New inspirations in swarm intelligence: A survey

Article *in* International Journal of Bio-Inspired Computation · January 2011

DOI: 10.1504/IJBIC.2011.038700 · Source: DBLP

2 authors:

Rafael Stubs Parpinelli
Santa Catarina State University
126 PUBLICATIONS **2,631** CITATIONS

SEE PROFILE

Heitor Silvério Lopes
Federal University of Technology of Paraná
274 PUBLICATIONS **5,199** CITATIONS

SEE PROFILE

New inspirations in swarm intelligence: a survey

R.S. Parpinelli*

Bioinformatics Laboratory,
Federal University of Technology – Paraná (UTFPR),
Curitiba (PR), 80230-901, Brazil
and
Applied Cognitive Computing Group,
Santa Catarina State University (UDESC),
Joinville (SC), 89223-100, Brazil
E-mail: parpinelli@joinville.udesc.br
*Corresponding author

H.S. Lopes

Bioinformatics Laboratory,
Federal University of Technology – Paraná (UTFPR),
Curitiba (PR), 80230-901, Brazil
E-mail: hslopes@utfpr.edu.br

Abstract: The growing complexity of real-world problems has motivated computer scientists to search for efficient problem-solving methods. Evolutionary computation and swarm intelligence meta-heuristics are outstanding examples that nature has been an unending source of inspiration. The behaviour of bees, bacteria, glow-worms, fireflies, slime moulds, cockroaches, mosquitoes and other organisms have inspired swarm intelligence researchers to devise new optimisation algorithms. This tutorial highlights the most recent nature-based inspirations as metaphors for swarm intelligence meta-heuristics. We describe the biological behaviours from which a number of computational algorithms were developed. Also, the most recent and important applications and the main features of such meta-heuristics are reported.

Keywords: swarm intelligence; new meta-heuristics; bio-inspired algorithms; problem solving methods; biological behaviours; social living beings.

Reference to this paper should be made as follows: Parpinelli, R.S. and Lopes, H.S. (2011) 'New inspirations in swarm intelligence: a survey', *Int. J. Bio-Inspired Computation*, Vol. 3, No. 1, pp.1–16.

Biographical notes: R.S. Parpinelli received his BSc in Computer Science (2000) from the Maringá State University (UEM), and MSc in Computer Science (2001) from the Federal University of Technology – Paraná (UTFPR). Currently, he is a PhD candidate in Computer Science at UTFPR. He is also a Professor in the Computer Science Department of the Santa Catarina State University (UDESC). His main areas of interest are data mining, bioinformatics and all kinds of bio-inspired algorithms.

H.S. Lopes received his degrees in Electrical Engineering (1984) and MSc in Biomedical Engineering (1990) from the Federal University of Technology – Paraná – UTFPR, and PhD in Electrical Engineering (1996) from the Federal University of Santa Catarina. He is presently working as an Associate Professor in the Department of Electronics of UTFPR. He is the Head of the Bioinformatics Laboratory of UTFPR, since its foundation in 1997. He is also a Researcher of the Brazilian National Research Council. His current research interests are evolutionary computation applications, data mining and bioinformatics.

1 Introduction

Swarm-based systems are inspired by the behaviour of some social living beings, such as ants, termites, birds, and fishes. Self-organisation and decentralised control are remarkable features of swarm-based systems that, such as in nature, leads to an emergent behaviour. Emergent behaviour is a property that emerges through local interactions among system components and it is not possible to be achieved by any of the components of the system acting alone (Bonabeau et al., 1999; Garnier et al., 2007).

In the beginning, the two mainstreams of the swarm intelligence area were: ant colony optimisation (Dorigo and

Stützle, 2004) and particle swarm optimisation (Kennedy and Eberhart, 2001; Poli et al., 2007).

The ACO meta-heuristics is inspired by the foraging behaviour of ants. The ants' goal is to find the shortest path between a food source and the nest. Each path constructed by the ants represents a potential solution to the problem being solved. When foraging for food, ants lay down a chemical substance called pheromone. Ants can locally communicate to each other by means of the pheromone trails deposited in the environment. This indirect communication system is called stigmergy. When an ant finds a path from a food source to the nest, it deposits certain amount of pheromone in the path biasing other ants to follow that path. This is known as positive feedback and it is the result of successive deposits of pheromone on the same path: as more ants use a path, more pheromone will be present, and consequently, more ants will be attracted to it. As a chemical substance, the pheromone evaporates along time, therefore, reducing attractiveness strength of the trail. From the combination of stigmergy, positive feedback and evaporation an emergent behaviour takes place in the ant colony, leading them to find the shortest path between a food source and the colony.[1]

The PSO meta-heuristics[2] is motivated by the coordinate movement of fish schools and bird flocks. The PSO is compounded by a swarm of particles. Each particle represents a potential solution to the problem being solved and the position of a particle is determined by the solution it currently represents. In PSO, particles are 'flown' through hyperdimensional search space. Changes to the position of the particles within the search space are based on the socio-cognitive tendency of individuals to emulate the success of other individuals. Each individual of a population has its own life experience and is able to evaluate the quality of its experience. As social individuals they also have knowledge about how well their neighbours have behaved. These two kind of information corresponds to the cognitive component (individual learning) and social component (cultural transmission), respectively. Hence, an individual decision is taken considering both the cognitive and the social components, thus, leading the population to an emergent behaviour of forage for food or escape from a predator.

Both methods above cited have been applied successfully in a vast range of problems (Clerc, 2006). In recent years, new swarm intelligence algorithms have appeared, inspired by bacterial foraging (BFO) (Passino, 2002), fireflies bioluminescense (Krishnanand and Ghose, 2009; Yang, 2008), slime moulds life cycle (Monismith and Mayfield, 2008), cockroaches infestation (Havens et al., 2008), mosquitoes host-seeking (Feng et al., 2009), bats echolocation (Yang, 2010a), and various bees algorithms (BAs), i.e., inspired by bees foraging (Karaboga, 2005; Pham et al., 2006a) and bees mating (Haddad and Afshar, 2004) (for a comprehensive review about algorithms inspired by only bee swarms (see Karaboga and Akay, 2009b). In spite of the swarm inspiration common to these approaches they have their own particular way to exploit and explore the search space of the problem.

This work aims at surveying the most recent inspirations in the field of swarm intelligence, reporting them in a concise way. The way each approach search the space of solutions and other features (i.e., biological inspiration, communication model) are presented in the paper.

The following section describes these new approaches, from the inspirative nature phenomenon to the corresponding meta-heuristics. Section 3 shows applications of these algorithms in the most different domains. Some important features concerning all algorithms are summarised and discussed in Section 4. Later, some general conclusions are presented.

2 New inspirations and meta-heuristics

The careful observation of the behaviour of some living beings can give us insights on how to map their natural behaviour into algorithmic routines. That is why the new meta-heuristics discussed in this work are nature-inspired algorithms. These new approaches are global optimisation meta-heuristics and they are basically composed by a selection of the best scheme and by a randomisation scheme. The former guides, the algorithm convergence to the optimality (exploitation) and the later avoids both the loss of diversity and the algorithm to get trapped in local optima (exploration). A good balance between exploitation and exploration may lead to the global optimality achievement.

Sections 2.1 to 2.8 present some natural or behavioural phenomena in specific living beings that inspired computational models for problem solving.

2.1 Bee foraging

Many social insects, such as ants and bees, spend most of their life in foraging for food. Honey bee colonies have a decentralised system to collect the food and can adjust the searching pattern precisely in order to enhance the collection of nectar (Seeley, 1995).

Bees can estimate the distance from the hive to food sources by measuring the amount of energy consumed when they fly, besides the direction and the quality of the food source. This information is shared with their nestmates by performing a waggle dance and trophallaxis (direct contact). The dance floor is the place in the hive where the coming back forager bees perform the waggle dance to recruit more foragers. Bees that decide foraging without any guidance from other bees are called scouts. Bees that attend to the waggle dance at the dance floor can decide which food source to go based on his its quality. The quality of a food source is proportional to the quantity of nectar found there, and this information is transmitted by changing the intensity of the waggle dance and through antennae contacts. The better the food source, the most intense is the dance and the contacts (Reinhard and Srinivasan, 2009).

Each forager bee can behave in three different ways after unloading the food: it can perform the waggle dance to recruit more foragers to the same food source; it can abandon the food source due to loss of available resources; or it can directly return to foraging.

The basic idea concerning the algorithms based on the bee foraging behaviour is that foraging bees have a potential solution to an optimisation problem in their memory (i.e., a configuration for the problem decision variables). This potential solution corresponds to the location of a food source and has an aggregated quality measure (i.e., value of the objective function). The food source quality information is exchanged through the waggle dance that probabilistically biases other bees to exploit food sources with higher quality.

