Geometrical applications in assessing the behavior of a robotic swarm

S. M. Bibic and E. C. Cipu

Abstract. Robotic swarms could solve, without the need for a central controller, a wide range of difficult applications in real-world environments. First part of the paper is dedicated to the connection between robotic swarm and membrane computing. The second part of our research is the development of a biologically inspired BIS approach for enabling the distributed security of a robotic swarm. Thus we propose in AIS an algorithm based on negative selection for Pcol automaton. In this paper we define a mathematical model for security issues of robotic swarms combining bio-inspired computational models with evolutive systems.

M.S.C. 2010: 68N30, 68T40, 92-08, 92B20, 93C85.
Key words: robotic swarm; negative selection algorithm; string metrics.

1 Introduction

The swarm robotics inspired from nature represents the study and design of collective robotic systems without relying on any form of centralized control. In a robot swarm which is composed of a large number of simple robots (or agents), the collective behavior of the robots results from local interactions between the robots and between the robots and the environment in which they act. The ideal robotic swarm is able to exhibit emergent intelligence and solve complex problems in a secure way. A robotic swarm may be hierarchically described in terms of *sub-swarms* (a subset of the swarm with its own objective and can be decomposed in several neighborhoods based on spatial or logical criteria) and neighborhoods.

Membrane computing (introduced by Gh. Paun, [33]) is an area of computer science that aims to synthesize ideas and computation models *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. The basic idea of P systems is to separate computing processes in different compartments (membranes) which are able to intercommunicate. In membrane computing there are evolution rules which allow objects to evolve, i.e. to transform themselves in other objects, to pass through a membrane, or to dissolve the membrane in which the specific object is placed. The objects are identified with symbols corresponding to a specific alphabet. This evolution takes

Applied Sciences, Vol.17, 2015, pp. 24-39.

place in parallel for all objects that are able to evolve at the current step. The result of computation in this parallel and distributed model is represented by the objects contained in a particular region.

There exists interesting analogies between P colonies and secure robotic swarms, see [3]: P colonies are based on the same ideas of swarm intelligence, P automata are combining features of classical automata and natural systems with a distributed architecture, the communication and processes are segregated in tasks and sub-tasks.

The structure of this paper is as follows. Next section will give 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 look at this problem which is based on membrane computing. Therefor an overview of P colonies is given in the final part of the section. Section 3 introduces our main theoretical contribution. Some conclusions and directions for future work are presented in section 4.

What we propose in this paper is to use concepts inspired by membrane computing, P colonies and bio-inspired computational models with evolutive systems to formalize this issue of intrusion detection and rejection and so to separate non-self robots from the original swarm.

2 Robotic swarms

A robotic swarm is 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.

Usually, a robotic swarm is composed of homogeneous robots, although there are a number of works involving heterogeneous robot swarms [38, 12]. Swarm robotics is a natural swarm intelligence system validated on natural examples, see for instance the relationship between a model of self-enhanced aggregation for social insects and the behavior of a cockroach-like-robotic swarm described in [17]. Also, there are relevant results in what concerns the conditions under which some complex social behaviors might result out of an evolutionary process, and robot swarms have been used to study the evolution of communication and collective decision making, see [35, 20]. 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 [31].

Lately, in the swarm robotics field, the research focuses on the development of tools and methods to solve real problems from engineering field [2]. However, the stringent issue of security of robotic swarm has been largely overlooked [3]. A secure robotic swarm must be able to detect foreign robots and expel them (intrusion detection). In an analogy with the human being 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 *nonself robots* (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).

P colonies

The concept of P *colony* is a generalization of a membrane computing model, [28] and took another source of inspiration from the notion of colonies of simple formal grammars (introduced by [27], see also [5], for basics of grammar systems, and [6], for eco-grammar systems), i.e. to use as simple as possible agents which are placed in a shared environment. The components or entities of a P colony include objects (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 [29].

Colonies of synchronizing agents (*CSA*), introduced in [8], consist of multi-sets of agents determined by objects that work together under global rules, i.e., all agents follow the same rules. Specific rules *CSA* are of two types: the evolution rules and synchronization rules. Application of synchronization rules changes the content of the two agents simultaneously. In [8] are determined some results related to computing power and robustness.

In [30], P colonies are analyzed in a parallel and sequential computing mode, depending on the number of agents acting in a single derivation step. In parallel mode each agent can apply any of its programs, chooses nondeterministically one of its programs and applies it, else an agent that is chosen nondeterministic may act. Calculation stops when no agent can not apply any of its programs to existing object.

