A mathematical model adapted to a P swarm to detect malevolent actions

C. E. Cipu and S. M. Bibic

Abstract. Robotic swarms could solve, without the need for a central controller, a wide range of difficult applications in real-world environments. In the first part of paper is extended the notion of a P colony to a colony of P colonies used for robotic swarms, called P swarms. The second part of our research is based on the basic principles of NSAs (negative selection algorithms, in AIS field) to develop an computing algorithm for Pcol automaton. In this paper we define a mathematical model bio-inspired for security issues of robotic swarms, which combining computational models with evolutive systems.

M.S.C. 2010: 68N30, 68T40, 92-08, 92B20, 93C85.
Key words: robotic swarm, string metrics, boundary functions.

1 Introduction

The field of study and design of the collective robotic systems without centralized control is called swarm robotics. Hierarchically, a robotic swarm may be described in terms of sub-swarms and neighborhoods. Thus, each sub-swarm is a subset of the swarm with its own objective and can be decomposed in several neighborhoods based on spatial or logical criteria on different P colonies. Inspired by the structure and functioning of living cells as well as by the way cells are organized in tissues or other higher forms of organization, in 2000 was introduced the concept of *membrane computing* (Gh. Păun, [22]). Later were defined the P systems, where the computing processes is separated in different compartments (membranes) which are able to inter-communicate. There exists interesting analogies between P systems (and their extensions, such as P colonies and Pcol automata) and secure robotic swarms see [1] which are exploited in this paper.

This paper is organized as follows. Section 2 gives an overview of swarm robotics in terms of concepts, applications and challenges. Security of robotic swarms is identified as a key issue in designing robotic swarms and the core of this paper is a new way to

BSG Proceedings 22. The International Conference "Differential Geometry - Dynamical Systems" DGDS-2014, September 1-4, 2014, Mangalia-Romania, pp. 18-27.
© Balkan Society of Geometers, Geometry Balkan Press 2015.

look at this problem which is based on membrane computing. Also, in this section are presented the concepts of P colonies and P swarms and some theoretical contributions. Next, Section 3 introduces string distance metrics, which were used in our study, and the simulation results are described are described. Finally, some conclusions and directions for future work are given in Section 4.

2 Robotic swarms

A robotic swarm is defined as a self-organizing multi-robot system characterized by simplicity of individuals, local sensing and communication capabilities, parallelism in task execution, robustness, scalability, flexibility and decentralized control. The collective behavior of the robot swarm emerges from the interactions between robots, respectively with the environment. In most cases, a robotic swarm is composed of homogeneous robots, although there are a number of works involving heterogeneous robot swarms [25]. Swarm robotics is a natural swarm intelligence system validated on natural instances, e.g., relationship between a model of self-enhanced aggregation for social insects and the behavior of a cockroach-like-robotic swarm described in [12]. There are significant results in what concerns the conditions under which some complex social behaviors might result out of an evolutionary process. Robot swarms are used to study the evolution of communication and collective decision making [14]. Collective robotic systems based on swarm intelligence concepts are fault tolerant, scalable and flexible [1].

Swarm robotics has been studied in the context of the following tasks: *aggregation* - which deals with spatially grouping all robots together in a region of the environment, *pattern formation* - position of the robots must follow a geometric pattern, *object clustering and assembling* - the robots must picking up objects and assemble them in a specific region, *path planning* - used for mapping or obstacle avoidance, or *consensus achievement and collective decision making*, see also [21]. Now, the research in the swarm robotics field it focuses on the development of tools and methods to solve real problems from engineering field [2]. One of the priority issues is related to security of the robotic swarm[3]. In this regard, a secure robotic swarm must be able to detect foreign robots and reject them (intrusion detection). By analogy with the immune system, where we denote by *non-self* that which is identified by the immune system as foreign to the body, in a robotic swarm one may speak of *self robots* (robots of the original swarm) and *non-self robots* (they are robots which have managed to breach the physical and/or logical barriers of the swarm and to join the swarm with the goal of imposing their own objectives).