Some algorithms inspired by bee foraging behaviour have been found in literature including bee system (Sato and Hagiwara, 1997), honey bee algorithm (Nakrani and Tovey, 2003), BeeHive (Wedde et al., 2004), virtual bee algorithm (Yang, 2005), bee colony optimisation (Teodorovic and Dell'Orco, 2005), bees swarm optimisation (Drias et al., 2005), artificial bee colony (ABC) algorithm (Karaboga, 2005), BA (Pham et al., 2006a), honey bee foraging (Baig and Rashid, 2007). The two most widely used bee foraging inspired algorithms are described next.

2.1.1 Bees algorithm

The BA was first introduced by Pham et al. (2005) applied to a benchmark of mathematical functions. In this seminal work, the comparison with other meta-heuristics (simplex method, stochastic simulated annealing, genetic algorithm and ant colony system) showed that BA outperformed them, regarding processing speed and accuracy of results, thus, suggesting that BA is a powerful optimisation approach.

In this algorithm, a bee is a *d*-dimensional vector containing the problem variables and represents a possible solution to an optimisation problem. Moreover, a solution represents a visited site (i.e., food source) and has a fitness value assigned. The fitness is computed according to the objective function being optimised. The algorithm balances exploration and exploitation by using scout bees that randomly search for new sites and use recruitment for neighbourhood search in sites with the higher fitness, respectively (Pham et al., 2006a).

The algorithm starts with *n* scout bees randomly placed in the search space of dimension *d*. Each solution $\vec{x}_i = [x_{i1}, x_{i2}, ..., x_{id}]$ is evaluated by a fitness function $f(\vec{x}_i), i = 1, ..., n.$ Bees that have the highest fitnesses are chosen as 'selected bees' and sites visited by them (elite sites) are chosen for neighbourhood search. The algorithm conducts searches in the neighbourhood of the selected sites, assigning more bees to search near to the best sites (recruitment). The BA parameters are: number of scout bees (*n*); number of selected sites (*m*), out of the *n* bees; number of elite sites (*e*), out from the *m* selected sites; number of bees recruited for the best e elite sites (*nep*); number of bees recruited for the other (*m* – *e*) selected sites (*nsp*); and the radius for neighbourhood search (*ngh*). The BA is shown in Algorithm 1. Further information about BA can be found in its repository.[3]

Algorithm 1 Bees algorithm (BA)

```
1  Parameters: n, m, e, nep, nsp, ngh
2  Initialise the bees population x_i randomly
3  Evaluate fitness f(x_i) of the population
4  while stop condition not met do
5      Select m sites from n
6      for each m do
7          Select nsp sites from m
8          for each nsp do
9              Perform neighbourhood search with radius ngh
10             Update bee position according to f()
11         end for
12     end for
13     Select e elite sites from m
14     for each e do
15         for each nep do
16             Perform neighbourhood search with radius ngh
17             Update bee position according to f()
18         end for
19     end for
20     Assign remaining (m – e) bees to search randomly and
       evaluate their fitness
21     Rank all the bees and find the current best
22 end while
23 Postprocess results and visualisation
```

2.1.2 ABC algorithm

The ABC algorithm was first proposed by Karaboga (2005) for solving multidimensional and multimodal optimisation problems. A recent work (Karaboga and Akay, 2009a) compared the ABC algorithm performance against other population-based algorithms (genetic algorithm, particle swarm optimisation, differential evolution and evolution strategies) upon several benchmark functions. Results showed that the performance of the ABC was better than or similar to those of the other algorithms. Another relevant work concerning the ABC algorithm analysed the tuning of control parameters (Akay and Karaboga, 2009a).

Algorithm 2 Artificial bee colony (ABC) algorithm

```
1  Parameters: n, limit
2  Initialise the food sources x_i randomly
3  Evaluate fitness f(x_i) of the population
4  while stop condition not met do
```

```
5     for i = 1 to n / 2 do {Employed phase}
6         Select k, j and r at random such that
          k ∈ {1, 2, …, n}, j ∈ {1, 2, …, d},
7         r ∈ [0, 1]
8         v⃗ = x_ij + r · (x_ij − x_kj)
9         Evaluate solutions v⃗ and x⃗_i
10        if f(v⃗) is better than f(x⃗) then
11            Greedy selection
12        else
13            count_i = count_i + 1
14        end if
15    end for
16    for i = n / 2 + 1 to n do {Onlooker phase}
17        Calculate selection probability
18            P(x⃗_k) = f(x⃗_k) / Σ_{k=1}^{n} f(x⃗_k)
19        Select a bee using the selection probability
20        Produce a new solution v⃗ from the selected bee
21        Evaluate solutions v⃗ and x⃗_i
22        if f(v⃗) is better than f(x⃗) then
23            Greedy selection
24        else
25            count_i = count_i + 1
26        end if
27    end for
28    for i = 1 to n do {Scout phase}
29        if count_i > limit then
30            x⃗_i = random
31        end if
32    end for
33    Memorise the best solution achieved so far
34 end while
35 Postprocess results and visualisation
```

The ABC algorithm begins with n solutions (food sources) of dimension d that are modified by the artificial bees. Each solution $\vec{x}_i = [x_{i1}, x_{i2}, ..., x_{id}]$ is evaluated by a fitness function $f(\vec{x}_i), i = 1, ..., n$. The bees aim at discovering places of food sources (regions in the search space) with high amount of nectar (good fitness). There are three types of bees: the scout bees that randomly fly in the search space without guidance; the employed bees that exploit the neighbourhood of their locations selecting a random solution to be perturbed; and the onlooker bees that use the population fitness to select probabilistically a guiding solution to exploit its neighbourhood. If the nectar amount of a new source is higher than that of the previous one in their memory, they update the new position and forget the previous one (greedy selection). If a solution is not improved by a predetermined number of trials, controlled by the parameter *limit*, then the food source is abandoned by the corresponding employed bee and it becomes a scout bee. The ABC is shown in Algorithm 2. More about the ABC algorithm can be found in the repository.[4]

The ABC algorithm attempts to balance exploration and exploitation using the employed and onlooker bees to perform local search, and the scout bees to perform global search, respectively.

2.2 Bee mating

The honey bee mating process is started when the queen flies in a journey called mating flight. The queen is the only sexually productive female in the colony. Hence, it is the mother of all future queens, drones and workers. The lifetime of the queen is around one to three years, in average (Winston, 1991).

The drones follow the queen and the mating takes place in the air during seven or more days. The sperm of the drones are stored into a small organ called spermatheca in the queen's abdomen. The queen uses this random mix of accumulated sperm to fertilise its eggs during its whole live (Winston, 1991).

The basic idea concerning the algorithms based on bee mating behaviour is that the queen is considered the best solution to an optimisation problem and during the mating flight, it selects drones probabilistically for reproduction so as to form the spermatheca. The spermatheca is, then, a pool of selected solutions. New broods are created by crossovering the genotypes of drones and the queen. Natural selection takes place by replacing weaker queens by fitter broods. The algorithm inspired by the bee mating behaviour is described in details next.

2.2.1 Marriage in MBO algorithm

The first bee mating algorithm, called marriage in honey-bees optimisation (MBO) algorithm, was presented by (Abbass, 2001), and was applied to propositional satisfiability problems, known as 3-SAT problems.

In the MBO the mating flight can be seen as a set of transitions in a state space (fitness landscape), where queens (solutions) moves between different states and mates probabilistically with the drone encountered at each state. The probability of mating depends on the queen's energy and speed, and the fitness of the drone. The workers are heuristics, such as local search, used to improve the solutions. Each queen $\vec{q}_i = [q_{i1}, q_{i2}, ..., q_{id}]$ is characterised by a genotype of dimension d, a speed at instant $t(S_i(t))$, an energy at instant $t(E_i(t))$, and a spermatheca with a defined capacity. The MBO algorithm has some user-defined parameters: the number of queens (Q); the queen's spermatheca size (M), representing the maximum number of matings in a single mating flight; the number of broods (B) that will be born from the queen; and a speed reduction factor (α). The MBO is shown in Algorithm 3 and it can be summarised in five main steps:

1 Mating flight, where a queen selects drones probabilistically to form the spermatheca. A drone is randomly selected from the spermatheca for the creation of broods.

