Attraction and Diffusion in Nature-Inspired Optimization Algorithms

Xin-She Yang[a], Suash Deb[b], Thomas Hanne[c], and Xingshi He[d]

a) School of Science and Technology, Middlesex University, Hendon Campus, London NW4 4BT, UK.b) Cambridge Institute of Technology, Cambridge Village, Tatisilwai, Ranchi-835103, Jharkhand, India.,
c) Corresponding author; University of Applied Sciences and Arts Northwestern Switzerland, Riggenbachstr. 16, 4600 Olten, Switzerland
d) School of Science, Xi'an Polytechnic University, No. 19 Jinhua South Road, Xi'an, China

[Accepted by ***Neural Computing and Applications***
http://link.springer.com/article/10.1007%2Fs00521-015-1925-9]

Abstract Nature-inspired algorithms usually use some form of attraction and diffusion as a mechanism for exploitation and exploration. In this paper, we investigate the role of attraction and diffusion in algorithms and their ways in controlling the behaviour and performance of nature-inspired algorithms. We highlight different ways of the implementations of attraction in algorithms such as the firefly algorithm, charged system search, and the gravitational search algorithm. We also analyze diffusion mechanisms such as random walks for exploration in algorithms. It is clear that attraction can be an effective way for enhancing exploitation, while diffusion is a common way for exploration. Furthermore, we also discuss the role of parameter tuning and parameter control in modern metaheuristic algorithms, and then point out some key topics for further research.

Key Words: Metaheuristics, Nature-Inspired Optimization Algorithms, Analysis of Algorithms, Attraction, Diffusion

1. Introduction

Optimization problems are often challenging to solve, and highly nonlinear problems often necessitate new optimization methods such as nature-inspired metaheuristic algorithms [20, 8]. Genetic algorithms (GA) and particle swarm optimization (PSO) [12] have been used in almost every area of science and engineering. During the last 10 years, new nature-inspired algorithms such as firefly algorithm (FA) and cuckoo search (CS) have also become popular and proved to be competitve compared to more established approaches [22, 24]. For example, CS was applied to engineering optimization and structural design problems, and obtained superior results [25, 7]. Other new algorithms such as the accelerated particle swarm optimization (APSO) and bat algorithm are also very promising [28, 29].

Two key components of any nature-inspired algorithm are exploitation and exploration, and different algorithms have different ways of realizing them. Most algorithms use some form of randomization or random walks to enhance exploration, while exploitation can be more deterministic. It is often very challenging to analyze and understand how these interacting components actually work and how they affect the efficiency of an algorithm.

This paper attempts to analyze attraction and diffusion as effective mechanisms for exploitation and exploration in nature-inspired algorithms. The novel idea of attraction was first introduced by Yang [20, 22] when he developed the firefly algorithm in 2007 and published a book chapter on this subject (Chapter 8) in 2008 [20]. The attraction action tends to aggregate all swarm agents into a smaller region, thus leading to quicker convergence. Random walks as a diffusion mechansim and thus a search mechanism have been used for many years, and can be traced back as early as the development of the Monte Carlo method. The main purpose of this paper is to gain further insight by analyzing these key components.

The paper is organized as follows: we first introduce the concepts of attraction and diffusion, and then analyze three algorithms: the firefly algorithm, the charged system search and the gravitational search algorithm. We then discuss parameter tuning and parameter control, and finally we discuss the implications and possible extension for further research.

2. Attraction in Nature-Inspired Algorithms

2.1 Exploitation and Exploration

In all nature-inspired algorithms, there are two key components: exploitation and exploration, or intensification and diversification [3]. Exploration means that the search space is sufficiently investigated on a rough level, while exploitation means that interesting areas (i.e., those aound local optima) are searched more intensively in order to allow for a good approximation to an optimum. The two concepts are characterized in [6] as follows: "Exploration is the process of visiting entirely new regions of a search space, whilst exploitation is the process of visiting those regions of a search space within the neighborhood of previously visited points." For proper exploration, new solutions generated by an algorithm should be sufficiently far from the existing solulutions so that all the regions can be accessible by the explorative moves. On the other hand, proper exploitation means that new solutions should use information gained from exisiting good solutions so that the rate of convergence can be sped up, without wasting too many new moves. Exploitation is often achieved by using intensive local search information such as gradients [21], such information is usually landscape based, which is closely linked with the optimization objective, while exploration is often carried out using some randomization techniques and random walks [21]. Depending on the step size (or the distance moved from existing solutions), such randomization can be either local (when step sizes are small) or global (when step sizes are large).