Eco-P colonies are extensions of P colonies with dynamical evolution of environments [7, 9], which are based on the assumption that objects can act on the environment not only through the exchange of objects, but also directly, by mutations of external objects. In [15] there are compared the parallel and sequential derivation modes, and the new rule types for programs of P colonies. They show that P colonies of size at most 5, which work in sequential derivation mode, with priority rules, respectively in maximal parallel derivation mode, with programs without priorities in communication rules are completely calculable.

P *colony automata* are combining properties of finite automata and P-colonies [34, 10, 9, 30]. An interesting application of P colonies automata for robot control is presented in [30].

According to those specified above, we will describe further the basic definitions and notations for a P-colony.

Definition 2.1 (P-colony of capacity k)**.** A P-colony of capacity k is a structure of the form

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

where: A is an alphabet (finite nonempty set of abstract symbols, its elements are called *objects*), $e \in A$ is the basic object (environment object) of the colony, $f \in A$ is the final object of the colony, $I_E \in (A - \{e\})^*$ is a finite set of objects initially

present in the environment and different from e, $(B_i)_{i=\overline{1,n}}$ are cells or agents, each B_i being of the form (O_i, P_i) where O_i is a multiset of k copies 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.

Each program $p_{i,j}$ consists of k rules not necessary different. There exist three basic types of rules: *evolution or rewriting rule* $a \to b$ (an internal object a will be transformed into an internal object b), *communication rule* $c \leftrightarrow d$ (an internal object c is replaced with d: c is sent outside the agent, to environment, instead of object d, which is existing in the environment) and *checking or priority rule* $c \leftrightarrow d/c' \leftrightarrow d'$, that expands the skills of agents [26] (in this case is assumed that 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), for a, b, c, d, c', $d' \in A$.

Definition 2.2 (Degree and height of a P-colony)**.** For a P colony (2.1) there are defined the following notions: capacity k, degree n and height h of Π, as follows: k is number of objects belonging to each agent, n express total number of agents and h represents the maximal number of programs of the agents in Π.

At the beginning of the computation, the environment contains an arbitrary number of copies of the basic object e (also, each agent contains k copies of e). At each step of computation, the content of the environment as well as of the agents will change. 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.

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 as

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

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.

P colonies can calculate everything a Turing machine can compute. Thus, the P colonies are completely computable, i.e. universal. The universality is satisfied only if the programs contain checking rules.

3 Assessing the behavior of a robotic swarm

3.1 Biological model of an immune algorithm

Biological models that could form the basis of new paradigms in the field of *Natural Computing* are different. Evolutionary behavior of ant colonies and swarms of particles, structure and functioning of the brain, are some of the most studied bio-inspired

sources; generated general methods for solving (algorithms) of classes of difficult problems. One of the latest techniques is based on principles of biological immune systems, see [22, 13, 39]. The most representative applications are: pattern recognition, approximation of functions, numerical optimization, analysis and classification of data, automatic learning, anomaly detection, computer security and computer networks, control and planning, & c.

Biological immune system (BIS) is a collection of cells, molecules and organs involved in recognizing and combating certain disorders of the human body, being able to perform complex tasks: pattern recognition, learning and memory, tolerance to noise, generating diversity or optimization. System complexity is comparable to that of the human brain. The immune system response may be: *primary* (the reaction to the first contact with pathogens) and *secondary* (the body has memory and reacts faster). So secondary immune response is faster and more intense than the primary due to immunological memory cells (*lymphocytes*).

There are two categories of elements of type antigen that can be identified by the immune system: *self antigen* (cells as part of the human body, inoffensive, which are recognized by the immune system) and *nonself antigen* (elements that are foreign to the body). The main feature in the correct functioning of the biological immune system is the distinction between the two types of antigens (self and nonself). The recognition is based on binding the antigen by receptors in cells that ensure the functioning of BIS. The recognition of antigen precedes the immune response, which represents all the actions performed by immune system in order to defend the human body. The steps of the immune response: capture of the antigen by macrophages, fragmentation and transfer of the antigenic information to the immune system. Study of the lymphocytes behavior leads to different applications: classification tasks, and detection of anomalies in the behavior of some systems (malware and the computer/network security).

