Applied Soft Computing 27 (2015) 490–503

Contents lists available at ScienceDirect

Applied Soft Computing

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

Performance evaluation of automatically tuned continuous optimizers on different benchmark sets

Tianjun Liao [a,*], Daniel Molina [b], Thomas Stützle [c]

[a] State Key Laboratory of Complex System Simulation, Beijing Institute of System Engineering, Beijing, China
[b] Dept. of Computer Engineering, University of Cádiz, Cádiz, Spain
[c] IRIDIA, CoDE, Université Libre de Bruxelles (ULB), Brussels, Belgium

ARTICLE INFO

Article history:
Received 29 January 2014
Received in revised form 3 October 2014
Accepted 5 November 2014
Available online 22 November 2014

Keywords:
Continuous optimization
Benchmark sets
Evolutionary computation
Swarm intelligence
Automatic parameter tuning

ABSTRACT

The development of algorithms for tackling continuous optimization problems has been one of the most active research topics in soft computing in the last decades. It led to many high performing algorithms from areas such as evolutionary computation or swarm intelligence. These developments have been sidelined by an increasing effort of benchmarking algorithms using various benchmarking sets proposed by different researchers.

In this article, we explore the interaction between benchmark sets, algorithm tuning, and algorithm performance. To do so, we compare the performance of seven proven high-performing continuous optimizers on two different benchmark sets: the functions of the special session on real-parameter optimization from the IEEE 2005 Congress on Evolutionary Computation and the functions used for a recent special issue of the Soft Computing journal on large-scale optimization. While one conclusion of our experiments is that automatic algorithm tuning improves the performance of the tested continuous optimizers, our main conclusion is that the choice of the benchmark set has a much larger impact on the ranking of the compared optimizers. This latter conclusion is true whether one uses default or tuned parameter settings.

1. Introduction

Continuous optimization problems arise in many important design and control problems in areas such as engineering, telecommunications, or computer science. In many cases, the resulting problems have to be treated as black-box problems, where no explicit mathematical formulation of the problem is available and, therefore, derivatives may not be computed [9]. Often, such problems have multiple optima, correlated variables and non-linear objective functions, making their solution very hard. Thus, as an alternative to classical continuous optimization techniques, direct-search (or also called derivative-free) methods have emerged [9].

An important source of direct-search methods is the area of soft computing. Many techniques such as evolutionary computation [4], memetic algorithms [44], ant colony optimization [12], particle swarm optimization [27], differential evolution [52] or artificial bee colonies [25] have been at the core of recent advancements when developing optimization techniques. All these techniques have been applied to tackle (black-box) continuous optimization problems. Such algorithms have been developed starting more than four decades ago and thousands of articles have been published providing new algorithmic ideas or algorithm designs and analyses of such algorithms.

This proliferation of algorithmic techniques and variants thereof, has made it difficult to identify which algorithms contribute to define the state-of-the-art. To address some shortcomings in the evaluation of (nature-inspired) continuous optimizers, several benchmarking efforts have been undertaken. Early benchmarking efforts such as the first international contest on evolutionary optimization [5] in 1996 had some but little impact. Later efforts, however, have been followed much more strongly, possibly because the number of available algorithms has grown very much. A noteworthy example is the special session on real parameter optimization of the 2005 IEEE Congress on Evolutionary Computation (CEC'05) [53]. It proposed a set of 25 benchmark problems and identified the covariance matrix adaptation evolution strategy (CMA-ES) [16] with occasional restarts with increasing population size (IPOP-CMA-ES) [1] as the best performing algorithm on this CEC'05 benchmark set. The impact this special session

* Corresponding author. Tel.: +86 18302330650.
E-mail addresses: liaotianjun@gmail.com (T. Liao), daniel.molina@uca.es (D. Molina), stuetzle@ulb.ac.be (T. Stützle).

http://dx.doi.org/10.1016/j.asoc.2014.11.006
1568-4946/

had and still has is illustrated by the more than 750 citations the technical report defining the benchmark functions received in google scholar by December 2013. Other efforts include a special session on large scale global optimization held at the IEEE CEC'08 conference. The benchmark set of this special session was later extended in various successor competitions. In particular, the special issue on large scale continuous optimization problems of the Soft Computing journal [40] extended this benchmark set to comprise a total of 19 freely scalable benchmark functions; we refer to this benchmark set in the following as the SOCO benchmark set. In this special issue, a memetic algorithm that combines a local search algorithm with a differential evolution algorithm in the multiple offspring (MOS) framework [30] was the best-performing algorithm; we refer to this algorithm as MOS in what follows.

If one takes a closer look into the results on these two benchmark sets, an interesting difference arises. In fact, on the SOCO benchmark set, IPOP-CMA-ES, the winner of the CEC'05 benchmarking competition, was used as an algorithm to set a performance baseline and the final winner, MOS, performed significantly better than IPOP-CMA-ES. This result reminds of an observation made by Whitley et al. [56], p. 245:

> "*Test functions are commonly used to evaluate the effectiveness of different search algorithms. However, the results of evaluation are as dependent on the test problems as they are on the algorithms that are the subject of comparison.*"

Paraphrasing this statement in terms of the current benchmark sets, it would say that the choice of the benchmark set has a strong impact on the final ranking of the algorithms tested. Unfortunately, MOS was not tested on the CEC'05 benchmark set and it is therefore unclear whether it would perform better than IPOP-CMA-ES also on the CEC'05 benchmark set.

