1

Swarm-Based Metaheuristic Algorithms and No-Free-Lunch Theorems

Xin-She Yang
National Physical Laboratory
United Kingdom

1. Introduction

Metaheuristic algorithms, especially those based on swarm intelligence (SI), form an important part of contemporary global optimization algorithms (Kennedy and Ebarhart, 1995; Yang, 2008; Auger and Teytaud, 2010; Auger and Doerr, 2010; Blum and Roli, 2003; Neumann and Witt 2010; Parpinelli and Lopes, 2011). Good examples are particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) and firefly algorithm (FA) (Yang, 2009). They work remarkably efficiently and have many advantages over traditional, deterministic methods and algorithms, and thus they have been applied in almost all area of science, engineering and industry (Floudas and Pardolos, 2009; Yang 2010a, Yang, 2010b; Yu et al., 2005).

The main characteristics of swarm intelligence is that multiple self-interested agents somehow work together without any central control. These agents as a population can exchange information, by chemical messenger (pheromone by ants), by dance (waggle dance by bees), or by broadcasting ability (such as the global best in PSO and FA). Therefore, all swarm-based algorithms are population-based. However, not all population-based algorithms are swarm-based. For example, genetic algorithms (Holland, 1975; Goldberg, 2002) are population-based, but they are not inspired by swarm intelligence (Bonabeau et al., 1999).

The mobile agents interact locally and under the right conditions they somehow form emergent, self-organized behaviour, leading to global convergence. The agents typically explore the search space locally, aided by randomization which increases the diversity of the solutions on a global scale, and thus there is a fine balance between local intensive exploitation and global exploration (Blue and Roli, 2003). Any swarm-based algorithms have to balance these two components; otherwise, efficiency may be limited. In addition, these swarming agents can work in parallel, and thus such algorithms are particularly suitable for parallel implementation, which leads to even better reduction in computing time.

Despite such a huge success in applications, mathematical analysis of algorithms remains limited and many open problems are still un-resolved. There are three challenging areas for algorithm analysis: complexity, convergence and no-free-lunch theory. Complexity analysis of traditional algorithms such as quick sort and matrix inverse are well-established, as these algorithms are deterministic. In contrast, complexity analysis of metaheuristics remains a challenging task, partly due to the stochastic nature of these algorithms. However, good results do exist, concerning randomization search techniques (Auger and Teytaud, 2010).

Convergence analysis is another challenging area. One of the main difficulties concerning the convergence analysis of metaheuristic algorithms is that no generic framework exists, though

substantial studies have been carried out using dynamic systems and Markov processes. However, convergence analysis still remains one of the active research areas with many encouraging results (Clerc and Kennedy, 2002; Trelea, 2003; Ólafsson, 2006; Gutjahr, 2002).

In optimization, there is a so-called 'no-free-lunch (NFL) theorem' proposed by Wolpert and Mcready (1997), which states that any algorithm will *on average* perform equally well as a random search algorithm over *all* possible functions. In other words, two algorithms A and B will on average have equal performance;' that is, if algorithm A performs better than B for some problems, then algorithm B will outperform A for other problems. This means that there is no universally superior algorithm for all types of problems. However, this does not mean that some algorithms are not better than other algorithms for some *specific* types of problems. In fact, we do not need to measure performance on average for all functions. More often, we need to measure how an algorithm performs for a given class of problems. Furthermore, the assumptions of the NLF theorem are not valid for all cases. In fact, there are quite a few no-free-lunch (NFL) theorems (Wolpert and Mcready, 1997; Igel and Toussaint, 2003). While in well-posed cases of optimization where its functional space forms finite domains, NFL theorems do hold; however, free lunches are possible in continuous domains(Auger and Teytaud, 2010; Wolpert and Mcready 2005; Villalobos-Arias et al., 2005).

In this chapter, we intend to provide a state-of-the-art review of the recent studies of no-free-lunch theory and also free lunch scenarios. This enables us to view the NLF and free lunch in a unified framework, or at least, in a convenient way. We will also briefly highlights some of the convergence studies. Based on these studies, we will summarize and propose a series of recommendations for further research.

2. Swarm-based algorithms

There are more than a dozen of swarm-based algorithms using the so-called swarm intelligence. For a detailed introduction, please refer to Yang (2010b), and for a recent comprehensive review, please refer to Parpinelli and Lopes (2011). In this section, we will focus on the main chararcteristics and the ways that each algorithm generate new solutions, and we will not discuss each algorithm in details. Interested readers can follow the references listed at the end of this chapter and also refer to other chapters of this book.

2.1 Ant algorithms

Ant algorithms, especially the ant colony optimization (Dorigo and Stütle, 2004), mimic the foraging behaviour of social ants. Primarily, it uses pheromone as a chemical messenger and the pheromone concentration as the indicator of quality solutions to a problem of interest. As the solution is often linked with the pheromone concentration, the search algorithms often produce routes and paths marked by the higher pheromone concentrations, and therefore, ants-based algorithms are particular suitable for discrete optimization problems.

The movement of an ant is controlled by pheromone which will evaporate over time. Without such time-dependent evaporation, the algorithms will lead to premature convergence to the (often wrong) solutions. With proper pheromone evaporation, they usually behave very well.

There are two important issues here: the probability of choosing a route, and the evaporation rate of pheromone. There are a few ways of solving these problems, although it is still an area of active research. Here we introduce the current best method.

For a network routing problem, the probability of ants at a particular node i to choose the route from node i to node j is given by

$$p_{ij} = \frac{\phi_{ij}^{\alpha} d_{ij}^{\beta}}{\sum_{i,j=1}^{n} \phi_{ij}^{\alpha} d_{ij}^{\beta}}, \tag{1}$$

where $\alpha > 0$ and $\beta > 0$ are the influence parameters, and their typical values are $\alpha \approx \beta \approx 2$. ϕ_{ij} is the pheromone concentration on the route between i and j, and d_{ij} the desirability of the same route. Some *a priori* knowledge about the route such as the distance s_{ij} is often used so that $d_{ij} \propto 1/s_{ij}$, which implies that shorter routes will be selected due to their shorter traveling time, and thus the pheromone concentrations on these routes are higher. This is because the traveling time is shorter, and thus the less amount of the pheromone has been evaporated during this period.

