Advances in Engineering Software 000 (2017) 1–29

Contents lists available at ScienceDirect

Advances in Engineering Software

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

Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems

Seyedali Mirjalili[a,*], Amir H. Gandomi[b,f], Seyedeh Zahra Mirjalili[c], Shahrzad Saremi[a], Hossam Faris[d], Seyed Mohammad Mirjalili[e]

[a] Institute for Integrated and Intelligent Systems, Griffith University, Nathan, QLD 4111, Australia
[b] BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI 488241, USA
[c] School of Electrical Engineering and Computing, University of Newcastle, Callaghan, NSW 2308, Australia
[d] Business Information Technology Department, King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
[e] Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, H3G1M8, Canada
[f] School of Business, Stevens Institute of Technology, Hoboken, NJ 07030, USA

ARTICLE INFO

Article history:
Received 25 December 2016
Revised 23 April 2017
Accepted 9 July 2017
Available online xxx

Keywords:
Optimization
Constrained optimization
Particle swarm optimization
Multi-objective optimization
Genetic algorithm
Heuristic algorithm
Algorithm
Benchmark

ABSTRACT

This work proposes two novel optimization algorithms called Salp Swarm Algorithm (SSA) and Multi-objective Salp Swarm Algorithm (MSSA) for solving optimization problems with single and multiple objectives. The main inspiration of SSA and MSSA is the swarming behaviour of salps when navigating and foraging in oceans. These two algorithms are tested on several mathematical optimization functions to observe and confirm their effective behaviours in finding the optimal solutions for optimization problems. The results on the mathematical functions show that the SSA algorithm is able to improve the initial random solutions effectively and converge towards the optimum. The results of MSSA show that this algorithm can approximate Pareto optimal solutions with high convergence and coverage. The paper also considers solving several challenging and computationally expensive engineering design problems (e.g. airfoil design and marine propeller design) using SSA and MSSA. The results of the real case studies demonstrate the merits of the algorithms proposed in solving real-world problems with difficult and unknown search spaces.

1. Introduction

Over the past decade, meta-heuristic techniques have become surprisingly very popular. This popularity is due to several main reasons: flexibility, gradient-free mechanism, and local optima avoidance of these algorithms. The first two advantages originate from the fact that meta-heuristics consider and solve optimization problems by only looking at the inputs and outputs. In other words, meta-heuristics assume an optimization problem as a black box. Therefore, there is no need to calculate derivative of the search space. This makes them highly flexible for solving a diverse range of problems. Since meta-heuristics belong to the family of stochastic optimization techniques, they benefit from random operators. This assists them to avoid local solutions when solving real problems, which usually have a large number of local optima. Due to these advantages, the application of meta-heuristics can be found in different branches of science and industry.

Meta-heuristic algorithms are classified into two dominant classes: evolutionary [1] and swarm intelligence [2] techniques. Evolutionary algorithms mimic the concepts of evolution in nature. The best and most well-regarded algorithm in this class is Genetic Algorithm (GA) [3]. This algorithm simulates the concepts of Darwinian theory of evolution. In GA, the optimization is initiated with a set of random solutions for a particular problem. After evaluating the solutions by the objective function, it modifies the variables of solutions based on their fitness value. Since the best individuals are given higher probability to involve in improving other solutions, the random initial solutions are very likely to be improved. There are several other evolutionary algorithms in the literature such as Differential Evolution (DE) [4], Evolutionary Strategy (ES) [5], and Evolutionary Programming (EP) [6,7], and Biogeography-Based Optimization (BBO) algorithm [8] as well.

Swarm intelligence techniques mimic the intelligence of swarms, herds, schools, or flocks of creatures in nature. The main foundation of these algorithms originates from the collective behaviour of a group of creatures. For instance, ants are able to collectively guarantee the survival of a colony without having a centralized control unit. In other word, no one tells ants where and

* 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.2017.07.002
0965-9978/

how a source food can be found, but they cooperatively find foods at even far distances from their nests. The two most popular algorithms in this class are Ant Colony Optimization (ACO) [9] and Particle Swarm Optimization (PSO) [10]. The ACO algorithm mimics the social behaviour of ants for finding the shortest path between the nest and a source food. The PSO algorithm simulates the collective behaviour of birds in navigating and hunting. Other swarm intelligence techniques in the literature are: Artificial Bee Colony (ABC) algorithm [11], Cuckoo Search (CS) algorithm [12], Firefly Algorithm (FA) [13], Bat Algorithm (BA) [14], Grey Wolf Optimizer (GWO) [15–17], Dolphin Echolocation (DE) [18], Whale Optimization Algorithm (WOA) [19], Fruitfly Optimization Algorithm (FOA) [20], and Harmony Search [21,22].