The receptors (antibodies) that correspond to a lymphocytes are generated by combining of gene sequences that are found in the body's self register. By combining sequences from the register, it ensures a wide variety of receptors. BIS model underpins the development of new computational techniques and construction of various evolutionary systems called AIS (*artificial immune systems*). To define and develop an AIS it will take into account: immunological mechanisms, their modeling to describe complex processes, understanding and exploiting a natural phenomenon that the immune response of an organism. Scheme of an AIS for solving a problem is the following: issue is interpreted as an antigen, and its solution is represented by antibodies generated from response. To develop an AIS algorithm are priority: data representation (the discrete representation, vector of binary values or discrete, or continuous representation, vector of real values), affinity between antibody and antigen which represents similarity between two configurations, threshold of acceptance the matching, choosing the appropriate model or algorithm (model of bone marrow, negative selection algorithm, clonal selection algorithm, artificial immune networks).

Negative selection algorithms (NSAs)

The first studies devoted to the implementation of an intrusion detection system inspired by BIS emerged in the late 90s and the central algorithm of the proposed

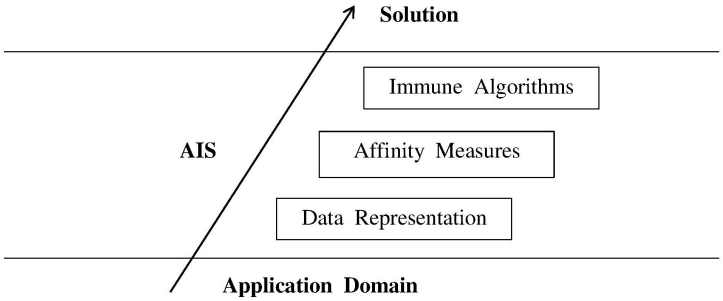

Figure 1: Determining the solution by using AIS-based model.

system is the one of *negative selection* that represents a process whereby the body is protected against self-reactive lymphocytes and uses the immune system's capacity to detect unknown antigens without reacting to the self-cells.

Negative selection principle (NS): in the thymus gland, T-cell maturation occurs, and this process resulted in the elimination of those T-cells that recognized the body's self elements. Virtually all T-cells leaving the thymus will enter the circulatory system having the property to tolerate the body's self elements. NS is a viable paradigm for pattern recognition by storing information about complementary set (*extraneous elements*) recognizable shapes (*self-elements*). *Negative selection algoritm* (NSA) is used for detection of anomalies or changes in AIS and consists of two steps: *generating or censoring stage* and *detecting or monitoring stage*. The NSA implies: *data representation* (the basis on that it can discuss the difference of various detector representations), *coverage estimate*, *affinity measure*, and *matching rules*.

Thus, different NSA are characterized by particular data representation schemes, matching rules and detector generation processes. The fundamental purpose of a NSA is to classify data. NSAs have been modified to handle data in string (alphabetic) representation and hybrid data, comprising both real-valued and string data representations. The detector generation and elimination mechanisms implemented in a NSA are a defining characteristic of the algorithm. For example, for string data representation are proposed randomized algorithms, i.e.,exhaustive algorithm and deterministic algorithms (linear time and greedy algorithm), and only random-based generation schemes have been implemented for real-valued vector data representation, see [25].

To date, numerous strategies were proposed for how the random generation of detectors are implemented. The classical approach is the random generation and elimination strategy. Among other approaches to detector generation one could include: evolutionary approaches such as genetic algorithms, one-shot randomized algorithms, optimization with aftermath adjustment [19, 25]. Distinction between (discrimination) *self* and *nonself* represents one of the major mechanisms in the complex immune system: these two terms are artificial labels given to the classification of data instances. Thus, *self* refers to standard incoming safe data, while *nonself* represents data considered malware or intrusive. First artificial NSA was designed as a change detection method (Forest et al., [16]). The most important characteristics of a NSA

are: negative representation of information, using some form of detector set as the detection mechanism and one-class classification.

Remark 3.1. Consider S the set of self-elements that can be recognized and will be protected or monitored (self-antigens). The set D is the set of detectors with the aim of identifying all antigens that do not belong to the set S, namely the extraneous antigens. The main steps of NSA are presented below and listed in Algorithm 3.1

a) generating stage:

the detectors are generated by a random process and censored by trying to match self-samples; those candidates that match are eliminated; the rest are kept as detectors;

b) detecting stage:

the set of detectors (collection of mature detectors retained in memory) is implemented in the detection phase; next step: each unknown data instance is analyzed by the detector set and classified as self or non-self; if the unknown data instance matches any detector in the detector set, then it is classified as non-self or anomaly; if the incoming data instance is not recognized by any detector, then it is assumed as its, i.e., self.