This probability formula reflects the fact that ants would normally follow the paths with higher pheromone concentrations. In the simpler case when $\alpha = \beta = 1$, the probability of choosing a path by ants is proportional to the pheromone concentration on the path. The denominator normalizes the probability so that it is in the range between 0 and 1.

The pheromone concentration can change with time due to the evaporation of pheromone. Furthermore, the advantage of pheromone evaporation is that the system could avoid being trapped in local optima. If there is no evaporation, then the path randomly chosen by the first ants will become the preferred path as the attraction of other ants by their pheromone. For a constant rate γ of pheromone decay or evaporation, the pheromone concentration usually varies with time exponentially

$$\phi(t) = \phi_0 e^{-\gamma t}, \tag{2}$$

where ϕ_0 is the initial concentration of pheromone and t is time. If $\gamma t \ll 1$, then we have $\phi(t) \approx (1 - \gamma t)\phi_0$. For the unitary time increment $\Delta t = 1$, the evaporation can be approximated by $\phi^{t+1} \leftarrow (1 - \gamma)\phi^t$. Therefore, we have the simplified pheromone update formula:

$$\phi_{ij}^{t+1} = (1 - \gamma)\phi_{ij}^{t} + \delta\phi_{ij}^{t}, \tag{3}$$

where $\gamma \in [0, 1]$ is the rate of pheromone evaporation. The increment $\delta\phi_{ij}^{t}$ is the amount of pheromone deposited at time t along route i to j when an ant travels a distance L. Usually $\delta\phi_{ij}^{t} \propto 1/L$. If there are no ants on a route, then the pheromone deposit is zero.

There are other variations to this basic procedure. A possible acceleration scheme is to use some bounds of the pheromone concentration and only the ants with the current global best solution(s) are allowed to deposit pheromone. In addition, certain ranking of solution fitness can also be used.

2.2 Bee algorithms

Bees-inspired algorithms are more diverse, and some use pheromone and most do not. Almost all bee algorithms are inspired by the foraging behaviour of honey bees in nature. Interesting characteristics such as waggle dance, polarization and nectar maximization are often used to simulate the allocation of the foraging bee along flower patches and thus different search

regions in the search space. For a more comprehensive review, please refer to Parpinelli and Lopes (2011).

Honeybees live in a colony and they forage and store honey in their constructed colony. Honeybees can communicate by pheromone and 'waggle dance'. For example, an alarming bee may release a chemical message (pheromone) to stimulate attack response in other bees. Furthermore, when bees find a good food source and bring some nectar back to the hive, they will communicate the location of the food source by performing the so-called waggle dances as a signal system. Such signaling dances vary from species to species, however, they will try to recruit more bees by using directional dancing with varying strength so as to communicate the direction and distance of the found food resource. For multiple food sources such as flower patches, studies show that a bee colony seems to be able to allocate forager bees among different flower patches so as to maximize their total nectar intake.

In the honeybee-based algorithm, forager bees are allocated to different food sources (or flower patches) so as to maximize the total nectar intake. The colony has to 'optimize' the overall efficiency of nectar collection, the allocation of the bees is thus depending on many factors such as the nectar richness and the proximity to the hive (Nakrani and Trovey, 2004; Yang, 2005; Karaboga, 2005; Pham et al., 2006)

Let $w_i(j)$ be the strength of the waggle dance of bee i at time step $t = j$, the probability of an observer bee following the dancing bee to forage can be determined in many ways depending on the actual variant of algorithms. A simple way is given by

$$p_i = \frac{w_i^j}{\sum_{i=1}^{n_f} w_i^j}, \tag{4}$$

where n_f is the number of bees in foraging process. t is the pseudo time or foraging expedition. The number of observer bees is $N - n_f$ when N is the total number of bees. Alternatively, we can define an exploration probability of a Gaussian type $p_e = 1 - p_i = \exp[-w_i^2/2\sigma^2]$, where σ is the volatility of the bee colony, and it controls the exploration and diversity of the foraging sites. If there is no dancing (no food found), then $w_i \to 0$, and $p_e = 1$. So all the bee explore randomly.

The virtual bee algorithm (VBA), developed by Xin-She Yang in 2005, is an optimization algorithm specially formulated for solving both discrete and continuous problems (Yang, 2005). On the other hand, the artificial bee colony (ABC) optimization algorithm was first developed by D. Karaboga in 2005. In the ABC algorithm, the bees in a colony are divided into three groups: employed bees (forager bees), onlooker bees (observer bees) and scouts. For each food source, there is only one employed bee. That is to say, the number of employed bees is equal to the number of food sources. The employed bee of an discarded food site is forced to become a scout for searching new food sources randomly. Employed bees share information with the onlooker bees in a hive so that onlooker bees can choose a food source to forage. Unlike the honey bee algorithm which has two groups of the bees (forager bees and observer bees), bees in ABC are more specialized (Karaboga, 2005; Afshar et al., 2007).

Similar to the ants-based algorithms, bee algorithms are also very flexible in dealing with discrete optimization problems. Combinatorial optimizations such as routing and optimal paths have been successfully solved by ant and bee algorithms. Though bee algorithms can be applied to continuous problems as well as discrete problems, however, they should not be the first choice for continuous problems.

2.3 Particle swarm optimization

Particle swarm optimization (PSO) was developed by Kennedy and Eberhart in 1995, based on the swarm behaviour such as fish and bird schooling in nature. Since then, PSO has generated much wider interests, and forms an exciting, ever-expanding research subject, called swarm intelligence. PSO has been applied to almost every area in optimization, computational intelligence, and design/scheduling applications.

