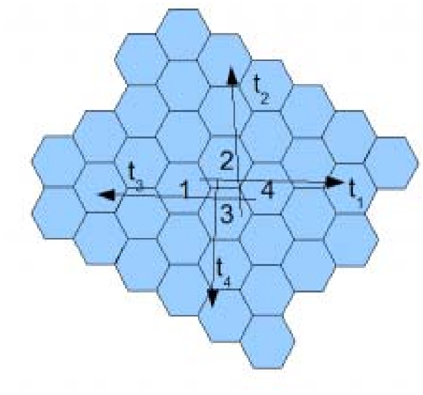

Figure 5**:** Hierarchy static priority

For instance, we can make a rule based on the direction of the movement. We can decide, for instance, that priority directions are "East, North, West, South". Then based on this rule, robot 1 would have the highest priority, followed by 3, 4 and finally 2 would have the lowest priority. In this case, the robot having the lowest priority would search for a place to go (E1) and accept to move in a place even if it is in the opposite direction of its destination.

This solution is the best one because it brakes the deadlock. A simple proof can be given:

It is well known that there are four conditions that are required for a deadlock:

1. Each resource can only be assigned to exactly one resource.
2. Processes can hold a resource and request more.
3. Resources cannot be forcibly removed from a process.
4. There must be a circular chain of processes, each waiting for a resource held by the next member of the chain.

Condition 4 can be avoided by using a static priority law because the robot having the lowest priority will be obliged to move and he will release (property 2) the resource which is the place occupied by the robot in the field.

The limit of this approach is that the priority rule is not necessarily easy to find with a limited range of sensors in the robot. This priority rule must be computed locally in each robot without communication. In the previous example, suppose that the geographic direction of north can be detected locally in each robot.

IV. Displacement of a Collaborative Disjoint Group

A. *The problem and its solution*

In this section, the objective is to create collective movement through difficulties, thus permitting robotic modules to climb on a first bloc, to cross on a second bloc and to get off this second bloc and finally reach the target.

In this study [17] we show that it is possible to organize local behaviour to obtain emergent behaviour satisfying the constraint.

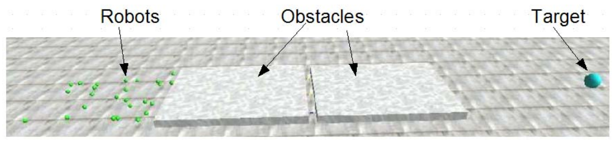

Figure 6**:** The collective displacement

With this local behaviour (figure 7) we obtain a solution to the problem respecting the hypothesis. The tests: difficulty detected, module aligned, module in front ... are all obtained by a local sensor getting information from the surrounding environment.

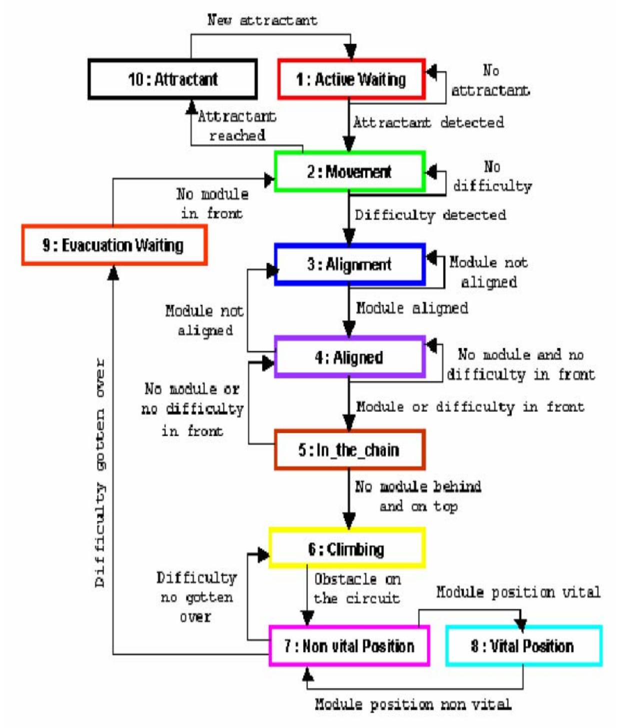

Figure 7**:** The local behaviour

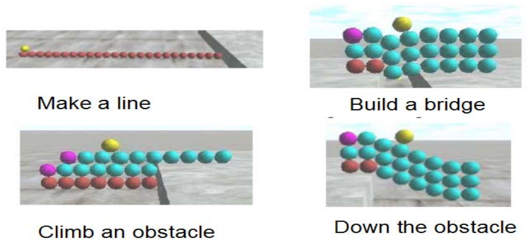

Figure 8**:** The emergent behaviour

Figure 8 gives some examples of the global emergent behaviour obtained with the local behaviour described in figure 7. Some full videos and explanations can be found in [9].

