European Journal of Operational Research 235 (2014) 569–582

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Automatically improving the anytime behaviour of optimisation algorithms

Manuel López-Ibáñez *, Thomas Stützle

IRIDIA, Université Libre de Bruxelles (ULB), CP 194/6, Av. F. Roosevelt 50, B-1050 Brussels, Belgium

ARTICLE INFO

Article history:
Received 16 May 2012
Accepted 17 October 2013
Available online 28 October 2013

Keywords:
Metaheuristics
Anytime algorithms
Automatic configuration
Offline tuning

ABSTRACT

Optimisation algorithms with good anytime behaviour try to return as high-quality solutions as possible independently of the computation time allowed. Designing algorithms with good anytime behaviour is a difficult task, because performance is often evaluated subjectively, by plotting the trade-off curve between computation time and solution quality. Yet, the trade-off curve may be modelled also as a set of mutually nondominated, bi-objective points. Using this model, we propose to combine an automatic configuration tool and the hypervolume measure, which assigns a single quality measure to a nondominated set. This allows us to improve the anytime behaviour of optimisation algorithms by means of automatically finding algorithmic configurations that produce the best nondominated sets. Moreover, the recently proposed weighted hypervolume measure is used here to incorporate the decision-maker's preferences into the automatic tuning procedure. We report on the improvements reached when applying the proposed method to two relevant scenarios: (i) the design of parameter variation strategies for MAX-MIN Ant System and (ii) the tuning of the anytime behaviour of SCIP, an open-source mixed integer programming solver with more than 200 parameters.

1. Introduction

Many optimisation algorithms are designed without a specific termination criterion, and generate a sequence of feasible solutions that are increasingly better approximations of the optimal solution. However, the performance of an algorithm is often crucially determined by the choice of the termination criterion and the parameters of the algorithm. If the parameter settings of an algorithm result in fast convergence to good solutions, this may prevent the algorithm from adequately exploring the search space to find better solutions if given ample time. On the other hand, parameter settings that give higher exploration capabilities may produce poor results if the termination criterion is too short. Hence, there is a trade-off between solution quality and the runtime of the algorithm that can be adjusted by appropriately setting the parameters of the algorithm.

In many practical scenarios, an optimisation algorithm may be terminated at an arbitrary time, and, upon termination, the algorithm returns the best solution found since the start of the run. In such scenarios, the termination criterion is not known in advance, and, hence, the algorithm should produce as high quality solutions as possible at any moment of its run time. Algorithms that show a better trade-off between solution quality and runtime are said to have a better *anytime behaviour* (Zilberstein, 1996).

There are two classical views when analysing the anytime behaviour (Hoos & Stützle, 2005). One view defines a number of termination criteria and analyses the quality achieved by the algorithm at each termination criterion. In this quality-over-time view, the anytime behaviour can be analysed as a series of plots of time-dependent solution quality distributions. A different view defines a number of target quality values and analyses the time required by the algorithm to reach each target. In this time-over-quality view, algorithms are often analysed in terms of a series of qualified runtime distributions.

In this paper, we consider a third view that does not favour time over quality or vice versa. Instead, this third view models the performance profile of an algorithm as a nondominated set in a multi-objective space. An algorithm has better anytime behaviour when it produces better nondominated sets, where "better" means better in terms of Pareto optimality. Surprisingly, this third view has received little attention (Hoos & Stützle, 2005; Chiarandini, 2005; den Besten, 2004), despite the important advances in theory and practice achieved in performance assessment of multi-objective optimisers in the last decade. Essentially, this model allows us to apply the same unary quality measures used in multi-objective optimisation to assign a single numerical value to the anytime behaviour of an algorithm's run. In this paper, we use the hypervolume measure as the unary quality measure for this purpose. The main reason is that the hypervolume is the quality measure with

* Corresponding author. Tel.: +32 (0) 2650 2745.
E-mail addresses: manuel.lopez-ibanez@ulb.ac.be (M. López-Ibáñez), stuetzle@ulb.ac.be (T. Stützle).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.10.043

the highest discriminatory power among the known unary quality measures (Zitzler, Thiele, Laumanns, Fonseca, & Grunert da Fonseca, 2003). In addition, recent work has made possible to describe user preferences in terms of a weighted hypervolume measure (Auger, Bader, Brockhoff, & Zitzler, 2009), and, hence, our proposal allows incorporating user preferences when analysing the anytime behaviour of an algorithm. Moreover, as shown in this paper, evaluating the anytime behaviour of an algorithm in terms of the hypervolume allows applying automatic algorithm configuration methods to find parameter settings of an algorithm that optimise the trade-off between quality and time.