The movement of a swarming particle consists of two major components: a social component and a cognitive component. Each particle is attracted toward the position of the current global best $\boldsymbol{g}^*$ and its own best location $\boldsymbol{x}_i^*$ in history, while at the same time it has a tendency to move randomly.

Let $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$ be the position vector and velocity for particle i, respectively. The new velocity and location updating formulas are determined by

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + \alpha\boldsymbol{\epsilon}_1[\boldsymbol{g}^* - \boldsymbol{x}_i^t] + \beta\boldsymbol{\epsilon}_2[\boldsymbol{x}_i^* - \boldsymbol{x}_i^t]. \tag{5}$$

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}, \tag{6}$$

where $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ are two random vectors, and each entry taking the values between 0 and 1. The parameters α and β are the learning parameters or acceleration constants, which can typically be taken as, say, $\alpha \approx \beta \approx 2$.

There are at least two dozen PSO variants which extend the standard PSO algorithm, and the most noticeable improvement is probably to use inertia function $\theta(t)$ so that $\boldsymbol{v}_i^t$ is replaced by $\theta(t)\boldsymbol{v}_i^t$ where $\theta \in [0, 1]$. This is equivalent to introducing a virtual mass to stabilize the motion of the particles, and thus the algorithm is expected to converge more quickly.

2.4 Firefly algorithm

Firefly Algorithm (FA) was developed by Xin-She Yang at Cambridge University (Yang,2008; Yang 2009), which was based on the flashing patterns and behaviour of fireflies. In essence, each firefly will be attracted to brighter ones, while at the same time, it explores and searches for prey randomly. In addition, the brightness of a firefly is determined by the landscape of the objective function.

The movement of a firefly i attracted to another more attractive (brighter) firefly j is determined by

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \beta_0 e^{-\gamma r_{ij}^2}(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) + \alpha_t\, \boldsymbol{\epsilon}_i^t, \tag{7}$$

where the second term is due to the attraction. The third term is randomization with α_t being the randomization parameter, and $\boldsymbol{\epsilon}_i^t$ is a vector of random numbers drawn from a Gaussian distribution or uniform distribution. Here $\beta_0 \in [0, 1]$ is the attractiveness at $r = 0$, and $r_{ij} = ||\boldsymbol{x}_i^t - \boldsymbol{x}_j^t||$ is the Cartesian distance. For other problems such as scheduling, any measure that can effectively characterize the quantities of interest in the optimization problem can be used as the 'distance' r. For most implementations, we can take $\beta_0 = 1$, $\alpha = O(1)$ and $\gamma = O(1)$.

Ideally, the randomization parameter α_t should be monotonically reduced gradually during iterations. A simple scheme is to use

$$\alpha_t = \alpha_0\delta^t, \quad \delta \in (0, 1), \tag{8}$$

where α_0 is the initial randomness, while δ is a randomness reduction factor similar to that used in a cooling schedule in simulated annealing. It is worth pointing out that (7) is essentially a random walk biased towards the brighter fireflies. If $\beta_0 = 0$, it becomes a simple random walk. Furthermore, the randomization term can easily be extended to other distributions such as Lévy flights.

2.5 Bat algorithm

Bat algorithm is a relatively new metaheuristic, developed by Xin-She Yang in 2010 (Yang, 2010c). It was inspired by the echolocation behaviour of microbats. Microbats use a type of sonar, called, echolocation, to detect prey, avoid obstacles, and locate their roosting crevices in the dark. These bats emit a very loud sound pulse and listen for the echo that bounces back from the surrounding objects. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. Most bats use short, frequency-modulated signals to sweep through about an octave, while others more often use constant-frequency signals for echolocation. Their signal bandwidth varies depends on the species, and often increased by using more harmonics.

Inside the bat algorithm, it uses three idealized rules:

1. All bats use echolocation to sense distance, and they also 'know' the difference between food/prey and background barriers in some magical way;
2. Bats fly randomly with velocity $\boldsymbol{v}_i$ at position $\boldsymbol{x}_i$ with a fixed frequency $f_{\min}$, varying wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their target;
3. Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum constant value $A_{\min}$.

BA has been extended to multiobjective bat algorithm (MOBA) by Yang (2011), and preliminary results suggested that it is very efficient.

2.6 Cuckoo search

Cuckoo search (CS) is one of the latest nature-inspired metaheuristic algorithms, developed in 2009 by Xin-She Yang and Suash Deb (Yang and Deb, 2009; Yang and Deb, 2010). 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. This algorithm was inspired by the aggressive reproduction strategy of some cuckoo species such as the *ani* and *Guira* cuckoos. These cuckoos lay their eggs in communal nests, though they may remove others' eggs to increase the hatching probability of their own eggs. Quite a number of species engage the obligate brood parasitism by laying their eggs in the nests of other host birds (often other species).

In the standard cuckoo search, the following three idealized rules are used:

- Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest;
- The best nests with high-quality eggs will be carried over to the next generations;
- The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in [0, 1]$. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest.

As a further approximation, this last assumption can be approximated by a fraction p_a of the n host nests are replaced by new nests (with new random solutions). Recent studies suggest that cuckoo search can outperform particle swarm optimization and other algorithms (Yang and Deb, 2010). These are still topics of active research.

There are other metaheuristic algorithms which have not been introduced here, and interested readers can refer to more advanced literature (Yang, 2010b; Parpinelli and Lopes, 2011).

3. Intensification and diversification

Metaheuristics can be considered as an efficient way to produce acceptable solutions by trial and error to a complex problem in a reasonably practical time. The complexity of the problem of interest makes it impossible to search every possible solution or combination, the aim is to find good feasible solution in an acceptable timescale. There is no guarantee that the best solutions can be found, and we even do not know whether an algorithm will work and why if it does work. The idea is to have an efficient but practical algorithm that will work most the time and is able to produce good quality solutions. Among the found quality solutions, it is expected some of them are nearly optimal, though there is often no guarantee for such optimality.