The most extensive use of exploitation is probably the use of the gradient information of the objective function of an optimization problem of interest. In fact, gradient-based methods such as the well-known Newton-Raphson method are among the most widely used and they are very efficient for local search, though they have the drawbacks that they could not provide the global optimality for multimodal optimization problems. For swarm intelligence based algorithms such as particle swarm optimization, it typically uses a forcing term (x_{best}-x_i) where x_{best} is the best solution found so far. This can be considered as a gradient term if we implicitly assume that the step size is fixed (or with a scaling factor). One of the exotic forms of exploitation is attraction which we will discuss in the next subsection.

On the other hand, exploration is usually carried out by randomization, in terms of either simply random numbers or variables, random walks, or more sophisticated methods such as Lévy flights [22, 24]. From the statistical point of view, diffusion is a random walk, and is thus equivalent to a search mechanism.

For ensuring that an algorithm can reach a globally optimal solution (or a Pareto-optimal solution in the case of a multiobjective optimization problem), there must be a positive probability that such a solution can be obtained from a given starting solution. For a continuous optimization problem, this usually requires certain regularity properties of the problem and a random mechanism with a positive probability density around the global optimum or along a feasible exploration way towards the optimum [11].

Exploration and exploitation can be considered two sides of the same coin. Exploration tends to increase of the probability of finding the global optimality, but affect the rate of convergence, whereas exploitation tends to speed up the convergence but may sacrafice the chance of finding the global optimality. In factm, too much exploration and too little exploitation may almost guarantee the access of global optimality, but can significantly slow the convergence, and thus may render the algorithm almost useless in practice because it may take almost infinitely long to achieve the guaranteed global optimality. At the other extreme, too much exploitation and too little exploration can make the algorithm converge towards a local optimum, largely depending on its initial, starting point, which effectively turn a global optimizer into a local optimizer, with little probability of achieving true global optimality. Therefore, it is obvious that an important issue is to balance exploitation and exploration during the search process so that an algorithm can produce a better, ideally an optimal, performance. However, there is no easy way to achieve this balance, and the fine balance itself is a tough optimization process, or an optimization problem of an optimizaiton algorithm.

2.2 Attraction as an Exploitation Mechanism

The basic idea of attraction is that solutions are attracted by other solutions when they are modified during the run of the algorithm. For a population of solutions, solutions can be ranked based on their fitness values or objective function values. If some moves allow the low-fitness solutions to move towards higher (or even highest) fitness solutions, the overall fitness of the population can improve with such moves. One possible mechanism is to use attraction where the highest solution acts as an attractor, whereas the low fitness solutions are essentially the attractees. How far each attraction move can be will depend on the attraction mechanism. Thus, attraction is a mechanism prevents a purely random walk of solution.

The novel idea of attraction via light intensity as an exploitation mechanism was first used by Yang [20] in the firefly algorithm (FA). In FA, the attractiveness (and light intensity) is intrinsically linked with the inverse-square law of light intensity variations and the absorption coefficient. As a result, the attraction β is calculated by $\beta = \beta_0 \cdot \exp[-\gamma \cdot r^2]$ where r is the distance between two solutions (or fireflies), β_0 is a constant corresponding to the attractiveness at the distance r=0, and $\gamma>0$ is the absorption coefficient [20, 22]. In essence, the brightest firefly (or the best solution) acts as a main attractor that exerts a 'magic' force so that other fireflies can move towards this best locationm, making the population potentially converge towards this current best location.

Other algorithms also used attraction based on the inverse-square laws, derived from nature. For example, the charged system search (CSS) used Coulomb's law [13], while the gravitational search algorithm (GSA) used Newton's law of gravitation [16].

