Applied Soft Computing 18 (2014) 185–195

Contents lists available at ScienceDirect

Applied Soft Computing

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

Automatic parameter tuning for Evolutionary Algorithms using a Bayesian Case-Based Reasoning system

E. Yeguas [a,*], M.V. Luzón [b], R. Pavón [c], R. Laza [c], G. Arroyo [b], F. Díaz [d]

[a] Department of Computing and Numerical Analysis, Maimonides Institute for Biomedical Research (IMIBIC), University of Córdoba, Córdoba, Spain
[b] Department of Computer Science, University of Granada, Granada, Spain
[c] Department of Computer Science, University of Vigo, Ourense, Spain
[d] Department of Computer Science, University of Valladolid, Segovia, Spain

ARTICLE INFO

Article history:
Received 27 February 2012
Received in revised form 18 June 2012
Accepted 18 January 2014
Available online 30 January 2014

Keywords:
Parameter tuning
Case-Based Reasoning
Bayesian Networks
Evolutionary Algorithms
Geometric constraint solving

ABSTRACT

The widespread use and applicability of Evolutionary Algorithms is due in part to the ability to adapt them to a particular problem-solving context by tuning their parameters. This is one of the problems that a user faces when applying an Evolutionary Algorithm to solve a given problem. Before running the algorithm, the user typically has to specify values for a number of parameters, such as population size, selection rate, and probability operators.

This paper empirically assesses the performance of an automatic parameter tuning system in order to avoid the problems of time requirements and the interaction of parameters. The system, based on Bayesian Networks and Case-Based Reasoning methodology, estimates the best parameter setting for maximizing the performance of Evolutionary Algorithms. The algorithms are applied to solve a basic problem in constraint-based, geometric parametric modeling, as an instance of general constraint-satisfaction problems.

The experimental results demonstrate the validity of the proposed system and its potential effectiveness for configuring algorithms.

1. Introduction

One of the major issues in applying metaheuristics and, in particular, an Evolutionary Algorithm (EA) [1], is how to choose an appropriate parameter setting. High computational requirements are needed to adjust the control parameters of the algorithm to improve its performance on a particular problem. Numerous studies focused on developing methods for automatically finding the best set of parameter values have been performed in [2] and more recently in [3].

Geometric constraint solving, as an instance of general constraint-satisfaction problems, is a basic problem in constraint-based, geometric parametric modeling in the field of Computer-Aided Design [4]. Geometric problems defined by constraints have an exponential number of solution instances. Generally, the user is only interested in one instance that fulfills the geometric constraints. This solution instance is called *the intended solution*. Selecting a solution instance amounts to selecting one among a number of different roots of a nonlinear equation or system of equations. The problem of selecting a given root has been named in [5] the *Root Identification Problem* (RIP).

In previous work [6,7] EAs have been shown as suitable techniques for solving the RIP, but the success and performance of EAs depends crucially on finding suitable parameter settings. Given the size of a problem and an EA, specific values must be assigned to the set of parameters which determine the evolution of the algorithm. Doing this by hand is a very time consuming process that does not ultimately guarantee finding satisfactory parameters. Users of EAs mostly rely on conventions, *ad hoc* choices, and experimental comparisons on a limited scale. Parameter settings are commonly chosen in practice by *trial and error*, tuned by hand [8], taken from other fields, by parameter tuning [9] or by adaptation and self-adaptation mechanisms (parameter control) [2,10,11].

Given the great variety of possible approaches, an alternative solution would be to design a system which automatically provides the correct values of the parameters to control the execution of a particular algorithm when applied to a specific problem. Such a system, referred to as *Bayesian Case-Based Reasoning* (CBR), follows the approach proposed in [12,13] and empirically evaluated to set a basic genetic algorithm [14] that solves the RIP in [15]. Although good results were obtained when the *Bayesian CBR system* was applied, it would be useful to adapt and study the behavior of the

* Corresponding author. Tel.: +34 957212289.
E-mail addresses: eyeguas@uco.es (E. Yeguas), luzon@ugr.es (M.V. Luzón), pavon@uvigo.es (R. Pavón), rlaza@uvigo.es (R. Laza), arroyo@ugr.es (G. Arroyo), fdiaz@infor.uva.es (F. Díaz).

1568-4946/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2014.01.032

system with the statistically most promising EAs in a given environment. In particular, recent studies on the applicability of a number of metaheuristics to the RIP have determined the algorithms that statistically obtain the best performance [16]. Such studies have originated a great deal of data for adapting, applying and evaluating the previously proposed *Bayesian CBR system* and comparing the results obtained.

The aim of this paper is twofold. First, carrying out an adaptation and a new empirical evaluation of the *Bayesian CBR system* when it is used to configure EAs, in general, that solve the RIP. Second, a comparison is made between the proposed approach and the method which obtains the statistically best parameter configuration known [16,17].

The paper is structured as follows. Section 2 briefly describes some preliminary aspects about parameter tuning. Section 3 presents the main issues about geometric constraint solving systems and the RIP. Section 4 demonstrates the profiling of the generic *Bayesian CBR system* according to the specific application domain, presents the experiments carried out and compares the results to the statistically best parameter configuration known. Finally, in Section 5, some conclusions and future work are discussed.

2. Parameter tuning

EAs comprise a wide class of solution methods that have been successfully applied to many optimization problems. The assessment of these methods is commonly based on experimental analysis but the lack of a methodology in these analyses limits the scientific value of their results. Despite the lack of theoretical foundation, the simplicity of EAs has attracted many researchers and practitioners. Many results in the literature indicate that metaheuristics, and in particular EAs, are state-of-the-art techniques for problems for which there is no efficient algorithm [18,19].

However, EAs do not always reach an optimal solution, even for long computation times. In addition, it is often difficult to obtain an analytical prediction of either the achievable solution quality within a given computation time or the time taken to find a solution of a given quality. The assessment of these conflicting performance measures is critical for the evaluation of EAs and their application to real problems. Given the lack of theoretical guidelines and the stochastic nature of most metaheuristics, such performance assessments are best carried out by experimentation over representative benchmarks [20–22].

In a parameter tuning framework, the experimental design consists of a set of techniques which comprise methodologies for adjusting the parameters of the algorithms depending on the settings used and results obtained [23,24]. A brief survey on these parameter tuning techniques is contained in [25]. The Design of Experiment (DoE) paradigm offers a way of retrieving optimal parameter settings. It is still a tedious task, but it is known to be a robust and well tested suite, which can be beneficial for giving reason to parameter choices besides human experience.

Finding a good set of parameter values is a complex optimization task with a nonlinear objective function, interacting variables, multiple local optima, noise (by the stochastic nature of the EA to be tuned), and a lack of analytic solvers. Ironically, it is precisely in these types of problems that EAs are very competitive heuristic solvers. It is therefore logical to use an evolutionary approach to optimize the parameters of an EA. The approaches following such idea are known as belonging to algorithmic parameter tuning, which even comprises search methods specifically designed for parameter tuning and parameter analysis, e.g., Sequential Parameter Optimization [26] or ParamILS, a local search approach for algorithm configuration [27]. Besides the principal parameter tuning algorithms, there are a number of useful 'add-ons', i.e., methods for increasing search efficiency, that are independent from the main tuner and can be combined with different tuning algorithms: racing, sharpening, etc. [9,28].

The main problems in parameter tuning are, amongst others, that mistakes in settings by users can be sources of errors or suboptimal performance, that it is usually very time consuming or that good values may become bad during the run [2]. Algorithm configuration is commonly (either implicitly or explicitly) treated as an optimization problem, where the objective function captures performance on a fixed set of benchmark instances.

In [12,13] a *Bayesian CBR system* was introduced as a general framework for solving the parameter tuning problem, conditioned by the current problem instance to be solved. The system estimates the best parameter configuration for maximizing the algorithm performance when it solves an instance-specific domain problem. The proposed system follows the CBR methodology [29–34]. CBR systems have had considerable success in a wide variety of problem solving tasks and domains, including parameter tuning [35,36]. Other characteristics that have made CBR a good solution for the tuning problem are:

- CBR is preferable when it is difficult to give explicit laws of behavior, but easy to give examples. In setting parameters for EAs, a number of past examples of good solutions usually exists.
- CBR is suitable when a completely accurate solution to the problem does not exist. Generally, in tuning an EA, a parameter configuration only works well with a subset of all instances.
- CBR is suitable when problems or cases tend to repeat themselves. Problem instances are very similar in most of the cases.
- Finally, CBR is applicable when it is easy to construct new cases from the solution of new problems. It is sufficient to run the EA with the suggested configuration for it to perform and create a new case.

Moreover, CBR has two interesting properties related to the parameter tuning problem: its *learning* capacity, that allows the system to adapt itself to changes and its capacity for *autonomy*, that allows the system to assist the user completely.

The design of the *Bayesian CBR system* has been carried out in two phases. The first one makes use of Bayesian Networks (BNs) [37–40] to model qualitative and quantitative relationships among the different algorithm parameters of interest to the user. The second phase integrates BNs within a CBR system to solve new domain problem instances taking into account relevant features and past experience of similar instances. A detailed description of the basic system can be found in [12,13].

3. Constraint-based geometric design

Computer-Aided Design systems were intended to present intelligent assistants to designers, but conventional geometric modelers were and are actually used only for activities that occur near the end of the design process, when analyzing and presenting the results. The computer program merely substitutes the drawing board, and does not support the designer in the time-consuming and error-prone calculations of coordinates and dimensions, or in different decisions that must be made. The program 'thinks' in a procedural way, but the declarative description of geometry is much closer to a human [41,42]. For this reason, the designer meets the requirement for numerous additional calculations by extending the design time and often presenting sources of errors. A new generation of geometric modelers offers the solution to this problem. The geometry can be described by defining relations (constraints) among particular geometric elements.