Algorithm 1 Basic negative selection algorithm

Data: set of self or normal data $S \subseteq U$, l, $r \in \mathbb{N}$, where l is the string length and r is a matching threshold

Result: set of detectors $D \subseteq U$

1 **begin**

2. Generate randomly string $d \in U$.
   If d doesn't match any string in S **then** $D := D \cup \{d\}$

3 Monitoring new sample $y \in U$ by permanently checking the detectors (in D) to y. **For** $d \in D$ **if** d recognizes the element $y \in U$ **then** classify it as nonself

4 **end**.

Another application of NSA is found in the literature dedicated to intrusion detection in computer networks. The BIS model provides some structural and functional similarities to those of a system that ensures the security of a computer network. Both systems monitors the body activity, respectively of the computer network, and by the specific actions reject potential attacks of intruders. The immune system and the intrusion detection system have the ability to recognize and distinguish between the two categories of items: self (body cells, respectively network activity) and foreign (antigens, respectively potential attacks).

Terminology in this area: *detector* (antibody), *self-samples* (self-cells), *incoming data instances* (antigen), *distance measure* (affinity measure in the shape-space: real-valued, integer, binary or symbolic), *r-contiguous*[1] *matching rule matching rule.*

[1] adjacent or neighboring

A significantly important factor in the performance of the NSA is the choice of matching rules implemented for data recognition. The choice of the matching rules or the threshold used in matching rules must be application specific and data representational dependent; the matching rule is a measure of distance, affinity or similarity that two data instances share.

3.2 Matching rules in data representation

Due to the fact that antigens and antibodies are characterized by their physicochemical binding properties, they are represented as points in an n-dimensional Euclidean space. In the terms of computational models, the notion of the affinity between antibodies and antigens is defined based on a distance measure between points in the *shape-space* or *representation space* (concept introduced by Perelson and Oster, [36]). Specifically, the distance between an antibody and an antigen is inversely proportional to the affinity between them; in some cases, coordinates are not given explicitly but the distance between antibodies and antigens is provided. Therefore, if Γ, X and $\Gamma_\varepsilon \subseteq \Gamma$ represent the shape-space, antigens, and affinity (coverage) of antibodies, respectively, then ε specifies a recognition threshold. Thus, if the affinity between an antibody and an antigen X is less than ε (i.e., the antigen lies inside the affinity region of an antibody), then the antigen is said to match (bind) the antibody (Balthrop et al., [4]).

The entities involved in immune algorithms are mainly B and T-cells, antibodies, and antigens. Representations that are used in most immune algorithms are: binary strings, strings over finite alphabets (other than binary), real-valued vectors, and hybrid representation where each entity consists of several features and each feature may be of a different type (e.g., integer, real value, boolean value, or categorical information). In literature are introduced different similarity or distance measures to define the notion of affinity between a T or B-cell and an antigen. The detector (T-cell model) is characterized by a set of attributes and a matching rule, based on a distance measure Gonzalez et al., [19]; it can be implemented as either a production rule or a neural network or a software agent and these implementations may be difficult to analyze (Hofmeyr, [22]). Usually, a detector describes a region of the shape - space and it may be defined by a formal representation of a subspace of the shape space. In the case of the binary representation, a detector may be defined by a binary string (*center* of the detector) and threshold value (*radius* of the detector); the detector will represent all the strings that are at a distance from the center, below considered threshold.

Notion of the matching rule represents the key concept in immune algorithms: is used both in detector generation (the mechanisms to generate the detectors), as well in detection. Regardless of representation, a matching rule M is formally defined as dMx and represents the affinity measure between detector d and data instance x if and only if x belongs to the set defined by the detector d, which it means that is necessary to define a notion of a point belonging to a set, see [25]. Unlike classical theory, a different notion like the one in fuzzy set theory may be used, when a degree of membership to the universal set is defined [19]. Some observations are required: for different data representations, the matching rules can be very different, the matching threshold applies the concept of partial matching: in representation scheme, the two

points which are considered matching are different, a partial matching rule is meant to be an approximation and a generalization, for real valued applications, the entire algorithm is meaningless without this generalization.

3.3 String matching rules

As previously shown, the matching rule which defines *matching* or *recognition*, and the *distance measure* that the former is based on, are the basics in any detection, classification, or recognition algorithms. Also, the choice of a matching rule depends on the representation scheme and type of data. For instance, NSAs were initially implemented to detect changes in string data, and in literature [21, 23, 37, 24] matching rules have been proposed for measuring the affinity of string data. In the following several matching rules are described in detail.