2 Creation of new broods (trial solutions) by crossoverring genotypes of drones and the queen.

3 Use of workers (heuristics) to conduct local search on broods (trial solutions).

4 Adaptation of workers' fitness based on the amount of improvement achieved on broods.

5 Replacement of weaker queens by fitter broods.

Algorithm 3 Marriage in honey-bees optimisation algorithm (MBO)

```
Parameters: Q, M, B, α
Initialise the population of queens q_i randomly
for each queen q_i do
    Use the workers to improve the queens' genotype
    Initialise spermatheca_i as empty
end for
for j = 1 to M do {mating-flight loop}
    t = 0
    for i = 1 to Q do
        Initialise E_i(t) and S_i(t) at random
        Initialise energy reduction step γ = 0.5E_i(t) / M
        Generate a drone D of dimension d at random
        while E(t) > 0 do
            Evaluate drone's genotype f(D) and queen's genotype f(q_i)
            if (rand < e^(-|f(q_i) - f(D)| / S_i(t))) and spermatheca_i < M then
                Add drone's sperm to spermatheca_i
            end if
            t = t + 1
            Update queen's internal energy: E_i(t + 1) = E_i(t) − γ
            Update queen's speed: S_i(t + 1) = αS_i(t)
            if rand < S_i(t) then
                Perturbate drone's genotype
            end if
        end while
    end for
    Generate B broods by crossover and mutation
    Use workers to improve the broods
    while the best brood is better than the worst queen do
        Replace the least-fittest queen with the best brood
        Remove the best brood from the brood list
    end while
end for
Postprocess results and visualisation
```

2.3 Bacterial chemotaxis

An Escherichia coli bacterium can move itself by rotating its flagella distributed around the cell body. When all flagella rotate counterclockwise they propel the bacterium along a trajectory, which is called run (or swim). When the flagella rotate clockwise, they pull on the bacterium in different directions and make the bacterium to tumble (Berg, 2003). The bacterium alternates between these two modes to search for nutrients in random directions. A proper combination of running and tumbling keeps the bacteria in places of higher concentration of nutrients. This foraging activity is called bacterial chemotaxis. This behaviour can be considered as an optimisation process that includes the exploitation of known resources and the exploration for new, potentially more valuable resources. The algorithm inspired by BFO is described next.

2.3.1 BFO algorithm

The BFO algorithm was first reported by Passino (2002) that applied the algorithm to the optimisation of a benchmark function.

Possible solutions to an optimisation problem are represented in the BFO algorithm by a colony of n bacteria of dimension d. Each solution $\vec{x}_i = [x_{i1}, x_{i2}, ..., x_{id}]$ is evaluated by a fitness function $f(\vec{x}_i), i = 1, ..., n$. The BFO consists of three main routines: chemotaxis, reproduction, and elimination-dispersal. In chemotaxis, a bacterium with random direction represents a tumble and a bacterium with the same direction of the previous step indicates a run. In reproduction, the health of each bacterium represents its fitness value. All bacteria are sorted according to their health status and only the first half of population survives. The surviving bacteria are split into two identical ones in order to form a new population. Thus, the population of bacteria is kept constant. The elimination-dispersal process is responsible for increasing the diversity of the population. The dispersion happens after a certain number of reproduction steps, when some bacteria are chosen according to a preset probability. Such bacteria are killed and new ones are randomly generated in another position within the search space. The exploitation of the search space is accomplished by both chemotaxis and reproduction steps, while the exploration is done by the elimination-dispersal step.

Algorithm 4 Bacterial foraging algorithm (BFO)

1 Parameters: n, N_c, N_s, N_{re}, N_{ed}, P_{ed}, $C(i)$ (i = 1, 2, …, n)
2 Initialise randomly the bacterial colony $\vec{x}_i$
3 **for** l = 1 **to** N_{ed} **do** {*Elimination-dispersal loop*}
4 **for** k = 1 **to** N_{re} **do** {*Reproduction loop*}
5 **for** j = 1 **to** N_c **do** {*Chemotaxis loop*}
6 **for** i = 1 to n **do**
7 Compute fitness $f\left(\vec{x}_i^{j,k,l}\right)$
8 Tumble: Generate random vector $\Delta(i) \in [-1, 1]^d$
9 Move: $\vec{\theta}_i^{j,k,l} = \vec{x}_i^{j,k,l} + C(i)\frac{\Delta(i)}{\sqrt{[\Delta(i)]^T \cdot \Delta(i)}}$
10 Compute $f\left(\vec{\theta}_i^{j,k,l}\right)$
11 $m = 0$
12 **while** $m < N_s$ **do** {*Run loop*}
13 **if** $f\left(\vec{\theta}_i^{j,k,l}\right) < f\left(\vec{x}_i^{j,k,l}\right)$ **then**
14 Update solution
15 Move again
16 **else**
17 $m = N_s$
18 **end if**
19 **end while**
20 **end for**
21 **end for**
22 **for** i = 1 **to** n **do**
23 $J_{health}^i = \sum_{j=1}^{N_c+1}\left(f\left(\vec{x}_i^{j,k,l}\right)\right)$
24 Sort bacteria by ascending values
25 Best half of the colony duplicates and replaces the worst part
26 **end for**
27 **end for**
28 **for** i = 1 **to** n **do**
29 **if** *rand* < P_{ed} **then**
30 Generate a new random bacterium i
31 **end if**
32 **end for**
33 **end for**
34 Postprocess results and visualisation

The parameters involved in the BFO algorithm are: the number of chemotactic steps (N_c); the number of run steps (N_s); the number of reproductive steps (N_{re}); the number of elimination-dispersal steps (N_{ed}); the probability of elimination (P_{ed}); and the size of the step taken in each run or tumble ($C(i)$, for each bacterium i). The BFO is shown in Algorithm 4.

2.4 Lampyridae bioluminescense

Lampyridae is a family of insects (order *Coleoptera*) that are capable to produce natural light (bioluminescense) to attract a mate or a prey. They are commonly called fireflies or lightning bugs. In the species *Lampyris noctiluca* the fireflies are also known as glow-worms and, despite of the name, they are not worms. In this species, it is always the female who glows, and only the male has wings. In other species, *Luciola lusitanica*, both male and female firefly may emit light and both have wings (Fraga, 2008; Shimomura, 2006).

If a firefly is hungry or looks for a mate its light glows brighter in order to make the attraction of insects or mates more effective. The brightness of the bioluminescent light depends on the available quantity of a pigment called *luciferin*, and more pigment means more light (Tyler, 2002).

Two optimisation algorithms were inspired by the bioluminescent behaviour and are described next.

2.4.1 Glow-worm swarm optimisation (GSO) algorithm

The GSO algorithm was first presented by Krishnanand and Ghose (2005) as an application to collective robotics.

In this algorithm, each glow-worm uses a probabilistic mechanism to select a neighbour that has a luciferin value associated with him and moves towards it. Glow-worms are attracted to neighbours that glow brighter. The movements are based only on local information and selective neighbour interactions. This enables the swarm to divide into disjoint subgroups that can converge to multiple optima of a given multimodal function.

The GSO is shown in Algorithm 5. It starts by placing a population of n glow-worms of dimension d randomly in the search space. Each solution $\vec{x}_i = \left[x_{i1}, x_{i2}, ..., x_{id}\right]$ is evaluated by a fitness function $f\left(\vec{x}_i\right), i = 1, ..., n.$ At the beginning, all the glow-worms contain an equal quantity of luciferin l_0 and the same neighbourhood range decision r_0. Each iteration consists of a luciferin update phase followed by a movement phase based on a transition rule. Other involved parameters are the luciferin decay constant (ρ), the luciferin enhancement constant (γ), the step size (s), the number of neighbours (n_t), the sensor range (r_s) and a constant value (β). The authors observed that the only two parameters that influences the algorithm behaviour are n and r_s.

Algorithm 5 Glow-worm swarm optimisation algorithm (GSO)

1 Parameters: n, l_0, r_0, ρ, γ, β, s, r_s, n_t
2 Generate the glow-worms population $\vec{x}_i$ randomly
3 **for** i = 1 **to** n **do**

4 Initialise luciferin $l_i(0) = l_0$
5 Initialise neighbourhood range $r_d^i(0) = r_0$
6 **end for**
7 $t = 1$
8 **while** stop condition not met **do**
9 **for** each glow-worm *i* **do** {*update luciferin*}
10 $l_i(t+1) = (1-\rho)\cdot l_i(t) + \gamma \cdot f(x_i(t))$
11 **end for**
12 **for** each glow-worm i **do** {*movement phase*}
13 Find neighbours $N_i(t)$
14 **for** each glow-worm $j \in N_i(t)$ **do**
15 Find probability $P_{ij}(t) = \frac{l_j(t)-l_i(t)}{\sum_{k\in N_i(t)} l_k(t)-l_i(t)}$
16 **end for**
17 Select glow-worm j using P_{ij}
18 Update glow-worm position with $x_i(t+1) = x_i(t) + s\left(\frac{x_j(t)-x_i(t)}{\|x_j(t)-x_i(t)\|}\right)$
19 Update decision range:
20 $r_d^i(t+1) = \min\left\{r_s, \max\left\{0, r_d^i(t) + \beta\cdot\left(n_t - |N_i(t)|\right)\right\}\right\}$
21 **end for**
22 $t = t + 1$
23 **end while**
24 Postprocess results and visualisation

2.4.2 *Firefly algorithm*

The firefly algorithm (FA) was proposed by Yang (2008) and the algorithm was applied to the optimisation of benchmark functions. The FA uses three main basic rules:

1 a firefly will be attracted by other fireflies regardless their sex

2 attractiveness is proportional to their brightness and decreases as the distance among them increases

3 the landscape of the objective function determines the brightness of a firefly.

