Swarm and Evolutionary Computation 27 (2016) 116–131

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Regular Paper

Benchmarking NLopt and state-of-the-art algorithms for continuous global optimization via $IACO_{\mathbb{R}}$

Udit Kumar, Sumit Soman, Jayadeva*

Department of Electrical Engineering, Indian Institute of Technology, Delhi, India

ARTICLE INFO

Article history:
Received 12 March 2015
Received in revised form
4 October 2015
Accepted 7 October 2015
Available online 22 October 2015

Keywords:
ACO
Global optimization
$IACO_{\mathbb{R}}$−Local Search
Mtsls1
NLopt
Hybrid $IACO_{\mathbb{R}}$

ABSTRACT

This paper presents a comparative analysis of the performance of the Incremental Ant Colony algorithm for continuous optimization ($IACO_{\mathbb{R}}$), with different algorithms provided in the NLopt library. The key objective is to understand how various algorithms in the NLopt library perform in combination with the Multi-Trajectory Local Search (Mtsls1) technique. A hybrid approach has been introduced for the local search strategy, by the use of a parameter that allows for probabilistic selection between Mtsls1 and the NLopt algorithm. In case of stagnation, a switch is made based on the algorithm being used in the previous iteration. This paper presents an exhaustive comparison on the performance of these approaches on Soft Computing (SOCO) and Congress on Evolutionary Computation (CEC) 2014 benchmarks. For both sets of benchmarks, we conclude that the best performing algorithm is a hybrid variant of Mtsls1 with BFGS for local search.

1. Introduction

The NLopt (Non-Linear Optimization) library (v2.4.2) [1] is a rich collection of optimization routines and algorithms, which provides a platform-independent interface for their use for global and local optimization. The library has been widely used for practical implementations of optimization algorithms as well as for benchmarking new algorithms.

The work in this paper is based on the $IACO_{\mathbb{R}}$−LS algorithm proposed by Liao et al. [2]. This algorithm introduced the local search procedure in the original $IACO_{\mathbb{R}}$ technique, specifically Mtsls1 by Tseng and Chen [3] for local search. The $IACO_{\mathbb{R}}$ was an extension of the $ACO_{\mathbb{R}}$ algorithm for continuous optimization with the added advantage of a variable size solution archive. The premise of our work lies in improving the local search strategy adopted by $IACO_{\mathbb{R}}$−LS, by allowing algorithms other than Mtsls1 to be used for local search.

We present a comparison of using various algorithms from the NLopt library for local search procedure in the $IACO_{\mathbb{R}}$−LS algorithm. In order to introduce a hybrid approach for local search, we use a parameter that probabilistically determines whether to use the Mtsls1 algorithm or the NLopt library algorithm. In case of stagnation, we switch between Mtsls1 or the NLopt algorithm based on the algorithm being used in the previous iteration. The objective is to rigorously analyze the effect of using various optimization algorithms in the local search procedure for $IACO_{\mathbb{R}}$−LS, as well as provide results on benchmark functions to enable a naive researcher to choose an algorithm easily. To the best of our knowledge, available works in literature have not provided exhaustive comparisons using optimization algorithm libraries on ant colony based approaches, other than Rios and Sahinidis [4]. However, surveys on state-of-the-art in multi-objective evolutionary algorithms [5], differential evolution [6] and real-parameter evolutionary multimodal optimization [7] have appeared in literature.

The rest of the paper is organized as follows. Section 2 discusses our hybrid approach which allows using Mtsls1 along with an NLopt library algorithm for local search phase of $IACO_{\mathbb{R}}$−LS. This is followed by a discussion on the NLopt library in Section 3. We present our results and a discussion in Section 4, followed by the conclusions in Section 5.

2. Hybrid local search using Mtsls1 and NLopt algorithms

We begin by introducing the Mtsls1 algorithm, and the motivation to develop a hybrid approach for local search. This is followed by a description of our algorithm which uses the hybrid local search using Mtsls1 and the algorithms from the NLopt library.

Many variants of the Ant Colony Optimization (ACO) algorithm have been developed for continuous global optimization [8,9]. These algorithms work in two phases, viz. exploration and

* Corresponding author.
E-mail addresses: udit.vlsi@gmail.com (U. Kumar), sumit.soman@gmail.com (S. Soman), jayadeva@ee.iitd.ac.in (Jayadeva).
URL: http://web.iitd.ac.in/~jayadeva (Jayadeva).

http://dx.doi.org/10.1016/j.swevo.2015.10.005
2210-6502/© 2015 Elsevier B.V. All rights reserved.

exploitation. The exploration phase searches in a global region to find an optimal solution, while the exploitation (or local search) phase aims at refining the global solutions by searching within a limited region. Usually, the Incremental Ant Colony Optimization ($IACO_{\mathbb{R}}$) algorithm [2] is used for the exploration phase. An acclaimed algorithm for the exploitation phase has been the Multi-Trajectory Local Search, or Mtsls1 algorithm [3], which exploits the search space across multiple paths. The approach has evolved to many variants, notable among which are the self-adaptive evolution by Zhao et al. [10], multi-objective optimization [11] and dynamic search trajectories by Snyman and Fatti [12].

