Particle Swarm Optimization with Protozoic Behaviour

Václav Snášel, Pavel Krömer, Ajith Abraham
IT4Innovations & Department of Computer Science
VŠB-Technical University of Ostrava
17. listopadu 12, Ostrava-Poruba, Czech Republic
Email: {vaclav.snasel,pavel.kromer,ajith.abraham}@vsb.cz

***Abstract*—Nature inspired algorithms implement successful optimization and adaptation strategies observed in the nature. Various bio-inspired algorithms mimic the behavioural patterns of plants, animals, their communities and their evolution. Surprisingly, the behavioural patterns and survival strategies of protozoa, one of the most prevalent and successful species on Earth, did not receive significant attention from the bio-inspired computing community until present time. This study proposes a new variant of Particle Swarm Optimization incorporating behaviour inspired by protozoa and evaluates the performance of such an algorithm on a set of well known test functions.**

***Index Terms*—Bio-inspired algorithms; particle swarm optimization; protozoic behaviour**

I. Introduction

Bio-inspired meta-heuristic algorithms use optimization strategies seen in the nature to solve practical search and optimization problems [1]. They mimic various natural phenomena including evolution and survival of the fittest (e.g. Evolutionary Algorithms [1]), behavioural patterns of individual organisms and collective intelligent behaviour of plants (e.g. Invasive Weed Optimization [2]), animals (e.g. Swarm Intelligent algorithms [1]). and human groups (e.g. Harmony Search [3]). The algorithms share a common goal of solving given problem but vary in the strategy they chose to explore (browse) problem space. The well known no free lunch theorem shows that different algorithms perform differently for different problems and classes of problems [4]. In conclusion, it advocates experimental evaluation of performance (speed of convergence, accuracy of results) of different algorithms for different problems, search for new variants of known algorithms and search for novel optimization algorithms.

Protozoa are a large group of unicellular organisms some of which exhibit an interesting behaviour [5], [6]. Motile protozoa hunt and feed like animals. However, their reproduction and survival strategy differs significantly from those of complex multicellular animals. The survival strategy of some protozoan species can be characterized by longevity (seemingly infinite lifespan), environment driven changes of reproduction strategies, and the presence of programmed cell death [5], [6], [7], [8]. Needless to say, such a behaviour appears to be a successful strategy as protozoa are an abundant, diverse, and versatile global family of organisms.

In this study we propose a modified Particle Swarm Optimization (PSO) [9], [1] algorithm extended by some elements of protozoic behaviour and evaluate the performance of protozoic PSO on a set of well-known test functions. The rest of this paper is structured in the following way: section II provides a brief introduction of protozoa and summary of computationally interesting reproductive behaviour of motile protozoa. Section III outlines related work and section IV describes the proposed modified protozoic PSO algorithm.The algorithm is experimentally evaluated in section V and conclusions are drawn in section VI.

II. Protozoa

Protozoa are unicellular eukaryotic microorganisms that lack cell walls. Some protozoa are motile at some stage of their life cycle and exhibit animal-like behaviour such as hunting and feeding [6], [10], [11]. Protozoa are diverse, ubiquitous, abundant, and have been recently proposed (due to these properties) as model organisms in microbiology [12]. They can be often found in aquatic environments. Ciliated protozoa *Tetrahymena thermophila* is shown in fig. 1. The cilia clearly visible on the picture are used to move the cell in its environment.

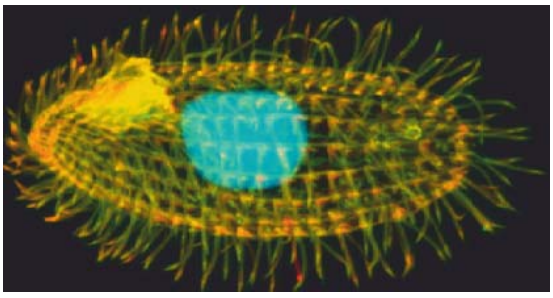

Fig. 1: *Tetrahymena thermophila*, a ciliated protozoa. Image from [12].