In other nature-inspired algorithms, the mechanism of attraction is possibly less obvious but nevertheless important for convergence. For instance, in evolutionary algorithms genetic information is exchanged between solutions from a given population. This mechanism which is called recombination and cross-over leads, together with a more likely selection of better solutions, to an increase of their genetic "material". This can be interpreted as an attraction towards the superior solutions. For a rigourous convergence analysis, usually further properties of the methods are required such as, for instance, elitism which means that best solutions are preserved during optimization runs [9, 11]. This mechanism also facilitates the attraction towards the best found solutions.

The main function of such an attraction is to enable an algorithm to converge quickly because these multi-agent systems evolve, interact and attract, leading to some self-organized behaviour and attractors. As the

swarming agents evolve, it is possible that their attractor states will move towards the true global optimum under appropriate conditions, though each algorithm may behave differently in terms of the ways of converging towards the global optimum.

2.3 Firefly Algorithm

The Firefly Algorithm (FA) was first developed by Yang in 2007 and was based on the flashing patterns and behaviour of fireflies, and the work on FA was published in 2008 [20, 22]. The movement of firefly i (corresponding to solution x_i) attracted to another, more attractive (brighter, attractor) firefly j (solution x_j) is determined by

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha \varepsilon_i^t, \qquad (1)$$

where β_0 is the attractiveness at the distance r_{ij}=0, and the second term is due to the attraction. The third term is randomization with α being the randomization parameter, and ε_i^t a vector of random numbers drawn from a Gaussian distribution, or other distributions, at time t. If β_0=0, it becomes a simple random walk. Furthermore, the randomization ε_i^t can easily be extended to other distributions such as Lévy flights.

The above updating Eq. (1) has a nonlinear attraction term (the second term). As the attraction decreases as the distance increase, the main attraction force can be visible in a domain of the radius of the order of $L = O\left(\frac{1}{\sqrt{\gamma}}\right)$. Outside this domain, the attraction force becomes very weak. Therefore, the above equation provides a novel and yet efficient mechanism to allow the agents to move and attract each other in the neighbourhood. Since the short-range attraction is stronger than long-range attraction, agents in the whole population can subdivide into multiple subswarms due to the strong local, short-range attractions. Such automatic division of the whole popula-

tion can be advantageous because each smaller subgroup or subswarm can swarm around a local optimum or mode in the multimodal, objective landscape. Obviously, among all the modes, there is the global optimum, and FA will usually find this global optimality very efficiently. Since this subdivision is purely based on the fact that short-distance attraction is stronger than long-distance attraction, there is no need to intervene by the user. The single update equation governs such behaviour, and the division is automatic and intrinsic. Therefore, we can say that , the concept of multi-swarm is intrinsically built into the firefly algorithm. Obviously, if the number of modes is m, then the population size n must be sufficient large ($n \gg m$) so that all the modes can be detected and obtained simultaneously.

However, the proper convergence and stability require good parameter setting, as it is true for almost all metaheuristic algorithms, an insufficient parameter setting or other circumstances may lead to premature convergence (i.e. convergence to a non-optimal solution) [1].

Compared with other algorithms, FA has three distinct advantages: 1) automatic subdivision of the whole population into subgroups; 2) the natural capability of dealing with multimodal optimization; 3) high ergodicity and diversity in the solutions. All these advantages make FA very efficient. This novel attraction mechanism is the first of its kind in the literature of nature-inspired computation and computational intelligence. This also motivated and inspired others to design similar or other types of attraction mechanisms. Whatever the attraction mechanism may be, from the metaheuristic point of view, the fundamental principles are the same: that is, they allow the swarming agents to interact with one another and provide a forcing term to guide the convergence of the population [20, 28].