The main components of any metaheuristic algorithms are: intensification and diversification, or exploitation and exploration (Blum and Roli, 2003; Yang, 2008; Yang, 2010b). Diversification means to generate diverse solutions so as to explore the search space on the global scale, while intensification means to focus on the search in a local region by exploiting the information that a current good solution is found in this region. This is in combination with the selection of the best solutions. Randomization techniques can be a very simple method using uniform distributions and/or Gaussian distributions, or more complex methods as those used in Monte Carlo simulations. They can also be more elaborate, from Brownian random walks to Lévy flights.

In general, intensification speeds up the convergence of an algorithm, however, it may lead to a local optimum, not necessarily the global optimality. On the other hand, diversification often slows down the convergence but increases the probability of finding the global optimum. Therefore, there is a fine balance beteween these seemingly competing components for any algorithm.

In ant and bee algorithms, intensification is usually achieved by pheromone and exchange of information so that all agents swarm together or follow similar routes. Diversification is achieved by randomization and probabilistic choices of routes. In PSO, intensification is controlled mainly by the use of the global best and individual best solutions, while diversification is plainly done using two random numbers or learning parameters.

For the standard FA, the global best is not used, though its use may increase the convergence rates for some problems such as unimodal problems or problems with some dominant modes. Intensification is subtly done by the attraction among fireflies and thus brightness is the information exchanged among adjacent fireflies. Diversification is carried out by the randomization term, either by random walks or by Lévy flights, in combination with a randomness-reduction technique similar to a cooling schedule in simulated annealing.

Intensification and diversification in the bat algorithm is controlled by a switch parameter. Intensification as well as diversification is also enhanced by the variations of loudness and

pulse rates. In this sense, the mechanism is relatively simple, but very efficient in balancing the two key components.

In the cuckoo search, things become more subtle. Diversification is carried out in two ways: randomization via Lévy flights and feeding new solutions into randomly chosen nests. Intensification is achieved by a combination of elitism and the generation of solutions according to similarity (thus the usage of local information). In addition, a switch parameter (a fraction of abandoned nests) is used to control the balance of diversification and intensification.

As seen earlier, an important component in swarm intelligence and modern metaheuristics is randomization, which enables an algorithm to have the ability to jump out of any local optimum so as to search globally. Randomization can also be used for local search around the current best if steps are limited to a local region. When the steps are large, randomization can explore the search space on a global scale. Fine-tuning the randomness and balance of local search and global search is crucially important in controlling the performance of any metaheuristic algorithm.

4. No-free-lunch theorems

The seminal paper by Wolpert and Mcready in 1997 essentially proposed a framework for performance comparison of optimization algorithms, using a combination of Bayesian statistics and Markov random field theories. Let us sketch Wolpert and Macready's original idea. Assuming that the search space is finite (though quite large), thus the space of possible objective values is also finite. This means that objective function is simply a mapping $f : \mathcal{X} \mapsto \mathcal{Y}$, with $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$ as the space of all possible problems under permutation.

As an algorithm tends to produce a series of points or solutions in the search space, it is further assumed that these points are distinct. That is, for k iterations, k distinct visited points forms a time-ordered set

$$\Omega_k = \left\{ \left(\Omega_k^x(1), \Omega_k^y(1) \right), ..., \left(\Omega_k^x(k), \Omega_k^y(k) \right) \right\}. \tag{9}$$

There are many ways to define a performance measure, though a good measure still remains debatable (Shilane et al., 2008). Such a measure can depend on the number of iteration k, the algorithm a and the actual cost function f, which can be denoted by $P(\Omega_k^y \| f, k, a)$. Here we follow the notation style in seminal paper by Wolpert and Mcready (1997). For any pair of algorithms a and b, the NFL theorem states

$$\sum_f P(\Omega_k^y | f, k, a) = \sum_f P(\Omega_k^y | f, k, b). \tag{10}$$

In other words, any algorithm is as good (bad) as a random search, when the performance is averaged *o*ver all possible functions.

Along many relevant assumptions in proving the NFL theorems, two fundamental assumptions are: finite states of the search space (and thus the objective values), and the non-revisiting time-ordered sets.

The first assumption is a good approximation to many problems, especially in finite-digit approximations. However, there is mathematical difference in countable finite, and countable infinite. Therefore, the results for finite states/domains may not directly applicable to

infinite domains. Furthermore, as continuous problem are uncountable, NFL results for finite domains will usually not hold for continuous domains (Auger and Teytaud, 2010).

The second assumption on non-revisiting iterative sequence is an over-simplification, as almost all metaheuristic algorithms are revisiting in practice, some points visited before will possibly be re-visited again in the future. The only possible exception is the Tabu algorithm with a very long Tabu list (Glover and Laguna, 1997). Therefore, results for non-revisiting time-ordered iterations may not be true for the cases of revisiting cases, because the revisiting iterations break an important assumption of 'closed under permutation' (c.u.p) required for proving the NFL theorems (Marshall and Hinton, 2010).

Furthermore, optimization problems do not necessarily concern the whole set of all possible functions/problems, and it is often sufficient to consider a subset of problems. It is worth pointing out active studies have carried out in constructing algorithms that can work best on specific subsets of optimization problems, in fact, NFL theorems do not hold in this case (Christensen and Oppacher, 2001).

These theorems are vigorous and thus have important theoretical values. However, their practical implications are a different issue. In fact, it may not be so important in practice anyway, we will discuss this in a later section.

5. Free lunch or no free lunch

The validity of NFL theorems largely depends on the validity of their fundamental assumptions. However, whether these assumptions are valid in practice is another question. Often, these assumptions are too stringent, and thus free lunches are possible.