3.3.1 Hamming distance

Speaking about the simplicity of computation, the *Hamming distance* (3.4) or *Hamming dissimilarity* (*H*-distance) is obviously the convenient choice for string data. The Hamming distance is defined as the minimum number of point mutations required to transform one string data instance into another, where a point mutation means to change a letter or a bit. Thus, *H*-distance between two strings of bits (binary integers) is the number of corresponding bit positions that differ, and it can be determined by using the exclusive (disjunction) OR operator (symbolized by $\oplus$ or XOR) on corresponding bits or, equivalently, by adding corresponding bits (base 2).

$$H(X, Y) = \sum_{i=1}^{n} \overline{X_i \oplus Y_i}\,, \tag{3.1}$$

where X, Y are binary n-dimensional vectors, i.e., $X, Y \in \{0,1\}^n$. *The fractional* Hamming distance or *relative distance* between two strings X and Y ($|X| = |Y| = n$) is given by formula

$$H_F(X, Y) = \frac{H(X, Y)}{n}\,. \tag{3.2}$$

A variation of the *Hamming distance* is the *Roger & Tanimoto distance* (*RT*-distance) which is defined as follows

$$RT(X, Y) = \frac{\sum_{i=1}^{n} \overline{X_i \oplus Y_i}}{\sum_{i=1}^{n} \overline{X_i \oplus Y_i} + 2\sum_{i=1}^{n} X_i \oplus Y_i} \geq r\,, \tag{3.3}$$

where $X, Y \in \{0,1\}^n$ and r is the threshold value, $r \in [0,1]$.

3.3.2 Edit distance

The *Levenshtein distance* (also called *edit distance*) is a generalization of Hamming distance, and represents a *string metric* (string similarity metric or string distance

function) for measuring the difference between two sequences. The *Levenshtein distance* (L-distance) calculates the least number of edit operations required for a single-character to change one string into the other string, using the dynamic programming method (tabular computation for $L(X,Y)$): *insertion* (i), *deletion* (d), *substitution* (s) and the cost is normally set to 1 for each of these operations

$$L\left(X[1\ldots i], Y[1\ldots j]\right) = \begin{cases} \max(i,\, j), & \text{if } \min(i,\, j) = 0 \\ \begin{cases} d : L(i-1,\, j) + 1 \\ i : L(i,\, j-1) + 1 \\ s : L(i,\, j) + 1_{X_i \neq Y_j} \end{cases}, & \text{otherwise} \end{cases}, \tag{3.4}$$

where $1_{X,Y}$ is an indicator function. Levenshtein similarity is computed by formula

$$S_L\left(X,\, Y\right) = 1 - \frac{L\left(X,\, Y\right)}{\max\{|X|,\, |Y|\}}\,. \tag{3.5}$$

Noted that Levenshtein distance (3.4) is *inversely* with Levenshtein similarity (3.5) between sequences.

3.3.3 Cosine similarity

Cosine similarity (CS) is other method of calculating similarities between strings. In this similarity metric, the attributes are vectors of the same length. The algorithm requires that each string is converted to a vector to get the degree between strings. Thus, if the value of *cosine similarity* is 1 it means that the strings are equal because the angle between the objects is 0 degrees, since two sequences point to the same point; otherwise, the two strings are totally different if the angle between the objects is 90 degrees. The distance is directly proportional to similarity between sequences

$$CS\left(X,\, Y\right) = \frac{\sum_{i=1}^{n} X_i \cdot Y_i}{\sqrt{\sum_{i=1}^{n} X_i \cdot \sum_{i=1}^{n} Y_i}}\,, \tag{3.6}$$

where X, Y are binary n-dimensional vectors.

3.3.4 Value difference metric

Value difference metric (VDM) is one of the usually used distance functions, which was improved to work with nominal attributes and is defined as

$$VDM\left(X,\, Y\right) = \sum_{i=1}^{n} vdm\left(X_i,\, Y_i\right) \cdot weight\left(X_i\right)\,, \tag{3.7}$$

where the modified value difference metric for nominal attributes $vdm\left(X_i,\, Y_i\right)$ is computed by formula

$$vdm\left(X_i,\, Y_i\right) = \sum_{a \in A} \left(P\left(a|X_i\right) - P\left(a|Y_i\right)\right)^k\,, \tag{3.8}$$

and

$$weight\,(X_i) = \sqrt{\sum_{a \in A} P\,(a|X_i)^k}\,. \tag{3.9}$$

$P\,(a|X_i)$ denotes the conditional probability that X_i be equal to the character a in the alphabet A, and k is a constant, usually 1 or 2.

3.3.5 Landscape-affinity matching

The *Lanscape-affinity* matching rule was introduced starting from the need to capture the characteristics of matching biochemical and physical structures, and approximate matching in the immune system. The input string and the antibody strings are sampled as bytes and converted into positive integer values to generate a landscape; two strings are then compared in a sliding window. The comparison is made in one of the following three ways to produce an affinity measure and the measure is checked against with a threshold.