This algorithm assumes that a population of *n* candidate solutions for an optimisation problem are agents of type firefly. These agents are vectors of dimension *d* representing the problem variables. Each solution $\vec{x}_i = \left[x_{i1}, x_{i2}, ..., x_{id}\right]$ is evaluated by a fitness function $f(\vec{x}_i), i = 1, ..., n$ that represents his quality. Each agent glows proportionally to its quality which, together with his attractiveness (β), dictate how strong it attracts other members of the swarm. Two other user-defined parameters are the maximum attractiveness value (β_0) and the absorption coefficient (γ) that determines the variation of attractiveness with increasing distance from communicated firefly. The FA is summarised in Algorithm 6.

Algorithm 6 Firefly algorithm (FA)

1 Parameters: n, β_0, γ
2 Initialise the fireflies population $\vec{x}_i$ randomly
3 Compute $f(\vec{x})$
4 **while** stop condition not met **do**
5 $i^{\min} = \arg\min_i \left\{f(\vec{x}_i)\right\}$
6 $\vec{x}_i^{\min} = \arg\min_{\vec{x}_i} \left\{f(\vec{x}_i)\right\}$
7 **for** $i = 1$ **to** n **do**
8 **for** $j = 1$ **to** n **do**
9 **if** $f(\vec{x}_j) < f(\vec{x}_i)$ **then** {*Move firefly i towards j*}
10 Calculate distance r_j
11 Obtain attractiveness: $\beta \leftarrow \beta_0 e^{-\gamma r_j}$
12 Generate a random solution $\vec{u}_i$
13 **for** $k = 1$ **to** d **do**
14 $x_{i,k} = (1-\beta)x_{i,k} + \beta x_{j,k} + u_{i,k}$
15 **end for**
16 **end if**
17 **end for**
18 **end for**
19 Generate a random solution $\vec{u}$
20 **for** $k = 1$ **to** d **do** {*Best firefly moves randomly*}
21 $x_{i^{\min},k} = x_{i^{\min},k} + u_k$
22 **end for**
23 Compute $f(\vec{x})$
24 Find the current best
25 **end while**
26 Postprocess results and visualisation

2.5 *Slime mould life cycle*

The cells of a slime mould are known to biologists as the amoeba *Dictyostelium discoideum* – Dd. Amoebae perform a random search for food and move using their pseudopodia[5] as sensors to detect nearby food sources. The pseudopod is not always completely accurate, but they work, most time, to direct the amoebae towards food when it is available (Kessin, 2001). Dd cells move independently until starvation, emitting a chemical substance known as cyclic adenosine monophosphate (cAMP). The last form that Dd undergo during starvation is a period of grouping together (aggregation). Thereafter, cells group together in streams and eventually form a mound. The mound forms a slime sheath about itself to protect from predatory multicellular organisms and to forage new regions. Once the mound is complete, cells orient themselves to form a head and a tail. At this point, the slug is not a multicellular organism, but a group of single-cell organisms working

towards a common goal (find food). This process continues until the slug reaches a location with resources or until resources have been depleted. If all resources have been depleted, the amoeba dies. However, if a source is available, culmination occurs and a fruiting body is formed. During culmination and formation of the fruiting body, spores are created and are dispersed by wind, birds or invertebrates. When spores arrive at another location, they may lie dormant for some time. After a period of dormancy, spores form new amoebae, and the life cycle of Dd begins again.

Some algorithms that are inspired by this behaviour have been found in literature including cell-based optimisation algorithm (Rothermich et al., 2003) that uses only a portion of the Dd life cycle and the slime mould optimisation algorithm (SMOA) (Monismith and Mayfield, 2008) that uses the life cycle as a whole. Next section describes the SMOA.

2.5.1 Slime mould optimisation algorithm

The SMOA was introduced by Monismith and Mayfield (2008), and was applied to the optimisation of benchmark functions. Although authors achieved good results compared to the known optimum values of the benchmarks, no comparisons were done with other algorithms.

The SMOA consists of an amoebae population of size n representing possible solutions for an optimisation problem of dimension d. Each solution $\vec{x}_i = [x_{i1}, x_{i2}, ..., x_{id}]$ is evaluated by a fitness function $f(\vec{x}_i), i = 1, ..., n$. In the algorithm, amoebae may take a number of states: vegetative amoeba, aggregating amoeba, mound, slug, fruiting body, and dispersal. In the algorithm, amoebae begin in the vegetative state, and are assigned to random positions in the search space. They are given time to search for local optima (i.e., food). Based on their initial positions, a mesh is formed using the approximate nearest neighbour algorithm (ε-ANN, where ε is the number of neighbours being considered) (Arya et al., 1998). In the vegetative state, amoebae perform a semi-random search. This is accomplished by the ability of amoebae to extend their pseudopodia towards multiple directions and, so, perform a local search. A parameter k represents the number of pseudopodia (and the number of directions explored by each amoeba). A simulation of starvation is necessary to start an aggregative state for the amoebae. This change is controlled by two parameters: $t_{unimproved}$ iterations without improving the best solution of an amoeba; and $t_{lifetime}$, the number of time steps since its last dispersal event. As the number of starving amoebae in one distinct lattice point increases above some preset threshold (A), the probability of forming a mound also increases. Once there is no more improvement in the slug movement, it must be dispersed to continue searching for new locations with better results. Through the algorithm, the communication is performed using cAMP trails. A high-level view of SMOA is shown in Algorithm 7. For a complete description (see Monismith and Mayfield, 2008).

Algorithm 7 Slime mould optimisation algorithm (SMOA)

```
1  Parameters: n, k, ε, A
2  Generate the amoeba population x_i randomly
3  Evaluate fitness f(x_i)
4  To all amoeba set the state to VEGETATIVE
5  Archive the best objective function value from the
   amoebae
6  Input the locations of each amoeba to ε-ANN
7  Create a mesh based on the results of ε-ANN
8  for each amoeba i do
9        switch amoebae state
10           case VEGETATIVE: Vegetative movement
11           case AGGREGATIVE: Aggregation
12           case MOUD: Moud formation
13           case DISPERSAL: Dispersal
14       end switch
15 end for
16 Postprocess results and visualisation
```

2.6 Cockroaches infestation

Cockroaches are insects of the order *Blattodea* and are one of the most ancient animals on earth, having appeared around for 350 million years.

Communication between cockroaches occurs mainly through chemical trails in their feces as well as emitting airborne pheromone for swarming and mating. Using these signs other cockroaches can follow the trails to discover sources of food and water, and places where other cockroaches are hiding (Bell et al., 2007). Cockroaches are mainly nocturnal insects and will run away when exposed to light. To decide which path to follow cockroaches basically use two pieces of information: how dark is the environment, and how many other cockroaches are there (Halloy et al., 2007). In other words, in an infestation the cockroaches will prefer to group themselves in darker places with other cockroaches.

Next section describes the cockroaches infestation algorithm.

2.6.1 Roach infestation optimisation

The roach infestation optimisation (RIO) was introduced by Havens et al. (2008). They applied RIO to benchmark functions and achieved competitive results compared to a standard PSO. Actually, RIO has some elements that resemble the traditional PSO algorithm.

In RIO algorithm, cockroaches agents are defined using three simple behaviours:

- cockroaches search for the darkest location in the search space and the fitness value is directly proportional to the level of darkness (find darkness phase)

- cockroaches socialise with nearby cockroaches (find friend phase)
- cockroaches periodically become hungry and leave the friendship to search for food (find food phase).

Algorithm 8 Roach infestation optimisation algorithm (RIO)

1 Parameters: n, $t_{\max}$, C_0, $C_{\max}$, A, t_{hunger}
2 Initialise the roaches population $\vec{x}_i$ and $\vec{v}_i$ randomly
3 **for** $i = 1$ **to** n **do**
4 Set $\vec{p}_i$ as $\vec{x}_i$
5 Initialise $hunger_i$ randomly between $\{0, t_{hunger} - 1\}$
6 **end for**
7 **for** $t = 1$ to $t_{\max}$ **do**
8 M = distances from one roach to each other
9 d_g = average distance from M
10 **for** $i = 1$ **to** n **do**
11 if $f(\vec{x}_i) < f(\vec{p}_i)$ **then** {*Find darkness phase : Update best roach location*}
12 $\vec{p}_i = \vec{x}_i$
13 **end if**
14 Compute the neighbours of roach i:
15 $\{j\} = \{k: 1 \le k \le n, k \ne I, M_{ik} < d_g\}$
16 N_i = number of neighbours $|\{j\}|$
17 **for** $q = 1$ **to** N_i **do** {*Choose a neighbour*}
18 **if** $rand < A$ **then**
19 $\vec{l}_i = \arg\min_k \{f(\vec{p}_k)\}, k = \{i, q\}$
20 **end if**
21 **end for**
22 **if** $hunger_i < t_{hunger}$ **then** {*Find friend phase*}
23 $\vec{v}_i = C_0\vec{v}_i + C_{\max}R_1(\vec{p}_i - \vec{x}_i) + C_{\max}R_2(\vec{l}_i - \vec{x}_i)$
24 $\vec{x}_i = \vec{x}_i + \vec{v}_i$
25 Increment $hunger_i$
26 **else** {*Find food phase*}
27 x_i = random food location. Random position.
28 $hunger_i = 0$
29 **end if**
30 **end for**
31 **end for**
32 Postprocess results and visualisation

In fact, RIO is a cockroach-inspired PSO in which a population of n cockroaches (possible solutions) of dimension d use darkness sense (fitness function evaluation) and neighbourhood communication to move through a search space updating their positions $\vec{x}_i$ and velocities $\vec{v}_i$. Each solution $\vec{x}_i = [x_{i1}, x_{i2}, ..., x_{id}]$ is evaluated by a fitness function $f(\vec{x}_i), i = 1, ..., n$. Each roach keeps its best location p_i found so far. The other tunable parameters involved are $t_{\max}$ that is the maximum number of iteration of the algorithm, C_0 that ponderates the relative velocity importance, $C_{\max}$ that ponderates the relative importance of both roach personal best and roach neighbour position, A that influences the neighbour choice, and t_{hunger} that defines the hunger interval. The RIO is shown in Algorithm 8.