2.1 P colonies

As a generalization of a membrane computing model, the concept of P *colony* is introduced in [18] and inspired by the notion of colonies of simple formal grammars ([17], basics of grammar systems [4], and eco-grammar systems [5], to use as simple as possible agents which are placed in a shared environment. Its components include objects (e.g., symbolic counterparts of chemical compounds), agents (internally structured active entities), and programs. So, in this computing model agents are single cells containing a small number of objects which are processed by different types of

programs [19, 6]. P colonies can be analyzed in parallel and sequential computing mode [11, 20], depending on the number of agents acting in a single derivation step. In the parallel derivation mode is assumed that each agent which can apply any of its programs must non-deterministically choose one and apply it at the same time with all the other agents; when using the program, all its rules are applied, each for different objects of the agent. In the sequential derivation mode one agent uses one of its programs at a time; in this case, contents of the agent and the environment can be changed in a single step. Calculation stops when no agent can apply any of its programs to existing object. *Pcol automata* are combining properties of finite automata and P colonies [23, 8, 7, 20]. In the following, the basic definitions and notations for a P colony will be presented.

Definition 2.1 (P-colony)**.** A P colony of capacity k, degree n and height h is a structure of the form

$$\Pi(k,\, n,\, h) = (\Sigma,\, e,\, f,\, I_E,\, B_1,\, B_2, \ldots,\, B_n)\ ,\ n \geq 1\,, \tag{2.1}$$

where Σ is an alphabet (finite nonempty set of abstract symbols, its elements are called *objects*), $e \in \Sigma$ is the basic object (environment object) of the colony, $f \in \Sigma$ is the final object of the colony, $I_E \in (\Sigma - \{e\})^*$ is a finite set of objects placed in the environment, $(B_i)_{i=\overline{1,n}}$ are cells or agents, each B_i has of the form $(O_i,\, P_i)$ where O_i is a multiset of the basic object e (the initial state of the agent or cell, and $|O_i| = k$) and $P_i = \{p_{i,1},\, p_{i,2}, \ldots,\, p_{i,k_i}\}$ is a finite set of programs. Thus, capacity k is number of objects belonging to each agent, n represents total number of agents and h represents the maximal number of programs of the agents in Π.

Each program $p_{i,j}$ consists of k rules (not necessary different). The basic types of rules are: *evolution or rewriting rule* (an internal object will be transformed into an internal object), *communication rule* (an internal object is sent outside the agent, to environment, instead of object, which is existing in the environment) and *checking or priority rule* (the communication rule can be chosen from two possibilities with the first one having higher priority: the agent checks the possibility to apply the communication rule having higher priority, else, the other rule can be applied [16]). A P colony works as follows: a computing algorithm (parallel or sequential derivation mode) is a sequence of transitions that begins with the initial states of the agents and of the environment. At each time step, any agent which can make use of any of its programs should use it, and when using a program, every rule has to be applied (separately) to the distinct objects of the agent. Using the programs $p_{i,j}$, actually a P colony evolves from a configuration to another, and the algorithm stops when no agent can use any program (`halt` condition is associated with the number of copies of the object f). The configuration of a P colony represents the content of the agents and the environment and formally it is expressed as a $(n+1)$-tuple (consisting of objects that appear within each agent and the environment) as

$$(w_1,\, w_2, \ldots, w_n;\, w_E \cdot e^{\omega})\ ,\ n \geq 1,\, \omega \geq 0\,, \tag{2.2}$$

where: w_i is a sequence of objects of the agent B_i, $i = \overline{1,n}$, and $|w_i| = k$ is the number of objects that make up the sequence w_i, w_E represents the sequence of objects that are placed in the environment which always contains an arbitrary number ω of copies of the basic object e.

