Synchronized dispersion of robotic swarms using XP colonies

Andrei George Florea, Cătălin Buiu
Department of Automatic Control and Systems Engineering
Politehnica University of Bucharest
Bucharest, Romania
{andrei.florea,catalin.buiu}@acse.pub.ro

Abstract – Robotic swarms are multi-robot systems with no central coordination. The swarm as a whole is capable of displaying complex behaviors even if the robots are very simple and limited in sensing capabilities. The interaction between robots is only local and may also take place indirectly through the environment. The main contribution of this paper is that it demonstrates that membrane computing (XP colonies in this case) is a valid instrument for modeling and control of robots in a swarm. More, as membrane systems are parallel and distributed models, this makes the use of XP colonies to control robots in a swarm a valuable approach to the control of swarms of hundreds and thousands of robots where the parallelization of the control is a must. A case study on the synchronized dispersion of robots in a swarm is presented together with demonstration videos.

Keywords -- swarm robotics; membrane computing; P/XP colonies; dispersion; synchronization

I. INTRODUCTION

A robotic swarm is a multi-robot system composed of very simple robots (agents). The size of the swarm can be up to hundreds and thousands of robots which together with the lack of centralised control make the control of such swarms challenging. Swarm intelligence principles say that the use of simple control mechanisms together with direct and indirect local interactions between the agents would generate complex macroscopic behaviors, such as self-assembly and pattern formation [1]. Robustness, scalability and flexibility are key features of swarm robotic systems [2]. Swarm robotic systems are able to display a wide range of nature inspired behaviors, such as: dispersion, flocking, foraging, collaborative search [3]. More, the use of swarm intelligence allows the dynamic task allocation [4] or the distributed localization [5] in swarm robotic systems. Stigmergy is a central concept in swarm robotics and can be defined as a mechanism of indirect coordination between agents in which a trace left by an agent in the environment will trigger subsequent actions of the agents in the swarm [6].

P colonies are representative for the membrane computing paradigm and are formal models that take further inspiration from grammar systems called colonies [7]. They are based on the same ideas as swarm robotics: the use of simple agents, placed in a shared environment, with simple interaction rules would generate emergent behaviors. This is why it appears to be natural the use of P colonies and their extensions to model and control swarms of robots. XP colonies add to the power of P colonies a stigmergic mechanism which as modeled through the use of exteroceptive rules [8] and enables a coordinated activity of the swarm.

This paper proposes the use of XP colonies to model and control a swarm of robots where the task for the robots is to disperse themselves in the environment in a synchronized way. The next section will describe the two basic models in our approach, P colonies and XP colonies, in a formal way. The third section of this paper will describe a series of experiments on simulated Kilobot robots, while the last section will give some concluding remarks and directions for further developments. The experiments are supported by online demonstration videos [9].

II. MEMBRANE COMPUTING

Membrane systems (or P systems) are parallel and distributed computing models inspired by the structure and functioning of biological membranes in a cell [10].

Definition 1. A P colony of capacity m is a construct:

$$\Pi=(A,e,f,B_1,K,B_m)$$

where A is an alphabet (a set of objects), e \(\in\) A is the basic object of the colony, \(f \in A\) is the final object of the colony, and \(B_i (i=1..n)\) are agents. Each \(B_i\) is of the form \((O_j,P_i)\), where \(O_j\) is a multiset of \(c\) copies of the basic object \(e\) (the initial state of the agent) and \(P_i = \{p_{i,1},p_{i,2},K,p_{i,k}\}\) is a finite set of programs. Each program \(p_{i,j}\) consists of \(m\) rules. There are two basic types of rules: evolution rules which take the form \(a \rightarrow b\) (rewriting object \(a\) into object \(b\)) and communication rules of the form \(c \leftrightarrow d\) (object \(c\) from the agent will move into the environment and object \(d\) from the environment
will move into the agent). A program where the agent can check for the presence in the agent of a given object \( c \) is a checking program and it is of the following form: \( <a \rightarrow b; c \leftrightarrow d'/c' \leftrightarrow d' > \). After the evolution rule \( a \rightarrow b \) is applied, if \( c \) is present in the agent, then \( c \) is exchanged with \( d \) from the environment; if not, the exchange \( c' \leftrightarrow d' \) will be performed.