2.7 Mosquito host-seeking

Mosquitoes are insects from the family *Culicidae*. Both male and female mosquitoes feed on nectar (or other sugar source). However, only the female of many species is also capable of taking blood. A blood meal is necessary to develop and nourish the eggs. To do so, female mosquitoes have to seek for animals or humans as possible sources of blood. This is known as host-seeking behaviour.

To find its blood host the female mosquito receives external environmental information from its sensory receptors. These sensory receptors respond to varying concentrations of attractants such as carbon dioxide (CO_2) and L-lactic acid in the air. A mosquito, upon a host, chooses a specific body region for feeding according to the skin temperature and humidity (Mehlhorn, 2001).

A swarm of mosquitoes randomly searches a host to attack. The host-seeking behaviour of a mosquito can be summarised in three main steps:

1 it looks randomly for CO_2 or a smelling substance
2 once identified the smell, it seeks towards a place of high-concentration of this smell
3 it lands when it feels the radiated heat of the host.

2.7.1 Mosquito host-seeking algorithm

The MHSA treats every entry of the TSP matrix as an artificial mosquito m_{ij}. Hence, the n-city TSP is transformed into host-seeking behaviour of a swarm of $n \times n$ artificial mosquitoes. A host is considered as an edge between cities for the TSP. Each entry m_{ij} of the TSP matrix (a mosquito) is a triple composed by d_{ij} (distance between cities i and j), x_{ij} (mosquito sex: 0 for male and 1 for female), and r_{ij} (distance between mij and the host). Each r_{ij} ranges from 0 to 1 as the artificial mosquito moves, and $r_{ij} = 1$ represents that the artificial mosquito m_{ij} is attacking the host and the shortest path passes through this host.

When all mosquitoes are in a state of equilibrium, all r_{ij} will be 1 or 0. The equilibrium state indicates that the swarm have found a possible solution for the problem. Hence, a solution for the TSP using the MHSA is the set of mosquitoes that have successfully attacked a host ($r_{ij} = 1$). For an extended description of the mosquito host-seeking algorithm (see Feng et al., 2009).

2.8 Bat echolocation

Several animals such as dolphins, shrews, most bats, and most whales use echolocation (also called as biosonar) for

navigation, communication and foraging. Most bats use echolocation at a certain degree and, among all the species, microbats use echolocation extensively. In microbats, echolocation is a type of sonar used to avoid close obstacles in the dark, detect prey, and locate roosting crevices (places to sleep over) (Altringham et al., 1998). During echolocation these microbats emit a series of short, high-frequency sounds and listen for the echo that bounces back from the surrounding objects. With this echo a bat can determine an object's size, shape, direction, distance, and motion. When hunting for a prey, the rate of pulse emission can be speed up to about 200 pulses per second when they fly near their prey and every pulse has a constant frequency. Moreover, the wavelengths of a pulse are in the same order (is proportional) of their prey sizes. The loudness also varies from the loudest when searching for prey and to a quieter base when homing towards the prey (Altringham et al., 1998). Next section describes the bat echolocation algorithm.

2.8.1 Bat algorithm

The bat algorithm (BA) was first presented in Yang (2010b). They applied to benchmark functions, and achieved better results compared to genetic algorithms and PSO. To date, no other application to real-world problems was found using the BA.

The basic idea behind the BA is that a population of n bats (possible solutions) of dimension d use echolocation to sense distance and fly randomly through a search space updating their positions $\vec{x}_i$ and velocities $\vec{v}_i$. Each solution $\vec{x}_i = \left[x_{i1}, x_{i2}, ..., x_{id}\right]$ is evaluated by a fitness function $f\left(\vec{x}_i\right), i = 1, ..., n.$ The bats' flight aims at finding food/prey (best solutions). Two other parameters are: the loudness decay factor (α) that acts in a similar role as the cooling schedule in the traditional simulated annealing optimisation method, and the pulse increase factor (γ) that regulates the pulse frequency. The properly update for the pulse rate (r_i) and the loudness (A_i) balances the exploitation and exploration behaviour of each bat, respectively. As the loudness usually decrease once a bat has found its prey/solution (in order to do not loss the prey), the rate of pulse emission increases in order to raise the attack accuracy. The BA pseudo-code is shown in Algorithm 9.

The BA is shown in Algorithm 9.

Algorithm 9 Bat algorithm (BA)

1 Parameters: n, α,γ
2 Initialise the bats population $\vec{x}_i$ and $\vec{v}_i$ randomly
3 Define pulse frequency $\vec{f}_i$ at $\vec{x}_i$
4 **for** i = 1 **to** n **do**
5 Initialise pulse rates r_i and loudness A_i
6 **end for**
7 Compute $f\left(\vec{x}_i\right)$
8 Find the current best $\vec{x}_*$
9 **while** stop condition not met **do**
10 **for** i = 1 **to** n **do**
11 Generate new solutions by adjusting:
12 Frequency: $\vec{f}_i = \vec{f}_{\min} + \left(\vec{f}_{\max} - \vec{f}_{\min}\right)\beta, \beta \in [0,1]$
13 Velocity: $\vec{v}_i^t = \vec{v}_i^{t-1} + \left(\vec{x}^t - \vec{x}_*^t\right)\vec{f}_i$
14 Location: $\vec{x}^t = \vec{x}_i^{t-1} + \vec{v}_i^t$
15 **if** *rand* > r_i **then**
16 Select a solution among the best solutions
17 Generate a local solution around the selected best solution
18 **end if**
19 Generate a new solution by flying randomly
20 **if** *rand* < A_i & $f\left(\vec{x}_i\right) < f\left(\vec{x}_*\right)$ **then**
21 Accept the new solutions
22 Increase $r_i : r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)]$
23 Decrease: $A_i : A_i^{t+1} = \alpha A_i$
24 **end if**
25 **end for**
26 Find the current best x_*
27 **end while**
28 Postprocess results and visualisation

3 Applications

For all the algorithms mentioned in the previous sections, a search in the literature was done to find applications in the most different domains. Although this search is not exhaustive, it covers the most relevant applications, and emphasises the applicability of those algorithms.

The BA (Section 2.1.1) was applied with success in some problems including training of multi-layered perceptron neural networks (Pham et al., 2006b), job shop scheduling optimisation (Pham et al., 2007a), data clustering (Pham et al., 2007b), multi-objective optimisation (Pham and Ghanbarzadeh, 2007), protein folding optimisation using the torsion angles model (Bahamish et al., 2008), optimisation of fuzzy logic controller parameters (Pham and Kalyoncu, 2009), peer-to-peer file sharing in mobile ad-hoc networks (Dhurandher et al., 2009), and interference suppression of linear antenna arrays (Guney and Onay, 2010).

Applications found in the literature using the ABC algorithm (Section 2.1.2) include: the generalised assignment problem optimisation (Baykasoğlu et al., 2007), energy distribution network configuration (Srinivasa et al., 2008; Linh and Anh, 2010), neural network training (Karaboga and Ozturk, 2009), multi-objective optimisation (Pawar et al., 2008), data clustering (Marinakis et al., 2009), solving integer programming benchmarks (Akay and Karaboga, 2009b), template matching in digital images (Chidambaram and Lopes, 2009), and signal model parameter extraction (Sabata et al., 2010).

The Marriage in MBO algorithm (Section 2.2.1), was applied to several problems in the literature, for instance: three-SAT problem optimisation (Abbass, 2001; Teo and Abbass, 2003), MAX-SAT problem optimisation (Benatchba et al., 2005), water resources management (Haddad and Afshar, 2004), non-linear constrained and unconstrained optimisation (with an updated version called honey bee mating optimisation algorithm – HBMO) (Haddad et al., 2006), stochastic dynamic programming (Chang, 2006), continuous optimisation (with an improved version of HBMO) (Afshar et al., 2007), data clustering (Fathian et al., 2007), stepped spillway optimum design (Haddad et al., 2008), and reconfiguration of multi-objective distribution feeder (Taher, 2009).