2.4 Charged System Search

The charged system search (CSS) was based on the fundamental Coulomb's law of charged systems [13]. Though its actual formulae are no longer similar to the actual attraction of charges, they still have some attraction term that was initially based on the physical mechanism. In the CSS, they used a normalized distance

$$r_{ij} = \frac{|| x_i - x_j ||}{|| (x_i + x_j)/2 - x_{best} || + \delta}, \qquad (2)$$

where δ is a small positive number to avoid singularity. However, this singularity may cause potential problems if not handled properly, especially when the current solution is already close to a local optimum. For example, when $x_i = x_j = x_{best}$, the distance would become 0, unless $\delta > 0$. In addition to this normalized distance, they also used the goodness (or badness) factor of attraction $p_{ij} \in \{0, 1\}$. This factor determines whether a solution (charged particle) attracts another charged particle. In general good solutions can act as attractors to attract bad solutions, but the opposite mechanism is also possible. While the first type of attraction mainly serves an exploitation purpose, the second type can allow for a better exploration if it is strong enough.

Due to the rescaled characteristics of the true distance, the attraction is also normalized, which gives a weakened force than its physical counter part. In addition, as there is no exponential term as that in FA, the combination of normalization and linearization make the CSS lose the subdivision capability. Though as one of the reviewers has pointed out, it may be possible to write the CSS as

$$x_i^{t+1} = x_i^t + A\left(x_j^t - x_i^t\right) + B, \qquad (3)$$

where A and B are constant, though A should be dependent on the distance. However, the form of variation of A has a peak value at r=1, which means the attraction is weaker when the normalized distance is less 1. This means that aggregation of short distance swarms are weaker and slower. Thus, the population cannot be split up into subgroups which may evolve more or less

independently. This can be seen in the overall behaviour of the CSS algorithm. At least, up to now, there is no subdivision of CSS observed in the literature. Despite these drawbacks, CSS can be efficient in solving optimization problems [13] and structural design optimization problems [31,32].

2.5 Gravitational Search Algorithm

The gravitational search algorithm (GSA) was based on Newton's law of gravitation [16], though the actual form of the algorithm was also somehow different from reality in the sense that the gravitational constant varies and the distances are rescaled. For example, in GSA, the attraction or force term was written as

$$F = G(t)\frac{M_i \times M_j}{R_{ij}(t)+\delta}(x_j - x_i), \qquad (4)$$

where R_{ij} is the Euclidean distance. Again, δ is a small positive number to avoid singularity. The mass M_i of each particle i is encoded and linked with the objective function through the following equations:

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)},$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)}. \qquad (5)$$

Here, $fit_i(t)$ is the fitness value of agent/particle x_i at time t, $worst(t)$ is the fitness of the worst agent/particle (x_{worst}) in generation t, and $best(t)$ is the fitness of the best agent/particle (x_{best}) in generation t. However, there is a potential singularity problem when the system fully converges; that is, in the special case of best(t)=worst(t) when all particles converge at the same

points for simple unimodal functions. In addition, they also used the velocity updating rules, similar to that of PSO, to calculate new positions and velocities. Though the gravitational constant G(t) can be treated as a constant, however, it can also be varied. In the standard GSA, there is no suggestion what is the best way to vary G(t). In the original GSA [16], the variation of G(t) obeys the following form $G(t) = G(t_0)\left(\frac{t_0}{t}\right)^{\beta}, \ \beta < 1,$ (6)

where $G(t_0)$ is the initial value of the gravitation constant at the time t_0. However, such variation may be physically realistic, but from the algorithmic point of view, it may not be useful. Late variants suggest the following variation

$$G(t) = G_0 \exp[-\alpha t / t_{max}], \quad (7)$$

Here, G_0 is the initial value of G, and α is a positive constant. Here, t_{max} is the maximum number of iterations [30]. . There are a few parameter in the GSA: population size (n), G_0, α and δ. The rescaling of the fitness and masses weaken the ability of subdivision, though the modification using niche GSA (called NSGA) can indeed bring back the multi-swarm capability [30], however, the complication of introducing more parameters such as the size of neighburs K and other niche-related parameters in this algorithm makes the tuning of the algorithm difficult. Even with these complicated issues, GSA can be efficient and can provide results comparable with PSO for function optimization problems.

3. Diffusion in Nature-Inspired Algorithms

Attraction mainly provides the mechanisms for exploitation only, but with proper randomization it is also possible to carry out some degree of exploration. However, the exploration is better analyzed in the framework of random walks and diffusive randomization.