B. *Why does it work ?: memory and scheduling*

In the previous problem we had a lot of deadlocks. Here, we can verify that we do not have this kind of problem.

The reason is that here we use both methods described in section 3: a local memory and a hierarchy based on a total order.

For instance, the memory counts the number of upward displacements to know what the level reached is (here limited at 3; see figure 8). And a total order is given because all the robots move one step in the same scheduling period. So this scheduling gives the whole system the property that no collisions between robots will appear until contact with an obstacle. This first collision will change definitively their behaviour by reaching a new state in the algorithm (figure 7).

This conjunction between local memory and total order is the key point.

V. Displacement of a collaborative joint group

In this new problem we address the displacement of a group of robot modules which are part of a reconfigurable robot [1, 4, 5, 6 9]. Here the global displacement of the robot is obtained by the set of reconfiguration of all the modules. The constraint is that the robot must not loss any module during the global movement.

This problem is a very complex problem, for instance if the robot is built with 10 modules having 8 potential elementary displacements (figure 9) then a 8^{10} number of possible solutions must be explored to see what would happen after one displacement of all modules in the robot.

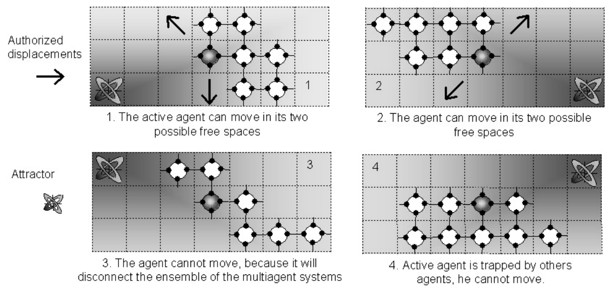
Authorized displacements
1
1. The active agent can move in its two possible free spaces
2
2. The agent can move in its two possible free spaces
Attractor
3
3. The agent cannot move, because it will disconnect the ensemble of the multiagent systems
4
4. Active agent is trapped by others agents, he cannot move.

Figure 9**:** The emergent behaviour

Here, again, as in the problem described in section 4 we can use reactive programming using the following algorithm.

A. *Local behavior*

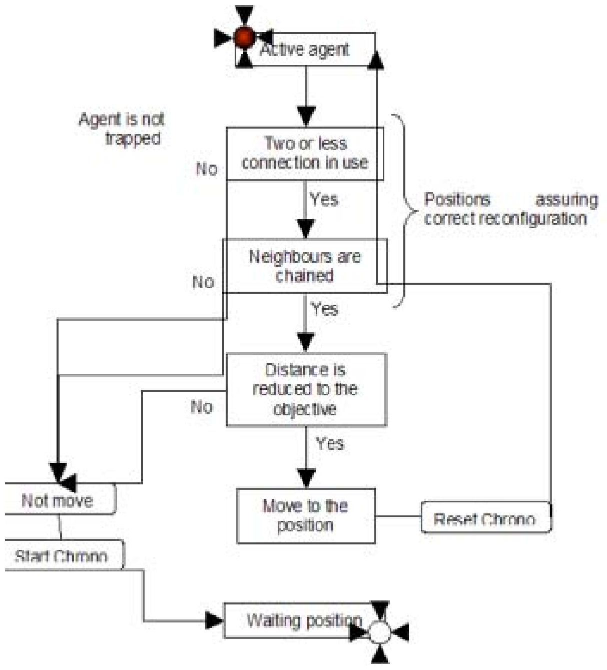

Figure 10: The local behavior

With this local behaviour we can succeed in the collective displacement. But, as we will show, the emergent behaviour depends on the scheduling policy.

B. *Simulation 1: move all steps*

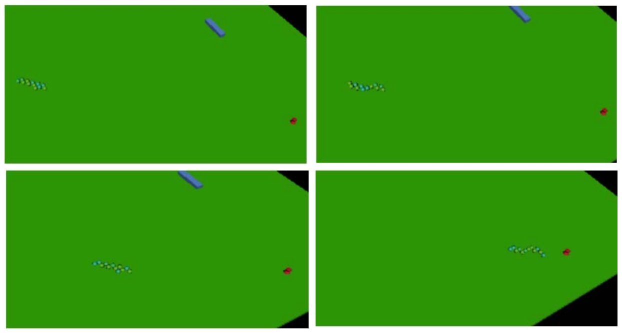

Figure 11**:** Moving all possible displacement

In this simulation when it is the turn of a module to move then he makes all the possible moves before handing over control to the next one. This means that each time that it is his turn, the module will reach, as a final position, the one having the lower gradient, so the nearest one to the target. The emergent effect is to build a "line" of modules.