5.1 Continuous free lunches

One of the assumptions is the non-revisiting nature of the k distinct points which form a time-ordered set. For revisiting points as they do occur in practice in real-world optimization algorithms, the 'closed under permutation' does not hold, which renders NFL theorems invalid (Schumacher et al., 2001; Marshall and Hinton, 2010). This means free lunches do exist in practical applications.

Another basic assumption is the finiteness of the domains. For continuous domains, Auger and Teytaud in 2010 have proven that the NFL theorem does not hold, and therefore they concluded that 'continuous free lunches exist'. Indeed, some algorithms are better than others. For example, for a 2D sphere function, they demonstrated that an efficient algorithm only needs 4 iterations/steps to reach the global minimum.

5.2 Coevolutionary and multiobjective free lunches

The basic NFL theorems concern a single agent, marching iteratively in the search space in distinct steps. However, Wolpert and Mcready proved in 2005 that NFL theorems do not hold under coevolution. For example, a set of players (or agents) in self-play problems can work together so as to produce a champion. This can be visualized as an evolutionary process of training a chess champion. In this case, free lunch does exist (Wolpert and Mcready, 2005). It is worth pointing out that for a single player, it tries to pursue the best next move, while for two players, the fitness function depend on the moves of both players. Therefore, the basic assumptions for NFL theorems are no longer valid.

For multiobjective optimization problems, things have become even more complicated. An important step in theoretical analysis is that some multiobjective optimizers are better than others as pointed out by Corne and Knowles (2003). One of the major reasons is that the archiver and generator in the multiobjective algorithms can lead to multiobjective free lunches.

Whether NFL holds or not, it has nothing to say about the complexity of the problems. In fact, no free lunch theorem has not been proved to be true for problems with NP-hard complexity (Whitley and Watson, 2005).

6. NFL theorems and meaning for algorithm developers

No-free-lunch theorems may be of theoretical importance, and they can also have important implications for algorithm development in practice, though not everyone agrees the real importance of these implications. In general, there are three kinds of opinions concerning the implications. The first group may simply ignore these theorems, as they argue that the assumptions are too stringent, and the performance measures based on average overall functions are irrelevant in practice (Whitley and Watson, 2005). Therefore, no practical importance can be inferred, and research just carries on.

The second kind is that NFL theorems can be true, and they can accept that the fact there is no universally efficient algorithm. But in practice some algorithms do performance better than others for a specific problem or a subset of problems. Research effort should focus on finding the right algorithms for the right type of problem. Problem-specific knowledge is always helpful to find the right algorithm(s).

The third kind of opinion is that NFL theorems are not true for other types of problems such as continuous problems and NP-hard problems. Theoretical work concerns more elaborate studies on extending NFL theorems to other cases or on finding free lunches (Auger and Teytaud, 2010). On the other hand, algorithm development continues to design better algorithms which can work for a wider range of problems, not necessarily all types of problems. As we have seen from the above analysis, free lunches do exist, and better algorithms can be designed for a specific subset of problems (Yang,2009; Yang and Deb, 2010).

Thus, free lunch or no free lunch is not just a simple question, it has important and yet practical importance. There is certain truth in all the above arguments, and their impacts on optimization community are somehow mixed. Obviously, in reality, the algorithms with problem-specific knowledge typically work better than random search, and we also realized that there is no universally generic tool that works best for all the problems. Therefore, we have to seek balance between speciality and generality, between algorithm simplicity and problem complexity, and between problem-specific knowledge and capability of handling black-box optimization problems.

7. Convergence analysis of metaheuristics

For convergence analysis, there is no mathematical framework in general to provide insights into the working mechanism, the complexity, stability and convergence of any given algorithm (He and Yu, 2001; Thikomirov, 2007). Despite the increasing popularity of metaheuristics, mathematical analysis remains fragmental, and many open problems concerning convergence analysis need urgent attention. In addition, many algorithms, though

efficient, have not been proved their convergence, for example, harmony search usually converges well (Geem, 2009), but its convergence still needs mathematical analysis.

7.1 PSO

The first convergence analysis of PSO was carried out by Clerc and Kennedy in 2002 using the theory of dynamical systems. Mathematically, if we ignore the random factors, we can view the system formed by (5) and (6) as a dynamical system. If we focus on a single particle i and imagine that there is only one particle in this system, then the global best $\boldsymbol{g}^*$ is the same as its current best $\boldsymbol{x}_i^*$. In this case, we have

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + \gamma(\boldsymbol{g}^* - \boldsymbol{x}_i^t), \quad \gamma = \alpha + \beta, \tag{11}$$

and

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}. \tag{12}$$

Considering the 1D dynamical system for particle swarm optimization, we can replace $\boldsymbol{g}^*$ by a parameter constant p so that we can see if or not the particle of interest will converge towards p. By setting $u_t = p - x(t+1)$ and using the notations for dynamical systems, we have a simple dynamical system

$$v_{t+1} = v_t + \gamma u_t, \quad u_{t+1} = -v_t + (1-\gamma)u_t, \tag{13}$$

or

$$Y_{t+1} = AY_t, \quad A = \begin{pmatrix} 1 & \gamma \\ -1 & 1-\gamma \end{pmatrix}, \quad Y_t = \begin{pmatrix} v_t \\ u_t \end{pmatrix}. \tag{14}$$

The general solution of this dynamical system can be written as $Y_t = Y_0 \exp[At]$. The system behaviour can be characterized by the eigenvalues λ of A

$$\lambda_{1,2} = 1 - \frac{\gamma}{2} \pm \frac{\sqrt{\gamma^2 - 4\gamma}}{2}. \tag{15}$$

It can be seen clearly that $\gamma = 4$ leads to a bifurcation. Following a straightforward analysis of this dynamical system, we can have three cases. For $0 < \gamma < 4$, cyclic and/or quasi-cyclic trajectories exist. In this case, when randomness is gradually reduced, some convergence can be observed. For $\gamma > 4$, non-cyclic behaviour can be expected and the distance from Y_t to the center $(0, 0)$ is monotonically increasing with t. In a special case $\gamma = 4$, some convergence behaviour can be observed. For detailed analysis, please refer to Clerc and Kennedy (2003). Since p is linked with the global best, as the iterations continue, it can be expected that all particles will aggregate towards the the global best.