From the Markov chain point of view, random walks and diffusion are both Markov chains because new solutions will primarily depend on the

current set of solutions and the way to generate new solutions (transition probability). In fact, Brownian diffusion such as the dispersion of an ink drop in water is a random walk. For example, the most fundamental random walks for an agent or solution x_i can be written in the following form:

$$x_i^{t+1} = x_i^t + \varepsilon \ , \qquad (8)$$

where t is an iterative counter of steps. Here, ε is a random number drawn from a Gaussian normal distribution with a zero mean. This gives an average diffusion distance of a particle or agent that is proportional to the square root of the finite number of steps t. That is, the distance is the order of $(D\ t)^{1/2}$ where D is the diffusion coefficient. To be more specific, the variance σ^2 of the random walks in a d-dimensional case (variance of the covered distances depending on the number of iterations, t) can be written as

$$\sigma^2(t) = |v_0|^2 t^2 + (2dD)t, \qquad (9)$$

where v_0 is the drifting velocity. This means it is possible to cover the whole search domain if t is sufficiently large. Therefore, the steps in the Brownian motions B(t) essentially follow a Gaussian distribution with a zero mean and time-dependent variance. A diffusion process can be viewed as a series of Brownian motion, which obeys a Gaussian distribution. For this reason, standard diffusion is often referred to as the Gaussian diffusion. If the motion at each step is not Gaussian, then the diffusion is called non-Gaussian diffusion. In this sense, the characteristics of diffusion (and thus the search behaviour) will largely depend on the probability distribution of the step sizes, and different distributions may result in different forms of random walks.

In general, random walks can take many forms. If the step lengths obey other distributions, we have to deal with more generalized random walks. A very special case is when step lengths obey the Lévy distribution, such a random walk is called Lévy flights or Lévy walks [14]. For example, when

steps are large, Lévy distribution can be approximated as a simple power-law

$$L(s) \sim |s|^{-1-\beta}, \qquad (10)$$

where $0 < \beta \leq 2$ is an index or exponent [20, 24]. This randomization technique has been used in cuckoo search (CS) which was found to be very efficient. One of the advantage of Lévy flights is that large steps or long jumps occasionally exist, enabling the algorithm with an ability to escape any local optima, and thus increasing the probability of finding the true global optimal solutions.

Cuckoo search (CS) was developed in 2009 by Yang and Deb [24, 25]. CS is based on the brood parasitism of some cuckoo species. In addition, this algorithm is enhanced by the so-called Lévy flights, rather than by simple isotropic random walks. Recent studies show that CS is potentially far more efficient than PSO and genetic algorithms [25, 7]. One of the key steps in CS is

$$x_i^{t+1} = x_i^t + \alpha L(\beta), \qquad (11)$$

where L is the step size drawn from a Lévy distribution, and α is a scaling factor. The other key steps in CS are the generation of new solutions and the replacement of the not-so-good solutions with a probability p_a. This can be represented mathematically as

$$x_i^{t+1} = x_i^t + s \otimes H(p_a - \varepsilon) \otimes (x_j^t - x_k^t), \qquad (12)$$

where x_i, x_j and x_k are three different solutions. Here *H(u)* is a Heaviside function of u, and ε is a random number drawn from a uniform distribution in [0,1]. In addition, s is the step size vector.

Lévy flights are more efficient than Brownian random walks in exploring the unknown, large-scale search space [23]. There are many reasons to explain this high efficiency, and one simple reason is that the variance of

consecutive Lévy moves is unbounded as it increases with iterations t in the following manner

$$\sigma^2(t) \sim t^{3-\beta}, \quad 1 \le \beta \le 2, \tag{13}$$

which is faster than that for Brownian walks [26, 15].

