From Swarm Simulations to Swarm Intelligence*

Andrew Schumann
University of Information Technology and Management in Rzeszow
Sucharskiego 2
35-225, Rzeszow, Poland
andrew.schumann@gmail.com

ABSTRACT

In self-organizing systems such as collective intelligent behaviors of animal or insect groups: flocks of birds, colonies of ants, schools of fish, swarms of bees, etc. there are ever emergent patterns which cannot be reduced to a linear composition of elementary subsystems properly. This reduction is possible only due to many repellents and an artificial environment. The emergent patterns are studied in the so-called swarm intelligence. In this paper we show that any swarm can be represented as a conventional automaton such as Kolmogorov-Uspensky machine, but with a very low accuracy because of deleting emergent phenomena. Furthermore, we show as well that implementing some unconventional algorithms of p-adic arithmetic and logic are much more applicable than conventional automata. By using p-adic integers we can code different emergent patterns.

Categories and Subject Descriptors

B.2 [**Arithmetic and Logic Structures**]: Miscellaneous; D.3.2 [**Language Classifications**]: [concurrent, distributed, and parallel languages]

General Terms

Theory

Keywords

swarm intelligence, Kolmogorov-Uspensky machine, p-adic valued logic

1. INTRODUCTION

Recently, a new approach in unconventional computing called swarm intelligence has developed [5], [14], [33]. For example, it was discovered experimentally that swarms of social insects [9] can solve complex computational problems in searching for food and in transporting sources and information due to massive-parallel behaviour with labour divisions.

Now there are many algorithms in simulating swarms: the Particle Swarm Optimization (PSO) [15], the Bacterial Foraging Optimization Algorithm (BFOA) [19], the Artificial Bee Colony (ABC) [12], the Cuckoo Optimization Algorithm (COA) [20], the Social Spider Optimization (SSO) [6], the Ant Colony Optimization (ACO) [8], etc.

In this paper we try to formulate swarms as labelled transition systems with the same set of labels (events, or actions): direction, splitting, fusion, repelling (Section 2). Then we show that these labelled transition systems can implement Kolmogorov-Uspensky machines, but with a low accuracy because of emergent patterns which occur if we have many states of appropriate transition system (Section 3). Then we propose p-adic arithmetic and logic to formalize emergent patterns of swarms (Section 4).

To sum up, we offer a general logical approach to swarm intelligence.

2. FROM EMERGENT COMPUTING TO SWARM COMPUTING

In conventional logic circuits some electrical properties of transistors are used. In particular, the voltage is managed to be in only one of two states: high (if the voltage runs the range from 2.8 to 5.0V) or low (if the voltage is in the range from 0 to 0.8V). The high state of voltage means '1' or logical true. The low state means '0' or logical false. The main idea of electric devices based on electrical properties of transistors is that the Boolean logic can be implemented with a very high accuracy. In this logic any complex logic expression is considered a composition of logical atoms (i.e., of simple logical propositions). In other words, there are no emergent phenomena. Let us recall that the emergence is detected, when new patterns of a highly structured collective behavior appear and these patterns cannot be reduced to a linear composition of simple subsystems.

Notably, emergent phenomena are key phenomena in all self-organizing systems such as collective intelligent behaviors of animal groups: flocks of birds, colonies of ants, schools of fish, swarms of bees, etc. Emergence is observed in the economy as well: macroeconomic fluctuations, traffic jams, hierarchy of cities, motion picture industry and mass protest behavior [18]. There are attempts to formalize the notion of emergence by algorithmic complexity theory. However, the Kolmogorov complexity function is not computable. There is no way to define the emergence by minimum linear compositions. Wolfram proposed a more useful approach in a

*This research is supported by FP7-ICT-2011-8.

.

BICT 2015, December 03-05, New York City, United States

DOI 10.4108/eai.3-12-2015.2262484

mathematical definition of emergency [34]. He showed that the behavior of one-dimensional cellular automata is divided into the following four cases:

- there are limit points of the system, i.e. we obtain a homogeneous state of the system;
- there are limit cycles of the system, i.e. we obtain separated periodic structures;
- there are chaotic attractors, i.e. we face chaotic patterns;
- there are complex localized structures.

In the last case we deal with an emergent phenomenon in the true sense. On the basis of the Wolfram's approach the so-called *emergent computing* has developed. In this kind of computing (i) the computation process is distributed over a set of parallel and autonomous processing units; (ii) each unit computes locally and can interact directly only with a small number of other units; (iii) there is a consistency among the processing units; (iv) there are more outputs, than inputs.

One of the most studied instance of emergent computing is represented by *swarm computing* [5], [15]. This computing is based on labor division in animal groups. Any swarm as a result of collective behavior, such as birds flocking or fish schooling, is a self-organizing system, where, on the one hand, each unit responds to local stimuli individually and, on the other hand, all together they accomplish a global task (transporting, eating, self-protecting, etc.).

There were proposed many swarm algorithms to simulate the behavior of insect or animal groups: the Particle Swarm Optimization (PSO) [15], the Bacterial Foraging Optimization Algorithm (BFOA) [19], the Artificial Bee Colony (ABC) [12], the Cuckoo Optimization Algorithm (COA) [20], the Social Spider Optimization (SSO) [6], the Ant Colony Optimization (ACO) [8].

In the PSO it is assumed that the particles (agents) know (i) their best position 'local best' (lb) and (ii) their neighborhood's best position 'global best' (gb). The next position is determined by velocity. Let $x_i(t)$ denote the position of particle i in the search space at time step t, where t is discrete. Then the position x_i is changed by adding a velocity to the current position:

$$x_i(t+1) = x_i(t) + v_i(t+1),$$

where $v_i(t+1) = v_i(t)+c_1r_1(lb(t)-x_i(t))+c_2r_2(gb(t)-x_i(t))$ and i is the particle index, c_1, c_2 are acceleration coefficients, such that $0 \leq c_1, c_2 \leq 2$, r_1, r_2 are random values (such that $0 \leq r_1, r_2 \leq 1$) regenerated every velocity update.