One goal of this article is to examine the impact the choice of the benchmark set has on the relative performance of current high performance continuous optimizers. We do so by choosing a number of algorithms that are derived from different search techniques and that have shown very good performance on at least one of the CEC'05 and SOCO benchmark sets. In particular, we include the already mentioned IPOP-CMA-ES [1] and MOS [30] algorithms as winners of the two benchmark competitions. In addition, we selected the following algorithms. MA-LSCh-CMA [42] is a memetic algorithm for continuous optimization based on local search chains; it was reported very high performance on the CEC'05 benchmark set. SaDE [48] is a self-adaptive differential evolution algorithm, which received the 2012 IEEE Transactions on Evolutionary Computation Outstanding Paper Award; IPSOLS [43] and IABCLS [3] are particle swarm optimization and artificial bee colony algorithms, which obtained the best performance in their respective fields on the SOCO benchmark set; UACOR [37] is a configurable framework of ant colony optimization for continuous domains that shows improved performance when compared to $IACO_R$ [33], which was the so far best performing ant colony optimization algorithm for continuous optimization on the above mentioned benchmark sets. Although some researchers have compared continuous optimizers from different metaheuristics [2,18,14,49], the aforementioned start-of-the-art continuous optimizers have not been evaluated together in the literature.

A second goal of this article is to examine the impact specific choices on the parameter settings have on the evaluation of continuous optimizers. Typically, parameter settings are defined by the algorithm designers based on preliminary experiments and considering a specific set of test problems. This may lead to implicit biases in the parameter tuning toward specific test functions or benchmark sets and an uneven effort in parameter tuning. Biases in algorithm tuning and uneven tuning effort are two problematic issues when comparing algorithms as one algorithm may seem superior to another one only because its parameters are more fine-tuned for a specific benchmark set. As an alternative, one may rely on automatically tuned parameter settings through the usage of automatic algorithm configuration techniques [7,23,24,39,46]. In fact, experience with these techniques has shown that often they are able to find improved parameter settings over manually tuned default settings and to make parameter tuning more efficient [7,23,22,24]. Therefore, here we compare the continuous optimizers using their default settings and based on parameters that are tuned by using an automatic algorithm configuration tool, irace [7,39], to give an unbiased comparison.

In a nutshell, the experimental results that we obtain indicate that (i) all algorithms may profit from the automatic algorithm configuration, (ii) the benchmark sets have a major impact on the ranking of the algorithms, and (iii) some algorithms exhibit poor scaling behavior with increasing dimension. We believe that these results also show that future benchmarking efforts need to consider larger and more systematically varied sets of benchmark problems and, in particular, the evaluation of the continuous optimizers on more benchmark problems stemming from real applications is desirable to steer future research efforts.

The article is organized as follows. Section 2 describes the seven continuous optimizers we test. Section 3 gives more background on automatic algorithm configuration and tuning. Next, Section 4 describes the benchmark sets and Section 5 the experimental setups. In Sections 6–8, we evaluate the optimizers using default and tuned parameters on the CEC'05 and SOCO benchmark sets. We end the article with a discussion of the obtained results and conclude in Section 9.

2. The tested continuous optimizers

In this section, we concisely introduce the seven continuous optimizers that we compare and justify their choice. Since these optimizers are well described in the literature, we refer to the original articles for their detailed description to keep the present article short.[1]

2.1. IPOP-CMA-ES

CMA-ES [16] is a (μ, λ)-evolution strategy that samples at each iteration new solutions based on a multivariate normal distribution. It adapts the covariance matrix of a normal search distribution during the execution to lead the sampling to the most promising part of the search space. Its design makes the algorithm invariant to the scaling of the evaluation function and to rotations of the variable's correlation. CMA-ES may be seen as a stochastic local search technique; IPOP-CMA-ES [1] provides an iterative restart scheme once CMA-ES is deemed to stagnate and at each restart increases the population size. It obtained the best performance among all the candidates in the special session on real parameter optimization of CEC'05 [53] and is since then seen as a state-of-the-art continuous optimizer.

Following [36], we consider seven parameters (a, b, c, d, e, f, g) for tuning. These parameter control some of the formulas that determine IPOP-CMA-ES's search behavior. In particular, these parameters are used in the following equations. The initial population size is $\lambda = 4 + \lfloor a \ln(D) \rfloor$, where D is the number of dimensions of the function to be optimized. The number of parent solutions is $\mu = \lfloor \lambda / b \rfloor$. The initial step-size is $\sigma^{(0)} = c(B - A)$, where $[A, B]^D$ is the initial search interval. The population size is multiplied by a

[1] The list of the parameters, their default and tuned values and their domains considered for tuning are given in Tables 2 and 3.