Now let us give a very crude but yet useful estimate of the number of steps needed for simple random walks for a given domain size W and dimension d. If we wish to achieve an accuracy (distance of an obtained solution from an optimum solution) of $\theta = 10^{-4}$, we can estimate the number of steps or iterations t_{max} needed by pure random walks. This is essentially the upper bound for

$$t_{\max} \approx \frac{W^2}{\theta^2 d}. \tag{14}$$

For example, for W=10 and d=100, we have

$$t_{\max} \approx \frac{1}{(10^{-4})^2 \times 100} \approx 10^8, \tag{15}$$

which is a huge number that is not easily achievable in practice. Even so, this number is still much smaller than a brute force search method. In reality, most metaheuristics require far fewer iterations.

It is straightfoward to show that Lévy flights are more efficient than standard random walks [23]. For Lévy flights, we have an estimate

$$t_{\max} \approx (\frac{W^2}{\theta^2 d})^{1/(3-\beta)}. \tag{16}$$

With β=1.5, we have $t_{\max} \approx 2 \times 10^5$. That is to say, Lévy flights can reduce the number of iterations by about two or three orders. For other cases, the reduction can be even more significant.

4. Parameter Tuning and Parameter Control

In the abvove discussion, our emphasis has been on the role of attraction and diffusion in controlling the characteristics and behaviour of nature-inspired algorithms. This is one side of the algorithmic ‘coin’. Even with the fixed mechanisms of attraction and diffusion, the performance of an algorithm will also be heavily influenced by its parameter settings.

Almost all algorithms have algorithm-dependent parameters. Some algorithms have fewer parameters than others, but the influence of each parameter can be sutble. For different parameter values or settings of an algorithm usually result in different performance of the algorithm. Based on both empirical observations and some preliminary theoretical analysis [20, 27], parameter settings can have some significant influence on the performance of an algorithm. Therefore, how to tune the parameters of an algorithm so as to maximize the performance of the algorithm is a major challenge.

One way of understanding the importance of parameter tuning is that such parameter values may implicitly linked to the ways how exploration and exploitation are carried out. It is almost theoretically impossible to figure out any explicit relationship between parameter values and exploration or exploitation. As we mentioned earlier, one of the most challenging issues for nature-inspired metaheuristic algorithms is probably to control exploration and exploitation properly, which is still an open question. Consequently, it is also a challenging task to control attraction and diffusion in algorithms that use such features so that the performance of an algorithm can be influenced in the right way. Ideally, we should have some mathematical relationship that can explicitly show how parameters can affect the performance of an algorithm, but this is an unsolved problem. In fact, apart from very simple cases under very strict, sometimes, unrealistic assumptions, there is no theoretical result at all.

As an algorithm is a set of interacting Markov chains, we can in general write an algorithm as

$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}^{t+1} = A[\mathrm{x_i}, \varepsilon(\mathrm{t}), \mathrm{p_1(t)}, ..., \mathrm{p_k(t)}] \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}^{t}, \qquad (17)$$

which generates a set of *n* new solutions $(\mathrm{x_1}, \ldots, \mathrm{x_n})^{t+1}$ from the current population of *n* solutions. This behaviour of the algorithm in question is largely determined by the eigenvalues of the matrix A that are in turn controlled by the parameters $\mathrm{p_k(t)}$ and the randomness vector $\varepsilon(\mathrm{t})$. From the Markovian theory, we know that the largest eigenvalue is typically 1, and therefore, the convergence rate of an algorithm is mainly controlled by the second largest eigenvalue λ_2 of A. However, it is in general extremely difficult to find this eigenvalue. Therefore, the tuning of parameters becomes a very challenging task.

In fact, parameter-tuning is an important topic under active research. The aim of parameter-tuning is to find the best parameter setting so that an algorithm can perform best for a wider range of problems. At the moment, paramerter-tuning is mainly carried out by detailed, extensive parametric studies, and there is no efficient method in general. In essence, parameter-tuning itself is an optimization problem which requires the tackling of higher-level optimization methods.

Related to parameter-tuning, there is another issue of parameter control. Parameter values after parameter-tuning are often fixed during iterations, while parameters should vary for parameter control. The idea of parameter control is to vary the parameters so that the algorithm of interest can provide the best convergence rate and thus may achieve the best performance. Again, parameter control is another tough optimization problem yet to be solved. In essence, both parameter tuning and control can be considered a higher-level optimization problem, and it is still not clear which has more influence on the performance of an algorithm. In addition, even with a good

set of parameter and its control sequence, such settings may also depend on the type of problem to be solved. In this case, parameter tuning and control not only depend on the algorithm but also depend on the type of objective landscape. All this makes it even more challenging.