Example 1. P **colony with reading tape rules**

Let's consider a P colony of the form

$$\Pi = (\{a,\, b\, c,\, d\},\, e,\, \varepsilon,\, (w,\, P)\,,\, F) \quad ,\; w = ea\,,$$

The programs are defined as follows

$$P = \Big\{ p_1 :< e \xrightarrow{T} a\,;\, a \leftrightarrow e >,\, p_2 :< a \xrightarrow{T} b;\, e \leftrightarrow a >,\, p_3 :< a \xrightarrow{T} b\,;\, b \leftrightarrow a >,$$
$$p_4 :< a \xrightarrow{T} c\,;\, b \to b >,\, p_5 :< b \xrightarrow{T} c\,;\, c \leftrightarrow b > \Big\}\,.$$

We are using two type of rules: *evolution* rule ($\square \to \triangle$) and *communication* rule ($\square \leftrightarrow \triangle$). The result of computation (final object) is of the form

$$F = \{(\varepsilon\,;\, u\; cb)\,,\, (\varepsilon\,;\, u\; ea) \mid\; u \in \{c\}^*\}\,.$$

Each step of computation is described in Table 1.

Step	Program	Cell	Unread part of tape	Env.
1	p_1	ea	aabbcc	ε
2	p_2	ae	abbcc	a
3	p_3	ea	bbcc	aa
4	p_4	ab	bcc	a
5	p_5	ba	cc	b
6	p_5	bc	c	b
7	-	cb	ε	e

Table 1. Configurations of the P colony from Example 1.

2.2 P swarms

A P swarm is a colony of several P colonies in order to allow for each member P colony to model a sub-swarm of the original swarm. The sub-swarms are forming and evolving dynamically in order to assure the security of the swarm and to expel intruders, which means that if the intruders are not recognized as partners in changing information in a direct peer-to-peer way or indirectly through the environment, then they are expelled from a logical point of view. If there are N robots in the swarm at the beginning, there will be N sub-swarms, P colonies (of one robot each) initially. Then based on some *adoption* rules, each sub-swarm will try to adopt robots in its sub-swarm. Those robots which will fail to do that will get adopted in other robots' sub-swarms. Consequently, there will be *abandoning* rules by which robots will not be members of the sub-swarm anymore. Finally, robots that will not get adopted by any of the P colonies will be considered adverse robots and any direct or indirect communication with these robots will cease. Formally speaking, a P swarm is a construct of the form

$$\mathbf{\Lambda} = (\Sigma,\, e,\, f,\, \Pi_i) \tag{2.3}$$

and Π_i are P colonies of the form (2.1). As a simple example, Fig. 1.a) presents a P swarm with 3 P colonies inside (each one corresponding to a sub-swarm) and Fig. 1.b) presents the modularity structure of a P colony, see also [1].

Rules inside each sub-swarm have the following form in order to assure the proposed functionalities: *adoption rule* $a \to ab$ (it means that automata a will adopt automata b in its sub-swarm) and *abandoning rule* $cd \to c$ (in this case automata c will expel automata d from its sub-swarm).

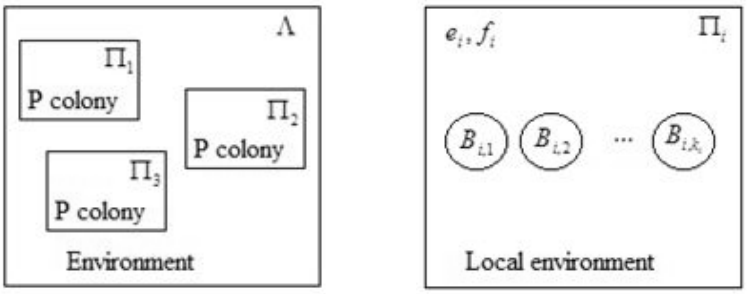

Figure 1. a) A simple P swarm $\not\geq$ with three P colonies inside (Π_1, Π_2, Π_3)**; b) The structure of a module of a P colony Π_i.**

3 String distance metrics. Relation between robots

The issue of similarity is aimed by searching of objects that are most similar to the query object, and each object is ranked in database. In other application areas, the technique is known as pattern matching or signature analysis. Some of its advantages are: one known active object becomes the search key, setting of the limits on output, possibility to determine the degree of similarity that it's quantified by similarity coefficient. In general, this method has a subjective nature.

