Swarm embodiment - a new way for deriving emergent behavior in artificial swarms

S. Kornienko, O. Kornienko, P. Levi

Institute of Parallel and Distributed Systems,
University of Stuttgart, Universitätsstr. 38, D-70569 Stuttgart, Germany

Abstract This paper concerns the emergent properties of collective robotic systems with imposed microscopic and macroscopic constraints. These constraints dramatically impact the emergent behavior of collective systems so that creating desired emergence becomes very challenged. In this paper we present the top-down swarm embodiment methodology that allows obtaining desired collective behavior in systematic way.

1 Introduction

In order to go beyond the current state of the art in the realization of robotic swarms, the European Commission has granted funding to the I-SWARM project. This project is going to produce a large group of micro-robots (about 1000 micro-robots with the proposed size $2 \times 2 \times 1$ mm) that are capable to mimic some aspects of social insects. Such a robot swarm is expected to perform a variety of applications, including micro assembling, biological, medical or cleaning tasks.

The micro-robots, due to small size, are very restricted in hardware capabilities, like distance measurement, navigation or communication. These constraints, arising on the microscopic level, hardly limit the emergent behavior. Not only microscopic, but also macroscopic constraints influence the collective behavior. This kind of constraints arises in technically useful behavior because of necessity to emerge collective activities in specific order, with specific parameters.

Both types of constraints change the problem of collective behavior. The question is not only to find the swarm-controlling mechanisms, allowing a solution of typical problems like foraging or division of labors [1]. The question becomes of how to modify and even how to create new mechanisms that generate the desired emergent behavior satisfying all constraints. Trough many scientific domains contribute to solution of this problem, we fail to find a common methodology that consolidate approaches from these domains. Without this, the derivation of swarm mechanisms is often performed in "trial and error" way. The given work suggests the top-down methodology using a swarm embodiment, that enable a systematical derivation of the desired emergent behavior for artificial swarm. The suggested methodology has been tested in real experiments with the group of micro-robots Jasmine (large prototype in the I-Swarm project).

2 Microscopic and macroscopic constraints

As mentioned in the previous section, artificial robotic swarms are constrained from microscopic and macroscopic sides. The microscopic constrains originate mainly from a construction of a robot (see e.g. the micro-robot Jasmine in Figure 1(r)). The most important constraints are the communication and percep-

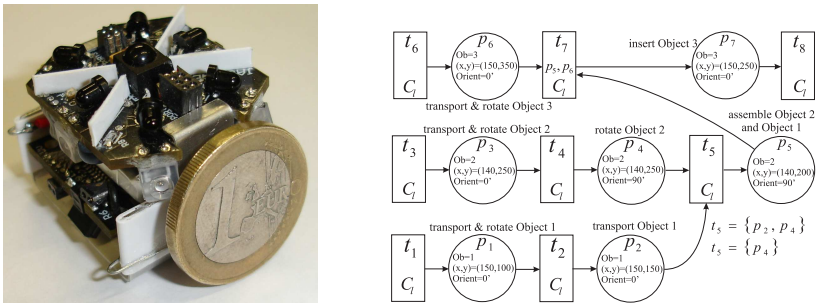

Figure 1. (r) Micro-robot Jasmine; **(l)** The assembling plan in the form of Petri-net. P_i are phases, where t_i are transitions with the shown conditions.

tion radius, type of sensors, time of autonomous work and so on. These individual capabilities essentially impact a group behavior.

Macroscopic constraints arise if the collective systems has to emerge the technically useful behavior. The appearance of these constraints can be demonstrated on a simple assembling example, where robots push three different objects into one defined construction. Assembling of the object has be performed only in the specified order, shown as the Petri-net in Figure 1(l), otherwise the desired construction will be not obtained. The plan consists of 7 steps, shown as the phases p_1-p_7 with the corresponding positions and rotation angles. The phases p_1, p_3 and p_6 can be started in parallel, other phases have to be proceeded sequentially. The phase p_7 can be started only if p_5, p_6 are finished.

