Applied Soft Computing 61 (2017) 1125–1138

Contents lists available at ScienceDirect

Applied Soft Computing

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

Full length article

Robust optimization using Bayesian optimization algorithm: Early detection of non-robust solutions

Marjan Kaedi [a], Chang Wook Ahn [b,*]

[a] Faculty of Computer Engineering, University of Isfahan, Hezar-Jerib Ave., Isfahan 81746-73441, Iran
[b] School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), 123 Cheomdangwagi-ro, Buk-gu, Gwangju 61005, Republic of Korea

ARTICLE INFO

Article history:
Received 7 August 2015
Received in revised form 1 March 2017
Accepted 26 March 2017
Available online 31 March 2017

Keywords:
Robust optimization
Bayesian optimization algorithm
Bayesian networks
Probabilistic robustness evaluation

ABSTRACT

Probabilistic robustness evaluation is a promising approach to evolutionary robust optimization; however, high computational time arises. In this paper, we apply this approach to the Bayesian optimization algorithm (BOA) with a view to improving its computational time. To this end, we analyze the Bayesian networks constructed in BOA in order to extract the patterns of non-robust solutions. In each generation, the solutions that match the extracted patterns are detected and then discarded from the process of evaluation; therefore, the computational time in discovering the robust solutions decreases. The experimental results demonstrate that our proposed method reduces computational time, while increasing the robustness of solutions.

1. Introduction

In many real-world optimization problems, uncertainty is often unavoidable [1]. In these problems, after finding the optimal solution (i.e., after finding the best values for the design variables), the values of the design variables or the environmental variables may slightly change. If the optimal solution is very sensitive to these changes, slight alterations in the design or environmental variables lead to significant reduction in the quality of the solution [1]. Therefore, the optimal solution cannot be used in practice [2,3]. Problems such as scheduling, vehicle routing optimization, and engineering design are examples of the situations [4–8]. In these situations, a solution which is not just optimum but also robust, should be found [8,9]. A robust solution can be thought of as a solution that can tolerate slight variation in design variables or environmental variables, and its quality does not significantly change with any minor alteration in the condition for which it was designed [6,10–12].

In this paper, we focus only on the change of design variables. In this case, a robust solution is defined as a solution that has high quality while the solutions in its neighborhood (i.e., solutions that are similar to it but slightly differ in some design variables) also have high quality [13]. To find a robust solution, the robustness of the candidate solutions should be evaluated and considered in the process of searching for the optima. This objective can be accomplished in two ways [11,14]. First, the objective function of the optimization problem can be replaced with a new objective function that simultaneously evaluates both the performance and robustness of the solutions. Second, a new objective function can be added whose responsibility is to evaluate the robustness of the solutions. As such, the optimization problem is converted to a multi-objective optimization problem.

In evaluating the robustness of a solution, there are two approaches: *probabilistic* and *deterministic* [11,14]. In the probabilistic approach, the probable changes that may occur in the future are considered, and the expected performance of the solution in all of the probable situations is calculated. This approach is appropriate for maximizing the average performance. In the deterministic approach, the worst situation (i.e., the worst change) that is likely to occur is considered, and a solution which behaves well in this worst case, is sought. In other words, the 'best worst-case performance' solution should be found. This approach is appropriate when the decision-maker is very risk averse or the situation risk is high [5].

Evolutionary algorithms have proven successful in finding robust solutions using the probabilistic robustness evaluation approach [11,15]. In these methods, in the fitness evaluation stage, the expected performance of each solution is considered as

* Corresponding author.
E-mail addresses: kaedi@eng.ui.ac.ir (M. Kaedi), cwan@gist.ac.kr (C.W. Ahn).

http://dx.doi.org/10.1016/j.asoc.2017.03.042
1568-4946/

the solution's fitness. In probabilistic robustness evaluation, the expected fitness of solution x is defined as follows [11]:

$$f_{\exp}(X) = \int_{-\infty}^{\infty} f(X+\delta).p(\delta)\ \ d\delta \tag{1}$$

where X denotes a design vector (i.e., a solution) of dimension d, $f(X)$ is the fitness of that solution, and δ is a disturbance that is distributed according to the probability density function $p(\delta)$.

Unfortunately, f is not available for many of the complex problems; therefore, Eq. (1) cannot be analytically computed for them [11]. Alternatively, f_{exp} should be estimated. The most common approach for estimation of f_{exp} is Monte Carlo integration (e.g., as used in [13,16,17]). To perform this estimation, a number of points in the neighborhood of solution x should be randomly sampled (i.e., sampling over a number of realizations of δ) [11,13]. This approach, known as the explicit averaging, requires a large number of additional fitness evaluations because each sample corresponds to one fitness evaluation [1]. Therefore, if the fitness evaluation is expensive, this approach is slow and impractical [11,13,18]. A number of methods have been proposed in the literature to reduce the number of fitness evaluations in the explicit averaging approach. These methods are reviewed in Section 2.

