European Journal of Operational Research 234 (2014) 597–609

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Continuous Optimization

A unified ant colony optimization algorithm for continuous optimization

Tianjun Liao[a,*], Thomas Stützle[b], Marco A. Montes de Oca[c], Marco Dorigo[b]

[a] State Key Laboratory of Complex System Simulation, Beijing Institute of System Engineering, 10 An Xiang Bei Li Rd., Beijing, China
[b] IRIDIA, Université Libre de Bruxelles, CP 194/6, Av. F. Roosevelt 50, B-1050 Brussels, Belgium
[c] Dept. of Mathematical Sciences, University of Delaware, 15 Orchard Rd., Newark, DE 19716, USA

ARTICLE INFO

Article history:
Received 11 August 2012
Accepted 8 October 2013
Available online 30 October 2013

Keywords:
Ant colony optimization
Continuous optimization
Automatic algorithm configuration

ABSTRACT

In this article, we propose UACOR, a unified ant colony optimization (ACO) algorithm for continuous optimization. UACOR includes algorithmic components from ACO$_{\mathbb{R}}$, DACO$_{\mathbb{R}}$ and IACO$_{\mathbb{R}}$-LS, three ACO algorithms for continuous optimization that have been proposed previously. Thus, it can be used to instantiate each of these three earlier algorithms; in addition, from UACOR we can also generate new continuous ACO algorithms that have not been considered before in the literature. In fact, UACOR allows the usage of automatic algorithm configuration techniques to automatically derive new ACO algorithms. To show the benefits of UACOR's flexibility, we automatically configure two new ACO algorithms, UACOR-s and UACOR-c, and evaluate them on two sets of benchmark functions from a recent special issue of the Soft Computing (SOCO) journal and the IEEE 2005 Congress on Evolutionary Computation (CEC'05), respectively. We show that UACOR-s is competitive with the best of the 19 algorithms benchmarked on the SOCO benchmark set and that UACOR-c performs superior to IPOP-CMA-ES and statistically significantly better than five other algorithms benchmarked on the CEC'05 set. These results show the high potential ACO algorithms have for continuous optimization and suggest that automatic algorithm configuration is a viable approach for designing state-of-the-art continuous optimizers.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Metaheuristics are a family of optimization techniques that have seen increasingly rapid development and have been applied to numerous problems over the past few years. A prominent metaheuristic is ant colony optimization (ACO). ACO is inspired by the ants' foraging behavior and it was first applied to solve discrete optimization problems (Dorigo & Stützle, 2004; Dorigo, Maniezzo, & Colorni, 1991, 1996). Only much later, adaptations of ACO to continuous optimization problems were introduced. Socha and Dorigo (Socha & Dorigo, 2008) proposed one of the now most popular ACO algorithms for continuous domains, called ACO$_{\mathbb{R}}$. It uses a solution archive as a form of pheromone model for the derivation of a probability distribution over the search space. Leguizamón and Coello (2010) proposed an extension of ACO$_{\mathbb{R}}$, called DACO$_{\mathbb{R}}$, that had the goal of better maintaining diversity during the search. Subsequently, Liao, Montes de Oca, Aydın, Stützle, and Dorigo (2011) proposed IACO$_{\mathbb{R}}$-LS, an incremental ant colony algorithm with local search for continuous optimization. IACO$_{\mathbb{R}}$-LS uses a growing solution archive as an extra search diversification mechanism and a local search to intensify the search. IACO$_{\mathbb{R}}$-LS was benchmarked on two prominent sets of benchmark functions for continuous optimization, obtaining very good results. These benchmark function sets are the ones proposed for a recent special issue of the Soft Computing journal (Herrera, Lozano, & Molina, 2010; Lozano, Molina, & Herrera, 2011) (we refer to this special issue as SOCO) and the special session on real parameter optimization of the 2005 IEEE Congress on Evolutionary Computation (CEC'05) (Suganthan et al., 2005).