7.2 Firefly algorithm

We now can carry out the convergence analysis for the firefly algorithm in a framework similar to Clerc and Kennedy's dynamical analysis. For simplicity, we start from the equation for firefly motion without the randomness term

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \beta_0 e^{-\gamma r_{ij}^2}(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t). \tag{16}$$

If we focus on a single agent, we can replace $\boldsymbol{x}_j^t$ by the global best g found so far, and we have

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \beta_0 e^{-\gamma r_i^2}(g - \boldsymbol{x}_i^t), \tag{17}$$

where the distance r_i can be given by the ℓ_2-norm $r_i^2 = ||g - \boldsymbol{x}_i^t||_2^2$. In an even simpler 1-D case, we can set $y_t = g - \boldsymbol{x}_i^t$, and we have

$$y_{t+1} = y_t - \beta_0 e^{-\gamma y_t^2} y_t. \tag{18}$$

We can see that γ is a scaling parameter which only affects the scales/size of the firefly movement. In fact, we can let $u_t = \sqrt{\gamma} y_t$ and we have

$$u_{t+1} = u_t[1 - \beta_0 e^{-u_t^2}]. \tag{19}$$

These equations can be analyzed easily using the same methodology for studying the well-known logistic map

$$u_t = \lambda u_t(1 - u_t). \tag{20}$$

Mathematical analysis and numerical simulations of (19) can reveal its regions of chaos. Briefly, the convergence can be achieved for $\beta_0 < 2$. There is a transition from periodic to chaos at $\beta_0 \approx 4$. This may be surprising, as the aim of designing a metaheuristic algorithm is to try to find the optimal solution efficiently and accurately. However, chaotic behaviour is not necessarily a nuisance, in fact, we can use it to the advantage of the firefly algorithm. Simple chaotic characteristics from (20) can often be used as an efficient mixing technique for generating diverse solutions. Statistically, the logistic mapping (20) for $\lambda = 4$ for the initial states in (0,1) corresponds a beta distribution

$$B(u, p, q) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} u^{p-1}(1-u)^{q-1}, \tag{21}$$

when $p = q = 1/2$. Here $\Gamma(z)$ is the Gamma function

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt. \tag{22}$$

In the case when $z = n$ is an integer, we have $\Gamma(n) = (n-1)!$. In addition, $\Gamma(1/2) = \sqrt{\pi}$. From the algorithm implementation point of view, we can use higher attractiveness β_0 during the early stage of iterations so that the fireflies can explore, even chaotically, the search space more effectively. As the search continues and convergence approaches, we can reduce the attractiveness β_0 gradually, which may increase the overall efficiency of the algorithm. Obviously, more studies are highly needed to confirm this.

7.3 Markov chains

Most theoretical studies use Markov chains/process as a framework for convergence analysis. A Markov chain is said be to regular if some positive power k of the transition matrix $\boldsymbol{P}$ has only positive elements. A chain is ergodic or irreducible if it is aperiodic and positive recurrent, which means that it is possible to reach every state from any state.

For a time-homogeneous chain as $k \to \infty$, we have the stationary probability distribution π, satisfying

$$\pi = \pi \boldsymbol{P}, \tag{23}$$

thus the first eigenvalue is always 1. This will lead to the asymptotic convergence to the global optimality θ_*:

$$\lim_{k \to \infty} \theta_k \to \theta_*, \tag{24}$$

with probability one (Gamerman, 1997; Gutjahr, 2002).

Now if look at the PSO and FA closely using the framework of Markov chain Monte Carlo, each particle in PSO or each firefly in FA essentially forms a Markov chain, though this Markov chain is biased towards to the current best, as the transition probability often leads to the acceptance of the move towards the current global best. Other population-based algorithms can also be viewed in this framework. In essence, all metaheuristic algorithms with piecewise, interacting paths can be analyzed in the general framework of Markov chain Monte Carlo. The main challenge is to realize this and to use the appropriate Markov chain theory to study metaheuristic algorithms. More fruitful studies will surely emerge in the future.

7.4 Other results

Limited results on convergence analysis exist, concerning finite domains, ant colony optimization (Gutjahr,2010; Sebastiani and Torrisi,2005), cross-entropy optimization, best-so-far convergence (Margolin, 2005), nested partition method, Tabu search, and largely combinatorial optimization. However, more challenging tasks for infinite states/domains and continuous problems. Many open problems need satisfactory answers.

On the other hand, it is worth pointing out that an algorithm can converge, but it may not be efficient, as its convergence rate could be typically low. One of the main tasks in research is to find efficient algorithms for a given type of problem.

8. Open problems

Active research on NFL theorems and algorithm convergence analysis has led to many important results. Despite this, many crucial problems remain unanswered. These open questions span a diverse range of areas. Here we highlight a few but relevant open problems.

Framework: Convergence analysis has been fruitful, however, it is still highly needed to develop a unified framework for algorithmic analysis and convergence.

Exploration and exploitation: Two important components of metaheuristics are exploration and exploitation or diversification and intensification. What is the optimal balance between these two components?

Performance measure: To compare two algorithms, we have to define a measure for gauging their performance (Spall et al., 2006). At present, there is no agreed performance measure, but what are the best performance measures ? Statistically?

Free lunches: No-free-lunch theorems have not been proved for continuous domains for multiobjective optimization. For single-objective optimization, free lunches are possible; is this true for multiobjective optimization? In addition, no free lunch theorem has not been proved to be true for problems with NP-hard complexity (Whitley and Watson, 2005). If free lunches exist, what are their implications in practice and how to find the best algorithm(s)?