The BFO algorithm (Section 2.3.1) has been used for some applications such as multivariate PID controller tuning (Kim and Cho, 2005a; Luo and Chen, 2010), power systems harmonic estimation (Mishra, 2005), power transmission loss optimisation (Tripathy et al., 2006), machine learning (Kim and Cho, 2005b), multi-objective optimisation (Hazra and Sinha, 2008), prediction of stock market indexes (Majhi et al., 2009), identification of dynamic and non-linear systems (Majhi and Panda, 2008, 2009), and optimisation of fuzzy controller (Alavandar et al., 2010). Some other applications are summarised in Das et al. (2009). Recently, an updated version of the algorithm named self-adaptive bacterial foraging optimisation (SABFO) was proposed by Chen et al. (2008). Results obtained upon several benchmark functions showed a significant improvement in performance over the original BFO, whilst similar or even superior performance compared to PSO and GA.

The GSO algorithm (Section 2.4.1) was applied in Krishnanand and Ghose (2009) to find multiple optima of multimodal benchmark functions. This work also conducted detailed parameter tuning experiments. The comparison with a Niched-PSO showed better results concerning the number of peaks found in almost all test functions. Another application using GSO is for hazard sensing in ubiquitous environments (Krishnanand and Ghose, 2008).

For the FA (Section 2.4.2), a recent work done by Lukasik and Zak (2009) performed an extensive set of empirical tests for parameter tuning, and proposed some extensions to the algorithm. Some comparisons against PSO on benchmark functions showed competitive results, suggesting that the FA is a powerful optimisation approach. Some extensions in the algorithm was proposed in Yang (2010a) combining Lévy flights with the FA search strategy. Since FA is a very recent meta-heuristic, to date no other application to real-world problems was found.

Concerning the SMOA (Section 2.5.1) no other applications to problem solving were found to date, besides that of the original paper (Monismith and Mayfield, 2008).

A single application of the RIO algorithm (Section 2.6.1) was found: it was used for pattern recognition in images, checking the posture and stability of elders while they are performing exercises (Havens et al., 2009).

Concerning the mosquito host-seeking algorithm (MHSA, Section 2.7.1) and the BA (Section 2.8.1), to date, no other application to real world problems was found, besides the original works for solving the symmetric travelling salesman problem (TSP) (Feng et al., 2009), and for solving benchmark functions (Yang, 2010b), respectively.

Most of the above-mentioned algorithms are very recent. Although few applications have appeared to date, we can notice a growing interest in bio-inspired computation.

4 Discussion

Table 1 summarises all algorithms in terms of biological inspiration, optimisation domain, mechanisms of exploitation and exploration, and the communication model for each approach.

The second column of Table 1 indicates the biological inspiration behind of each algorithm. All the inspiring behaviours have in common two facts: they are compounded by a distributed society/population of individuals where the control is also distributed among the individuals (there is no centralised control); and the individuals' decision-making is stochastic and based only on local information, without knowledge of the global pattern/solution (communications among them are localised). Moreover, the society-level behaviour transcends the behaviour of a single individual, leading to an emergent behaviour through self-organisation.

The third column of Table 1 classifies the algorithms according to the domain they were first applied, either continuous or discrete optimisation. This classification was done concerning the algorithms only in their first canonical publication. An approach inserted in the continuous domain is characterised by solving problems where the variables to be optimised in the objective function can assume only real values. On the other hand, the discrete domain is characterised by solving problems where the variables to be optimised in the objective function are restricted to assume only discrete values, such as integers. Moreover, some algorithms were further adapted to handle the other optimisation domain, different from the original. It is the case, e.g., of BA in Pham et al. (2007a), ABC in Akay and Karaboga (2009b), and MBO in Afshar et al. (2007).

Table 1 Meta-heuristics summary

Algorithm	*Inspiration*	*First applied to…*	*Mechanism of exploitation*	*Mechanism of exploration*	*Communication model*
BA	Bee foraging	Continuous optimisation	Neighbourhood search in good food sources	Random search of scout bees	Broadcast-like
ABC	Bee foraging	Continuous optimisation	Neighbourhood search carried by employed and onlooker bees	Random search of scout bees	Broadcast-like
MBO	Bee mating	Discrete optimisation	Neighbourhood search in queens and broods carried by workers	Spermatheca creation	Direct
BFO	Bacterial foraging	Continuous optimisation	Chemotaxis and reproduction steps	Elimination-dispersal step	Direct
GSO	Firefly bioluminescense	Continuous optimisation	Glow-worm position update	Find neighbour phase dictated by sensor range	Broadcast-like
FA	Firefly bioluminescense	Continuous optimisation	Firefly movement according to attractiveness	Random move of the best firefly	Broadcast-like
SMOA	Amoebae foraging	Continuous optimisation	Vegetative state	Dispersal state	Stigmergic
RIO	Cockroaches infestation	Continuous optimisation	Find friend phase	Find food phase	Broadcast-like
MHSA	Mosquito foraging	Discrete optimisation	Host attraction	Mosquitoes interaction	Broadcast-like
BA	Bat echolocation	Continuous Optimisation	Low loudness and high pulse rate values	High loudness and low pulse rate values	Broadcast-like

An interesting fact about the third column is that most approaches can be applied to continuous optimisation. This gives us an insight that they could be applied together to the same problem, without major modifications, in order to promote co-evolution. The co-evolution can occur when migrations of individuals from one population biases positively the evolution of another population that receives the individuals. Hence, each approach can be viewed as an island evolving with its own strategies upon a migration topology.

The fourth and fifth columns of Table 1 show the exploitation and exploration mechanisms for each approach, respectively. All the mentioned algorithms use exploration and exploitation procedures in their own particular way to seek for the global optimum value of an optimisation problem. According to the no free-lunch theorem (Wolpert and Macready, 1997), it is not possible to point which is the best approach without considering a specific problem. At a higher level, currently it is not possible to point which algorithms are more efficient for generic classes of problems, such as continuous optimisation or discrete optimisation. Therefore, systematic studies comparing the performance of the swarm intelligence algorithms presented here are still missing and this will be a future research direction.

The communication model (sixth column of Table 1) classifies the approaches regarding how their individuals communicate to each other. The communication can be:

1 Broadcast-like: The information propagates throughout the environment to some limited extent and/or is made available for a short time such as the bees waggle dance, and the fireflies glow intensity.

2 Direct: The communication is done through antennation, reproduction, trophallaxis (food or liquid exchange), and mandibular contact.

3 Stigmergic/indirect: Occurs when one individual modifies the environment and other individual responds asynchronously to the changes in the environment at a later time.

The communication models for the well-known ACO and PSO meta-heuritics are indirect through pheromone trails in the environment and broadcast-like through the social component of particles, respectively. In the BA approach, best sites are broadcasted and recruitment takes place. The ABC algorithm uses the waggle dance to disseminate the information. The MBO algorithm uses the spermatheca formation as a direct communication strategy. The BFO algorithm uses the reproduction phase to propagate the information. Both GSO and FA approaches use the glow intensity as broadcast-like communication strategy. The SMOA uses indirect communication through cAMP trails in the environment. The RIO algorithm broadcasts the information using the find friend phase. In the MHSA, broadcast-like communication occurs through the radiated heat of the host. Finally, the BA approach uses the same PSO strategy that is based on a social component.

5 Conclusions and future work

This work presented a review of the most recent developments in the field of swarm intelligence. Going beyond the traditional ACO and PSO meta-heuristics, we focused on emergent works that take inspiration from the behaviour of social organisms, but are not much explored, to date.

In the same way as in ACO and PSO, nature observation has lead to these new algorithms. This highlights the fact that the nature is an unending source of inspiration for computer scientists, and many other approaches will appear in the future.

The computer science community have already learned about the importance of emergent behaviours for complex problem solving. As shown in this work, learning about the collective behaviour of living beings can provide interesting and useful swarm-based meta-heuristics. The studies that have been done to date show the potential of these new approaches to effectively find good solutions to many types of practical optimisation problems.

In fact, there is no 'best' approach, independently of specific context (Wolpert and Macready, 1997). Different implementations will be more adequate for different problems, either leading to better solutions, or improved speed. Moreover, the convenience of a particular approach does not depend only on the problem: different methods will be more useful for different people, depending on their experience and expertise.

A straightforward future research is the comparison of performance of these approaches upon a specific problem in a specific domain, such as numerical optimisation of mathematical functions (continuous optimisation), combinatorial optimisation (discrete optimisation), or multi-objective optimisation (either continuous or discrete optimisation). Such work will focus on unveiling the strengths and weakness of the several algorithms, as well as trying to point out their applicability for different classes of problems, from the user viewpoint.

In evolutionary computation in general, and in swarm intelligence, in particular, the use of mechanisms for self-adaptation of parameters is scarce, but a subject of current research. Self-adaptation of parameters could be useful, e.g., for the BFO algorithm that has seven parameters, and for the GSO algorithm that has nine parameters to be tuned by the user. It is not a trivial task to set all these parameters to achieve the best performance for a specific problem. Hence, the design of strategies to fine-tune or to reduce the number of parameters of the swarm intelligence algorithms presented in this paper is another topic pointed for future research in this area.