In this article, we propose a ACO algorithm for continuous optimization that combines algorithmic components from ACO$_{\mathbb{R}}$, DACO$_{\mathbb{R}}$ and IACO$_{\mathbb{R}}$-LS. We call this algorithm Unified ACO for continuous optimization (UACOR). It is unified, because from UACOR, we can instantiate the original ACO$_{\mathbb{R}}$, DACO$_{\mathbb{R}}$ and IACO$_{\mathbb{R}}$-LS algorithms by using specific combinations of the available algorithmic components and parameter settings. However, we can also obtain combinations of algorithm components that are different from any of the already proposed combinations; in other words, from UACOR we can instantiate new continuous ACO algorithms that have not been proposed or tested before.

The flexibility of UACOR makes possible the use of automatic algorithm configuration tools to generate new, high-performing continuous ACO algorithms. Here, we follow such an approach and use Iterated F-race (Birattari, Yuan, Balaprakash, & Stützle, 2010), an automatic algorithm configuration tool, as implemented in the irace package (López-Ibáñez, Dubois-Lacoste, Stützle, & Birattari, 2011) for configuring new high-performing ACO algorithms for continuous optimization from UACOR. With automatic

* Corresponding author. Tel.: +86 18302330650.
E-mail addresses: tliao@ulb.ac.be (T. Liao), stuetzle@ulb.ac.be (T. Stützle), mmontes@math.udel.edu (M.A. Montes de Oca), mdorigo@ulb.ac.be (M. Dorigo).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.10.024

configuration tools, algorithm parameters are defined using a kind of machine learning approach in which an algorithm is first trained on a set of problem instances and later deployed. We use as training sets low dimensional versions of the functions in the SOCO and CEC'05 benchmark sets and configure two new ACO variants: UACOR-s is configured on the SOCO benchmark (the -s suffix stands for SOCO) set and UACOR-c on the CEC'05 benchmark set (the -c suffix stands for CEC). UACOR-s and UACOR-c are then tested on higher dimensional versions of the SOCO and CEC'05 benchmark functions. The results show that (i) UACOR-s is competitive or superior to all the 19 algorithms benchmarked on the SOCO function set and that (ii) UACOR-c is superior to IPOP-CMA-ES (Auger & Hansen, 2005) and statistically significantly better than other five recent state-of-the-art algorithms benchmarked on the CEC'05 function set. These experimental results show (i) the high potential of ACO algorithms for continuous optimization and (ii) the high potential of an algorithm design approach that is based on the combination of algorithm frameworks and automatic algorithm configuration. In fact, there are few researches that give evidence for the latter point. For instance, KhudaBukhsh, Xu, Hoos, and Leyton-Brown (2009) proposed SATenstein and instantiated a new state-of-the-art local search algorithm for the SAT problem; López-Ibáñez and Stützle (2010) configured a multi-objective ACO algorithm that outperformed previously proposed multi-objective ACO algorithms for the bi-objective traveling salesman problem; Dubois-Lacoste, López-Ibáñez, and Stützle (2011) configured new state-of-the-art algorithms for five variants of multi-objective flow-shop problems. More recently, the ideas behind the combination of algorithm frameworks and automatic algorithm configuration techniques have been extended to the programming by optimization paradigm (Hoos, 2012). This article is the first to automatically configure a continuous optimizer framework.

The article is organized as follows. Section 2 introduces ACO for continuous domains, reviews the three continuous ACO algorithms underlying UACOR, and identifies their algorithmic components in a component-wise view. Section 3 describes UACOR. In Section 4, we automatically configure UACOR to instantiate UACOR-s and UACOR-c and in Section 5, we evaluate their performance. We conclude and give directions for future work in Section 6.

2. ACO algorithms for continuous optimization

2.1. ACO metaheuristic

The Ant Colony Optimization (ACO) metaheuristic (Dorigo & Stützle, 2004) defines a class of optimization algorithms inspired by the foraging behavior of real ants. In ACO algorithms, artificial ants are stochastic procedure for constructing candidate solution that exploit a pheromone model and possibly available heuristic information on the problem being tackled. The pheromone model consists of a set of numerical values, called pheromones, that are modified at each iteration in order to bias ants toward the most promising regions of the search space; the heuristic information, if available, captures a priori knowledge on the particular problem instance being solved.

The main algorithmic components of the ACO metaheuristic are the ants' solution construction and the update of the pheromone information. "Daemon actions" are procedures that carry out tasks that cannot be performed by single ants. A common example is the activation of a local search procedure to improve an ant's solution or the application of additional pheromone modifications derived from globally available information about, for example, the best solutions constructed so far. Although daemon actions are optional, they can greatly improve the performance of ACO algorithms.