One of the possible PSO algorithms can be exemplified by the bird flocking [21], [22]. In flocks 'local best' and 'global best' of birds are defined by the following three rules: (i) collision avoidance (birds fly away before they crash into one another); (ii) velocity matching (birds fly about the same speed as their neighbors in the flock); and (iii) flock centering (birds fly toward the center of the flock as they perceive it). So, the position of a bird i at time t is given by its placement x_i at time $t-1$ shifted by its current velocity v_i. This v_i is determined by the rules (i) – (iii).

All the algorithms PSO, BFOA, ABC, COA, SSO, ACO are used to simulate swarms of different insects or animals. Let us try to answer the question, whether swarms can be considered an unconventional computer, i.e. whether swarms can calculate.

Each swarm is a natural transition system

$$TS = (S, E, T, I),$$

where:

- S is the non-empty set of states;
- E is the set of events;
- $T \subseteq S \times E \times S$ is the transition relation;
- $I \subseteq S$ is the set of initial states.

Any transition system is a labeled graph with nodes corresponding to states from S, edges representing the transition relation T, and labels of edges corresponding to events from E.

Let us consider some examples.

2.1 Ant Colony Transitions

Any ant colony is a swarm of ants localized first at the nest. This nest can be regarded as an initial state of ant colony transitions. Then ants use a special mechanism called *stigmergy* to build up all transitions. Stigmergy (stigma + ergon) means 'stimulation by work'. This mechanism has the following steps [8]:

- At first ants are looking for food randomly, laying down pheromone trails.
- If ants find food, they return to the nest, leaving behind pheromone trails. So there is more pheromone on the shorter path than on the longer one.
- Ants prefer to go in the direction of the strongest pheromone smell. As a consequence, the concentration of pheromone is so strong on the shorter path, that all the ants prefer this path.

Thus, stigmergy allows ants to transport food to their nest in a remarkably effective way. Food localizations are considered new states and ant roads to food places are regarded as transitions. Let $P_{ant} = \{n\}$ be an initial state of ant transitions (i.e. the nest), $A_{ant} = \{a_1, a_2, \ldots, a_j\}$ be a set of food pieces (attractants) localized at different places, $V_{ant} = \{r_1, r_2, \ldots, r_i\}$ be a set of ant roads. So the ant colony transition system, $TS_{ant} = (S_{ant}, E_{ant}, T_{ant}, I_{ant})$, can be defined as follows:

- $\sigma : P_{ant} \cup A_{ant} \to S_{ant}$ assigning a state to each original point of the ant colony as well as to each attractant;
- $\tau : V_{ant} \to T_{ant}$ assigning a transition to each ant road;
- $\iota : P_{ant} \to I_{ant}$ assigning an initial state to the nest.

Each event of the set of events E_{ant} is assigned to ant transitions in accordance with the following types of ant expansion:

- *direction* (the ants move from one state / attractant / initial point to another state / attractant),
- *fusion* (the ants move from different states / attractants to the same one state / attractant),

- *splitting* (the ants move from one state / attractant / initial point to different states / attractants),
- *repelling* (the ants stop to move in one direction).

The system TS_{ant} can be used to solve the Travelling Salesman Problem formulated as follows: given a list of cities and the distances between each pair of cities, we must to define the shortest possible route that visits each city exactly once and returns to the origin city. This problem is NP-hard. Let P_{ant} be considered the origin city, A_{ant} be a set of all other cities, and V_{ant} be a set of all connections between cities. The Travelling Salesman Problem can be solved, because the ants lay down pheromone trails faster on the shortest path so that the shortest path gets reinforced with more pheromone to attract more future ants. As a result, pheromone trails on the edges between cities depend on the distance: more shorter, more attracting. This allows ants to find shorter tours of cities.

2.2 Bee Colony Transitions

A bee colony is another example of swarm intelligence [13]. The bee nest is an initial state of bee colony transitions. Any bee colony exploits a mechanism called *waggle dance* to optimize the food transporting to the nest. This mechanism is as follows. In the nest there is an area for communication among bees. At this area the bees knowing, where the food source is precisely, exchange the information about the direction, distance, and amount of nectar on the related food source by a waggle dance. The direction of waggle dancing bees shows the direction of the food source in relation to the Sun, the intensity of the waggles is associated to the distance, and the duration of the dance shows the amount of nectar. Due to this form of communication the bee colony transitions are built up by the following steps [12]:

- There are two kinds of bees: employed and unemployed. Employed bees know exactly, where a particular food source (nectar) is, and visit just this source. Unemployed bees do not know and seek a food source. The unemployed bees are divided into the following two groups: scouts and onlookers. A scout bee carries out search for new food sources without any guidance. An onlooker bee follows the instruction of a waggle dancing bee and visits the food source for the first time. An employed bee visits this source many times. So, the first step in constructing the bee transporting system is in sending scout bees.
- Then onlookers are sent.
- At the next step the food source is exploited by employed bees.
- An employed bee tests if the nectar amount of the new food source is higher than that of the previous one. If it is so, the bee memorizes the new place and forgets the old one. If the nectar amount decreased or exhausted and the employed bee dos not know a new place, this bee become an unemployed bee. So, at this step the employed bees exchange the nectar information of the food sources to change their decision.

