Swarm and Evolutionary Computation 26 (2016) 97–119

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Regular Paper

Opposition-based Magnetic Optimization Algorithm with parameter adaptation strategy

Mahdi Aziz [a,*], Mohammad-H. Tayarani-N. [b]

[a] Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Vali Asr St., Iran
[b] Department of Electrical and Computer Science, University of Birmingham, Birmingham, UK

ARTICLE INFO

Article history:
Received 7 December 2014
Received in revised form
16 July 2015
Accepted 3 September 2015
Available online 16 September 2015

Keywords:
Parameter adaptation strategy
Opposition-Based Learning
Magnetic Optimization Algorithm
Numerical optimization problems

ABSTRACT

Magnetic Optimization Algorithm (MOA) has emerged as a promising optimization algorithm that is inspired by the principles of magnetic field theory. In this paper we improve the performance of the algorithm in two aspects. First an Opposition-Based Learning (OBL) approach is proposed for the algorithm which is applied to the movement operator of the algorithm. Second, by learning from the algorithm's past experience, an adaptive parameter control strategy which dynamically sets the parameters of the algorithm during the optimization is proposed. To show the significance of the proposed parameter adaptation strategy, we compare the algorithm with two well-known parameter setting techniques on a number of benchmark problems. The results indicate that although the proposed algorithm with the adaptation strategy does not require to set the parameters of the algorithm prior to the optimization process, it outperforms MOA with other parameter setting strategies in most large-scale optimization problems. We also study the algorithm while employing the OBL by comparing it with the original version of MOA. Furthermore, the proposed algorithm is tested and compared with seven traditional population-based algorithms and eight state-of-the-art optimization algorithms. The comparisons demonstrate that the proposed algorithm outperforms the traditional algorithms in most benchmark problems, and its results is comparative to those obtained by the state-of-the-art algorithms.

1. Introduction

Inspired by the principles of attraction among magnetic particles, MOA is a population based algorithm that belongs to the group of swarm intelligence algorithms. In MOA, the candidate solutions are some magnetic particles that are scattered across the search space. In this respect, each magnetic particle has a measure of mass and magnetic field according to its fitness. In this scheme, the fitter magnetic particles have higher magnetic field and greater mass. In terms of interaction, these particles are located in a lattice-like population and apply a long range force of attraction to their neighbors. Unlike Particle Swarm Optimization (PSO) algorithm in which each particle utilizes only the best experience of the best neighboring particle(s) or the best particle in the population, in MOA each magnetic particle uses the best experience of all its neighboring particles, including the inferior ones. In order to improve the performance of the algorithm, an OBL [1] approach is proposed in this paper in which by calculating the opposite population of the current population at each iteration, the algorithm tries to find fitter solutions. OBL has been used to solve many optimization problems [2–4] and has been employed in several population based algorithms [5–9].

MOA has shown promising results when applied to numerical benchmark functions [10,11] and to a wide range of optimization problem including travelling salesman problem [12] and multilayer perception training [13]. Similar to other population based algorithms, the performance of the MOA depends on appropriately setting its parameters [10,11]. Although there is a systematic way of setting the parameters of MOA [10,11], it is computationally expensive. The parameter setting technique [14,10,15,16] provides appropriate values for control parameters; however, the algorithm designer needs to set the control parameters for each problem prior to the search process.

To improve the performance of the algorithm in this aspect, several parameter setting approaches have recently been proposed. The F-Race algorithm firstly proposed for tackling the model selection problem [17] is among them. The algorithm is an automatic parameter configuration algorithm that was firstly used by [18] to automatically set the parameters of Ant Colony algorithm. Then a new version of the algorithm called iterated F-Race was utilized in some optimization algorithms [19–21]. Iterated F-Race determines the most appropriate parameter configuration of an algorithm using the non-parametric Friedman's two-way analysis of

* Corresponding author. Tel.: +98 9156523782.

http://dx.doi.org/10.1016/j.swevo.2015.09.001
2210-6502/

variance by ranks. Acting like a hill climbing stochastic procedure, iterated F-Race performs a few race among the candidate configurations on a stream of instances in order to find the best candidate configuration. First a set of configurations with uniform random values are initialized. Then, at each iteration, all configurations are evaluated according to Friedman test. If the first Friedman test shows that at least one configuration is significantly different from any other configurations in the race, the second Friedman test is applied to eliminate the candidates that are remarkably worse than other configurations. The race proceeds with the surviving configurations and continues until only one candidate configuration remains in the race or the certain number of iteration is reached. Although the method is successful in setting parameters of algorithms, specially when an algorithm has a number of parameters [19,21], it can be prohibitively expensive for large-scale optimization problems.