A. Fission and Conjugation

Some ciliated protozoa, including *Tetrahymena thermophila*, exhibit an interesting behaviour (slightly varying for different species) [6]. Their reproduction is based on

978-1-4799-0652-9/13 $31.00 © 2013 IEEE
DOI 10.1109/SMC.2013.347

a repeating asexual divide-growth cycle during which the protozoans create almost identical clones of themselves in a process called fission. Asexually reproducing protozoans are subject to senscence (degradation) due to accumulation of lethal genes. Fission rate of protozoans depends on the environment (availability of food), clonal age of the cell (decreases in time), and genetics of the cell [6].

Sexuality is in protozoa decoupled from reproduction and represents a qualitative rather than quantitative change (in contrast with fission). Two different protozoa participate in sexual conjugation and form two identical new cells. During the process, important parts of the cell including the macronucleus responsible for functioning of the cell are regenerated. The conjugation has for many species of protozoa repairing and rejuvenating effect and resets the fission rate [6].

The behaviour of protozoa is driven by environmental conditions. When the food is abundant and environmental conditions are good, protozoa culture progresses by fission. When the environmental conditions worsen, sexual conjugation starts to happen and genetically new rejuvenated cells are created [6]. In extremely hostile environments, some species of protozoa form resistant cysts that can survive until the living conditions improve [5].

The combination of fission and conjugation allows achieving clonal longevity and even a state similar to immortality (infinite lifespan) in which a culture of protozoa, given good environmental and nutrient conditions, flourishes for thousands of generations [6].

Reproduction by fission and conjugation is a specific survival strategy that (along with some other specific features) works very well for protozoa. In this work we enhance the PSO with protozoic behaviour in order to obtain a more robust version of PSO backed by a solid biological inspiration.

III. Related Work

Despite its success, the survival strategy of protozoa and its properties have been investigated by just a couple of studies from the optimization and computer science point of view.

The computational prospects of processes taking place in protozoan cell after conjugation were discussed in [13]. The sole optimization model based on protozoa known to the authors was introduced in 2012 by McCaffrey [14]. The Simulated Protozoa Optimization (SPO) is a self-adapting multi-agent optimization algorithm in which a population of agents finds optimal or sub-optimal solutions to given problem. The agents in SPO have position, velocity, and genotype (agent-specific control parameters). Simulated motility is a form of local search with random perturbations (agent moves in the direction of the best gradient in the neighborhood but can make a random error). The population of SPO agents has constant size. The fission is implemented by generating a copy of individual agent that can replace less fit agents in the population. Old agents die and are replaced by randomly generated individuals.

Conjugation is in SPO used to adapt control parameters of interacting agents. The level of environmental stress that triggers conjugation is expressed using agents current fitness and global best and global worst fitness of the population. Partners for conjugation are selected based on the closeness (similarity) of the agents.

The study [14] has compared the performance of SPO with PSO, Genetic Algorithms, and Bacteria Foraging Optimization algorithms on a series of test functions. It was shown that SPO performs better than other algorithms.

This study takes a different approach. The well known mechanics of PSO is used as a basis for a new extended PSO algorithm that is extended by protozoic behaviour. Based on the level of environmental stress, PSO particles reproduce either asexually by fission (with random errors) or sexually by conjugation.

IV. Particle Swarm Optimization with Protozoic Behaviour

The PSO algorithm is a global population-based search and optimization algorithm based on the simulation of swarming behavior of birds within a flock, schools of fish and even human social behavior [9], [1]. PSO uses a population of motile candidate particles characterized by their position x_i and velocity v_i inside the $n-$dimensional search space they collectively explore. Each particle remembers the best position (in terms of fitness function) it visited y_i and knows the best position discovered so far by the whole swarm $\bar{y}$. In each iteration, the velocity of particle i is updated according to [1]:

$$v_i^{t+1} = v_i^t + c_1 r_1^t (y_i - x_i^t) + c_2 r_2^r (\bar{y}^t - x_i^t) \tag{1}$$

where c_1 and c_2 are positive acceleration constants and r_1 and r_2 are vectors of random values sampled from uniform distribution. Vector y_i^t represents the best position known to particle i in iteration t and vector $\bar{y}^t$ is the best position visited by the swarm at time t.