Definition 3.1 ([13])**.** Let's consider two data objects $\{X, Y\}$[1], where X_i is the number of objects present in X and absent in Y, Y_j is the number of objects absent in X and present in Y, $\alpha = X \cap Y$ is the number of objects common to both data, and $\beta = X \backslash Y$ and $\gamma = Y \backslash X$ are the numbers of objects absent from both data. In this respect, the similarity measure is given by the number $S(X, Y) = \Delta(\alpha, \beta, \gamma)$ and measure the present and the absent matches, respectively $D(X, Y) = 1 - S(X, Y)$ represents measure the corresponding mismatches, i.e., dissimilarity.

In literature [15, 9] are proposed several models to measure the similarity between strings, based on the properties of a metric (non-negativity,identity, symmetry, triangle inequality). For example, well-known string metrics: *Hamming* distance (gives minimum number of point mutations required to transform one string data instance into another) and several extensions to this basic metric (*fractional Hamming distance* or *relative distance* and *Rogers & Tanimoto* distance), *Levenshtein* or *edit* distance (gives minimum number of character-level operations *insertion*, *deletion*, or *substitution* needed to transform one string into the other), *cosine similarity* (string

[1] i.e., characteristics, features

is converted to vector to get degree between strings) or more complex metric *Pearson correlation coefficient* (measures the correlation coefficient for two strings).

Considering a P swarm with a default collective behavior, in order to describe the interaction between two or more agents, the main problem is to measure the distance between them, namely to compare their characteristic configurations. Thus, the aim is to obtain a model of a robotic swarm for recognizing self/nonself actions. In this way, we pursue a swarm that must be grouped in a limited region of the environment. Formally speaking, to calculate the distance between two robots is equivalent with to measure the similarity between their characteristic configurations. Practically, this can be determined by adapting specific formulas of the distance-based family of similarity metrics for calculating the distance between strings. A configuration of a Pcol automaton is defined by a string containing words and letters from its alphabet. For this purpose, we define the distance between two agents as follows

Definition 3.2 (*Distance between two robots).* The distance between two agents $B_{i,j}$ from Π_i and $B_{k,l}$ from Π_k is given by the relation

$$\Delta\left(B_{i,j}, B_{k,l}\right) = \Delta\left(\{e_{ij}f_{ij}\}, \{e_{kl}f_{kl}\}\right) , \tag{3.1}$$

over all configurations obtained at the same step.

Definition 3.3 (*Relation between two robots).* We say that $B_{i,j}$ from Π_i and $B_{k,l}$ from Π_k are related (i.e., they will adopt each other), and is denoted by $B_{i,j} \circledcirc B_{k,l}$, if $\Delta\left(B_{i,j}, B_{k,l}\right) \leq \varepsilon$, where $0 \leq \varepsilon \leq 1$. Otherwise, $B_{i,j}$ and $B_{k,l}$ are rejecting each other.

Remark 3.4. If $\varepsilon = 0$, the relation between robots " $\circledcirc$" is reflexive, symmetric and transitive.

In fact, in order to determine the distance between robots means computing the distance between multi-token strings, beyond what is reduced. A specific distance between robots must be no grater then a fixed value. We define the problem as follows: for A, B non-disjunctive sets of strings of the same length over some alphabet Σ, let consider function of similarity $\Psi_\delta : A \times B \to I$ and $\Psi_\delta(X, Y) = 1 - X \circledcirc Y$, $I \subseteq \mathbb{R}_+$, and δ is the threshold value, $\delta \in [0, 1]$. If $\delta \leq \Psi_\delta(X, Y) \leq 1$, then X and Y are similar, else they don't match.