An agent starts transportation or rotation only if its position coincides with the position of an object. This object has not to be processed in this moment by other agents. Moreover the operations defined by the plan have to be applied to the given object only one time (two last problems can be solved by marking). We denote these restrictions as the local restrictions C_l. Each robot looks for objects Ob_i in its own neighborhood and mote to them and reads the mark. If the local and global restrictions are satisfied, the agent executes the required activities. The local rules of an agent have the following form:

- `Ob=look for (visible objects); read mark (Ob);`
- `if (constraints(Ob)) do (Activity);`

Agents can start assembling from different initial phases of the plan. Two generated agent-agent cooperation patterns with different initial phases are shown in Figures 2(r) and 2(l). Since these patterns are of different length, we can choose a short assembly by adding rules as e.g.

- `at choice → chose phase with smaller number;`

The cooperation, shown in Figure 2, emerges without being *preprogrammed*. Emergence arises because of *interactions* between agents that are controlled by *local rules*. The local and global constraints are not only introduced by these rules, but also through parameterization. In the assembly, each operation is *parameterized* by data from the plan. Without knowing these parameters, agents cannot accomplish assembly. Moreover, the emergent cooperation can be of different efficiency and we face the problem of *optimizing the emergent behavior*.

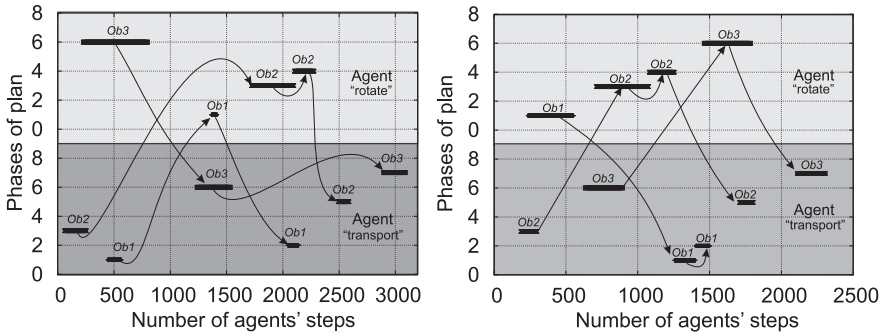

Figure 2. *Examples of emergent "agent-agent" cooperation, generated by the local rules.* **(r)** *Initial phases of the plan are* $(Ag_1)_{init} = p_6$ *and* $(Ag_2)_{init} = p_3$*;* **(l)** *Initial phases are* $(Ag_1)_{init} = p_1$ *and* $(Ag_2)_{init} = p_3$.

The microscopic and macroscopic constraints as well as parameterization and optimization of the emergent behavior appear on the swarm level. All these constraints are closely related with one another and hardly limit the emergent properties of collective systems. In trying to derive the desired emergence, we permanently confront with these constraints so that we identify *the problem of constrained emergent behavior as one of the main problems in artificial swarms* (from the viewpoint of controlling). Without systematic procedure, that allows involving constraints into the collective behavior, the derivation of desired emergence is performed mostly "by trial and error".

3 Top-down methodology

The local rules are in charge of artificial self-organization that appears the desired emergent behavior. There are two strategy to derive such rules. At the bottom-up strategy, the local rules are first programmed into each agent. This rule-based programming [2] originates from the domain of parallel and distributed computing. The general problem of bottom-up approach is that we cannot say in advance, which emergent behavior will be generated by the chosen rules. Especially if this behavior is bounded by constraints. The origin of this problem lies in enormous complexity of nonlinearly interacting system. As pointed out by some authors (e.g. [3]) "*A true emergent phenomenon is one for which the optimal means of prediction is simulation*". It means that *in the worst case* we

have to perform really many simulations, gradually changing the local rules, till we receive the desired collective behavior.

Another methodology, consisting in the top-down strategy, shown in Figure 3.