Assume that $P_{bee} = \{n\}$ is an initial state of bee transitions (i.e. the bee nest), $A_{bee} = \{a_1, a_2, \ldots, a_j\}$ is a set of food sources (attractants) localized at different places, $V_{onlooker} = \{r_1, r_2, \ldots, r_k\}$ is a set of onlooker bee roads, and $V_{employed} = \{r_1, r_2, \ldots, r_l\}$ is a set of employed bee roads. Then the bee colony transition system, $TS_{bee} = (S_{bee}, E_{bee}, T_{bee}, I_{bee})$, can be defined thus:

- $\sigma : P_{bee} \cup A_{bee} \to S_{bee}$ assigning a state to each original point of the bee colony as well as to each attractant;
- $\tau : V_{onlooker} \cup V_{employed} \to T_{bee}$ assigning a transition to each bee road;
- $\iota : P_{bee} \to I_{bee}$ assigning an initial state to the bee nest.

In the set of events E_{bee} there are the following types of labels for transitions:

- *direction* (the onlooker or employed bees move from one state / attractant / initial point to another state / attractant),
- *fusion* (the onlooker or employed bees move from different states / attractants to the same one state / attractant),
- *splitting* (the onlooker or employed bees move from one state / attractant / initial point to different states / attractants),
- *repelling* (the onlooker or employed bees stop to move in one direction).

The system TS_{bee}, if we use only $V_{employed}$ as a set of roads, can solve the Travelling Salesman Problem, also. Thereby, P_{bee} is examined as the origin city, A_{bee} is a set of all other cities, and $V_{employed}$ is a set of all connections between cities. The point is that the greater the number of iterations in sending onlooker or employed bees, the higher the influence of the distance in attracting the bees to appropriate food sources. As a result, the shorter distance seems to be more attracting for employed bees.

There is another NP-hard problem that can be solved by TS_{bee}, the so-called Generalized Assignment Problem formulated as follows: there are a number of agents and a number of tasks, each agent has a budget and each task assumes some cost and profit; we must find an assignment in which all agents do not exceed their budget and total profit of the assignment is maximized. In the case of the bee colony, the bees are regarded as agents, the nectar sources as tasks, the amount of nectar as profit, and the distance as cost. Hence, in this interpretation the bee colony can solve the Generalized Assignment Problem.

2.3 Paenibacillus vortex Transitions

Notice that there are bacteria with intelligent and successful swarming strategies such as *Paenibacillus vortex* [4]. These strategies help *Paenibacillus vortex* bacteria to carry out a cooperative colonization of new territories. When *Paenibacillus vortex* is inoculated on hard agar surfaces with peptone, it develops complex colonies of vortices. When it is inoculated on soft agar surfaces, it organizes a special network of swarms with intricate internal traffic. In contrast to *Escherichia coli*, the *Paenibacillus vortex* swarms are sensitive to chemotaxis (attractants). As a consequence, the *Paenibacillus vortex* network is built up on the basis of interactions between swarms allowing them to transport nutrients, spores and other organisms [11], [29].

Let $P_{one\ P.vortex} = \{n_1, n_2, \ldots, n_m\}$ be a set of initial states of *Paenibacillus vortex* transitions, $A_{one\ P.v.} = \{a_1, a_2, \ldots, a_j\}$ be a set of attractants (nutrient gradients), $V_{one\ P.v.} = \{r_1, r_2, \ldots, r_k\}$ be a set of paths for each *Paenibacillus vortex* bacterium. Then the *Paenibacillus vortex* transition system,

$$TS_{one\ P.v.} = (S_{one\ P.v.}, E_{one\ P.v.}, T_{one\ P.v.}, I_{one\ P.v.}),$$

is as follows:

- $\sigma : P_{one\ P.v.} \cup A_{one\ P.v.} \to S_{one\ P.v.}$ assigning a state to each original point of the *Paenibacillus vortex* population as well as to each attractant;
- $\tau : V_{one\ P.v.} \to T_{one\ P.v.}$ assigning a transition to each path of each *Paenibacillus vortex* bacterium;
- $\iota : P_{one\ P.v.} \to I_{one\ P.v.}$ assigning initial states to initial positions of *Paenibacillus vortex* bacteria.

The four types of labels for transitions in the set of events $E_{one\ P.v.}$:

- *direction*: the *Paenibacillus vortex* bacterium moves from one state / attractant / initial point to another state/attractant presented by a nutrient gradient;
- *tumbling*: the *Paenibacillus vortex* bacterium can tumble for a while;
- *repelling*: the *Paenibacillus vortex* bacterium stops to move in one direction, as it avoids some chemical concentrations;
- *repropuction*: the health bacterium splits into two bacteria.

Paenibacillus vortex bacteria can be organized in swarms. The locomotion in a swarm can be explained hydrodynamically by collisions among the bacteria and by the boundary of the layer of lubricant collectively generated by them. However, interactions among swarms can be considered intelligent. Each swarm has a snake-like formation. It looks for food and can cross each other's trail. When food is detected, swarms change their direction. The *Paenibacillus vortex* swarms can split and fuse in accordance with topology of nutrients.

So, for these swarms we can propose another transition system

$$TS_{P.v.\ swarm} = (S_{P.v.\ swarm}, E_{P.v.\ swarm}, T_{P.v.\ swarm},$$

$$I_{P.v.\ swarm}),$$

where

- $\sigma : P_{P.v.\ swarm} \cup A_{P.v.\ swarm} \to S_{P.v.\ swarm}$, where $P_{P.v.\ swarm} = \{n_1, n_2, \ldots, n_l\}$ is a set of initial states of *Paenibacillus vortex* swarm transitions and $A_{P.v.\ swarm} = A_{one\ P.v.}$ is a set of attractants (nutrient gradients). The function σ assigns a state to each original point of the *Paenibacillus vortex* swarms as well as to each attractant;
- $\tau : V_{P.v.\ swarm} \to T_{P.v.\ swarm}$, where $V_{P.v.\ swarm} = \{r_1, r_2, \ldots, r_k\}$ is a set of paths for each *Paenibacillus vortex* swarm. The function τ is to assign a transition to each path of each *Paenibacillus vortex* swarm;
- $\iota : P_{P.v.\ swarm} \to I_{P.v.\ swarm}$ is to assign initial states to initial positions of *Paenibacillus vortex* swarms.