Example 2. Relation between robots.

In the following, we consider two strings of the same length, i.e., $s^1 = (aabbcd)$ and $s^2 = (ababcd)$. Thus, in Table 2, approximating values of string similarity, respectively dissimilarity between the sequences s^1 and s^2, are presented. In Table 2., we illustrated step by step the calculus of distances `LD` and `LCS` (longest common substrings distance). The `LCS` metric recursively finds and removes the longest common substring in the two strings compared. If we denote $t = lcs(s^1, s^2)$ the first longest common substring for s^1 and s^2, then the string denoted by s^1_{-x} is obtained via removing from s^1 the first occurrence of x in s^1 (set to 2), and `LCS` metric is computed as

$$LCS\left(s^1, s^2\right) = \begin{cases} 0, & \text{if } |t| \leq k \\ |t| + LCS\left(s^1_{-t}, s^2_{-t}\right), & \text{else} \end{cases} , \tag{3.2}$$

String distance function	Similarity value	Dissimilarity value
Fractional Hamming distance	-	0.0833
Rogers & Tanimoto distance	-	0.1538
Hamming distance	-	4
Levenshtein (edit) distance	0.6667	-
Cosine Similarity	0.7368	-
Pearson Correlation Coefficient	0.8258	-

Table 2. The value of distance between sequences s^1 and s^2.

where the value $k \in \{2, 3\}$ and time complexity of `LCS` is $O\left(|s^1| \cdot |s^2|\right)$. It's possible to expand `LCS` formula by additional weighting of the $|t|$. The rule is to penalize longest common substrings which do not match the beginning of a token in at least one of the compared strings and t is assigned the weight

$$w_t = \frac{|t| + y - \max(x,\, y)}{|t| + y}\,, \tag{3.3}$$

where y represents the maximum number of non-whitespace characters, which precede the first occurrence of t in s^1 or s^2. From the point of view of the robotic swarm

LD	**-**	**a**	**a**	**b**	**b**	**c**	**d**
-	0	1	2	3	4	5	6
a	1	0	1	2	3	4	5
b	2	1	1	1	2	3	4
a	3	2	1	2	2	3	4
b	4	3	2	1	2	3	4
c	5	4	3	2	2	2	3
d	6	5	4	3	3	3	2

LCS$^{(1)}$	**-**	**a**	**a**	**b**	**b**	**c**	**d**
-	0	0	0	0	0	0	0
a	0	1	1	1	1	1	1
b	0	1	1	2	2	2	2
a	0	1	2	2	2	2	2
b	0	1	2	3	3	3	3
c	0	1	2	3	3	4	4
d	0	1	2	3	3	4	5

LCS$^{(2)}$	**-**	**a**	**b**	**a**	**b**	**c**	**d**
-	0	0	0	0	0	0	0
a	0	1	1	1	1	1	1
a	0	1	1	2	2	2	2
b	0	1	2	2	3	3	3
b	0	1	2	2	3	3	3
c	0	1	2	2	3	4	4
d	0	1	2	2	3	4	5

Table 3. (a) Simulation the *edit*-distance with backtrace; (b-c) Longest common subsequence of s^1 and s^2.