In this paper, we present a new method to reduce the number of fitness evaluations in the explicit averaging approach in evolutionary algorithms. In our proposed method, we present a heuristic method to detect the non-robust solutions, specifically the solutions that seem to be sensitive to changes in design variables. Once these non-robust solutions are detected, they are discarded from the process of finding their expected performance because they are not promising as the final solution. The expected fitness of the remaining solutions is then calculated through the explicit averaging, and the robust solutions are selected to be reproduced in the next generation.

Among all evolutionary algorithms, we focus on the Bayesian optimization algorithm (BOA) in our proposed method. There are two reasons in selecting BOA. For one thing, BOA is an estimation of distribution algorithm (EDA), which applies Bayesian networks to evolve the solution population. In BOA, the Bayesian network contains abstract knowledge about the problem's solutions. This knowledge can be useful for detecting sensitive solutions. Second, to the best of our knowledge, no other research has been accomplished on applying BOA to find robust solutions. Of course, most of the methods presented for finding robust solutions through genetic algorithms can be applied to BOA. However, no method has been presented to apply the specific BOA features to find the robust solutions.

The rest of this paper is organized as follows. In Section 2, previous works related to computational time reduction of the explicit averaging method are reviewed. In Section 3, the BOA is described. Our proposed method for reducing the computational time of the explicit averaging method in BOA is presented in Section 4. In Section 5, our proposed method is evaluated and the experimental results are reported. Conclusions are presented in Section 6.

2. Related works

Averaging over samples in the neighborhood of a solution (i.e., explicit averaging) is a common approach that has been used for evaluating the robustness degree of solutions (e.g., [16,17,19–21]). However, the explicit averaging has the drawback of requiring a large number of fitness evaluations, which leads to increasing the computational time. Several methods have been presented so far to decrease the computational time needed for fitness evaluations in the explicit averaging. These methods have been classified in three groups [5,11]. In this section, we review these works.

2.1. Reducing the number of neighbors that should be sampled and evaluated

In the first group of methods, some sophisticated sampling techniques are used to generate some samples in the neighborhood of a solution. In this way, fewer but targeted samples can be produced and evaluated in the neighborhood of each solution. Therefore, the time needed for approximating the expected fitness of each solution can be reduced. In [22,23], the chance-constrained genetic algorithm is presented. It employs Latin hypercube sampling (LHS) to produce the neighbors of each solution. Compared to standard Monte Carlo sampling, using LHS was shown to result in the use of smaller sample sizes. Being inspired by this research, other smart sampling methods have been presented based on LHS to reduce the number of neighbors that should be sampled [24,25].

As another idea, in [1], the polynomial chaos that is a polynomial representation of a Gaussian random process has been used for sampling the solutions.

2.2. Reducing the number of solutions that should be sampled and evaluated

In this group of methods, the solutions that appear more promising are detected. The focus then primarily becomes finding the expected fitness of these promising solutions; therefore, by approximating the expected fitness of only the promising solutions, the time in finding the robust solution is reduced. Detection of the promising solutions (i.e., the solutions that seem to be robust) can be accomplished on the basis of criteria such as solution fitness [13] or solution fitness variance [26].

2.3. Accelerating the fitness evaluation of each solution

Acceleration of the fitness evaluation of solutions can be achieved by two approaches. One is to avoid the redundant evaluation of the repetitive solutions. For example, the former solutions and their evaluations can be stored in memory and then applied for estimating the evaluation of the current solutions [26]. The second approach is to construct the local approximation model for the fitness function and estimate the expected fitness of solutions [11,27–31]. This approximation model is called meta-model or surrogate [27].■

In some researches, a hybrid method is proposed by combining the three mentioned approaches. For example, in [25] an archive of the evaluated solutions has been used to accelerate the fitness evaluation, the full neighborhood of only the high fitness solutions have been sampled, and LHS has been used as the sampling method.

Our proposed work places in the second approach. For evolutionary robust optimization, BOA was selected as a successful evolutionary algorithm. In our method, we strive to reduce the number of solutions that should be evaluated when we want to find the robust solution via BOA.

3. Bayesian optimization algorithm

BOA evolves a population of candidate solutions by constructing Bayesian networks and sampling them [32,33]. A Bayesian network is a directed acyclic graph (DAG) that represents probabilistic relationships among a set of random variables. In this graph, each node is related to a variable and the edges correspond to conditional dependencies. A set of conditional probability tables is used for presenting the conditional dependencies of variables [34].

BOA randomly constructs the initial population with a uniform distribution for all possible solutions. The population is then updated in several iterations. Each iteration consists of five stages