Using the top-down strategy, the derivation of local rules starts from a definition of the macroscopic pattern Ω and the corresponding constraints. Macroscopic constraints are incorporated directly into Ω, where as microscopic ones influence the so-called "distributing" transformation. The pattern Ω can be defined in statical way, as shown in Figure 1(l), or in evolutionary way, as e.g. to optimize some value. If the global pattern is determined evolutionary, we can use for "distributing" transformation several heuristical algorithms to produce a sequence of agents steps S_k that allows achieving the pattern Ω [4].

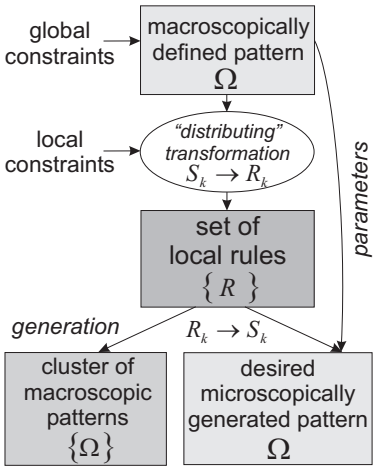

Figure 3. *Top-down strategy of derivation of local rules.*

From agent's viewpoint, each agent Ag_k has a sequence of activities S_k allowing the common group to accomplish Ω. Remark, that all constraints as well as communication are implicitly contained in S_k. Now we analyze S_k in order to derive the local rules R_k, that can generate this sequence of activities. More generally, to derive the local rules R_k, we can calculate Kolmogorov complexity of sequence S_k (finding the smallest grammar [5]). In this way can formally derive the set of these rules that defines a cooperation between agents and allows the agents' group jointly to solve the common task Ω.

3.1 Embodied top-down computational approach

In this section, we show that local rules can be obtained in the mentioned top-down way. Moreover these rules can be integrated (embodied) into specific motion/sensor systems. We consider here a *spatial clusterization* as a basic form of the macroscopic patterns Ω. Particulary, such a cluster can be a n-polygonal shape determined by distances D between corresponding corners. We introduce the local connectivity degree L_{cd} as the number of neighbor agents within the visibility radius R_{vis}. Global connectivity degree $G_{cd} = \sum L^i_{cd}$ is the sum of all local L^i_{cd} and the global compactness Φ is defined as $\Phi = \sum_i^N \sum_j^N D_{ij}$, where D_{ij} is a distance between agent i and agent j, N is the number of agents. In this way, the macroscopic pattern Ω is determined as $min\ (\Phi)$. The evolutionary algorithms, optimizing Φ, can have the following form

- `do {one step in all directions; calculate global compactness;}`
- `choose the step which minimum of global compactness;`

In the case of obtaining n-polygonal spatial formations we introduce additional constrains:

```
-   D=distance(itself-target[pattern]); {do (virtual Activities);
-     D[i]=distance(itself and target[from pattern]);}
-   j=find minimal(delta=D-D[j]); do (Activity j);
```

This algorithm produces a sequence of agents steps S_k allowing building the shapes from Ω. Now we analyze S_k to derive the same behavior, but without using Φ. In Figure 4 (r), (m) and (l) we plot the global connectivity and compactness, the command of the robot motion controller and, finally, local sensor information. We see that the behavior of whole swarm consists of two phases,

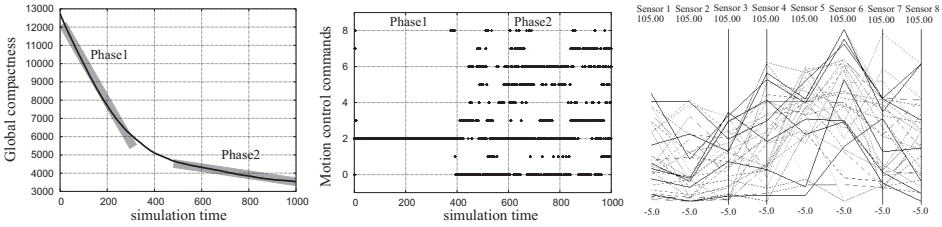

Figure 4. (r) *Global compactness Φ as the function of time;* **(m)** *Commands (in the 8-directional DOF motion system) of an agent in the same time range;* **(l)** *Sensor information (local connectivity degree L_{cd} in each direction) represented in parallel hierarchical coordinates when an agent decides to move in the direction "6".*