Difference matching rule

$$f_{difference} = \sum_{i=1}^{n} |X_i - Y_i|\ . \tag{3.10}$$

Slope matching rule

$$f_{slope} = \sum_{i=1}^{n} |(X_{i+1} - Y_{i+1}) - (X_i - Y_i)|\ . \tag{3.11}$$

Physical matching

$$f_{physical} = \sum_{i=1}^{n} \left[(X_i - Y_i) + 3|\mu|\right]\ ,\ \mu = \min_i (X_i - Y_i)\ . \tag{3.12}$$

3.3.6 R-contiguous bits matching

The *rcb* matching rule is defined as follows, if x and y are equal-length strings defined over a finite alphabet, xMy is true if x and y agree in at least r contiguous locations [14] (e.g., if $x = \{aebcda\}$ and $y = \{adbcea\}$, then xMy is true for $r \leq 4$ and false otherwise). In the binary strings case, the detector d is specified by a binary string c and threshold value r matches a string x if *rcb* of c matches the corresponding bits (at the same positions) of x. Although, it was initially implemented to asses approximate matching between two strings, the *rcb* rule was proved to be an useful tool mathematically speaking and also, an appropriate model for approximate T-cells matching. The parameter r determines the detector's degree of specificity; for a small value r, it results that the detector is more general.

R-chunk matching rule represents a generalization of the *rcb* matching rule (see [4, 14]), a detector is specified by a binary string c and parameter r. An r-chunk detector d is said to match a string x if all bits of c are equal to the r bits of x in the window specified by c. Unlike the *rcb* matching rule, the r-chunk approach allows the

use of detectors of any size, what results in improvement of the ability of the detectors to cover the self-space. Compared with *rcb* matching rule, in this case the matching window is specified for each individual detector; a group of r-chunk detectors that cover all possible windows has the same effect as an *rcb* detector.

Mathematical description of the algorithm, an element in the shape-space $x = (x_1 x_2 \dots x_m)$ and a detector $d = (k; d_1 d_2 \dots d_r)$, $r \leq m$ and $k \leq m - r + 1$, match according to the r-chunk rule $\Leftrightarrow x_i = d_i$, $(\forall)\, i = \overline{k, k + r - 1}$. That means, the element x matches with detector d if, at position k, there is a sequence of length r where all the characters are identical.

Remark 3.2. The matching rule is an important component in a NSA. It defines when a data item is considered to match or to be recognized by detectors and depends on the representation scheme of the data item and the detectors. Thus, data representation is one of fundamental differences between most models and algorithms. It decides and limits the possible matching rules, the generation mechanism of detectors, and the detection process. In context of our model data representation is string representation over higher alphabet.

Using *Gray code*, two consecutive strings differ only by 1 bit; thereby, affinity in problem space is to some extent maintained in shape-space. A real number $x \in [0, 1]$ which can be represented in a binary encoding using the transformation *floor*$(255x+0.5)$, and then encoding it in 8 bits. Due to this, the binary encoding is not most suitable method for generalization, and matching rules should accurately represent the data proximity in the problem space. Also, matching rules have effects in searching the shape-space, *rcb* matching rule produced a grid-like shape, while r-chunk matching rule generated similar but simpler shapes, *Hamming distance* and *R&T* matching rules produced a fractal-like shape. For example, the shape of areas covered by *rcb* and r-chunk matching rules were not affected by changing encoding from binary to Gray, because similarity between two real values is not reflected in their binary representations [19, 25, 14].

3.4 Distances between agents

We are interested to obtain a model to assess the behavior of a robotic swarm in two practical examples. The robotic swarm must perform: (a) to follow the leader: the swarm has a leading robot and the other robots must follow this leader's actions and trajectory and (b) to group in a limited region of the environment. In this respect, self robots will be those robots which (a) will follow the actions of the leader in a precise manner and (b) will be grouped in the same region of the environment (b). To distinguish between self and non-self robots in these two cases means to compute a distance between agents that is further reduced to compute a distance between strings in our P-colonies based model.

In the first case, after simulating the behavior of the leader by generating all his configurations step by step we compare them with those of the following robots and we will consider that all of agents that have a distance to the leader greater than a default value are non-self robots. In the second case a specific distance between robots must be not greater then a fixed value.

Definition 3.3 ([18])**.** Let's consider two data objects $\{X, Y\}$ (characteristics, features, etc.), 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 sets. 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 $D(X,Y)$ represents measure the corresponding mismatches, i.e., dissimilarity.