Probably the earliest approach to parameter control in nature-inspired algorithms is the 1/5 rule worked out by Rechenberg already in the early 1970s for the evolution strategy algorithm [17]. Based on a theoretical model, he found that about 1/5 of newly generated solutions should be better than their parent solutions. To achieve this, the mutation rates should be appropriately increased or decreased. Many other concepts of parameter control have been developed for evolutionary algorithms, including approaches for self-adaptation [2]. Self-adaptation means that parameters are not controlled by some fixed rule or according to a given schedule but by means of the algorithm logic itself. As the problem of good parameter selection can be formulated as an own optimization problem, it is possible to treat it just like the original optimization problem, e.g., by means of mutation, recombination, and selection in the case of evolutionary algorithms. In that context, the problem of parameter tuning is often denoted as meta-optimization [4] or meta decision making [10] and approaches for dealing with the the problem of choosing, constructing, or tuning optimization algorithms are also known as hyper-heuristics [5].

An example of adapting parameters of an evolutionary algorithm with a particular focus on a better balance of exploration and exploitation is given by Tan et al. [18, 19]. Another more recent example of parameter control can be found in the bat algorithm. Here, some basic form of parameter control has been attempted and found to be very efficient [29]. By controlling the loudness and pulse emission rate, BA can automatically switch from explorative moves to local exploitation that focuses on the promising regions when the global optimality may be nearby. Similarly, the

cooling schedule in simulated annealing can be considered as a form of basic parameter control.

On the other hand, eagle strategy (ES) is a two-stage iterative strategy with iterative switches [26, 27]. ES starts with a population of agents in the explorative mode, and then switches to the exploitation stage for local intensive search. Then it starts again with another set of explorative moves and subsequently turns into a new exploitation stage. This iterative, restart strategy has been found to be very efficient.

Both parameter-tuning and parameter control are under active research. More efficient methods are urgently needed for this purpose.

5. Conclusions

Nature-inspired metaheuristic algorithms can have many different forms of realization or manifestion in terms of exploration and exploitation, or diversification and intensification. Attraction can be an efficient form of exploitation, while randomization is usually efficient for exploration. In this paper, we have reviewed the ways of attraction in nature-inspired algorithms, including firefly algorithm, charged system search and gravitational search algorithm. We also discussed diffusion as a form of randomization technique for exploration in metaheuristics.

At the same time, parameter-tuning and parameter control are also extremely important for balancing exploration and exploitation and, thus, to control the behaviour and performance of an algorithm. Although, often investigated in the past, both parameter tuning and parameter control are tough, higher-level, optimization problems that require further research in the future.

Despite the above in-depth review and discussions, there are still many topics that will be very useful to explore. For example, it will be useful to carry out the comparison and application of current algorithms in order to

study large-scale problems in real-world applications. After all, real-world problems are nonlinear, and good solutions to such problems could have an important impact by which both academic research and industries would greatly benefit. Obviously, solving large-scale problems can be very challenging but will be extremely useful.

Another key research area should focus on the validation of current methods and the development of new methods or framework for tuning and controlling parameters in metaheuristic algorithms so that they can solve more challenging problems more efficiently.

References

[1] Arora S, Singh S (2013) The firefly optimization algorithm: convergence analysis and parameter selection. International Journal of Computer Applications 69(3): 48-52

[2] Beyer H-G (1995) Toward a theory of evolution strategies: self-adaptation. Evolutionary Computation 3(3): 311-347

[3] Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35: 268-308

[4] Branke J, Elomari JA (2012) Meta-optimization for parameter tuning with a flexible computing budget. In: Soule T (ed) Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (GECCO '12), ACM, New York, pp 1245-1252

[5] Burke E, Gendreau K, Hyde M, Kendall M, Ochoa G, Özcan E, Qu R (2013) Hyper-heuristics: a survey of the state of the art. J Oper Res Soc 64(12): 1695-1724

[6] Črepinšek M, Liu S-H, Mernik M (2013) Exploration and exploitation in evolutionary algorithms: A survey. ACM Comput. Surv. 45(3): 35:1-35:33

[7] Gandomi AH, Yang XS, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Engineering with Computers 29(1): 17-35

[8] Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison Wesley, Reading, Mass.