Another two trends of research could be either to explore the concept of co-evolution using these new approaches working as islands upon a migratory topology, or to explore the development of hybrid systems where some properties of one approach are combined with those of another one.

Hopefully, plenty of other collective behaviours remain in the shadow waiting to be investigated, such as dragonflies hunting, bees pollination, butterflies mating, ants nest building, locusts collective motion, and others. In the near future, these research opportunities will, possibly, lead to novel algorithms and problem-solving methodologies.

Acknowledgements

The authors would like to thank UDESC (Santa Catarina State University) and FUMDES program for the financial support to R.S. Parpinelli; as well as to the Brazilian National Research Council (CNPq) for the Research Grant No. 309262/2007-0 to H.S. Lopes.

References

Abbass, H. (2001) 'MBO: marriage in honey bees optimization – a haplometrosis polygynous swarming approach', in *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, IEEE Press, pp.207–214.

Afshar, A., Haddad, O., Marino, M. and Adams, B. (2007) 'Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation', *Journal of the Franklin Institute*, pp.452–462.

Akay, B. and Karaboga, D. (2009a) 'Parameter tuning for the artificial bee colony algorithm', in *1st International Conference on Computational Collective Intelligence – Semantic Web, Social Networks & Multiagent Systems*, October.

Akay, B. and Karaboga, D. (2009b) 'Solving integer programming problems by using artificial bee colony algorithm', *XI. Conferences on Advances in Artificial Intelligence by the Italian Association for Artificial Intelligence*, December.

Alavandar, S., Jain, T. and Nigam, M.J. (2010) 'Hybrid bacterial foraging and particle swarm optimisation for fuzzy precompensated control of flexible manipulator', *International Journal of Automation and Control*, Vol. 4, No. 2, pp.234–251.

Altringham, J., McOwat, T. and Hammond, L. (1998) *Bats: Biology and Behaviour*, Oxford Univesity Press.

Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R. and Wu, A. (1998) 'An optimal algorithm for approximate nearest neighbor searching', in *Journal of the ACM*, Vol. 45, pp.891–923.

Bahamish, H., Abdullah, R. and Salam, R. (2008) 'Protein conformational search using bees algorithm', in *Proceedings of the 2008 Second Asia international Conference on Modelling & Simulation*, IEEE Computer Society.

Baig, A. and Rashid, M. (2007) 'Honey bee foraging algorithm for multimodal & dynamic optimization problems', in *GECCO'07: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pp.169–169.

Baykasoğlu, A., Ozbakir, L. and Tapkan, P. (2007) 'Artificial bee colony algorithm and its application to generalized assignment problem', in Chan, F.T.S. and Tiwari, M.K. (Eds.): *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, Itech Education and Publishing, December, pp.532–564.

Bell, W.J., Roth, L. and Nalepa, C. (2007) *Cockroaches: Ecology, Behavior, and Natural History*, The Johns Hopkins University Press.

Benatchba, K., Admane, L. and Koudil, M. (2005) 'Using bees to solve a data mining problem expressed as a maxsat one', in *Proceedings of IWINAC'2005, International Work Conference on the Interplay between Natural and Artificial Computation*, pp.212–220.

Berg, H. (2003) *E. Coli in Motion*, Springer-Verlag, NY.

Bonabeau, E., Dorigo, M. and Theraulaz, G. (1999) *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press.

Chang, H. (2006) 'Converging marriage in honey-bees optimization and application to stochastic dynamic programming', *Journal of Global Optimization*, Vol. 35, pp.423–441.

Chen, H., Zhu, Y. and Hu, K. (2008) 'Self-adaptation in bacterial foraging optimization algorithm', in *Proceedings of 2008 3rd International Conference on Intelligent System and Knowledge Engineering*, pp.1026–1031.

Chidambaram, C. and Lopes, H. (2009) 'A new approach for template matching in digital images using an artificial bee colony algorithm', in *World Congress on Nature and Biologically Inspired Computing (NaBIC 2009)*.

Clerc, M. (2006) *Particle Swarm Optimization*, ISTE Press.

Das, S., Biswas, A., Dasgupta, S. and Abraham, A. (2009) *Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications, Volume 203/2009 of Studies in Computational Intelligence*, Springer Berlin/Heidelberg, pp.23–55.

Dhurandher, S., Singhal, S., Aggarwal, S., Pruthi, P., Misra, S. and Woungang, I. (2009) 'A swarm intelligence-based p2p file sharing protocol using bee algorithm', in *International Conference on Computer Systems and Applications*, May, pp.690–696.

Dorigo, M. and Stützle, T. (2004) *Ant Colony Optimization*, MIT Press.

Drias, H., Sadeg, S. and Yahi, S. (2005) 'Cooperative bees swarm for solving the maximum weighted satisfiability problem', in *IWAAN International Work Conference on Artificial and Natural Neural Networks*, pp.318–325.

Fathian, M., Amiri, B. and Maroosi, A. (2007) 'Application of honey-bee mating optimization algorithm on clustering', *Applied Mathematics and Computation*, Vol. 190, No. 2, pp.1502–1513.

Feng, X., Lau, F.C.M. and Gao, D. (2009) 'A new bio-inspired approach to the traveling salesman problem', in *Complex Sciences*, *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, February, Vol. 5, pp.1310–1321, Springer Berlin Heidelberg.

Fraga, H. (2008) 'Firefly luminescence: a historical perspective and recent developments', *Journal of Photochemical & Photobiological Sciences*, Vol. 7, pp.146–158.

Garnier, S., Gautrais, J. and Theraulaz, G. (2007) 'The biological principles of swarm intelligence', *Swarm Intelligence*, June, Vol. 1, No. 1, pp.3–31.

Guney, K. and Onay, M. (2010) 'Bees algorithm for interference suppression of linear antenna arrays by controlling the phase-only and both the amplitude and phase', *Expert Systems with Applications: An International Journal*, Vol. 37, No. 4, pp.3129–3135.

Haddad, O. and Afshar, A. (2004) 'MBO algorithm, a new heuristic approach in hydrosystems design and operation', in *1st International Conference on Managing Rivers in the 21st Century*, pp.499–504.

Haddad, O., Afshar, A. and Mariño, M. (2006) 'Honeybees mating optimization (HBMO) algorithm: a new heuristic approach for water resources optimization', *Water Resources Management*, Vol. 20, No. 5, pp.661–680.

Haddad, O., Mirmomeni, M. and Mariño, M. (2008) 'Optimal design of stepped spillways using the HBMO algorithm', *Civil Engineering and Environmental Systems*.

Halloy, J., Sempo, G., Caprari, G., Rivault, C., Asadpour, M., Tache, F., Said, I., Durier, V., Canonge, S., Amé, J.M., Detrain, C., Correll, N., Martinoli, A., Mondada, F., Siegwart, R. and Deneubourg, J.L. (2007) 'Social integration of robots into groups of cockroaches to control self-organized choices', *Science*, November, Vol. 318, No. 5853, pp.1155–1158.

Havens, T., Alexander, G., Abbott, C., Keller, J., Skubic, M. and Rantz, M. (2009) 'Contour tracking of human exercises', in *IEEE Workshop on Computational Intelligence for Visual Intelligence*, April, pp.22–28.

Havens, T., Spain, C., Salmon, N. and Keller, J. (2008) 'Roach infestation optimization', *IEEE Swarm Intelligence Symposium*, September, pp.1–7.

Hazra, J. and Sinha, A. (2008) 'Environmental constrained economic dispatch using bacteria foraging optimization', in *Joint International Conference on Power System Technology and IEEE Power India Conference, 2008. POWERCON 2008*, pp.1–6.

Karaboga, D. (2005) 'An idea based on honey bee swarm for numerical optimization', Technical report, Erciyes University, Engineering Faculty, Computer Engineering Department.

Karaboga, D. and Akay, B. (2009a) 'A comparative study of artificial bee colony algorithm', *Applied Mathematics and Computation*, Vol. 214, pp.108–132.

Karaboga, D. and Akay, B. (2009b) 'A survey: algorithms simulating bee swarm intelligence', *Artificial Intelligence Review*, October.

Karaboga, D. and Ozturk, C. (2009) 'Neural networks training by artificial bee colony algorithm on pattern classification', *Neural Network World*, Vol. 19, No. 3, pp.279–292.

Kennedy, J. and Eberhart, R. (2001) *Swarm Intelligence*, Morgan Kaufmann.

Kessin, R. (2001) *Dictyostelium: Evolution, Cell Biology, and the Development of Multicellularity*, Cambridge University Press, Cambridge, UK.

Kim, D. and Cho, C. (2005a) 'Adaptive tuning of PID controller for multivariable system using bacterial foraging based optimization', in *AWIC 2005*, pp.231–235.

Kim, D. and Cho, C. (2005b) 'Bacterial foraging based neural network fuzzy learning', in *Proceedings of the 2005 Indian International Conference on Artificial Intelligence*, pp.2030–2036.

