Advances in Engineering Software 105 (2017) 30–47

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Research paper

Grasshopper Optimisation Algorithm: Theory and application

Shahrzad Saremi[a,b], Seyedali Mirjalili[a,b,*], Andrew Lewis[a]

[a] School of Information and Communication Technology, Griffith University, Nathan, Brisbane, QLD 4111, Australia
[b] Griffith College, Mt Gravatt, Brisbane, QLD 4122, Australia

ARTICLE INFO

Article history:
Received 29 October 2016
Accepted 10 January 2017
Available online 31 January 2017

Keywords:
Optimization
Optimization techniques
Heuristic algorithm
Metaheuristics
Constrained optimization
Benchmark
Algorithm

ABSTRACT

This paper proposes an optimisation algorithm called Grasshopper Optimisation Algorithm (GOA) and applies it to challenging problems in structural optimisation. The proposed algorithm mathematically models and mimics the behaviour of grasshopper swarms in nature for solving optimisation problems. The GOA algorithm is first benchmarked on a set of test problems including CEC2005 to test and verify its performance qualitatively and quantitatively. It is then employed to find the optimal shape for a 52-bar truss, 3-bar truss, and cantilever beam to demonstrate its applicability. The results show that the proposed algorithm is able to provide superior results compared to well-known and recent algorithms in the literature. The results of the real applications also prove the merits of GOA in solving real problems with unknown search spaces.

1. Introduction

The process of finding the best values for the variables of a particular problem to minimise or maximise an objective function is called optimisation. Optimisation problems exist in different fields of studies. To solve an optimisation problem, different steps need to be taken. Firstly, the parameters of the problem should be identified. Based on the nature of the parameters, problems may be classified as continuous or discrete. Secondly, the constraints that are applied to the parameters have to be recognised [1]. Constraints divide the optimisation problems into constrained and unconstrained. Thirdly, the objectives of the given problem should be investigated and considered. In this case, optimisation problems are classified into single-objective versus multi-objective problems [2]. Finally, based on the identified types of parameters, constraints, and number of objectives a suitable optimiser should be chosen and employed to solve the problem.

Mathematical optimisation mainly relies on gradient-based information of the involved functions in order to find the optimal solution. Although such techniques are still being used by different researchers, they have some disadvantages. Mathematical optimisation approaches suffer from local optima entrapment. This refers to an algorithm assuming a local solution is the global solution, thus failing to obtain the global optimum. They are also often ineffective for problems with unknown or computationally expensive derivation [3]. Another type of optimisation algorithm that alleviates these two drawbacks is stochastic optimisation [4].

Stochastic methods rely on random operators that allow them to avoid local optima. They all start optimisation process by creating one or a set of random solutions for a given problem. In contrast to mathematical optimisation techniques, they do not need to calculate the gradient of a solution, just evaluating the solutions using the objective function(s). Decisions as to how to improve the solutions are made based on the calculated objective values. Therefore, the problem is considered as a black box, which is a very useful mechanism when solving real problems with unknown search spaces. Due to these advantages, stochastic optimisation techniques have become very popular over the past two decades [5].

Among stochastic optimisation approaches, nature-inspired, population-based algorithms are the most popular [6]. Such techniques mimic natural problems-solving methods, often those used by creatures. Survival is the main goal for all creatures. To achieve this goal, they have been evolving and adapting in different ways. Therefore, it is wise to seek inspiration from nature as the best and oldest optimiser on the planet. Such algorithms are classified into two main groups: single-solutions-based and multi-solution-based. In the former class, a single random solution is generated and improved for a particular problem. In the latter class, however, multiple solutions are generated and enhanced for a given problem. Multi-solution-based algorithms are more popular than single-solution-based methods, as the literature shows [7].

Multi-solution-based algorithms intrinsically have higher local optima avoidance due to improving multiple solutions during optimisation. In this case, a trapped solution in a local optimum can be assisted by other solutions to jump out of the local

* Corresponding author.
E-mail address: seyedali.mirjalili@griffithuni.edu.au (S. Mirjalili).
URL: http://www.alimirjalili.com (S. Mirjalili)

http://dx.doi.org/10.1016/j.advengsoft.2017.01.004
0965-9978/

optimum. Multiple solutions explore a larger portion of the search space compared to single-solution-based algorithms, so the probability of finding the global optimum is high. Also, information about the search space can be exchanged between multiple solutions, which results in quick movement towards the optimum. Although multi-solution-based algorithms have several advantages, they require more function evaluations.

The most popular single-solution-based algorithms are hill climbing [8] and simulated annealing [9]. Both algorithms follow a similar idea, but the local optima avoidance of SA is higher due to the stochastic cooling factor. Other recent single-solution-based algorithms are Tabu Search (TS) [10,11], and Iterated Local Search (ILS) [12]. The most popular multi-solutions-based algorithms are Genetic Algorithms (GA) [13], Particle Swarm Optimisation (PSO) [14], Ant Colony Optimisation (ACO) [15], and Differential Evolution (DE) [16]. The GA algorithm was inspired by the Darwinian theory of evolution. In this algorithm, solutions are considered as individuals and the parameters of solutions take the place of their genes. Survival of the fittest individuals is the main inspiration of this algorithm where the best individuals tend to participate more in improving poor solutions. The PSO algorithm simulates the foraging of herds of birds or schools of fishes. In this algorithm the solutions are improved with respect to the best solutions obtained so far by each of the particles and the best solution found by the swarm. The ACO algorithm mimics the collective behaviour of ants in finding the shortest path from the nest to the source of foods. Finally, DE utilises simple formulae combining the parameters of existing solutions to improve the population of candidate solutions for a given problem.