Each event of the set of events $E_{P.v.\ swarm}$ is assigned to swarm motions according to the following types of maneuvers:

- *direction*: the *Paenibacillus vortex* swarm moves from one state / attractant / initial point to another state / attractant,
- *fusion*: the *Paenibacillus vortex* swarms move from different states / attractants / initial points to the same one state/attractant,
- *splitting*: the *Paenibacillus vortex* swarm moves from one state / attractant / initial point to different states / attractants (for the experimental details see [10]),
- *repelling*: the *Paenibacillus vortex* swarm stops to move in one direction if it faces a repellent.

As we see, the *Paenibacillus vortex* transition system for swarms can solve the Travelling Salesman Problem, too.

2.4 Conclusion: Towards Physarum machines

In the project *Physarum Chip Project: Growing Computers From Slime Mould* [2] supported by FP7 we are going to design an unconventional computer on plasmodia of *Physarum polycephalum*. Notice that *Physarum polycephalum* is a one-cell organism whose plasmodia behave as a swarm organizing a network for transporting sources and information. In order to simulate *Physarum polycephalum* networks we have proposed *Physarumsoft* [28], a software tool for programming *Physarum* computing and simulating *Physarum* expansions.

Taking into account the fact that any plasmodium can be considered a typical intelligent swarm, we can use *Physarumsoft* for demonstrating computational powers of different swarms: ant colonies, bee colonies, and *Paenibacillus vortex* swarms. Indeed, let

$$TS_{P.polycephalum} = (S_{P.polycephalum}, E_{P.polycephalum},$$

$$T_{P.polycephalum}, I_{P.polycephalum})$$

be a transition system for plasmodia and this system is defined standardly. Let f be a mapping from $S_{P.polycephalum}$ to $S_\star$ and from $I_{P.polycephalum}$ to $I_\star$, where $\star \in \{ant, bee, P.v.\ swarm\}$. Assume that all transitions denoted by $\longrightarrow$ are the same for all systems: $TS_{P.polycephalum}, TS_{ant}, TS_{bee}, TS_{P.v.\ swarm}$. For example, we have the same direction, fusion, splitting, and repelling. The function f is a homomorphism if and only if

- for all $s \in (S_{P.polycephalum} \cup I_{P.polycephalum})$, if $s \longrightarrow s'$, for some $s' \in S_{P.polycephalum}$, then $f(s) \longrightarrow f(s')$;
- for all $s \in S_{P.polycephalum}$, if $f(s) \longrightarrow t$, for some $t \in S_\star$, then there exists $s' \in S_{P.polycephalum}$ with $s \longrightarrow s'$ and $f(s') = t$.

If $f : S_{P.polycephalum} \cup I_{P.polycephalum} \to S_\star \cup I_\star$ is a homomorphism as well as $f^{-1} : S_\star \cup I_\star \to S_{P.polycephalum} \cup I_{P.polycephalum}$ is a homomorphism, then f is an isomorphism. So, for any $TS_\star$, where $\star \in \{ant, bee, P.v.\ swarm\}$,

there is an isomorphism from $S_{P.polycephalum}$ to $S_\star$ and from $I_{P.polycephalum}$ to $I_\star$. This means that it is enough to study the *Physarum* machine $TS_{P.polycephalum}$ to know computational properties of different swarms: ant colonies, bee colonies, *Paenibacillus vortex* swarms, etc.

According to our previous study of $TS_{P.polycephalum}$ we know that it is impossible to define *Physarum* transitions as atomic acts [26], [25]. For instance, under the same conditions, the plasmodium can follow splitting or direction, fusion or direction, etc. Nevertheless, with a low accuracy we can implement some conventional algorithms in $TS_{P.polycephalum}$.

3. CONVENTIONAL ALGORITHMS ON PHYSARUM MACHINES

It is known that, theoretically, Turing machines and Kolmogorov-Uspensky machines [17], [32] have the same expressibility power. In other words, the class of functions computable by these machines is the same. For the first time A. Adamatzky [1] experimentally showed that the *Physarum* machine $TS_{P.polycephalum}$ can be represented as a kind of Kolmogorov-Uspensky machines. Hence, we can implement conventional algorithms in $TS_{P.polycephalum}$.

Let us show that $TS_{P.polycephalum}$ can be considered a Kolmogorov-Uspensky machine. Let $\Gamma = S_{P.polycephalum} \cup I_{P.polycephalum}$ be an alphabet, $k = |E_{P.polycephalum}|$ a natural number. We say that a tree is (Γ, k)-tree, if one of nodes is designated and it is called *root* and all edges are directed. Each node is labelled by one of signs of Γ and each edge from the same node is labelled by different numbers $\{1, \ldots, k\}$ (so, each node has not more than k edges). We see that by this definition of (Γ, k)-tree, the plasmodium grows from the one active zone (so, we simulate the expansion from the one nest of ants or bees, or from the one inoculation of *Paenibacillus vortex* swarms), where all attractants are labelled by signs of Γ and protoplasmic tubes (roads of ants or roads of bees) are labelled by numbers of $\{1, \ldots, k\}$. Thus, $TS_{P.polycephalum}$ (as well as TS_{ant}, TS_{bee}, or $TS_{P.vortex\ swarm}$) can be represented as a (Γ, k)-tree.