Next result supplies necessary conditions that must be fulfilled so that a distance coefficient represents a metric distance.

Definition 3.4. A metric on a nonempty set Π is a function (also called a distance function) $\Delta : \Pi \times \Pi \to \square_+$ which is required to satisfy the following conditions:

i) $\Delta(X,Y) \geq 0$, $(\forall)\, X, Y \in \Pi$ (non-negativity)

ii) $\Delta(X,Y) = 0 \Leftrightarrow X = Y$ (identity)

iii) $\Delta(X,Y) = \Delta(Y,X)$, $(\forall)\, X, Y \in \Pi$ (symmetry)

iv) $\Delta(X,Z) \leq \Delta(X,Y) + \Delta(Y,Z)$, $(\forall)\, X, Y, Z \in \Pi$ (triangle inequality),

As we mentioned, when considering a P swarm with a desired collective behavior, in order to describe the interactions between two or more agents, not necessarily from the same colony, the problem goes down to measure the distance between them, namely to compare their characteristic configurations. A configuration of a PCol automaton is defined by a string containing words and letters from its alphabet. For this purpose, we introduce the distance between two agents as follows and further demonstrate a proposition.

Definition 3.5 (Distance between two agents)**.** The distance between two agents $B_{i,j}$ from Π_i and $B_{k,l}$ from Π_k is given by the relation

$$\Delta(B_{i,j}, B_{k,l}) = \Delta(\{e_{ij} f_{ij}\}, \{e_{kl} f_{kl}\}) \,, \tag{3.13}$$

over all configurations obtained at the same step.

Definition 3.6 (Relation between two agents)**.** Two agents $B_{i,j}$ from Π_i and $B_{k,l}$ from Π_k are related, and this is denoted by $B_{i,j} \circledcirc B_{k,l}$, if $\Delta(B_{i,j}, B_{k,l}) \leq \varepsilon$, where $0 \leq \varepsilon \leq 1$. Otherwise, $B_{i,j}$ and $B_{k,l}$ are rejecting each other.

Proposition 3.1. *If $\varepsilon = 0$, the relation between agents "$\circledcirc$" is reflexive, symmetric and transitive.*

Proof. Taking into account Definition (3.4) and relation (3.13) $\Rightarrow$ results immediately that $\Delta(B_{i,j}, B_{i,j}) = 0$. It's assumed that $\Delta(B_{i,j}, B_{i,k}) = 0$, for $B_{i,j}, B_{i,k} \in \Pi_i$. If $\Delta(B_{i,k}, B_{i,j}) \neq 0$ and according to Definitions (3.4 - 3.6) $\Rightarrow B_{i,k}, B_{i,j} \notin \Pi_i$ which is a contradiction. Also, if $B_{i,j}, B_{i,k} \in \Pi_i$ and $\Delta(B_{i,j}, B_{p,q}) = 0 = \Delta(B_{i,k}, B_{p,q}) \Rightarrow B_{p,q} \in \Pi_i$. □

Formally speaking, to calculate the distance between two agents 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.

4 Conclusions

We have introduced a bio-inspired computational model based on a new idea of compute distances between agents using distances over strings. This model could be used to assess the behavior of a robotic swarm such as discriminating between self and non-self robots. The distance between agents is the based on a measure of dissimilarity between strings representing configurations of the respective automata. Directions for extending this research include the development of a P colony and P automata simulator, the proposal of new 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.

Acknowledgements. This work was supported by a 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 (NEWSWARM, http://newswarm.buiu.net/).

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. DOI: 10.1007/s11721-012-0075-2.

[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] J. Balthrop, F. Esponda, S. Forrest, M. Glickman *Coverage and generalization in an artificial immune system*, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Morgan Kaufmann, New York, 2002; 3-10.

[5] E. Csuhaj-Varju, J. Dassow, J. Kelemen, G. Paun, *Grammar systems: a grammatical approach to distribution and cooperation*, Topics in Computer Mathematics 5 (1994). ISBN: 2-88124-957-4.

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

[7] 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.

[8] M. Cavaliere, R. Mardare, S. Sedwards, *A multiset-based model of synchronizing agents: computability and robustness*, Theoretical Computer Science, 391 (2008), 216-238. DOI: 10.1016/j.tcs.2007.11.009.

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

[10] 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.

[11] L. Cienciala, L. Ciencialova, M. Langer, *Modularity in P colonies with checking rules*, Lecture Notes in Computer Science, LNCS 7184 (2012), 104-120.