2.2. ACO for continuous domains

After the initial proposals of ACO algorithms for combinatorial optimization problems (Dorigo & Stützle, 2004; Dorigo et al., 1991, Dorigo, Maniezzo, & Colorni, 1996), several ant-inspired algorithms for continuous optimization problems were proposed (Bilchev & Parmee, 1995; Dréo & Siarry, 2004; Hu, Zhang, & Li, 2008; Hu, Zhang, Chung, Li, & Liu, 2010; Monmarché, Venturini, & Slimane, 2000). However, as explained in Socha and Dorigo (2008), most of these algorithms use search mechanisms different from those used in the ACO metaheuristic. The first algorithm that can be classified as an ACO algorithm for continuous domains is ACO$_\mathbb{R}$ (Socha & Dorigo, 2008). In ACO$_\mathbb{R}$, the discrete probability distributions used in the solution construction by ACO algorithms for combinatorial optimization are substituted by probability density functions (PDFs) (i.e., continuous probability distributions). ACO$_\mathbb{R}$ uses a solution archive (Guntsch & Middendorf, 2002) for the derivation of these PDFs over the search space. Additionally, ACO$_\mathbb{R}$ uses sums of weighted Gaussian functions to generate multimodal PDFs.

Fig. 1 shows a sketch of a solution archive and the Gaussian functions that form the PDFs from which ACO$_\mathbb{R}$ samples values to generate candidate solutions. The solution archive keeps track of a number of complete candidate solutions for a problem, and, thus, it can be seen as an explicit memory of the search history.

DACO$_\mathbb{R}$ (Leguizamón & Coello, 2010) and IACO$_\mathbb{R}$-LS (Liao et al., 2011) are two more recent ACO algorithms for continuous optimization, which also use a solution archive and generate PDFs using sums of weighted Gaussian functions. Since the algorithmic components of UACOR are derived from the ACO$_\mathbb{R}$, DACO$_\mathbb{R}$ and IACO$_\mathbb{R}$-LS, the next sections describe their operation.

2.2.1. ACO$_\mathbb{R}$

ACO$_\mathbb{R}$ initializes the solution archive with k solutions that are generated uniformly at random. Each solution is a D-dimensional vector with real-valued components $x_i \in [x_{\min}, x_{\max}]$, with $i = 1, \ldots, D$. In this paper, we assume that the optimization problems are unconstrained except possibly for bound constraints of the D real-valued variables x_i. The k solutions of the archive are kept sorted according to their quality (from best to worst) and each solution $\boldsymbol{S}_j$ has associated a weight ω_j. This weight is calculated using a Gaussian function as:

$$\omega_j = \frac{1}{qk\sqrt{2\pi}} e^{\frac{-(rank(j)-1)^2}{2q^2k^2}}, \tag{1}$$

where $rank(j)$ is the rank of solution $\boldsymbol{S}_j$ in the sorted archive, and q is a parameter of the algorithm. By computing $rank(j) - 1$, the best solution receives the highest weight.

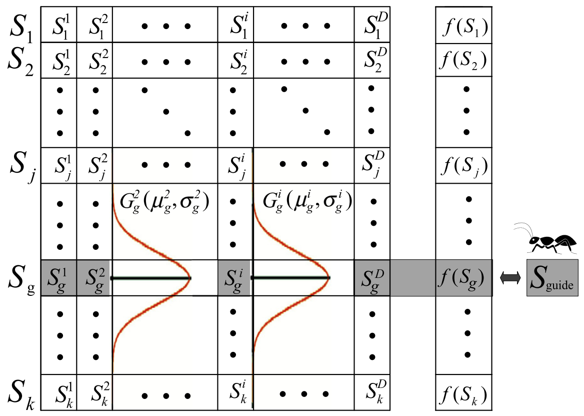

Fig. 1. The structure of the solution archive and the Gaussian functions used to generate PDFs in ACO$_\mathbb{R}$.

Download English Version:

https://daneshyari.com/en/article/478194

Download Persian Version:

https://daneshyari.com/article/478194

Daneshyari.com