The position of particle i is updated by [1]:

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{2}$$

The basic (*gbest*) PSO according to [9], [1] is summarized in Algorithm 1.

Algorithm 1: Summary of *gbest* PSO

```
Create population of M particles with random position and velocity;
Evaluate an objective function f ranking the particles in the population;
while Termination criteria not satisfied do
    for i ∈ {1, ..., M} do
        Set personal and global best position:
        if f(x_i) < f(y_i) then
            y_i = x_i
        end
        if f(x_i) < f(ȳ) then
            ȳ = x_i
        end
        Update velocity of i by (1) and position of i by (2);
    end
end
```

PSO is useful for dealing with problems in which the solution can be represented as a point or surface in an $n-$dimensional space. Candidate solutions (particles) are

placed in this space and provided with an initial (random) velocity. Particles then move through the solution space and are evaluated using some fitness function after each iteration. Over time, particles are accelerated towards those locations in the problem space which have better fitness values.

A. *proto*PSO, a PSO with protozoic behaviour

The *proto*PSO extends the traditional PSO algorithm in the following way:

- each particle i knows its *age* $\mathcal{A}(i)$ and *vitality* $\mathcal{V}(i)$
- based on its current vitality, each particle spawns offspring via *fission* (if the vitality of the particle is good) or performs *conjugation* if the vitality is bad
- particles older than *MAX_AGE* iterations are *removed* from the population
- if the population size falls bellow *M*, particles are *revived* with random age from the interval $[0, MAX\_AGE]$.

Particles in the initial *proto*PSO population have random position, random velocity, and random age drawn from the interval $[0, MAX\_AGE]$. The age of each particle is incremented in every iteration of the algorithm. The vitality of each particle $\mathcal{V}(i)$ is in each iteration updated using an arbitrary function $\nabla\mathcal{V}(i)$. The function used in this study was:

$$\nabla\mathcal{V}(i) = \left(f(x_i^{t-1}) - f(x_i)\right) \cdot iteration \tag{3}$$

where x_i^{t-1} is the previous position of particle i and $iteration$ is the sequential number of current iteration. Particles with good vitality (i.e. $\mathcal{V}(i) > 0$) get the chance to produce offspring particles by fission. Fission frequency of the real protozoa depends not only on the amount of food but also on the age of the cell. This is achieved in *proto*PSO by allowing the fission only to particles for which holds that $iteration$ modulo $\mathcal{A}(i)$ is equal to 0. That means that older particles spawn offspring less frequently than young particles. Offspring particle inherits velocity and position from its parent with 20% chance of random modification:

$$v_o(j) = \begin{cases} v_i(j) & \text{if } (rand(1) > 0.8) \\ v_i(j) \cdot rand(1) & \text{otherwise} \end{cases} \tag{4}$$

$$x_o(j) = \begin{cases} x_i(j) & \text{if } (rand(1) > 0.8) \\ x_i(j) \cdot rand(1) & \text{otherwise} \end{cases} \tag{5}$$

for all $j \in \{0, 1, \ldots, n\}$. In the previous, n is the dimension of problem space, $v_o(j)$ is the $j-$th velocity coordinate of the offspring particle, $x_o(j)$ is the $j-$th position coordinate of the offspring particle, $v_i(j)$ is the $j-$th velocity coordinate of parent particle, $x_i(j)$ is the $j-$th position coordinate of parent particle, and $rand(1)$ is uniform random number from the range $[0, 1]$. The age of offspring particle $\mathcal{A}(o)$ is defined as a function of particle fitness so that good particles produce longer living offspring particles. In this study was used $\mathcal{A}(o)$ defined by:

$$\mathcal{A}(o) = \min(f(x_i), MAX\_AGE) \tag{6}$$

Conjugation of i creates in *proto*PSO two new particles (exconjugants) by a recombination of two parent particles (conjugants). Second parent p is selected randomly and the velocity and position of the exconjugants is defined by:

$$v_{e1} = v_{e2} = v_i + rand(1) \cdot (v_i - v_p) \tag{7}$$

$$(x_{e1}, x_{e2}) = x_i \otimes x_p \tag{8}$$

where $\otimes$ is single point crossover of position vectors. The age of both exconjugants is set 0 (i.e. conjugation increases the lifespan of particles) and the exconjugants immediately replace their parents in the population.