that are characterized by different slope of compactness. In the first phase, global compactness rapidly decreases (the global connectivity increases). All neighbor agents during the first phase move in the same direction. In this way, they build small clusters with a homogeneous direction of motion. The size of these local clusters grows whereas the number of them decreases. In the second phase, the rate of building decays and robots no longer move homogeneously. In this phase agents primarily decrease distances only in the cluster. As follows from Figure 4(l), agents decide to move in the direction with the most high local connectivity degree.

In trying to reproduce this behavior (without using Φ), we faced the question of how to replace the gradient (introduced by the global compactness Φ). In the experiments we used two values: the degree of local connectivity L_{cd} and a biologically motivated mechanism based on pAMP-gradient waves, emitted by the fungi *Dictyostelium discoideum* during aggregation phase [6]. In the last case, instead of the pAMP-gradient waves we introduce the following dynamical value k_n based on the L_{cd}: $k_{n+1} = log(\sum_{i=1}^{all\ neighbors} k_n^i)$ with $k_0 = L_{cd}$, where n is the simulation step. The value k_n grows the more rapidly, the larger the cluster is. Based on the values L_{cd} or k_n, robots can decide where the larger cluster is and move in this direction.

The algorithm reproducing the one-cluster-building behavior has the following form (D-direction of motion, nR - neighbor robots with highest Lcd or k_n, Dist - distance to nR, CP - adjustment parameter):

```
if (Lcd==0) D=rand; else D=(D of nR); if (D(nR)>CP) D=(D to nR);
```

where (D=(D of nR) and D=(D to nR) are the mentioned two phases of motion. In Figure 5 we compare the global compactness for the cases of single-phase (only the second phase) motion and two-phases motion based on L_{cd} and k_n.

As followed from Figure 5, the one-phase motion, that is intuitively the most evident one, builds only small local clusters without bringing them into the bigger one. Both two-phases mechanisms perform building the cluster. However the efficiency of L_{cd} and k_n based mechanisms is different. The biologically motivated mechanism requires less time (and energy) to converge.

During derivation of local rules R_k we assume some basic functionality F_b, like message transmission or environmental sensing. However the perfectly working simulative sensors essentially differ from real ones. In this way the swarm behavior, generated by R_k, often diverges from our expectations. To get round this problem, we involved the embodiment concept. This says that the same functionality can be implemented in many different ways: Rolf Pfeiffer demonstrated that an "intelligent behavior" can even be implemented when using only some properties of materials [7]. Embodied functionally is also often implemented in some "unusual" way. For example a robot can get a distance to neighbors by sending an IR-impulse and measuring a reflected light. However distances can also be obtaining during communication by measuring a signal intensity. This simple trick saves time and energy: such an unusual functionality is a typical sight of embodiment.

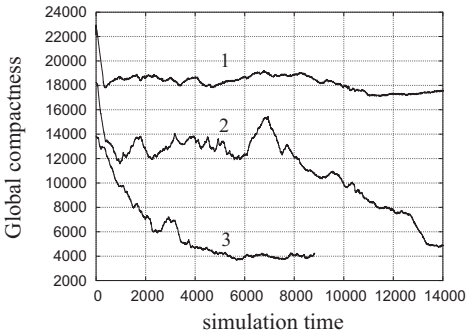

Figure 5. *Comparison of the global compactness for the cases of:* ***(1)*** *single-phase motion,* ***(2)*** *two-phases motion based on L_{cd} and* ***(3)*** *two-phases motion based on k_n. All curves represent typical cases of behavior.*

More generally, embodiment means that the system possesses the desired functionality F_b, but this functionality is in a latent form, "it is not appeared". This offers a way of how to get basic functionality for the local rules R_k: the local rules have to influence the hardware development of a robot. The swarm embodiment takes then the following form: definition of the macroscopic pattern Ω and the corresponding microscopic/macroscopic constraints; derivation of the local rules R_k; trade off between required functionality and adjustment of hardware; change of the hardware. The local rules have always been considered as a pure software components, however now they are a combination between software and hardware. We can say that in this way *the local rules for the whole swarm behavior are embodied into each individual robot.*

The embodiment in sense "*hardware* $\rightarrow$ *rules*" has been demonstrated in the work [8]. There we analyzed a dependence between agent's movement and sensor data for the derived S_k. Optimizing local rules to specific motion system,

the "top-down"-derived rules can be of 5-20% more efficient than corresponding "bottom-up" rules (see Figure 6(r)). The embodiment in sense "*rules* → *hardware*" has been demonstrated in the work [9]. In that work we considered context-awareness-related collective capabilities of interacting robotic group and incorporate several local rules into specific sensor system of real micro-robots. The achieved results essentially improve collective robotic behavior and reduced required communication and computational efforts.

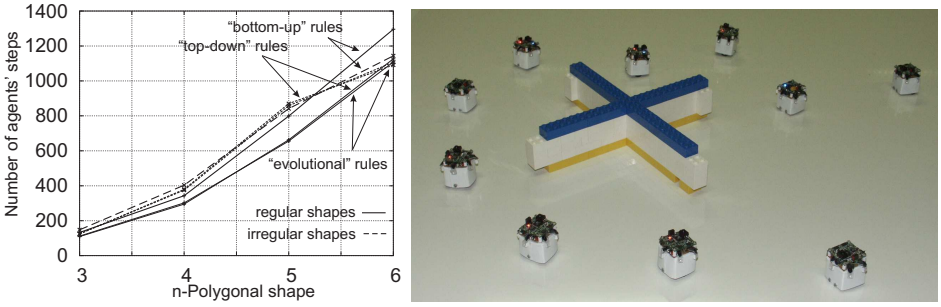

Figure 6. (r) *Comparison between efficiency (the number of steps, needed to reproduce the shapes) of "top-down" and "bottom-up" rules;* **(l)** *Preliminary experiments with a small group of micro-robots Jasmine.*

We performed several preliminary experiments with a small group of micro-robots Jasmine. The goal was an "embodied top-down" collective perception and spatial information processing (see Figure 3(l)). The development of collective behavior involved a definition of macroscopic patterns, derivation of local rules and redesigning of hardware components. As demonstrated by these experiments, the proposed approach allows creating a specific group's behavioral pattern, whereas robotic behavior still remained flexible (not predetermined). In the further works we will expand this to more generic behavioral types and test in a large robotic swarm.

1. Bonabeau E, Dorigo M, Theraulaz G: Swarm intelligence: from natural to artificial systems. New York, Oxford University Press, 1999.
2. Roma G-C, Gamble RF, Ball WE: Formal Derivation of Rule-Based Programs, IEEE Trans. Softw. Eng., 19(3):277-296, 1993.
3. Darley V: Emergent Phenomena and Complexity. In Proc. of Alive IV, 1994.
4. Kornienko S, Kornienko O, Levi P: Multi-agent repairer of damaged process plans in manufacturing environment. In Proc. of IAS-8, Amsterdam, 485-494, 2004.
5. Charikar M, Lehman E, Liu D, et al.: Approximating the smallest grammar. In Proc. of the 34th ACM symposium on Theory of computing, 792–801, 2002.
6. Haken H: Advanced synergetics. Springer ,Berlin, 1983.
7. Pfeifer R, Iida F: Embodied artificial intelligence: Trends and challenges. In Iida (ed) et al. Embodied artificial intelligence, 1-26, Springer, 2004.
8. Kornienko S, Kornienko O, Levi, P: Generation of desired emergent behavior in swarm of micro-robots. In Proc. of ECAI 2004, Valencia, Spain, 2004.
9. Kornienko S, Kornienko O, Levi P: Collective AI: context awareness via communication. In: Proc. of IJCAI 2005, Edinburgh, 2005.