(Γ, k)-*Physarum complex* is any initial finite digraph which is connected (i.e. each vertex is accessible from the initial one by a directed path), each node is labelled by one of signs of Γ, and each edge from the same node is labelled by different numbers $\{1, \ldots, k\}$. The set of all vertices of (Γ, k)-*Physarum* complex U is denoted by $v(U)$.

The *r-neighborhood* of (Γ, k)-complex is represented by a (Γ, k)-complex which consists of edges and vertices of initial complex that are accessible from initial vertex by a directed path that is not longer than r. Notice that r can be arbitrary. Any property of (Γ, k)-complex which is dependent just of r-neighborhood is called *r-local property of* (Γ, k)-*complex*. Hence, we can ever project *Physarum* transitions (using attractants and repellents) for inducing different numbers r and appropriate local properties.

A program of *Physarum* Kolmogorov-Uspensky machine is any r-local action transforming some (Γ, k)-complexes of growing plasmodia into other (Γ, k)-complexes of growing plasmodia:

$$U \to \langle W, \gamma, \delta \rangle,$$

where U, W are (Γ, k)-*Physarum* complexes, γ is a mapping from $v(U)$ to $v(W)$, δ is an injection from $v(U)$ into $v(W)$. The algorithm of transformation complexes $S \to S^*$ is as follows [17], [32]:

- r-Neighborhood of complex S is the same as of U.
- $v(S') = v(S \backslash U) \cup v(W)$.
- If $b \in U$, $a \in S \backslash U$, there is $\langle a, b \rangle$ in S and $\gamma(b)$ is defined, then $\langle a, \gamma(b) \rangle$ is an edge in S' with the same number as $\langle a, b \rangle$.
- If $a \in U$, $b \in S \backslash U$, there is $\langle a, b \rangle$ in S and $\delta(a)$ is defined, then $\langle \delta(a), b \rangle$ is an edge in S' with the same number as $\langle a, b \rangle$ (due to injectivity of δ we have different numbers for different edges from the same vertex).
- The initial vertex of W is an initial vertex of S' and we delete in S' all vertices (with appropriate edges) which are not accessible from the initial one. In this way we obtain S^*.

The simpler version of Kolmogorov-Uspensky machines is represented by Schönhage's storage modification machines [23], [24].

Unfortunately, the computational complexity of implementations Kolmogorov-Uspensky machines on the *Physarum polycephalum* medium is very high. The point is that not every computable functions can be simulated by plasmodium behaviors:

- first, the plasmodium has a free will and can make different decisions under the same conditions;
- second, the plasmodium follows emergent patterns which are fully eliminated in conventional automata such as Kolmogorov-Uspensky machines, although these patterns are natural for occupying many attractants.

Thus, swarm intelligence can be reduced to conventional automata, but with very low accuracy.

4. UNCONVENTIONAL ALGORITHMS ON PHYSARUM MACHINES

Let us take a set of edges, $X = \{0, 1, \ldots, p-1\}$, where $p - 1 = |E_{P.polycephalum}|$, from each node. The set of alphabet $\Gamma = S_{P.polycephalum} \cup I_{P.polycephalum}$ is identified with attractants. Hence, we have assumed that at each step of plasmodium propagation there are not more than $p-1$ neighboring attractants which can be directly occupied. This means that our universe is p-adic and the plasmodium transition system can be coded by p-adic integer. Let us remember that the set of p-adic integers is denoted by $\mathbf{Z}_p$ and each p-adic integer $n \in \mathbf{Z}_p$ has the following meaning:

$$n = \sum_{i=0}^{\infty} a_i \cdot p^i,$$

where $a_i \in \{0, 1, \ldots, p-1\}$, and the following notation:

$$n = \ldots a_i a_{i-1} \ldots a_1 a_0.$$

For each transition $s \longrightarrow s'$, the state $s' \in \Gamma$ is called the child of $s \in \Gamma$. For each two transitions $s \longrightarrow s'$ and $s' \longrightarrow s''$, the state $s'' \in \Gamma$ is called the grandchild of $s \in \Gamma$. Let us consider just strings $\gamma_0 \gamma_1 \ldots \gamma_k$, where γ_1 is a grandchild for γ_0, γ_2 is a grandchild for γ_1, $\ldots$, γ_k is a grandchild for γ_{k-1}. Let each string $\gamma_0 \gamma_1 \ldots \gamma_k$ have a numeric value $[\gamma_0 \gamma_1 \ldots \gamma_k] \in \mathbf{Z}_p$ which is defined as follows:

- Let $[\gamma_0]$ denote an integer $\leq p-1$ for the node $\gamma_0 \in X$. This integer is equal to the number of children for γ_0. If γ_0 has no grandchildren, then its value is coded by a p-adic integer $\ldots 0000[\gamma_0]$, see figures 1, 2, 3.
- Let $[\gamma_0]$ and $[\gamma_1]$ denote some integers $\leq p-1$ for the nodes $\gamma_0, \gamma_1 \in X$, where γ_1 is a grandchild of γ_0. The integer $[\gamma_0]$ is equal to the number of children for γ_0 and the integer $[\gamma_1]$ is equal to the number of neighbours for γ_1 occupied by the plasmodium. If γ_1 has no grandchildren, then the value of $\gamma_0\gamma_1$ is coded by a p-adic integer $\ldots 0000[\gamma_1][\gamma_0]$.
- $\ldots$
- Let $[\gamma_0]$, $[\gamma_1]$, $\ldots$, $[\gamma_k]$ denote some integers $\leq p-1$ for the nodes $\gamma_0, \gamma_1, \ldots, \gamma_k \in X$, respectively, where γ_1 is a grandchild of γ_0, γ_2 is a grandchild of γ_1, $\ldots$, γ_k is a grandchild of γ_{k-1}. The integer $[\gamma_0]$ is equal to the number of children for γ_0, the integer $[\gamma_1]$ is equal to the number of neighbours for γ_1 occupied by the plasmodium, $\ldots$, the integer $[\gamma_k]$ is equal to the number of neighbours for γ_k occupied by the plasmodium. If γ_k has no grandchildren, then the value of $\gamma_0\gamma_1\ldots\gamma_k$ is coded by a p-adic integer $\ldots 0000[\gamma_k]\ldots[\gamma_1][\gamma_0]$. See figure 4.