Automatic parameter tuning: For almost all algorithms, algorithm-dependent parameters require fine-tuning so that the algorithm of interest can achieve maximum performance. At the moment, parameter-tuning is mainly done by inefficient, expensive parametric studies. In fact, automatic self-tuning of parameters is another optimization problem, and optimal tuning of these parameters is another important open problem.

Knowledge: Problem-specific knowledge always helps to find an appropriate solution? How to quantify such knowledge?

Intelligent algorithms: A major aim for algorithm development is to design better, intelligent algorithms for solving tough NP-hard optimization problems. What do mean by 'intelligent'? What are the practical ways to design truly intelligent, self-evolving algorithms?

9. Concluding remarks

SI-based algorithms are expanding and becoming increasingly popular in many disciplines and applications. One of the reasons is that these algorithms are flexible and efficient in solving a wide range of highly nonlinear, complex problems, yet their implementation is relatively straightforward without much problem-specific knowledge. In addition, swarming agents typically work in parallel, and thus parallel implementation is a natural advantage.

At present, swarm intelligence and relevant algorithms are inspired by some specific features of the successful biological systems such as social insects and birds. Though they are highly successful, however, these algorithms still have room for improvement. In addition to the above open problems, a truly 'intelligent' algorithm is yet to be developed. By learning more and more from nature and by carrying out ever-increasingly detailed, systematical studies, some truly 'smart' self-evolving algorithms will be developed in the future so that such smart algorithms can automatically fine-tune their behaviour to find the most efficient way of solving complex problems. As an even bolder prediction, maybe, some hyper-level algorithm-constructing metaheuristics can be developed to automatically construct algorithms in an intelligent manner in the not-too-far future.

10. References

[1] Afshar, A., Haddad, O. B., Marino, M. A., Adams, B. J., (2007). Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation, *J. Franklin Institute*, 344, 452-462.

[2] Auger, A. and Teytaud, O., Continuous lunches are free plus the design of optimal optimization algorithms, Algorithmica, 57, 121-146 (2010).

[3] Auger, A. and B. Doerr, B., *Theory of Randomized Search Heuristics: Foundations and Recent Developments*, World Scientific, (2010).

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

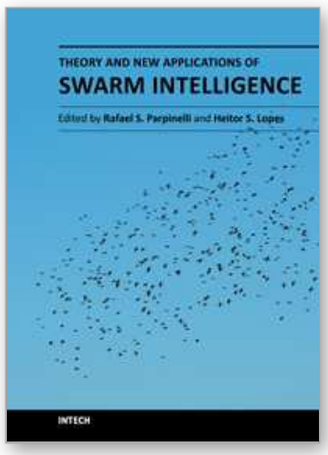
[5] Blum, C. and Roli, A. (2003) Metaheuristics in combinatorial optimisation: Overview and conceptural comparision, *ACM Comput. Surv.*, Vol. 35, 268-308.

[6] Clerc, M. and J. Kennedy, J., The particle swarm - explosion, stability, and convergence in a multidimensional complex space, *IEEE Trans. Evolutionary Computation*, 6, 58-73 (2002).

[7] Corne D. and Knowles, J., Some multiobjective optimizers are better than others, *Evolutionary Computation*, CEC'03, 4, 2506-2512 (2003).

[8] Christensen, S. and Oppacher, F., (2001). Wath can we learn from No Free Lunch? in: *Proc. Genetic and Evolutionary Computation Conference* (GECCO-01), pp. 1219-1226 (2001).
[9] Dorigo, M. and Stütle, T., *Ant Colony Optimization*, MIT Press, (2004).
[10] Floudas, C. A. and Pardolos, P. M., *Encyclopedia of Optimization*, 2nd Edition, Springer (2009).
[11] Gamerman, D., *Markov Chain Monte Carlo*, Chapman & Hall/CRC, (1997).
[12] Glover, F. and Laguna, M. (1997) *Tabu Search*, Kluwer Academic Publishers, Boston.
[13] Goldberg, D. E., *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Addison-Wesley, Reading, MA, (2002).
[14] Gutjahr, W. J., ACO algorithms with guaranteed convergence to the optimal solution, *Information Processing Letters*, 82, 145-153 (2002).
[15] Gutjahr, W. J., Convergence Analysis of Metaheuristics *Annals of Information Systems*, 10, 159-187 (2010).
[16] He, J. and Yu, X., Conditions for the convergence of evolutionary algorithms, *J. Systems Architecture*, 47, 601-612 (2001).
[17] Henderson, D., Jacobson, S. H., and Johnson, W., The theory and practice of simulated annealing, in: *Handbook of Metaheuristics* (Eds. F. Glover and G. A. Kochenberger), Kluwer Academic, pp. 287-319 (2003).
[18] Holland, J., *Adaptation in Natural and Artificial systems*, University of Michigan Press, Ann Anbor, (1975).
[19] Igel, C. and Toussaint, M., (2003). On classes of functions for which no free lunch results hold, *Inform. Process. Lett.*, 86, 317-321 (2003).
[20] Karaboga, D., (2005). An idea based on honey bee swarm for numerical optimization, Technical Report TR06, Erciyes University, Turkey.
[21] Kennedy, J. and Eberhart, R. (1995) 'Particle swarm optimisation', in: *Proc. of the IEEE Int. Conf. on Neural Networks*, Piscataway, NJ, pp. 1942-1948.
[22] Kirkpatrick, S., Gellat, C. D., and Vecchi, M. P. (1983) 'Optimisation by simulated annealing', *Science*, 220, 671-680.
[23] Nakrani, S. and Tovey, C., (2004). On Honey Bees and Dynamic Server Allocation in Internet Hosting Centers. *Adaptive Behaviour*, 12(3-4), 223-240.
[24] Neumann, F. and Witt, C., *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*, Springer, (2010).
[25] Margolin, L., On the convergence of the cross-entropy method, *Annals of Operations Research*, 134, 201-214 (2005).
[26] Marshall, J. A. and Hinton, T. G., Beyond no free lunch: realistic algorithms for arbitrary problem classes, WCCI 2010 IEEE World Congress on Computational Intelligence, July 1823, Barcelona, Spain, pp. 1319-1324 (2010).
[27] Ólafsson, S., Metaheuristics, in: *Handbook on Simulation* (Eds. Nelson and Henderson), Handbooks in Operation Reserch and Management Science VII, Elsevier, pp. 633-654 (2006).
[28] Parpinelli, R. S., and Lopes, H. S., New inspirations in swarm intelligence: a survey, *Int. J. Bio-Inspired Computation*, 3, 1-16 (2011).
[29] Pham, D.T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., and Zaidi, M., (2006). The Bees Algorithm Ű A Novel Tool for Complex Optimisation Problems, Proceedings of IPROMS 2006 Conference, pp.454-461
[30] Schumacher, C., Vose, M., and Whitley D., The no free lunch and problem description length, in: *Genetic and Evolutionary Computation Conference*, GECCO-2001, pp. 565-570 (2001).