The summary of *proto*PSO is shown in Algorithm 2.

Algorithm 2: Summary of *proto*PSO

```
Create population of M particles with random position, velocity, and age;
Evaluate an objective function f ranking the particles in the population;
while Termination criteria not satisfied do
    for i ∈ {1, ..., M} do
        Set personal and global best position:
        if f(x_i) < f(y_i) then
            y_i = x_i
        end
        if f(x_i) < f(ȳ) then
            ȳ = x_i
        end
        Update velocity of i by (1) and position of i by (2);
        A(i) ++;
        V(i) += ∇V(i);
        if V(i) > 0 then
            fission(i);
        else
            conjugation(i);
        end
        if A(i) > MAX_AGE then
            remove(i);
        end
    end
    if ACTIVE_PARTICLES < M then
        revive();
    end
end
```

V. Experiments

A series of computational experiments was conducted in order to verify the usefulness of PSO with protozoic behaviour.

A. Test problem

Several well known and widely used real-parameter optimization test functions [15] were selected as test problems. The functions were used because they are popular test problems with high computational requirements and known optima. The test functions used in this study are summarized in table I. PSO and *proto*PSO were used to search for optima of test functions with dimension $n = 20$. The search was terminated after each algorithm performed 1e6 fitness function evaluations.

$$f_1(\mathbf{x}) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e \tag{9}$$

Algorithm	Test function					
	f_1	f_2	f_3	f_4	f_5	f_6
PSO	6.96E-015	*2.49E-187*	*1.34E-185*	22.1544	5.44268	1731.38
$proto$PSO	*4.00E-015*	5.84E-102	5.59E-093	*0*	*2.61463*	*1125.16*

TABLE II: Average results of PSO and $proto$PSO after 1e6 fitness function evaluations (lower is better).

Function	Name	Equation	Parameter range
f_1	Ackley	(9)	[-30, 30]
f_2	De Jong	(10)	[-5.12, 5.12]
f_3	Griewank	(11)	[-600, 600]
f_4	Rastrigin	(12)	[-5.12, 5.12]
f_5	Rosenbrock	(13)	[-30, 30]
f_6	Schwefel	(14)	[-500, 500]

TABLE I: Test functions.

where n is problem dimension, $\mathbf{x} = (x_1, \ldots, x_n)$ is parameter vector, $e \approx 2.71828$ is Euler's number, and $\exp(a) = e^a$ is exponential function.

$$f_2(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \tag{10}$$

$$f_3(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) \tag{11}$$

$$f_4(\mathbf{x}) = 10 \cdot n + \sum_{i=1}^{n} \left(x_i^2 - 10 \cdot \cos(2\pi x_i)\right) \tag{12}$$

$$f_5(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100 \cdot (x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right) \tag{13}$$

$$f_6(\mathbf{x}) = \sum_{i=1}^{n} -x_i^2 \cdot \sin(\sqrt{|x_i|}) \tag{14}$$

B. Experiment settings

PSO and $proto$PSO were executed with the the same shared parameters: inertia weight was set to 0.729, cognitive weight and social weight was set to 1.49445 and population size was set to 100. The Mersenne Twister [16] algorithm initialized with current system time was used as pseudo-random number generator. The maximum age of $proto$PSO particles was set to 10. The execution of both PSO and $proto$PSO was terminated after 1000000 fitness function evaluations. Beacuse of the stochastic nature of the algorithms, 30 independent runs were performed and presented results are averages from the 30 runs.

C. Experiment results

The comparison of the results obtained by PSO and $proto$PSO is shown in table II. It can be seen that $proto$PSO has found better average solutions than traditional PSO for 4 out of 6 test functions. The solutions for Rastrigin, Rosenbrock, and Schwefel function obtained by $proto$PSO were significantly better than those found by PSO while the solutions for Ackley, Griewank, and De Jong function were comparable or slightly worse. The results suggest that the optimization strategies inspired by protozoa might be a helpful approach to some problems.