behavior considered, the major problem is to control of cooperative mobile robots. A possible solution is the development of control algorithms navigating of a swarm of robots into a predefined 2D shape while avoiding inter-member collisions [10]. The swarm model with basic artificial forces is based on a mathematical analysis of the

swarm behavior, the following aspects being considered: agents have identical physical properties (e.g., mass, mobility, etc.), instantaneous and error free localization capabilities, the communication network of the members can transmit data to all the members within the group instantaneously (i.e., without delay), and are acting on a flat surface without obstacles. With respect with these requirements, are defined the dynamics of a robotic swarm, artificial forces as exponential functions in order to navigate the robots and which have own types of control parameters (magnitude and response factors), and behavior of the robotic swarm guided by these forces. In the paper [10] is introduced a modified controller to avoid obstacles, change agent size and actuator limitations, and are formulated the additional constraints affecting the performance of the control algorithm together with an implementation procedure. In [24], modifications of the control algorithm are specifically designed to introduce the repulsion force; its components have the following properties: one avoids the robot from collision, while the other pulls the robot to escape from the local minima nearby large obstacles. The main purpose is to design a moving shape, sensitive to a potential change.

Definition 3.5. We define the 2D shape contour Γ as

$$x(t) = a[b(n-1)cos(t) + c\,cos((n-1)t)],\; y(t) = a[b(n-1)sin(t) - c\,sin((n-1)t)]\,,$$

where the parameters n, a, b, c have the following interpretation: $n \cdot b$ describes the number of edges in contour, $c < 1$-makes the edges round ($c = 0$ means a circle), and $a > 0$ defines the overall size of the shape.

In our study we consider $a = 1, b = 1, n = 6, c \in [0, 1)$, see Fig. 3, where the "■" point is the center and the "•" points are the robots, inside or outside the the 2D shape contour Γ.

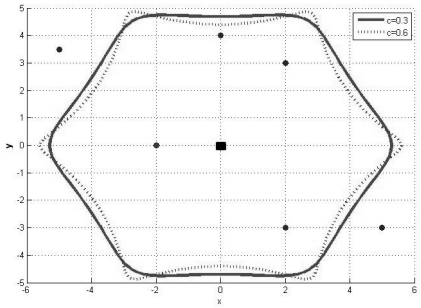

Figure 2. $2D$ **shape contour** Γ**.**

4 Conclusions

In this paper, we investigated the usefulness of some string distance metrics to compute distances between robots. The bio-inspired computational model represents a new idea and it can be applied to assess the behavior of a robotic swarm such as discriminating between self and non-self robots. In fact, the computing algorithm is

very simple and based on evaluating similarity/dissimilarity between strings which represent configurations of the respective automata. The results presented in Table 3. represent a starting point for developing a workable model of computation. In this direction, the research is focused on the development of a P colony and P automata simulator, finding a method to use distances between agents for other applications such as path planning and collective decision making and to test this new model on a real robotic swarm.

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

References

[1] G. Beni, *From swarm intelligence to swarm robotics*, Swarm Robotics, Lecture Notes in Computer Science, LNCS 3342 (2005), 1-9.

[2] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, *Swarm robotics: a review from the swarm engineering perspective*, Swarm Intelligence, 7, 1 (2013), 1-41.

[3] C. Buiu, M. Gansari, *A new model for interactions between robots in a swarm*, Proceedings of ECAI 2014, 6th International Conference on Electronics, Computers, and Artifical Intelligence, Bucharest, Romania, October 23-25, 2014.

[4] E. Csuhaj-Varju, J. Dassow, J. Kelemen, G. Paun, *Grammar systems: a grammatical approach to distribution and cooperation*, Topics in Computer Mathematics 5, 1994.

[5] E. Csuhaj-Varju, J. Kelemen, A. Kelemenova, G. Paun, *Eco(grammar) systems*, Proc. 12-th European Meeting on Cybernetics and Systems Research, Vienna, 1994, 941-948.

[6] E.Csuhaj-Varju, M. Margenstern, G. Vaszil, *P colonies with a bounded number of cells and programs*, Lecture Notes in Computer science, LNCS 4361 (2006), 352-366.

[7] L. Cienciala, L. Ciencialova, *Eco-P colonies*, Lecture Notes in Computer Science, LNCS 5957 (2010), 201-209.