Evidently that according to this definition if in the string $\gamma_0\gamma_1\ldots\gamma_k$ each γ_i $(0 \leq i \leq k)$ has no neighboring attractants occupied by the plasmodium, then $[\gamma_0\gamma_1\ldots\gamma_k] = 0 \in \mathbf{Z}_p$ and if in the string $\gamma_0\gamma_1\ldots\gamma_k$ each γ_i $(0 \leq i \leq k)$ has all neighboring attractants occupied by the plasmodium, then $[\gamma_0\gamma_1\ldots\gamma_k] = \sum_{i=0}^{k}(p-1)\cdot p^i \in \mathbf{Z}_p$. The strings $[\gamma_0\gamma_1\ldots\gamma_k] \neq 0$ are called non-empty.

Figure 1: At each step of plasmodium propagation, there are only two attractants which can be occupied. At the first time step the plasmodium expansion is coded by the 3-adic integer $\ldots 000001$.

Let us analyze the case when we have two strings $\gamma_0\gamma_1\ldots\gamma_k$ and $\gamma_0\gamma'_1\ldots\gamma'_m$ started from the same state γ_0. Suppose that γ_{i_k} $(0 \leq i_k \leq k)$ and γ'_{i_m} $(0 \leq i_m \leq m)$ have some neighboring attractants occupied by the plasmodium. This means that we face a splitting of the plasmodium at the node γ_0. Assume that there is not more splitting for nodes from $\gamma_0\gamma_1\ldots\gamma_k$ and $\gamma_0\gamma'_1\ldots\gamma'_m$ and only $\gamma_0\gamma_1\ldots\gamma_k$ and $\gamma_0\gamma'_1\ldots\gamma'_m$ are non-empty. Then the transition system is coded by the set

$$\{[\gamma_0\gamma_1\ldots\gamma_k], [\gamma_0\gamma'_1\ldots\gamma'_m]\}.$$

Another similar situation is observed when we have two strings $\gamma_0\gamma_1\gamma_2\ldots\gamma_k$ and $\gamma_0\gamma_1\gamma'_2\ldots\gamma'_m$ started from the same state γ_0 and with the splitting at the node γ_1. If only $\gamma_0\gamma_1\gamma_2\ldots\gamma_k$ and $\gamma_0\gamma_1\gamma'_2\ldots\gamma'_m$ are non-empty, then the transition system is coded by the set

$$\{[\gamma_0\gamma_1\gamma_2\ldots\gamma_k], [\gamma_0\gamma_1\gamma'_2\ldots\gamma'_m]\}.$$

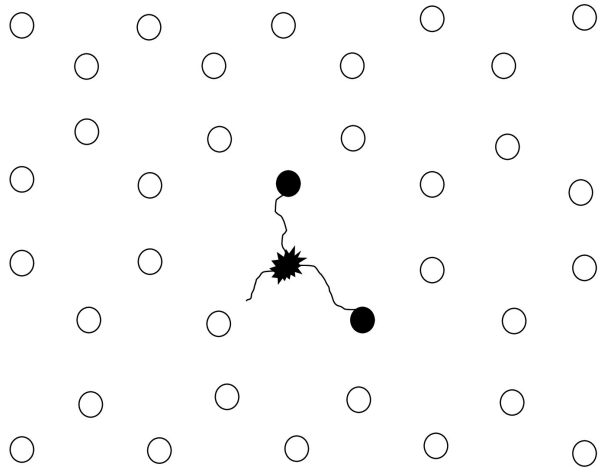

Figure 2: At each step of plasmodium propagation, there are only three attractants which can be occupied. At the first time step the plasmodium expansion is coded by the 4-adic integer $\ldots 000002$.

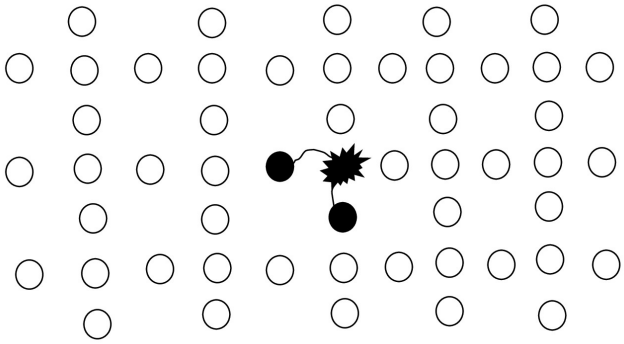

Figure 3: At each step of plasmodium propagation, there are not more than four attractants which can be occupied. At the first time step the plasmodium expansion is coded by the 5-adic integer $\ldots 000002$.