Regardless of the difference between evolutionary and swarm intelligence techniques, the common is the improvement of one or a set of solutions during optimization. If an algorithm improves only one solution, it is called individualist algorithm. If a set of solutions is improved, it is referred as a collective algorithm. Individualist algorithms are beneficial because of the low number of required function evaluation and simplicity of the overall optimization process. However, the probability of local optima stagnation is very high. Collective algorithms are able to avoid local solutions better and exchange information about the search space. However, such techniques require more number of function evaluations. Some of the individualist algorithms are Tabu Search (TS) [6,23], hill climbing [24], Iterated Local Search (ILS) [25], and Simulated Annealing (SA) [26], Variable Neighborhood Search (VNS) [27], and Guided Local Search [28]. The well-known collective algorithms are GA, ACO, PSO, DE, and ES.

Despite the merits of the proposed algorithms in the literature, it has been proved by the No-Free-Lunch (NFL) [29] that none of these algorithms are able to solve all optimization problems. In other words, all meta-heuristics perform similar when solving all optimization problems. This theorem reveals the importance of new and specific algorithms in different fields because effectiveness of an algorithm in solving a set of problems does not guarantee its success in different sets of test problems. This is the motivation of this paper, in which a new meta-heuristic optimization algorithm is first proposed for solving single-objective problems and then extended to a multi-objective version. The rest of the paper is organized as follows.

Section 2 reviews the literature and relevant works. Section 3 presents the inspiration and mathematical model proposed. The Salp Swarm Algorithm (SSA) and Multi-objective Salp Swarm Algorithm (MSSA) are proposed in this section as well. The qualitative and quantitative results of both algorithms on a variety of benchmark functions are presented and discussed in Section 4. Both SSA and MSSA are employed to solve several challenging real problems in Section 5. Finally, Section 6 concludes the work and suggest several future research directions.

2. Related works

This section reviews the state-of-the-art in the field of stochastic optimization. There are many branches in this field such as single-objective, multi-objective, constrained, dynamic, surrogate-assisted, many-objective, and so on. Since the algorithms proposed solve single- and multi-objective optimization problems, the main focus of this section is on the challenges and related works in single- and multi- objective optimization fields.

2.1. Single-objective optimization problems

As its name implies, single-objective optimization deals with one objective. This means there is only one objective to be minimized or maximized. This type of optimization might be subject to a set of constraints as well. The constraints are divided to two categories: equality and inequality. Single-objective optimization is formulated as a minimization problem as follows (without the loss of generality):

$$Minimize: \ F(\vec{x}) = \{f_1(\vec{x})\} \tag{2.1}$$

$$Subject\ to: \ g_i(\vec{x}) \geq 0, \ i = 1, 2, \ldots, m \tag{2.2}$$

$$h_i(\vec{x}) = \ 0, \ i = 1, 2, \ldots, p \tag{2.3}$$

$$lb_i \leq x_i \leq ub_i, \ i = 1, 2, \ldots, d \tag{2.4}$$

where d is the number of variables, p is the number of equality constraints, m is the number of inequality constrained, lb_i is the lower bound of the ith variable, and ub_i indicates the upper bound of the ith variable.

The set of variables, objectives, range of variables, and constraints create a search space/landscape. This search space exists in a d-dimensional space where d is the number of variables. For 1D, 2D, and 3D problems, we can easily draw the search space in a Cartesian coordinate system and observe their shapes. However, it is not possible to draw dimensions greater than 3 because they are beyond the dimensions that we experience every day. Therefore, a large number of variables is the first challenge when solving optimization problems.

The range of variables confides the search space and is varied. The variables themselves can be continuous or discrete, in which they create either a continuous or a discrete search space. In a former case, there is an infinite number of points between each two points in the search space. In the latter case, however, there is a finite set of points between two points. Finding the global optimum in a continuous space is different from a discrete one, and each of them has their own challenges. Although most of the optimization problems come with range of variables, there are some problems that do not have a specific range to be considered during optimization. An example is the problem of training Neural Networks (NNs) [30]. The connection weights and biases can be any real number. Solving such problems also need special consideration. For instance, an optimizer might start with an initial range and then expand it during optimization.

The constraints limit the search space even further. They create gaps in the search space because the solutions in those regions are not suitable for the problem. For instance, the thickness of a propeller blade cannot go below a certain number due to the fragility. A set of constraints can even split the search space to different separated regions. The solutions that violate the constrained regions are called infeasible solutions. In contrast, the solutions inside the constrained areas are called feasible solutions. In the literature, there are two terms for the parts of the search space that are inside and outside the constrained areas: feasible and infeasible regions. A constrained search space has the potential to make an algorithm ineffective despite its good performance in an unconstrained search space. Some of the real problems such as Computational Fluid Dynamic problems have dominated infeasible regions. Therefore, optimization techniques should be equipped with suitable operators [31] to handle constraints as well.

Another challenge when solving optimization problems is the presence of local solutions. The search space that the variables, objective function, and constraints create may be very simple or complicated. In most of the works in the literature, the number of local solutions is considered as the main difficulty for optimization algorithms. In a single-objective search space there is one best solution (the so-called global optimum) that returns the best objective value. However, there are usually many other solutions that return values close the objective value of the global optimum. This kind of solutions are called local solutions because they are locally the best solution if we consider the search space in their vicinity, but they are not the best solution globally when considering