[8] L. Cienciala, L. Ciencialova, E.C. Varju, G. Vaszil, *PCol automata: recognizing strings with P colonies*, Proceedings of the 8th Brainstorming Week on Membrane Computing, Sevilla, Spain, February 1-5, 2010, 65-76. ISBN: 978-84-614-2357-6.

[9] L.N. de Castro, *Fundamentals of natural computing: basics concepts, algorithms, and applications*, CRC Press 2007.

[10] S.W. Ekanayake, P.N. Pathirana, S. Sedwards, *Formations of robotic swarm: an artificial force based approach*, International Journal of Advanced Robotic Systems, IJARS 7, 3 (2010), 173-190.

[11] R. Freund, M. Oswald, *P colonies working in the maximally parallel and in the sequential mode*, Pre-proc. 1st Int. Workshop on Theory and Appl. of P Systems, Timisoara, Romania, Sept. 26-27, 2005, 49-56.

[12] S. Garnier, C. Jost, R. Jeanson, J. Gautrais, M. Asadpour, G. Caprari, G. Theraulaz , *Aggregation behaviour as a source of collective decision in a group of cockroach-like robots*, Advances in Artificial Life, Lecture Notes in Computer Science, LNAI 3630 (2005), 169-178.

[13] J. Gasteiger, T. Engel, *Chemoinformatics: a textbook*, Wiley-VCH 2003.

[14] J. Halloy, et al., *Social integration of robots into groups of cockroaches to control self-organized choices*, Science 318, 5853 (2007), 1155-1158.

[15] R.W. Hamming, *Error detecting and error correcting codes*, Bell System Technical Journal 26, 2 (1950), 147-160.

[16] J. Kelemen, A. Kelemenova, G. Paun, *The power of cooperation in a biochemically inspired computing model: P colonies. Preview of P colonies: a biochemically inspired computing model*, Workshop and Tutorial proc. 9-th Int. Conf. on the Simulation and Synthesis of Living Systems, ALIFE IX, Boston 2004, 82-86.

[17] J. Kelemen, A. Kelemenova, *A grammar-theoretic treatment of multiagent systems*, Cybernetics and Systems 63, 6 (1992), 621-633.

[18] J. Kelemen, A. Kelemenova, *On P colonies, a simple bio-chemicallyinspired model of computation*, Proc. 6-th Int. Symp. Hungarian Researchers on Computational Intelligence, Budapest, Nov. 18-19, 2005, 40-56.

[19] A. Kelemenova, *P colonies*, The Oxford handbook of Membrane Computing, 2010, 584-593.

[20] M, Langer, L. Cienciala, L. Ciencialova, M. Perdek, A. Kelemenova, *An application of th PCol automata in robot control*, Proceedings of the 11th Brainstorming Week on Membrane Computing, Sevilla, Spain, February 4-8, 2013, 153-164.

[21] A. Liekna, J. Grundspenkins, *Towards practical application of swarm robotics: overview of swarm tasks*, Proc. 13-th Int. Conf. Engineering for Rural Development, Jelgava, Latvia, May 29-30, 2014.

[22] G. Paun, *Computing with membranes*, Journal of Computer and System Sciences 61, 1 (2000), 108-143.

[23] G. Paun, G. Rozenberg, A. Salomaa, *The Oxford handbook of membrane computing*, Oxford University Press 2010.

[24] L. Qin, Y. Zha, Q. Yin, Y. Peng, *Formation control of robotic swarm using bounded artificial forces*, The Scientific World Journal, 2013. Article ID 194280, 15 pages.

[25] Y. Tan, Z. Zhong-Yang, *Research advance in swarm robotics*, Defense Technology 9, 1 (2013), 18-39.

Author's address:

Simona Mihaela Bibic, Elena Corina Cipu
University Politehnica of Bucharest, Faculty of Applied Sciences,
Department of Applied Mathematics,
Splaiul Independentei 313, Bucharest 060042, Romania.
E-mail: simona_bibic@mathem.pub.ro , corina_cipu@mathem.pub.ro