Let A_{γ_0} be a set of all strings started from γ_0 and this set be coded by $[A_{\gamma_0}]$, a set of p-adic integers obtained for each string from A_{γ_0}. Now we can define compositions of two sets A_{γ_0} and $A_{\gamma'_0}$, where $\gamma_0 \neq \gamma'_0$:

- If no strings from A_{γ_0} have joint nodes with some strings from $A_{\gamma'_0}$, then we have $A_{\gamma_0,\gamma'_0} = A_{\gamma_0} \cup A_{\gamma'_0}$ and this system is coded by $[A_{\gamma_0}] \cup [A_{\gamma'_0}]$.
- If some strings from A_{γ_0} have joint nodes with some strings from $A_{\gamma'_0}$, then we have $A_{\gamma_0,\gamma'_0} = A_{\gamma_0} + A_{\gamma'_0}$ and this set contains all strings started from γ_0 and started from γ'_0. The system A_{γ_0,γ'_0} is coded by $[A_{\gamma_0,\gamma'_0}]$.

By induction, we can define sets $A_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$.

Notably, the plasmodium expansion is time dependent. So we can consider sets $A^t_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$ at $t = 0, 1, \ldots$ Let us define logical operations over the same sets $A^t_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$ with different t:

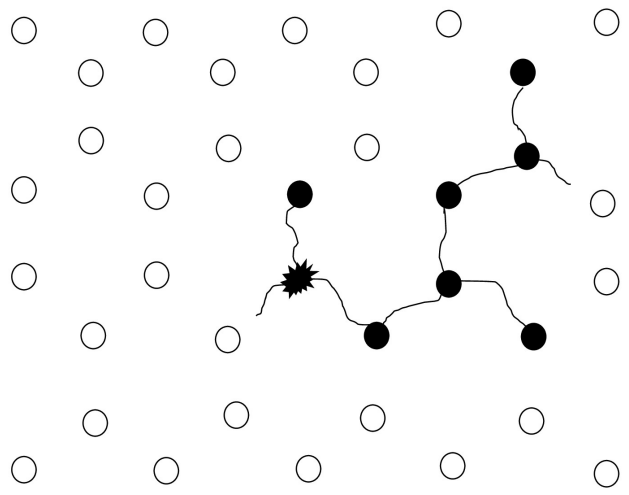

Figure 4: At each step of plasmodium propagation, there are not more than four attractants which can be occupied. At the time step $t > 0$ the plasmodium expansion is coded by the 4-adic integer ... 00000232.

conjunction $A^{t=k}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0} \wedge A^{t=l}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$**:** Notice that strings from $A^{t=k}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$ and $A^{t=l}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$, where $k \neq l$, are the same, but they can be coded by different p-adic integers at $t = k$ and $t = l$. Let us consider each string $\gamma_0\gamma_1\gamma_2\ldots\gamma_m$. Let $[\gamma_0\gamma_1\gamma_2\ldots\gamma_m]_k$ be a p-adic numerical value of $\gamma_0\gamma_1\gamma_2\ldots\gamma_m$ at $t = k$ and $[\gamma_0\gamma_1\gamma_2\ldots\gamma_m]_l$ be a p-adic numerical value of $\gamma_0\gamma_1\gamma_2\ldots\gamma_m$ at $t = l$. Then we define $\min([\gamma_0\gamma_1\gamma_2\ldots\gamma_m]_k, [\gamma_0\gamma_1\gamma_2\ldots\gamma_m]_l)$ digit by digit. The set $A^{t=k}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0} \wedge A^{t=l}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$ contains such minimum for each string.

disjunction $A^{t=k}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0} \vee A^{t=l}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$**:** Let us consider each string $\gamma_0\gamma_1\gamma_2\ldots\gamma_m$. Let $[\gamma_0\gamma_1\gamma_2\ldots\gamma_m]_k$ be a p-adic numerical value of $\gamma_0\gamma_1\gamma_2\ldots\gamma_m$ at $t = k$ and $[\gamma_0\gamma_1\gamma_2\ldots\gamma_m]_l$ be a p-adic numerical value of $\gamma_0\gamma_1\gamma_2\ldots\gamma_m$ at $t = l$. Then we define $\max([\gamma_0\gamma_1\gamma_2\ldots\gamma_m]_k, [\gamma_0\gamma_1\gamma_2\ldots\gamma_m]_l)$ digit by digit. The set $A^{t=k}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0} \vee A^{t=l}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$ contains such maximum for each string.

negation $\neg A^{t=k}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$**:** Let us define the universe $\Omega_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$ as a set of all possible strings started from the nodes γ_0, γ'_0, ..., γ''_0. A numerical value of each string $\gamma_0\gamma_1\ldots\gamma_m$ from $\Omega_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$ is maximal: $[\gamma_0\gamma_1\ldots\gamma_m] = \sum_{i=0}^{m}(p-1)\cdot p^i$. Then $\neg A^{t=k}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0} = \Omega_{\gamma_0,\gamma'_0,\ldots,\gamma''_0} \cap A^{t=k}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$ and it is coded by a set of p-adic integers $\sum_{i=0}^{m}(p-1)\cdot p^i - [\gamma_0\gamma_1\gamma_2\ldots\gamma_m]_k$ for each string $\gamma_0\gamma_1\gamma_2\ldots\gamma_m \in \neg A^{t=k}_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$.

By using this logic we can define an expansion strategy of the plasmodium on $\gamma_0\gamma_1\gamma_2\ldots\gamma_m$ in the universe $A_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}$:

$P(\gamma_0\gamma_1\gamma_2\ldots\gamma_m \in A_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}) = \bigwedge_{k=0}^{k=m}\sum_{t=0}^{\infty}(max([\gamma_k]_t, [\gamma_k]_{t+1}))\cdot p^i$.

$P(A_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}) = \{P(\gamma_0\gamma_1\gamma_2\ldots\gamma_m)$: $\gamma_0\gamma_1\gamma_2\ldots\gamma_m \in A_{\gamma_0,\gamma'_0,\ldots,\gamma''_0}\}$.