[31] Sebastiani, G. and Torrisi, G. L., An extended ant colony algorithm and its convergence analysis, *Methodoloy and Computating in Applied Probability*, 7, 249-263 (2005).
[32] Shilane, D., Martikainen, J., Dudoit, S., Ovaska, S. J. (2008) 'A general framework for statistical performance comparison of evolutionary computation algorithms', *Information Sciences*, Vol. 178, 2870-2879.
[33] Spall, J. C., Hill, S. D., and Stark, D. R., Theoretical framework for comparing several stochastic optimization algorithms, in:*Probabilistic and Randomized Methods for Design Under Uncertainty*, Springer, London, pp. 99-117 (2006).
[34] Thikomirov, A. S., On the convergence rate of the Markov homogeneous monotone optimization method, *Computational Mathematics and Mathematical Physics*, 47, 817-828 (2007).
[35] Villalobos-Arias, M., Coello Coello, C. A., and Hernández-Lerma, O., Asymptotic convergence of metaheuristics for multiobjective optimization problems, *Soft Computing*, 10, 1001-1005 (2005).
[36] Whitley, D. and Watson, J. P., Complexity theory and the no free lunch theorem, in: *Search Methodolgies*, pp. 317-339 (2005).
[37] Wolpert, D. H. and Macready, W. G. (1997), No free lunch theorems for optimisation, *IEEE Transaction on Evolutionary Computation*, Vol. 1, 67-82.
[38] Wolpert, D. H. and Macready, W. G., Coevolutonary free lunches, *IEEE Trans. Evolutionary Computation*, 9, 721-735 (2005).
[39] Yang, X. S., (2005). Engineering optimization via nature-inspired virtual bee algorithms, IWINAC 2005, *Lecture Notes in Computer Science*, 3562, pp. 317-323.
[40] Yang, X. S. (2008), *Nature-Inspired Metaheuristic Algorithms*, Luniver Press.
[41] Yang, X. S. (2009), Firefly algorithms for multimodal optimisation, *Proc. 5th Symposium on Stochastic Algorithms, Foundations and Applications, SAGA 2009*, Eds. O. Watanabe and T. Zeugmann, Lecture Notes in Computer Science, Vol. 5792, 169-178.
[42] Yang, X. S. (2010a), Firefly algorithm, stochastic test functions and design optimisation, *Int. J. Bio-Inspired Computation*, 2, 78–84.
[43] Yang, X. S. (2010b), *Engineering Optimization: An Introduction with Metaheuristic Applications*, John Wiley and Sons, USA (2010).
[44] Yang, X. S., (2010c), A New Metaheuristic Bat-Inspired Algorithm, in: *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)* (Eds. J. R. Gonzalez et al.), Studies in Computational Intelligence, Springer Berlin, 284, Springer, 65-74 (2010).
[45] Yang X. S. and Deb S., (2009). Cuckoo search via Lévy flights, *Proceeings of World Congress on Nature & Biologically Inspired Computing* (NaBIC 2009, India), IEEE Publications, USA, pp. 210-214.
[46] Yang X. S. and Deb, S. (2010) Engineering optimisation by cuckoo search, *Int. J. Math. Modelling & Num. Optimisation*, Vol. 1, 330-343.
[47] Yang X. S., (2011), Bat algorithm for multi-objective optimisation, *Int. J. Bio-Inspired Computation*, 3 (5), 267-274 (2011).
[48] Yu L., Wang S. Y., Lai K.K. and Nakamori Y, Time Series Forecasting with Multiple Candidate Models: Selecting or Combining? *Journal of Systems Science and Complexity*, 18, No. 1, pp1-18 (2005).



Theory and New Applications of Swarm Intelligence

Edited by Dr. Rafael Parpinelli

ISBN 978-953-51-0364-6
Hard cover, 194 pages
Publisher InTech
Published online 16, March, 2012
Published in print edition March, 2012

The field of research that studies the emergent collective intelligence of self-organized and decentralized simple agents is referred to as Swarm Intelligence. It is based on social behavior that can be observed in nature, such as flocks of birds, fish schools and bee hives, where a number of individuals with limited capabilities are able to come to intelligent solutions for complex problems. The computer science community have already learned about the importance of emergent behaviors for complex problem solving. Hence, this book presents some recent advances on Swarm Intelligence, specially on new swarm-based optimization methods and hybrid algorithms for several applications. The content of this book allows the reader to know more both theoretical and technical aspects and applications of Swarm Intelligence.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Xin-She Yang (2012). Swarm-Based Metaheuristic Algorithms and No-Free-Lunch Theorems, Theory and New Applications of Swarm Intelligence, Dr. Rafael Parpinelli (Ed.), ISBN: 978-953-51-0364-6, InTech, Available from: http://www.intechopen.com/books/theory-and-new-applications-of-swarm-intelligence/swarm-based-metaheuristic-algorithms-and-no-free-lunch-theorems

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen