Engineering Applications of Artificial Intelligence 25 (2012) 1009–1021

Contents lists available at SciVerse ScienceDirect

Engineering Applications of Artificial Intelligence

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

A computational intelligence algorithm for expensive engineering optimization problems

Yoel Tenne[1]

ARTICLE INFO

Article history:
Received 11 January 2011
Received in revised form
15 July 2011
Accepted 16 March 2012
Available online 10 May 2012

Keywords:
Expensive optimization problems
Evolutionary algorithms
Model-selection modeling
Classification

ABSTRACT

The modern engineering design optimization process often replaces laboratory experiments with computer simulations, which leads to *expensive black-box optimization problems*. Such problems often contain candidate solutions which cause the simulation to fail, and therefore they will have no objective value assigned to them, a scenario which degrades the search effectiveness. To address this, this paper proposes a new computational intelligence optimization algorithm which incorporates a classifier into the optimization search. The classifier predicts which solutions are expected to cause a simulation failure, and its prediction is used to bias the search towards solutions for which the simulation is expected to succeed. To further enhance the search effectiveness, the proposed algorithm continuously adapts during the search the type of model and classifier being used. A rigorous performance analysis using a representative application of airfoil shape optimization shows that the proposed algorithm outperformed existing approaches in terms of the final result obtained, and performed a search with a competitively low number of failed evaluations. Analysis also highlights the contribution of incorporating the classifier into the search, and of the model and classifier selection steps.

1. Introduction

In the modern design optimization process, researchers replace laboratory experiments with *computer simulations* to reduce the duration and cost of the design process. Such a simulation-driven setup transforms the design process into an optimization problem having three distinct features (Tenne et al., 2010a):

- The simulation acts as an objective function, assigning a candidate design its corresponding objective value. However, the simulation is often a legacy code or a commercial software available only as an executable, and so there is no analytic expression defining how candidate designs are mapped to their corresponding objective values. Such a *black-box function* precludes the use of optimizers which require an analytic function.
- Each simulation run is *computationally expensive*, that is, it requires anywhere from minutes to weeks of CPU time, and this severely restricts the number of permissible evaluations.
- Both the real-world physics being modeled, and the numerical simulation process, may yield an objective function which has a complicated, multimodal landscape.

Accordingly, such optimization scenarios are commonly termed as *expensive black-box optimization problems*, and a diverse range of optimization algorithms have been proposed to effectively address the challenges above (Tenne et al., 2010a).

However, expensive optimization problems introduce another difficulty, namely, the simulation may 'crash' and fail to return an objective value for some candidate designs. We refer to such designs as *simulator-infeasible* (SI), while those for which the simulation completes successfully and provides the objective value are termed *simulator-feasilbe (SF)*. SI designs have two main implications on the optimization search:

- Since they do not have a corresponding objective value, the objective function becomes discontinuous, which exacerbates the difficulty of the optimization, and
- Such designs can consume a large share of the limited computational budget, without providing any objective values, and can thus lead to search stagnation and a poor final result.

A literature survey, for example, as indicated by Poloni et al. (2000), Jin et al. (2002), and Büche et al. (2005), to name a few, shows that SI candidate designs are common in real-world optimization problems, and so it is important to effectively handle them. Existing strategies either discard such vectors altogether, or assign them a penalized objective value and then incorporate them into the model. However, both these strategies have significant demerits, for example, they discard information which can be beneficial to the search, or they result in a model with a severely deformed landscape.

To address these issues, we propose in this study a new computational intelligence algorithm which incorporates a classifier into

[1] Formerly with the Faculty of Science and Engineering, Kyoto University, Japan.
E-mail address: y.tenne1@yahoo.com

0952-1976/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.engappai.2012.03.009

the optimization search. The role of the classifier is to predict if a candidate design is SI or not, and the proposed algorithm leverages on this prediction to bias the search to designs predicted to be SF. However, the effectiveness of the algorithm strongly depends on the type of the model and classifier used, but the limited computational budget implies that it is impractical to experiment with different models and classifiers. As such, the proposed algorithm also incorporates rigorous statistical methods to select an optimal model and classifier from a family of candidates. Both the model *and* the classifier types are *continuously* adjusted during the search to maximize the search effectiveness. To the best of our knowledge, such a computational intelligence algorithm, combining both a classifier and a model, and the ability to adapt both of them, is new. We evaluate the proposed algorithm using a real-world application of airfoil shape optimization. A rigorous analysis shows that the proposed algorithm is both effective, namely it outperforms existing approaches in terms of the final result obtained, and is efficient, that is, it performs a search with a competitively low number of SI vectors, so it minimizes the computer resources wasted on failed evaluations. Analysis also highlights the contribution of incorporating the classifier, and of the model and classifier selection during the search.

The remainder of this paper is as follows: Section 2 provides the pertinent background information, Section 3 describes in detail the proposed algorithm, and Section 4 provides a rigorous performance analysis. Lastly, Section 5 concludes this paper.

2. Background

2.1. Expensive optimization problems and computational intelligence algorithms

Expensive optimization problems arise in diverse domains across engineering and science, and as mentioned in Section 1, a typical scenario is engineering design optimization problems driven by computer simulations. Fig. 1 shows the layout of such problems, where the simulation effectively acts as a black-box function, namely it assigns objective values to candidate designs, while the analytic expression of this function is unknown. In such optimization problems, candidate designs are represented as vectors of design variables and are provided as inputs to the simulation.

Also as mentioned, the resultant objective function often has a complicated, multimodal landscape, a scenario in which gradient-based optimizers may converge to a poor local optimum. This has motivated using computational intelligence (CI) optimizers in such problems, as they tend to be more explorative and hence are typically more resilient to multimodal landscapes. Such optimizers typically employ a population of candidate solutions and manipulate the latter both deterministically and stochastically. One such CI optimizer which is widely used is the EA, which

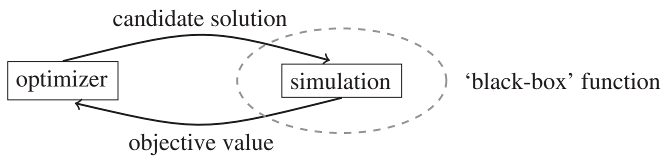

Fig. 1. The layout of an expensive black-box optimization problem. The optimization algorithm generates candidate solutions, and these are evaluated by the simulation to obtain their corresponding objective values. The optimizer treats the simulation as a black-box function, that is, having no analytic expression.

typically applies the following three operators inspired by the theory of evolution (Tenne et al., 2010a):

- *Selection*: The vectors with the best objective value are selected as *parents*.
- *Recombination*: Two parents are selected, typically at random, and their vectors are combined to yield an offspring. This is repeated several times to generate a population of offspring.
- *Mutation*: Offspring are selected at random, and some of their vector components are randomly changed.

The offspring population is then evaluated, and the fittest members, namely those with the best objective values, are taken to be the population of the next generation. The process then repeats until some termination criterion is met, such as no improvement in the best function value, or if the maximum number of generations has been reached. The EA emphasizes the fittest solutions, and this leads to a gradual adaptation of the population to the function landscape and convergence to an optimum. Algorithm 1 gives a pseudocode of a baseline EA.

Algorithm 1. A baseline evolutionary algorithm.

```
initialize a population of solutions;
evaluate each solution in the population;
/* main loop
repeat
    select a group of solutions and designate them as parents;
    recombine the parents to create a population of offspring;
    mutate some of the offspring;
    evaluate the offspring;
    select the best solutions as the population
    of the next generation;
until convergence, or maximum number of generations reached
```

Since CI optimizers rely on a population of solutions, and do not utilize gradient information, they often require many thousands of function evaluations to obtain a satisfactory solution. This is a major obstacle in applying them to expensive optimization problems, where the number of function evaluations is severely restricted. An established strategy to circumvent this issue is to employ a model which serves as a computationally cheaper approximation of the true expensive function. Models are typically interpolants trained with previously evaluated vectors, and variants include artificial neural networks (ANN), Kriging, polynomials, and RBF (Simpson et al., 2001). However, while models alleviate the bottleneck of expensive evaluations, they introduce several new challenges:

- *Model management*: Due to the small number of permissible expensive function evaluations, only a small number of vectors will be available to train the model, which results in an inaccurate model. This can hamper the optimization search, and if the model accuracy is poor, the optimizer may even converge to a false optimum, namely an optimum of the model which is a not an optimum of the true expensive function (Jin et al., 2002). Accordingly, it is necessary to *manage* the model and ensure its accuracy during the search. To accomplish this, the proposed algorithm leverages on the TR approach which originated in the field of nonlinear programming (Conn et al., 1997), and in which the optimization proceeds by a sequence of *trial steps*, where each is constrained to the TR, namely the region where the model is assumed to be accurate. Based on the success of the trial step, that is, whether a better solution has been found or not, the TR and model are updated, and the sequence repeats until some termination condition is met. A strong merit of the TR approach is that it ensures asymptotic