Using the feedback received from the search process, parameter adaptation techniques adjust the parameters of algorithms adaptively. According to [22–24], depending on how the received feedback is used, there are three major types of parameter setting strategies: deterministic parameter control, self-adaptive parameter control and adaptive parameter control. The deterministic parameter setting approaches are those that do not receive any feedback from the optimization process and set their parameters prior to the search process via trial and error. The original version of MOA is an example of this type of strategy. Self-adaptive parameter setting strategy attempts to evolutionarily adjust the parameters of algorithms; to do so, they often adopt recombination operators such as mutation and crossover to select the optimal parameter configuration. This approach has shown remarkable success in iteratively making the individuals more adapted to the problems. For example, reference [25] proposed a new Differential Evolution (DE) algorithm that uses a self-adaptive parameter strategy for the population size, mutation rate and crossover rate. The parameter adaptation strategy refers to the parameter setting, which uses feedback received from the search process to dynamically set the parameters of the problem. Several state-of-the-art algorithms such as JADE [22], SaDE [26], jDE [27] and Memetic algorithm with adaptive local search [28] can be categorized into this group. The proposed algorithm, which dynamically adjusts its control parameters in the course of the optimization, also belongs to this category.

Being adaptable to the properties of the problem usually enhances the ability of algorithms to find good parameters without spending time on the trial and error parameter setting procedure. Therefore, parameter adaptation strategy can help the algorithm discover a good parameter value while enhancing the convergence performance. JADE as one of the powerful DE algorithms that employs the parameter adaptation strategy showed remarkable success in tackling several small-scale optimization problems [22]. JADE has two control parameters that sets them adaptively. In this paper, we develop the idea used in JADE for adaptively setting the control parameters of the proposed algorithm. The difference between the proposed algorithm and JADE is that our algorithm optimizes the control parameters individually. When the control parameters are investigated and set together (similar to JADE), they may not provide high-quality results for large-scale benchmark problems. This is because it cannot be ensured which parameter is responsible for improving the quality of the solution and so unnecessary changes in the value of a parameter may occur. Instead, if separately evaluated and set, the parameters can be more appropriately adjusted which results in better performance.

The contribution of this paper is summarized into the following aspects:

- A new version of MOA using a run-time adaptation strategy for dynamically setting the parameters of the algorithm is proposed.
- A new approach that is based on the opposite number principle is developed and added to the algorithm to improve its performance.
- The proposed parameter adaptation strategy is compared with two famous parameter setting techniques including systematic parameter setting and F-Race algorithm.
- A set of most powerful optimizers including Genetic Algorithm (GA) [29], PSO [30], DE [31], Evolution Strategy (ES) [32], Fast Evolution Strategy (FES) [33], Evolutionary programming (EP) [34] and Fast Evolutionary Programming (FEP) [35], Memetic Algorithm with Solis Wet local search (MASW) [36], Memetic Algorithm with Subgrouping Solis Wet local search (MASSW) [36], Cooperatively Coevolving Particle Swarms Optimization (CCPSO2) [37], JADE [22], Three Stages Memetic Exploration (3SOME) [38], Parallel Memetic Structure (PMS) [39], Biogeography Based Optimization (BBO) [40], Opposition-based Differential Evolution (ODE) [41] and Covariance Matrix Adaptation Evolution Strategy (CMAES) [42] are used to be compared with the proposed algorithm on 27 standard benchmark functions.

The rest of this paper is organized as follows. Section 2 discusses the background of the proposed algorithm, including the OBL and the original version of MOA. Section 3 introduces the proposed algorithm. Section 4 evaluates the proposed parameter adaptation strategy, by studying the control parameters and comparing the proposed strategy with two well-known strategies. Section 5 provides a comparison between the proposed algorithm and the original version of MOA, seven popular population-based and nine state-of-the-art algorithms. Section 6 concludes this paper.

2. Background

In this section, a general overview of the key components of the proposed algorithm is presented, concentrating on the Opposition-Based Learning scheme and the original version of MOA.

2.1. Opposition-Based Learning

Population-based algorithms often initialize the population randomly, thus the chance of sampling better regions in the search space is not higher. However, there are several ways to enhance the probability of detecting better regions. One is Opposition-Based Learning (OBL). By employing OBL at the initialization phase of algorithms, the likelihood of finding better solutions increases. Furthermore, an algorithm can employ the OBL approach during its search process to increase its chances of finding better solutions [41].

The concept of OBL was proposed by Tizhoosh in [1]. In this paper, we first explain the concept of opposition numbers. Let $x \in [a, b]$ be a real number, then the opposite number $\breve{x}$ is defined as

$$\breve{x} = a + b - x.$$

The definition can be extended to an N-dimensional search space [1] as follows. Let $P = (x_1, x_2, \ldots, x_N)$ represent a point in an N-dimensional space. The opposition vector in this space is defined as

$$\breve{x}_i = a_i + b_i - x_i.$$

Heretofore, the OBL has extensively been used to solve many optimization problems [2–4] and has been employed in several population-based algorithms [41,6,7,9]. This encouraged us to employ the method in MOA to speed up the convergence speed