The similarity of both classes of nature-inspired algorithms is the improvement of solutions until the satisfaction of an end criterion and the division of optimisation process into two phases: exploration versus exploitation [17]. Exploration refers to the tendency for an algorithm to have highly randomised behaviour so that the solutions are changed significantly. Large changes in the solutions cause greater exploration of the search space and consequently discovery of its promising regions. As an algorithm tends toward exploitation, however, solutions generally face changes on a smaller scale and tend to search locally. A proper balance of exploration and exploitation can result in finding the global optimum of a given optimisation problem.

The literature shows that there are many recent swarm intelligence optimisation techniques such as Dolphin Echolocation (DEL) [18,19], Firefly Algorithm (FA) [20,21], Bat Algorithm (BA) [22], and Grey Wolf Optimizer (GWO) [3]. DEL and BA mimic echolocation of dolphins in finding prey and bats navigating respectively. However, FA simulates the mating behaviour of fireflies in nature. Cuckoo Search (CS) [23,24] is another recent algorithm in this field, in which the reproductive processes of cuckoos are employed to propose an optimisation algorithm. The GWO is also a swarm-based technique that models the hunting mechanism of grey wolves.

There are also other algorithms with different inspiration in the literature. For instance, State of Matter Search (SMS) [25,26] uses the concepts of different phases in matter to optimise problems and the Flower Pollination Algorithm (FPA) [27] has been inspired by the survival and reproduction of flowers using pollination. There is a question here as to why we need more algorithms despite the many algorithms proposed so far.

The answer to this question is in the No Free Lunch (NFL) theorem [28] that logically has proven that there is no optimisation technique for solving all optimisation problems. In other words, algorithms in this field perform equally on average when considering *all* optimisation problems. This theorem, in part, has motivated the rapidly increasing number of algorithms proposed over the last decade and is one of the motivations of this paper as well. The next section proposes a new algorithm mimicking the behaviour of grasshopper swarms. There are a few works in the literature that have tried to simulate locust swarm [29–33]. The current study is an attempt to more comprehensively model grasshopper behaviours and propose an optimisation algorithm based on their social interaction.

Due to their simplicity, gradient-free mechanism, high local optima avoidance, and considering problems as black boxes, nature-inspired algorithms have been applied widely in science and industry [34–36]. Therefore, we also investigate the application of the proposed algorithm in solving real problems. The rest of the paper is organised as follows:

The Grasshopper Optimisation Algorithm is proposed in Section 2. Section 3 presents and discusses the results on the optimisation test beds and inspects the behaviour of the proposed algorithm. Section 4 contains the application of the proposed method in the field of structural design optimisation. Finally, Section 5 concludes the work and suggests several directions for future studies.

2. Grasshopper Optimisation Algorithm (GOA)

Grasshopper are insects. They are considered a pest due to their damage to crop production and agriculture. The life cycle of grasshoppers is shown in Fig. 1. Although grasshoppers are usually seen individually in nature, they join in one of the largest swarm of all creatures [37]. The size of the swarm may be of continental scale and a nightmare for farmers. The unique aspect of the grasshopper swarm is that the swarming behaviour is found in both nymph and adulthood [38]. Millions of nymph grasshoppers jump and move like rolling cylinders. In their path, they eat almost all vegetation. After this behaviour, when they become adult, they form a swarm in the air. This is how grasshoppers migrate over large distances.

The main characteristic of the swarm in the larval phase is slow movement and small steps of the grasshoppers. In contrast, long-range and abrupt movement is the essential feature of the swarm in adulthood. Food source seeking is another important characteristic of the swarming of grasshoppers. As discussed in the introduction, nature-inspired algorithms logically divide the search process into two tendencies: exploration and exploitation. In exploration, the search agents are encouraged to move abruptly, while they tend to move locally during exploitation. These two functions, as well as target seeking, are performed by grasshoppers naturally. Therefore, if we find a way to mathematically model this behaviour, we can design a new nature-inspired algorithm.

The mathematical model employed to simulate the swarming behaviour of grasshoppers is presented as follows [39]:

$$X_i = S_i + G_i + A_i \tag{2.1}$$

where X_i defines the position of the *i-th* grasshopper, S_i is the social interaction, G_i is the gravity force on the *i-th* grasshopper, and A_i shows the wind advection. Note that to provide random behaviour the equation can be written as $X_i = r_1 S_i + r_2 G_i + r_3 A_i$ where r_1, r_2, and r_3 are random numbers in [0,1].

$$S_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} s\left(d_{ij}\right) \widehat{d_{ij}} \tag{2.2}$$

where d_{ij} is the distance between the *i-th* and the *j-th* grasshopper, calculated as $d_{ij} = |x_j - x_i|$, s is a function to define the strength of social forces, as shown in Eg. (2.3), and $\widehat{d_{ij}} = \frac{x_j - x_i}{d_{ij}}$ is a unit vector from the ith grasshopper to the jth grasshopper.

The s function, which defines the social forces, is calculated as follows:

$$s(r) = f e^{\frac{-r}{l}} - e^{-r} \tag{2.3}$$