Example 1. A simple P colony of capacity two, one agent and two programs (each having one evolution rule and one communication rule) is defined in [11] as:

\[
\Pi_x = (l_f, e, f, (e, e), \langle e, f \rangle, e \rightarrow f : e \leftrightarrow l_f, >, l_f \rightarrow e ; f, e \leftrightarrow e)\]  

Initially there are \( n \) objects \( f \) and one object \( l_f \) in the environment and two objects \( e \) inside the agent initially. Therefore, only the first program can be executed, resulting in the transformation of \( e \) into \( f \) and the exchange of the other internal \( e \) with the external \( l_f \). In the next step, only the second program can be executed: \( l_f \) will be transformed into an \( e \), and the internal \( f \) will be exchanged with an external \( e \). So, there will be \( n+1 \) objects \( f \) in the environment and no \( l_f \). Consequently, neither of the two programs can be applied anymore and the computation halts.

XP colonies were introduced in [12] as extensions of the P colonies. An XP colony allows a stigmergic mechanism based on exteroceptive rules of the form \( c \leftrightarrow d' \), where an object \( c \) appearing inside the agent will move into the global environment (the environment shared by multiple XP colonies) and another object \( d' \) appearing in the global environment will move into the agent. An exteroceptive checking program is of the following form \( <a \rightarrow b; c \leftrightarrow d'/c' \leftrightarrow d' > \) and will allow the object \( c' \) from the agent to be exchanged with object \( d' \) from the global environment if object \( c \) is not present in the agent or object \( d \) is not present in the global environment. P swarms are defined in [8] as colonies as XP colonies that communicate using the stigmergic mechanism allowed by the use of exteroceptive rules.

III. CASE STUDY

We now present a case study of signaling and synchronization between robots controlled with XP colonies. This will allow the robots to disperse in the environment in a synchronized way. First, the Kilobot robot will be described briefly, and then a series of two experiments will be presented. Demonstration videos for all the experiments are available at [9].

A. Kilobot robot

The Kilobot robot is an ideal robot for swarm robotics experiments, as it is a low-cost, basic robot which was designed to allow researchers to test distributed control algorithms on swarms of hundreds and thousands of robots [13]. A Kilobot robot (Fig. 1) has two vibration motors which allow differential drive, an infrared communication system which allows neighbor-to-neighbor communication and distance sensing, and an ambient light sensor.

A P colony with five agents is proposed (Fig. 2) to model the Kilobot robot and it is described in detail in [12]. All the remaining experiments are performed using V-REP, which is a robot simulator with an integrated development environment [14].

B. Basic scenario

In this first two-robot scenario, the first robot moves forward 15 steps and then signals the second robot to start moving left through the use of exteroceptive communication rules and by exchanging objects using the global environment. This concept enables objects to be transmitted from colony to colony in a P swarm in the same way as they are transmitted from agent to agent in a P colony. The control code is shown in Fig. 3 and a demonstration video is available at [9].

The first robot moves forward 15 steps until it has consumed all \( f \) objects. At this point, the checking rule at line 4 returns one \( e \) object because there are no more \( f \) objects in the environment. Because of the presence of both an \( e \) object and an \( m_s \) object in the \( \text{command} \) agent, the program at line 6 is executed and prepares the signal object \( s_L \) for sending. Finally, the program at line 7 simultaneously sends the \( m_0 \) object to the P colony environment in order to stop the robot and also sends the signal object \( s_L \) to the global P swarm environment.

Initially, the second robot has to wait for the signal to start moving left. Due to the way in which the simulation loop is constructed, the second XP colony will stop (and never restart again) if it does not execute any program during the waiting period imposed by the synchronization.
Synchronized dispersion of robotic swarms using XP colonies

Figure 2. Kilobot P colony-based modular controller structure

```c
1 xp_nimura_straight = |
2     command = (c, e);
3         < c->e, e->>c L, >
4         < c->l_m_d, e->>c e, >
5         < l_m_e, e->>c e, >
6         < c->l_m_e, e->>c e, >
7         < s->l_m_e, l_m_e, e->>c e, >;
8 |
9 xp_nimura_left = |
10     command = (c, e);
11         < c->e, e->>c L, >
12         < c->l_m_d, e->>c e, >
13         < l_m_e, e->>c e, >
14         < c->l_m_e, e->>c e, >
15         < s->l_m_e, l_m_e, e->>c e, >;
16         < e->e, e->>e R, >
17         < s->l_m_e, l_m_e, e->>c e, >;
18         < e->e, e->>e R, >;
19         heartbeat = (c, e);
20         < c->e, e->>c L, e->>c R, >
21         < s->l_m_e, l_m_e, e->>c e, >;
22         < e->e, e->>e R, >;
23         < s->l_m_e, l_m_e, e->>c e, >;
24         < e->e, e->>e R, >;
25    ;
```

Figure 3. Summary of the Lulu input file that defines two XP colonies that interact through the use of exteroceptive rules and the global P swarm environment.

For this reason, we have defined the **heartbeat** agent, which has the sole purpose of keeping the XP colony alive by continually exchanging one object (W) with the environment. This agent will exit the loop as soon as the signal object (s_L) is picked up from the global P swarm environment by the **command** agent [through the exteroceptive rule (=<>) on line 7] and published in the local XP colony environment. After the signal has been received, the second robot can start consuming f objects and turning left.

This technique of exchanging signal objects with the use of exteroceptive rules can be regarded as a simple and efficient MPI (Message Passing Interface) that allows the development of more complex swarm algorithms that depend on direct and indirect communication between robots.

C. Synchronizing groups of robots and individual robots

In the second experiment we implement two types of synchronizations, group to group and robot to robot, in an attempt to serialize two levels of coordination among members of a swarm. Screenshots of the different stages of the simulation are presented in Fig. 4.

Figure 4. Screenshots from different stages of the simulation: at simulation start (a), after the group synchronized movement ends (b), during the individual movement of the first group, the third robot is preparing to start (c), at the simulation end (d)

The objective is to move 9 robots (aligned as in Fig. 4a) in groups of three, one group at a time, for fifteen steps. The groups are numbered top-down from 0 to 2. Once this first stage is finished (Fig. 4b), in the second stage, each robot from the top-most group (0) starts an individual move of fifteen steps forward, again one after another. Because the first group has to make two displacements, we use two counter variables, f and g, that are decreased at each movement step, when in the group or individual stages respectively.

In order to achieve inter-group signaling, each group of three robots executes the same P colony out of a P swarm of three P colonies and each robot from group i-1 sends a signal into the P swarm global environment. Robots from group i constantly poll the global environment for one of these signal objects. Due the fact that there are three emitters and three receivers, all robots from group i are signalled, independently of the order in which signals are received. For example, each robot from the group 0 (group_forward) that finishes the group movement (all f objects are consumed) will publish a signal object using program (3). Robots from group 1 (group_left) will check the global environment using program (4). From then on, the signal object is processed and transmitted in the P colony environment to trigger any other waiting agent, in this case the heartbeat agent that has to be stopped.

$$< s_{from\_g\_forward}, e->e, e->e > \quad \text{(3)}$$

$$< e->e, e->e, s_{from\_g\_forward}, e->e > \quad \text{(4)}$$

On the other hand, for the individual movement, this scalable (id independent) signaling method was not applicable because the robots had to start in order. For this reason, the my_id_%id object was defined in the P colony environment of the first group with the meaning that at runtime, this object will be replaced by for e.g my_id_2 if the executing robot’s id is 2. This concept enabled the development of rules that take into account the identity of the robot, such as (5). Similarly, the my_id_ * object is expanded to all other objects that correspond to robot ids from the P colony, excepting the executing robot’s id. Due to the fact that in the P colony environment, each robot has an object that corresponds to the unique robot id, we can simply test if this is the first robot from the first group (group_forward) and it has received a signal from the last group (group_right).

$$< e->individual, e->e, e->my\_id_0, e->a\_from\_g\_right\_to\_forward_0 > \quad \text{(5)}$$

Part of the control code is given in Fig. 5 where the definition of the command agent from the forward moving group is given. The primary task of this agent is to subtract f objects and move each robot from the group forward (lines 32-35). This agent is also responsible for the communication with other P colonies, depending on the id of the executing robot. For robot 0 for e.g the program on line 11 would check if the last robot from group 2
Andrei George Florea, Cătălin Buiu

(group_right) has finished moving and if so, start the individual movement. Robot 1 on the other hand will be waiting for a signal from robot 0 (line 16) in order to trigger the individual movement.

The full control code for this use case scenario is available at [9].

```python
1 group_right
2 if robot 0 has finished moving and if so, start the individual movement. Robot 1 on the other hand will be waiting for a signal from robot 0 (line 16) in order to trigger the individual movement.
3
4 The full control code for this use case scenario is available at [9].
```

IV. CONCLUSIONS AND FURTHER DEVELOPMENTS

This paper presented a membrane computing based approach for the synchronized dispersion of robots in a swarm. XP colonies are used to model and control Kilobot control to achieve the proposed task. The use of ectoreceptive rules in XP colonies allows the synchronization of the robots in the swarm through a stigmergic mechanism. Demonstration videos for various synchronized dispersion patterns for the swarm support this approach. Further developments will include testing all the algorithms of a swarm of 100 real Kilobot robots, and the hardware implementation of a XP colony based controller.

ACKNOWLEDGMENT

This work was supported by the grant of the Ministry of National Education CNCS-UEFISCDI, project number PN-II-ID-PCE-2012-4-0239, Biomicspried techniques for robotic swarms security (Contract #2/30.08.2013).

REFERENCES