To bring perspective to the discussion on motivating our approach, we briefly mention the working of previous ACO algorithms. The original $ACO_{\mathbb{R}}$ approach used a fixed-size solution archive and aimed at iteratively improving the solution quality. New solutions were constructed along each dimension of the given problem as the algorithm progressed along iterations. This was done by estimating the multi-modal Probability Density Functions (PDFs) using weighted Gaussians. As an improvement to this algorithm, the $IACO_{\mathbb{R}}$ approach used a variable size solution archive, whose growth was regulated using a parameter in the algorithm. Initial solutions for this approach were randomly chosen, and these were then moved to the best solutions by estimation of a weighted difference between the new and present best solutions. This approach also chose solutions for subsequent generation of possible solutions in a probabilistic fashion. Subsequently, the $IACO_{\mathbb{R}}-LS$ approach introduced the local search phase in this direction to improve the solution quality. This also necessitated the use of parameters to set the maximum number of local iterations and exit criteria for local search, which was required when the solutions became stagnated due to local search. An algorithm for the local search was the Mtsls1.

Mtsls1 searches along one dimension, and optimum value of one dimension is used as starting point for the next dimension. At each dimension, Mtsls1 tries to move by a step size s along one dimension, and evaluates the change in the function value. If the function value decreases, then new point is used for optimization along the next dimension, If the function value increases, then algorithm goes back to the starting point and moves by a factor of the step size, $0.5{*}s$ towards negative direction and evaluates the function. Again the function value is compared and based on minimum value of the function, the optimum point is provided.

We propose an hybrid local-search approach which incorporates the non-gradient based Mtsls1, along with an algorithm from the NLopt library as part of the $IACO_{\mathbb{R}}-LS$ technique. Our approach offers a choice between selecting either of the two, based on a probabilistically determined choice. This is indicated by the algorithm parameter $P(nlopt)$. In case this probabilistic choice fails to provide any improvement after a specific number of iterations, we switch the algorithm being used based on the algorithm used in the previous iteration. The parameters $ctr_{localsearch}$ and $thresh_{localsearch}$ have been used in our algorithm to implement this. We select a different local search algorithm when $ctr_{localsearch}$ crosses $thresh_{localsearch}$. This ensures that our local-search approach does not stagnate, and also gives our approach an "adaptive" flavor. Our hybrid approach is summarized in Fig. 1.

All algorithms from NLopt library are used as part of the hybrid local search approach. The Nlopt algorithms meant for global optimization are allowed as many function evaluations as set for global search, but for local search, maximum allowed function evaluations in a single local search call is set to 160. It may be noted here that the method for approximating the gradient is directly linked to algorithm's ability to escape local minima. Solomon [13] opines that "if, however, the gradient is estimated by independent trials with a distance along each axis, the difference between both classes of algorithms almost vanishes." Hence, our

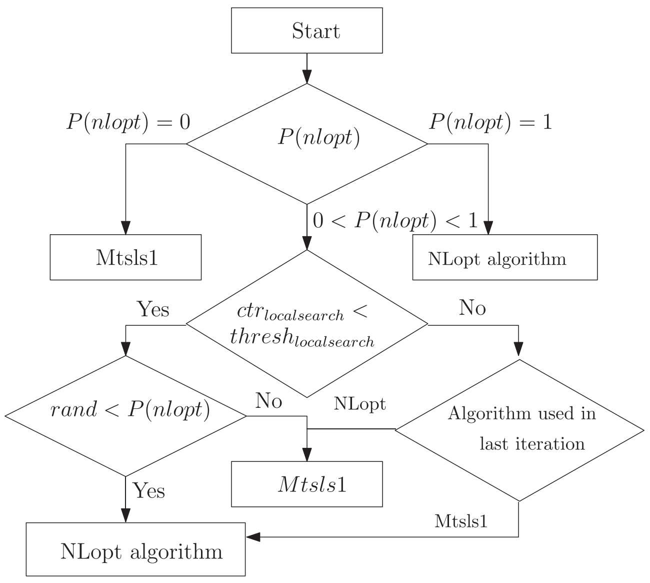

Fig. 1. Illustration of our hybrid approach.

computations of gradient are based on approximating the derivative by using central differences. By using this method, the maximum number of function evaluations would effectively be $(2{*}n\ (for\ gradient)+1\ (function\ evaluation){*}160)$ for local search, for a n-dimensional problem. Our approach is illustrated in Algorithm 1; note that hybrid local-search approach is incorporated at Steps (10)–(28), for additional details reader may refer to [3].

Algorithm 1. Hybrid $IACO_{\mathbb{R}}$ algorithm.

```
1:  procedure HYBRID $IACO_{\mathbb{R}}$ (Probability ($p$), constant parameter ($\zeta$), initial archive size ($\alpha$), growth ($\gamma$), maximum archive size ($\alpha_{max}$), function tolerance ($\tau$), maximum failure ($Fail_{max}$), maximum stagnation iterations ($Siter_{max}$), dimensions ($N$), termination criteria ($Tc$), probability of switching to NLopt algorithm ($P(nlopt)$), maximum local search iterations ($thresh_{localsearch}$))
2:    Initialize $\alpha$ solutions
3:    Evaluate initial solutions
4:    while ($Tc$ not satisfied) do
5:      if ($Fail_{(i,best)} < Fail_{max}$) then
6:        Local search from $Sol_{best}$
7:      else if ($Fail_{(i,random)} < Fail_{max}$) then
8:        Local search from $Sol_{random}$
9:      end if
10:     if $P(nlopt) = = 0$ then          ▹ Local Search
11:       Use Mtsls1
12:     els if $P(nlopt) = = 1$ then
13:       Use NLopt algorithm
14:     else
15:       if $ctr_{localsearch} < thresh_{localsearch}$ then
16:         if $rand() < P(nlopt)$ then
17:           Use NLopt algorithm
18:         else
19:           Use Mtsls1
20:         end if
21:       else
22:         if Last iteration used Mtsls1 then
23:           Use NLopt algorithm
24:         else
25:           Use Mtsls1
26:         end if
27:       end if
```