In this case, the algorithm works well under a fixed or random scheduling between the modules' turns. The problem is that in a real robot this approach means that we impose a synchronisation between robots because we have to wait for the robot to finish its turn before giving control to another one. However, we do not want to suppose that communication between modules is available.

C. Simulation 2: move one step

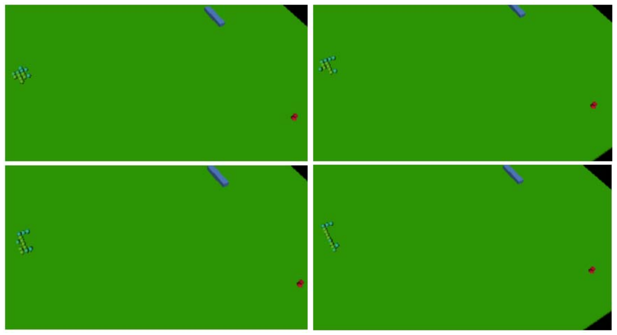

Figure 12**:** One displacement per module

In this simulation each robot in control will run only one displacement with the same previous algorithm. The emergent behaviour is then that some local minima appears. The reason is given by the symmetry of the potential field that gives two ways to reach the way to the goal. Then, modules having only a partial view of the situation (and having no memory of previous moves) will finally be blocked in a deadlock situation because the last one in the chain cannot move because he would brake the chain of robots into two pieces and this is not acceptable..

D. Using memory to avoid deadlock?

Even with a local memory in which the modules memorise the last moves made, thus the last values of the gradient, this kind of deadlock still appears. The solution is then to accept to make a move in a wrong direction as described in section 3.C. Then, the deadlock is momentarily avoided but appears later as shown in the following figure 13.

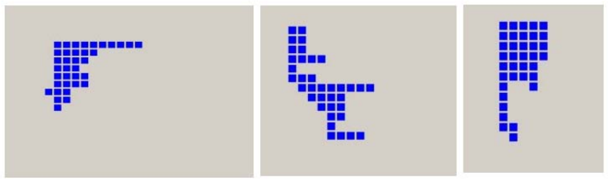

Figure 13**:** Various emerging deadlocks

On this simulation the target is on the right side of the picture. The first blocking situation appears in the simulation on the right-hand side picture of figure 13 where the modules at the end and down have no reason to move anymore because they all have reached the highest value of the gradient. If we modify the rule and accept that the modules can move up on a same level of gradient then the 2 deadlocks on the 2 left-hand side pictures of figure 13 appear. The fact that the 2 simulations do not give the same result is based on the difference of the scheduling policy over the running modules.

E. How to avoid deadlock

Again, the solution is to use two techniques together: memory and a total order of the problem.

The idea is to introduce behaviour states as proposed in section 4.A.

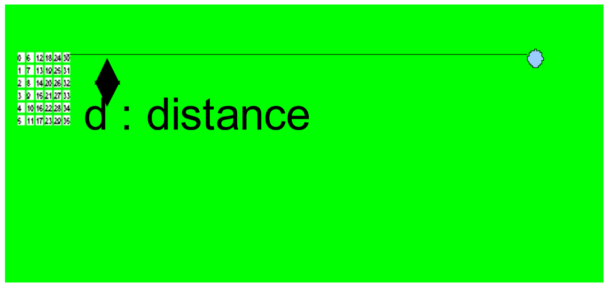

Figure 14**:** Introducing a total order

Based on the previous work we introduce in the behaviour of each robot the notion of distance “d” to the optimal line to reach the target (black line on figure 14). The d value is calculated during the module's movement. Each time that the module goes north or south the value of d is increased or decreased (we suppose that the module knows the direction of the target is east). This is quite realistic for real reconfigurable robots because during their displacement they are able to measure the evolution of the potential field. With this d data we link 3 different elementary behaviours:

1. One will be specific to modules on the optimal line (in figure 14 they have the numbers 0,6,12,18,24,30). Here the module will stay without moving except if he is the last one then he will accept to go on the second line to go in front.

2. The second behaviour will be for those on the second line 1,7,13,... In this case modules will only accept to move along the first line to reach the head of this first line and take this place.

3. For all the rest of the group, here their behaviour is to move east to find a place where they can decrease the d value.