5. CONCLUSIONS

In any intelligent swarm behaviour there are emergent patterns that cannot be reduced to linear combinations of subsystems properly. Therefore conventional algorithms such as Kolmogorov-Uspensky machines have very low accuracy of their implementations on swarm systems (Section 3). Nevertheless, we can define a p-adic valued logic that can describe a massive-parallel behavior of swarms (Section 4).

Acknowledgment

This research is supported by FP7-ICT-2011-8.

6. REFERENCES

[1] A. Adamatzky, Physarum machine: implementation of a Kolmogorov-Uspensky machine on a biological substrate. *Parallel Processing Letters*, 17:455–467, 2007.

[2] A. Adamatzky, V. Erokhin, M. Grube, Th. Schubert, A. Schumann, Physarum Chip Project: Growing Computers From Slime Mould. *International Journal of Unconventional Computing*, 8(4):319–323, 2012.

[3] A. Adamatzky. *Physarum Machines: Computers from Slime Mould*. World Scientific Series on Nonlinear Science, Series A, 2010.

[4] G. Ariel, A. Shklarsh, O. Kalisman, C. Ingham, and E. Ben-Jacob. From organized internal traffic to collective navigation of bacterial swarms. *New Journal of Physics*, 15:125019 2013.

[5] E. Bonabeau, M. Dorigo, G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

[6] E. Cuevas, M. Cienfuegos, D. Zaldivar, M. Perez-Cisneros. A swarm optimization algorithm inspired in the behavior of the social-spider. *Expert Systems with Applications*, 40(16):6374–6384, 2013.

[7] N. C. Darnton, L. Turner, S. Rojevsky, and H. C. Berg. Dynamics of bacterial swarming. *Biophysics Journal*, 98:2082–2090, 2010.

[8] M. Dorigo, T. Stutzle. *Ant Colony Optimization*. MIT Press, 2004.

[9] D. Gordon. The Organization of Work in Social Insect Colonies. *Complexity*, 8(1):43–46, 2003.

[10] C. J. Ingham, E. Ben-Jacob. Swarming and complex pattern formation in Paenibacillus vortex studied by imaging and tracking cells. *BMC Microbiology*, 8(36), 2008.

[11] C. J. Ingham, O. Kalisman, A. Finkelshtein, E. and Ben-Jacob. Mutually facilitated dispersal between the nonmotile fungus Aspergillus fumigatus and the swarming bacterium Paeni bacillus vortex. *Proceedings of the National Academy of Sciences of the United States of America*, 108(49):19731–19736, 2011.

[12] D. Karaboga. *An Idea Based on Honey Bee Swarm for Numerical Optimization*. Technical Report-TR06. Engineering Faculty, Computer Engineering Department, Erciyes University, 2005.

[13] D. Karaboga, B. Akay, A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation*, 214(1):108–132, 1 August 2009.

[14] I. Kassabalidis, M.A. El-Sharkawi, R.J. II Marks, P. Arabshahi, A.A. Gray. Swarm intelligence for routing in communication networks, In *Global Telecommunications Conference, GLOBECOM'01*, 6, pages 3613–3617, IEEE, 2001.

[15] J. Kennedy, R. Eberhart. *Swarm intelligence*. Morgan Kaufmann Publishers, Inc., San Francisco, CA., 2001.

[16] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, vol. 4, pages 1942–1948, December 1995.

[17] A. N. Kolmogorov. On the concept of algorithm. *Uspekhi Matematicheskich Nauk*, 8(4):175–176, 1953.

[18] C. Lee, Emergence and Universal Computation. *Metroeconomica*, 55(2&3):219–238, 2004.

[19] K. M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems*, 22(3):52–67, IEEE, 2002.

[20] R. Rajabioun. Cuckoo Optimization Algorithm. *Applied Soft Computing*, 11:5508–5518, 2011.

[21] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21:25–34, 1987.

[22] R. G. Reynolds. An introduction to cultural algorithms. In *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 131–139, 1994.

[23] A. Schönhage, *Real-time simulation of multi-dimensional Turing machines by storage modification machines*. Project MAC Technical Memorandum 37, MIT, 1973.

[24] A. Schönhage. Storage modification machines. *SIAM Journal of Computation*, 9:490–508, 1980.

[25] A. Schumann. Towards context-based concurrent formal theories. *Parallel Processing Letters*, 25:1540008, 2015.

[26] A. Schumann, A. Adamatzky. The double-slit experiment with Physarum polycephalum and p-adic valued probabilities and fuzziness. *International Journal of General Systems*, 44(3):392–408, 2015.

[27] A. Schumann and A. Adamatzky. Physarum Spatial Logic. *New Mathematics and Natural Computation*, 7(3):483–498, 2011.

[28] A. Schumann, K. Pancerz. Towards an Object-Oriented Programming Language for Physarum Polycephalum computing: A Petri Net Model Approach. *Fundamenta Informaticae*, 133(2-3):271–285, 2014.

[29] A. Shklarsh, A. Finkelshtein, G. Ariel, O. Kalisman, C. Ingham, and E. Ben-Jacob. Collective navigation of cargo-carrying swarms. *Interface Focus*. 2:689–692, 2012.

[30] R. E. Tarjan, *Reference machines require non-linear time to maintain disjoint sets*. STAN-CS-77-603, March 1977.

[31] L. Turner, R. Zhang, N. C. Darnton, and H. C. Berg. Visualization of flagella during bacterial swarming. *Journal of Bacteriology*, 192:3259–3267, 2010.

[32] V. U. Uspensky, Kolmogorov and mathematical logic. *Journal of Symbolic Logic*, 57:385–412, 1992.

[33] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian. Self-adaptive learning based particle swarm optimization. *Information Sciences*, 181(20):4515–4538, 2011.

[34] S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1–35, 1984.