[9] Hanne T (1999) On the convergence of multiobjective evolutionary algorithms. European Journal of Operational Research 117(3): 553-564

[10] Hanne T (2001) Intelligent strategies for meta multiple criteria decision making. Springer, Berlin

[11] Hanne T (2007) A multiobjective evolutionary algorithm for approximating the efficient set. European Journal of Operational Research 176(3): 1723-1734
[12] Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proc. of IEEE International Conference on Neural Networks, Piscataway, NJ., pp 1942-1948
[13] Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mechanica 213(3-4): 267-289
[14] Mantegna RN (1994) Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes. Phys. Rev. E 49: 4677-4683
[15] Pavlyukevich I (2007) Lévy flights, non-local search and simulated annealing. J. Computational Physics 226: 1830-1844
[16] Rashedi E, Nezamabadi-pour H, Saryazdi S (2009): GSA: a gravitational search algorithm. Information Science 179, 13, 2232-2248
[17] Rechenberg I (1973) Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (Evolution strategy: optimization of technical systems based on concepts from biological evolution). Fromman-Holzboog, Stuttgart, Germany
[18] Tan KC, Goh CK, Yang YJ, Lee TH (2006) Evolving better population distribution and exploration in evolutionary multi-objective optimization. European Journal of Operational Research 171(2): 463-495
[19] Tan KC, Chiam SC, Mamun AA, Goh CK (2009) Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. European Journal of Operational Research 197(2): 701-713
[20] Yang XS (2008) Nature-inspired metaheuristic algorithms. Luniver Press, Bristol, UK
[21] Yang XS (2008b) Introduction to computational mathematics. World Scientific Publishing, Singapore
[22] Yang XS (2009) Firefly algorithms for multimodal optimization. In: Watanabe O, Zeugmann T (eds), Proc. 5th Symposium on Stochastic Algorithms, Foundations and Applications, SAGA 2009, Springer, Heidelberg Berlin, pp 169-178
[23] Yang XS (2011) Metaheuristic optimization: algorithm analysis and open problems. In: Pardalos PM, Rebennack S (eds.), Experimental Algorithms, Springer, Berlin Heidelberg, Germany, pp 21-32
[24] Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009, India), IEEE Publications, USA, pp 210-214
[25] Yang XS, Deb S (2010) Engineering optimization by cuckoo search. Int. J. Mathematical Modeling and Numerical Optimization 1(4): 330-343
[26] Yang, XS, Deb S (2010b) Eagle strategy using Lévy walk and firefly algorithms for stochastic optimization. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, Berlin, pp 101-111

[27] Yang XS, Deb S (2012): Two-stage eagle strategy with differential evolution, Int. J. Bio-Inspired Computation 4(1): 1-5
[28] Yang XS, Deb S, Fong S (2011) Accelerated particle swarm optimization and support vector machine for business optimization and applications, In: Networked Digital Technologies (NDT2011), Communications in Computer and Information Science, vol 136, Springer, Berlin, pp 53-66
[29] Yang XS, Gandomi AH (2012) Bat algorithm: a novel approach for global engineering optimization. Engineering Computations 29(5): 464-483
[30] Yazdani S, Nezamabadi-pour H, Kamyab S, (2014). A gravitational search algorithm for multimodal optimization, Swarm and Evolutionary Computation, 14(1),1-14.
[31] Talatahari S and Jahani Y, (2015). Hybrid charged system search-particle swarm optimization for designof single-layer barrel vault structures, Asian Journal of Civil Engineering, 16(2015)515-533.
[32] Kaveh A and Talatahari S, (2012). Charged system search for optimal design of frame structures, Applied Soft Computing, 12(2012) 382-393.