With this simple general algorithm, we obtain a very stable emergent global behaviour even with a random scheduling as shown in figure 15.

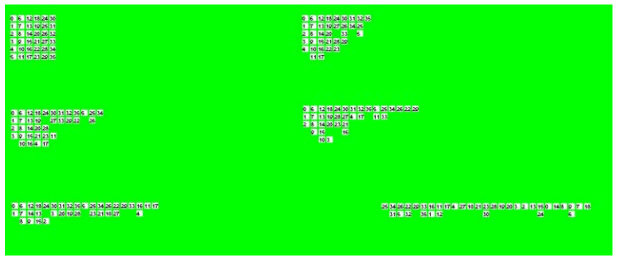

Figure 15**:** 6 snapshots of the evolution of the simulation.

VI. Conclusion

In this work, we show that without sharing any information on the global situation of the multi-robot system, we cannot avoid having deadlock by having all the robots unable to move or having a set of robots in oscillation.

To avoid this problem, we propose to introduce a general order on the environment that guaranties to build a hierarchy of behaviors between the robots. We present here some samples of this general order: one based on some events (obstacle detection section 4) or a direction in space (section 3) or a position in the space (section 5).

The price of this total order is usually to introduce a new sensor in the robot module: contact sensor for an event, magnetic sensor for the direction of north or some kind of GPS to have the position in the space.

With this approach the robot can decide locally, without any communication with the other robots, what is the best behaviour to run in a given situation. With this, the emergent behaviour of the group of robots is to reach the target.

Acknowledgment

This work has been supported by the French program Robea from the CNRS.

References

[1] H. Bojinov, A. Casal, T. Hogg, Multiagent control of selfreconfigurable robots, *Proceedings of the 4th InternationalConference on MultiAgent Systems*, Boston, Massachusetts, USA, 2000.

[2] R. C. Arkin, Behaviour-Based Robotics. *Cambridge, MA: MIT Press 1998*.

[3] R. A. Brooks, A robust layered control system for a mobile robot. *Journal of Robotics and Automation*. Vol. 2, pp. 14-23 1986.

[4] C. Jones, M.J. Mataric, From local to global behavior in intelligent self-assembly, *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA 2003)*, Taipei, Taiwan, September 2003.

[5] M. Yim, Y. Zhang, J. Lamping, E. Mao, Distributed control for 3D TABLE II metamorphosis, *Journal of Autonomous Robots* 10, 2001.

[6] J. Kubica, A. Casal, T. Hogg, Complex behaviors from local rules in modular self-reconfigurables robots, *Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA 2001)*, Seoul, Korea, May 2001.

[7] E. Yoshida, S. Murata, H. Kurokawa, K. Tomita, S. Kokaji, A distributed reconfiguration method for 3-D homogeneous structure, *Advanced Robotics*, 13-4, pp. 363-380, 2000.

[8] H. Bojinov, A. Casal, T. Hogg, Emergent structures in modular selfreconfigurable robots, *Proceedings IEEE International Conference on Robotics and Automation (ICRA 2000)*, San Francisco, pp. 1734-1741.

[9] B. Ashfield, Distributed Deadlock Detection in Mobile Agent Systems, Ashfield, B.: Distributed Deadlock Detection in Mobile Agent Systems, M.C.S. Thesis, Carleton University, (2000).

[10] S. Murata, H. Kurokawa, E. Yoshida, K. Tomita, S. Kokaji, A 3-D self-reconfigurable structure, *Proceedings of the 1998 IEEE International Conference on Robotics and Automation (ICRA 1998)*, Leuven, Belgium, May 1998.

[11] www-valoria.univ-ubs.fr/Dominique.Duhaut/maam/, *MAAM project web site.*

[12] www.swiss.ai.mit.edu/projects/amorphous/, *Amorphous computingweb site*.

[13] L. Clement, R. Nagpal, Self-assembly and self-repairingtopologies, *Workshop on Adaptability in Multi-Agent Systems*, Robocup Australian Open, January 2003.

[14] J. Zucker, Self-healing structures in amorphous computing, *International Conference on Complex Systems*, May 2000.

[15] K. Stoy, R. Nagpal, Self-Reconfiguration Using Directed Growth, *7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, France, June23-25, 2004.

[16] D. Duhaut, Study of communication in a multi-agent system, *IEEE/SMC'95 Conference, Vancouver, Canada, October 22-25, 1995*

[17] V.Montreuil, D. Duhaut, A. Drogoul A collective moving algorithm in modular robotics: contribution of communication capacities In *6th IEEE International Symposium on Computational Intelligence in Robotics and Automation* june 27-30 2005 Espoo Finlande