Krishnanand, K. and Ghose, D. (2005) 'Detection of multiple source locations using a glowworm metaphor with applications to collective robotics', in *Proceedings of the IEEE Swarm Intelligence Symposium*, pp.84–91.

Krishnanand, K. and Ghose, D. (2008) 'Glowworm swarm optimization algorithm for hazard sensing in ubiquitous environments using heterogeneous agent swarms', in *Soft Computing Applications in Industry*, Vol. 226, pp.165–187, Springer-Verlag.

Krishnanand, K. and Ghose, D. (2009) 'Glowworm swarm optimization for simulataneous capture of multiple local optima of multimodal functions', *Swarm Intelligence*, Vol. 3, No. 2, pp.87–124.

Linh, N. and Anh, N. (2010) 'Application artificial bee colony algorithm (ABC) for reconfiguring distribution network', *Second International Conference on Computer Modeling and Simulation*, January, Vol. 1, pp.102–106.

Lukasik, S. and Zak, S. (2009) 'Firefly algorithm for continuous constrained optimization tasks', in *1st International Conference on Computational Collective Intelligence*.

Luo, Y. and Chen, Z. (2010) 'Optimization for pid control parameters on hydraulic servo control system based on the novel compound evolutionary algorithm', *Second International Conference on Computer Modeling and Simulation*, January, Vol. 1, pp.40–43.

Majhi, B. and Panda, G. (2008) 'Nonlinear system identification based on bacterial foraging optimization technique', *International Journal of Systemics, Cybernetics and Informatics*, April, pp.44–50.

Majhi, B. and Panda, G. (2009) 'A hybrid functional link neural network and bacterial foraging approach for efficient identification of dynamic systems', *International Journal of Applied Artificial Intelligence in Engineering Systems*, January, Vol. 1, No. 1, pp.91–104.

Majhi, R., Panda, G., Majhi, B. and Sahoo, G. (2009) 'Efficient prediction of stock market indices using adaptive bacterial foraging optimization (ABFO) and BFO based techniques', *Expert Systems with Applications*, August, Vol. 36, No. 6, pp.10097–10104.

Marinakis, Y., Marinaki, M. and Matsatsinis, N. (2009) 'A hybrid discrete artificial bee colony – GRASP algorithm for clustering', *International Conference on Computers and Industrial Engineering*, pp.548–553.

Mehlhorn, H. (2001) 'Mosquitoes', in *Encyclopedic Reference of Parasitology, Biology, Structure, Function*, 2nd ed., pp.378–384, Springer Verlag.

Mishra, S. (2005) 'A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation', *IEEE Trans. Evolutionary Computation*, Vol. 9, No. 1, pp.61–73.

Monismith, D. and Mayfield, B. (2008) 'Slime mold as a model for numerical optimization', in *IEEE Swarm Intelligence Symposium*, pp.1–8.

Nakrani, S. and Tovey, C. (2003) 'On honey bees and dynamic allocation in an internet server colony', in *Proceedings of 2nd International Workshop on the Mathematics and Algorithms of Social Insects*.

Passino, K. (2002) 'Biomimicry of bacterial foraging for distributed optimization and control', *IEEE Control Systems Magazine*, pp.52–67.

Pawar, P., Rao, R. and Shankar, R. (2008) 'Multiobjective optimization of electro-chemical machining process parameters using artificial bee colony (ABC) algorithm', in *Advances in Mechanical Engineering (AME-2008)*, December.

Pham, D. and Ghanbarzadeh, A. (2007) 'Multiobjective optimisation using the bees algorithm', in *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007)*.

Pham, D. and Kalyoncu, M. (2009) 'Optimisation of a fuzzy logic controller for a flexible single-link robot arm using the bees algorithm', *7th IEEE International Conference on Industrial Informatics*, pp.475–480.

Pham, D., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S. and Zaidi, M. (2005) 'The bees algorithm', Technical report, Cardiff University, UK.

Pham, D., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S. and Zaidi, M. (2006a) 'The bees algorithm – a novel tool for complex optimisation problems', in *Proceedings of IPROMS*, pp.454–461.

Pham, D., Koc, E., Ghanbarzadeh, A. and Otri, S. (2006b) 'Optimisation of the weights of multi-layered perceptrons using the bees algorithm', in *Proc 5th International Symposium on Intelligent Manufacturing Systems*.

Pham, D., Koc, E., Lee, J. and Phrueksanant, J. (2007a) 'Using the bees algorithm to schedule jobs for a machine', in *Proc Eighth International Conference on Laser Metrology, CMM and Machine Tool Performance*, pp.430–439.

Pham, D., Otri, S., Afify, A., Mahmuddin, M. and Al-Jabbouli, H. (2007b) 'Data clustering using the bees algorithm', in *Proc. 40th CIRP Int. Manufacturing Systems Seminar*.

Poli, R., Kennedy, J. and Blackwell, T. (2007) 'Particle swarm optimization: an overview', *Swarm Intelligence*, June, Vol. 1, No. 1, pp.33–57.

Reinhard, J. and Srinivasan, S. (2009) 'The role of scents in honey bee foraging and recruitment', *Food Exploitation by Social Insects: Ecological, Behavioral, and Theoretical Approaches*, CRC Press, 1st ed., pp.165–182.

Rothermich, J.A., Wang, F. and Miller, J.F. (2003) 'Adaptivity in cell based optimization for information ecosystems', in *The Congress on Evolutionary Computation*, Vol. 1, pp.490–497.

Sabata, S.L., Udgatab, S.K. and Abraham, A. (2010) 'Artificial bee colony algorithm for small signal model parameter extraction of mesfet', *Engineering Applications of Artificial Intelligence*.

Sato, T. and Hagiwara, M. (1997) 'Bee system: finding solution by a concentrated search', in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 4(C), pp.3954–3959.

Seeley, T. (1995) *The Wisdom of the Hive*, Harvard University Press.

Shimomura, O. (2006) *Bioluminescense: Chemical Principles and Methods*, World Scientific Publishing.

Srinivasa, R., Narasimham, S. and Ramalingaraju, M. (2008) 'Optimization of distribution network configuration for loss reduction using artificial bee colony algorithm', *International Journal of Electrical Power and Energy Systems Engineering (IJEPESE)*, Vol. 1, No. 2.

Taher, N. (2009) 'An efficient hybrid evolutionary algorithm based on PSO and HBMO algorithms for multi-objective distribution feeder reconfiguration', *Energy Conversion and Management*, Vol. 50, No. 8, pp.2074–2082.

Teo, J. and Abbass, H. (2003) 'A true annealing approach to the marriage in honey-bees optimization algorithm', *International Journal of Computational Intelligence and Applications*, Vol. 3, No. 2, pp.199–211.

Teodorovic, D. and Dell'Orco, M. (2005) 'Bee colony optimization – a cooperative learning approach to complex transportation problems', *Advanced OR and AI Methods in Transportation*, pp.51–60.

Tripathy, M., Mishra, S., Lai, L. and Zhang, Q. (2006) 'Transmission loss reduction based on FACTS and bacteria foraging algorithm', in *Proceedings of the 2006 Parallel Problem Solving from Nature*, Vol. 4139, pp.222–231.

Tyler, J. (2002) *The Glow-worm*, Privately published.

Wedde, H., Farooq, M. and Zhang, Y. (2004) 'Beehive: an efficient fault-tolerant routing algorithm inspired by honey bee behavior', in Dorigo, M. (Ed.): *Ant Colony Optimization and Swarm Intelligence*, pp.83–94, Springer Berlin.

Winston, M. (1991) *The Biology of the Honey Bee*, Harvard University Press.

Wolpert, D. and Macready, W. (1997) 'No free lunch theorems for optimization', *IEEE Trans. Evolutionary Computation*, Vol. 1, No. 1, pp.67–82.

Yang, X. (2005) 'Engineering optimizations via nature-inspired virtual bee algorithms', in Yang, J. and Alvarez, J. (Eds.): *IWINAC 2005*, LNCS, pp.317–323, Springer-Verlag.

Yang, X. (2008) *Firefly Algorithm*, Chapter 8. Natureinspired Metaheuristic Algorithms. Luniver Press.

Yang, X. (2010a) *Firefly Algorithm, Lévy Flights and Global Optimization*, Research and Development in Intelligent Systems XXVI, pp.209–218, Springer, London.

Yang, X. (2010b) 'A new metaheuristic bat-inspired algorithm', in *Nature Inspired Cooerative Strategies for Optimization (NISCO 2010)*, Studies in Computational Intelligence, Vol. 284, pp.65–74, Springer Berlin.

Notes

1 ACO Repository: available at http://iridia.ulb.ac.be/~mdorigo/ACO/.

2 PSO Repository: available at http://www.particleswarm.info.

3 BA Repository: available at http://www.bees-algorithm.com.

4 ABC Repository: available at http://mf.erciyes.edu.tr/abc/.

5 A pseudopod is an extension of the cytoplasm of unicellular organisms that imitates a foot and is used to move the cell.