[12] M. Dorigo, D. Floreano, L.M. Gambardella, et al., *Swarmanoid: A novel concept for the study of heterogeneous robotic swarms*, IEEE Robotics and Automation Magazine, 20, 4 (2013), 60-71. DOI: 10.1109/MRA.2013.2252996.

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

[14] D. Dasgupta, L.F. Nino, *Immunogical computation: theory and applications*, CRC Press, 2009.

[15] R. Freund, M. Oswald, *P colonies working in the maximally parallel and in the sequential mode*, Pre-proceedings of the 1st International Workshop on Theory and Application of *P* Systems, Timisoara, Romania, September 26-27, 2005, 49-56.

[16] S. Forrest, A. Perelson, L. Allen, R. Cherukuri, *Self-nonself discrimination in a computer*, Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, Los Alamitos, CA, 202-212. IEEE Computer Society Press, 1994.

[17] 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.

[18] J. Gasteiger, T. Engel, *Chemoinformatics: a textbook*, Wiley-VCH, 2003. ISBN: 978-3-527-30681-7.

[19] F. Gonzalez, D. Dasgupta, L.F. Nino, *A randomized real-valued negative selection algorithm*, Proceedings of the 2nd International Conference on Artificial Immune Systems, ICARIS 2003, Edinburgh, UK, September 1-3, 2003.

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

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

[22] S.A. Hofmeyr, S. Forrest, *Architecture for an artificial immune system*, Evolutionary Computation, 8, 4 (2000), 443-473.

[23] N.C. Jones, P.A. Pevzner, *An introduction to bioinformatics algorithms*, MIT Press, 2004. ISBN: 978-0-262-10106-6.

[24] D. Jurafsky, J.H. Martin, *Speach and language processing, 2nd Edition*, Prentice Hall, 2009. ISBN-13: 978-0131873216.

[25] Z. Ji, D. Dasgupta *Revisiting negative selection algorithms*, Evolutionary Computation Journal 15, 2 (2007), 223-251.

[26] 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 proceedings of the 9th International Conference on the Simulation and Synthesis of living systems, ALIFE IX, Boston 2004, 82-86. ISBN-13: 978-0262661836.

[27] J. Kelemen, A. Kelemenova, *A grammar-theoretic treatment of multiagent systems*, Cybernetics and Systems 63, 6 (1992), 621-633. DOI: 10.1080/01969729208927485.

[28] J. Kelemen, A. Kelemenova, *On P colonies, a simple bio-chemicallyinspired model of computation*, Proceedings of the 6th International Symposium of Hungarian Researchers on Computational Intelligence, Budapest, November 18-19, 2005, 40-56.

[29] A. Kelemenova, *P colonies*, The Oxford handbook of Membrane Computing, 2010, 584-593. ISBN: 978-0-19-955667-0.

[30] 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. ISBN: 978-84-940691-9-2.

[31] A. Liekna, J. Grundspenkins, *Towards practical application of swarm robotics: overview of swarm tasks*, Proceedings of the 13th International Conference Engineering for Rural Development, Jelgava, Latvia, May 29-30, 2014.

[32] S. Mitri, D. Floreano, L. Keller, *The evolution of information suppression in communicating robots with conflicting interests*, Proceedings of the National Academy of Sciences, PNAS 106, 37 (2009), 15786-15790. DOI:10.1073/pnas.0903152106.

[33] G. Paun, *Computing with membranes*, Journal of Computer and System Sciences 61, 1 (2000), 108-143. ISSN: 0022-0000.

[34] G. Paun, G. Rozenberg, A. Salomaa, *The Oxford handbook of membrane computing*, Oxford University Press (2010). ISBN: 978-0-19-955667-0.

[35] L.E. Pineda, A.E. Eiben, M. van Steen, *Evolving communication in robotic swarms using on-line, on-board, distributed evolutionary algorithms*, Applications of Evolutionary Computation, Lecture Notes in Computer Science, LCNS 7248 (2012), 529-538.

[36] A.S. Peresoln, G.F. Oster, *Theoretical studies of clonal selection: minimal antibody repertoire size and reliability of self-nonself discrimination*, Journal of Theoretical Biology, 81 (1979), 645-670.

[37] W. Trappe, L.C. Washington, *Introduction to cryptography with coding theory, 2nd Edition*, Prentice Hall, 2006. ISBN-13: 978-0131862395.

[38] Y. Tan, Z. Zhong-Yang, *Research advance in swarm robotics*, Defence Technology 9, 1 (2013), 18-39. DOI: 10.1016/j.dt.2013.03.001.

[39] ***, http://www.artificial-immune-systems.org/index.shtml

Authors' 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