VI. Conclusions and Future Work

This work proposed and evaluated a new variant of PSO enhanced with behaviour inspired by protozoa named $proto$PSO. The algorithm combines the proven PSO-based optimization strategies (collective movement of a swarm of particles with local and social search) with innovative concepts implementing protozoic behaviour such as cellular fission, conjugation, and ageing. The algorithm was evaluated on a set of test functions.

The computational experiments have shown that $proto$PSO delivered better solutions for 4 out of 6 test functions. This is an encouraging result suggesting that protozoic behaviour and optimization strategies can improve the performance and results of existing nature-inspired optimization algorithms. However, the fact that $proto$PSO was able to find better results than the traditional PSO only for some test functions indicates that nature-inspired algorithms with protozoic behaviour are subject of the no free lunch theorem [4].

Future work on this topic will include the implementation of protozoic behaviour for more nature-inspired algorithms and their evaluation. Next, the relation between fission, conjugation, and ageing and exploration/exploitation tendencies in nature-inspired algorithms will be investigated and nature-inspired algorithms with protozoic behaviour will be compared to advanced self-adaptive variants of themselves. Finally, a new nature-inspired algorithm mimicking closely the complex behaviour and survival strategies of protozoa will be designed and evaluated.

Acknowledgement

This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and by the Bio-Inspired Methods: research, development and knowledge transfer project, reg. no. CZ.1.07/2.3.00/20.0073 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic. This work was also partially supported by the Grant of SGS No. SP2013/70, VŠB - Technical University of Ostrava, Czech Republic.

References

[1] A. Engelbrecht, *Computational Intelligence: An Introduction, 2nd Edition*. New York, NY, USA: Wiley, 2007.

[2] A. Mehrabian and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecological Informatics*, vol. 1, no. 4, pp. 355 – 366, 2006.

[3] Z. W. Geem, J.-H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[4] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 1, pp. 67–82, August 2002.

[5] I. Alcamo and J. Warner, *Schaum's Outline of Microbiology 2/e (e-book)*. Schaum's Outline Series, McGraw-Hill Education, 2009.

[6] G. Bell, *Sex and Death in Protozoa: The History of Obsession*. Cambridge University Press, 1988.

[7] I. Bruchhaus, T. Roeder, A. Rennenberg, and V. T. Heussler, "Protozoan parasites: programmed cell death as a mechanism of parasitism," *Trends in Parasitology*, vol. 23, no. 8, pp. 376 – 383, 2007.

[8] M. A. Fuertes, P. A. Nguewa, J. Castilla, C. Alonso, and J. M. Pérez Martín, "Programmed cell death in protozoa: An evolutionary point of view. the example of kinetoplastid parasites," in *Programmed Cell Death in Protozoa*, Molecular Biology Intelligence Unit, pp. 1–6, Springer New York, 2008.

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, pp. 1942–1948 vol.4, 1995.

[10] "Protozoan," *Encyclopædia Britannica. Encyclopædia Britannica Online.*, 2013. Available from http://www.britannica.com/EBchecked/topic/480488/protozoan, Accessed 14 May. 2013.

[11] "Microbiology," *Encyclopædia Britannica. Encyclopædia Britannica Online.*, 2013. Available from http://www.britannica.com/EBchecked/topic/380246/microbiology/216166/Fungi, Accessed 14 May. 2013.

[12] D. Montagnes, E. Roberts, J. Luke, and C. Lowe, "The rise of model protozoa," *Trends in Microbiology*, vol. 20, no. 4, pp. 184 – 191, 2012.

[13] M. Daley, "On the processing power of protozoa," in *Logic and Theory of Algorithms* (A. Beckmann, C. Dimitracopoulos, and B. Löwe, eds.), vol. 5028 of *Lecture Notes in Computer Science*, pp. 152–153, Springer Berlin Heidelberg, 2008.

[14] J. McCaffrey, "Simulated protozoa optimization," in *Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on*, pp. 179–184, 2012.

[15] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 Special Session on Real Parameter Optimization," tech. rep., Nanyang Technological University, 2005.

[16] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, pp. 3–30, Jan. 1998.