Recent advances in automatic configuration of algorithms (also called offline parameter tuning) have shown that such methods can save a significant amount of human effort and improve the performance of optimisation algorithms, when designing and evaluating new algorithms and when tuning existing algorithms to specific problems (Birattari, 2009; Bartz-Beielstein, 2006; Hutter, Hoos, Leyton-Brown, & Stützle, 2009; Eiben & Smit, 2011; Hutter, Hoos, & Leyton-Brown, 2011; Hoos, 2012). Our proposal here is to combine automatic configuration with the use of the hypervolume as a surrogate measure of anytime behaviour in order to enable the automatic configuration of algorithms in terms of anytime behaviour.

In the scenario described above, where the algorithm does not know its termination criterion in advance, techniques such as parameter adaptation are often applied to improve the anytime behaviour of the algorithm (Eiben, Michalewicz, Schoenauer, & Smith, 2007; Aine, Kumar, & Chakrabarti, 2009; Stützle et al., 2012). However, designing such parameter adaptation strategies is an arduous task, and they usually add new parameters to the algorithm that need to be tuned. The method proposed in this paper will help algorithm designers to compare and fine-tune such parameter adaptation strategies to find the settings that improve the anytime behaviour of the algorithm on the problem at hand. Indeed, the first case study reported here derives from our own efforts on designing parameter adaptation strategies for ant colony optimisation algorithms. This experience motivated us to develop the method proposed here, since the classical trial-and-error approach for designing such strategies proved extremely time-consuming.

The second case study reported here deals with a different scenario, in particular, a general purpose black-box solver (SCIP (Achterberg, 2009)) with a large number of parameters. The default parameter settings of such solvers are tuned for solving problem instances to optimality as fast as possible. However, in some practical scenarios, users may not want to wait until a problem instance is solved to optimality, and may decide to stop the solver at an arbitrary time. Using our method for fine-tuning the parameters of the solver with respect to anytime behaviour, users can improve the quality of the solutions found when the solver is stopped before reaching optimality, without knowing in advance the particular termination criterion.

The outline of the paper is as follows. Section 2 provides a background on automatic algorithm configuration, summarises the state of the art and describes the automatic configuration method (`irace`) used throughout this paper. Section 3 introduces the two classical views of the analysis of anytime algorithms and the less-explored multi-objective view. In Section 4, we describe our proposal in detail. We explain the benefits of using the hypervolume to evaluate the anytime behaviour of an algorithm in the context of an automatic configuration method. We discuss the choice of reference point and how to combine `irace` with the hypervolume measure. An additional section summarises related work and highlights the differences with our proposed approach. Section 5 describes our first case study, where we apply this proposal to the design of parameter adaptation strategies for MMAS. Section 6 discusses how our proposal enables a decision maker to incorporate preferences regarding the anytime behaviour of an algorithm to the automatic configuration procedure. A second case study is considered in Section 7, where we tune the anytime behaviour of SCIP. Finally, Section 8 provides a summary of our results and discusses possible extensions of the present work.

2. Preliminaries: automatic algorithm configuration

This section is a brief introduction to automatic algorithm configuration. We define the algorithm configuration problem, give an overview on the state of the art of automatic configuration methods, and describe `irace`, the automatic configuration method used throughout this paper. A more detailed and formal introduction is available from the literature referenced here and in the extended version of the paper (López-Ibáñez & Stützle, 2012a).

2.1. The algorithm configuration problem

Most algorithms for computationally hard optimisation problems have a number of parameters that need to be set. As an example, ACO algorithms (Dorigo & Stützle, 2004) often require the user to specify not only numerical parameters like the evaporation factor and the number of ants, but also components like the type of heuristic information and update method. Another example is mixed-integer programming solvers, such as SCIP (Achterberg, 2009), which often have a large number of configurable parameters affecting the main algorithm used internally, e.g., selecting among different branching strategies. The process of designing complex algorithms from a framework of algorithm components can be seen as an algorithm configuration problem (KhudaBukhsh, Xu, Hoos, & Leyton-Brown, 2009; Montes de Oca, Stützle, Birattari, & Dorigo, 2009; López-Ibáñez & Stützle, 2012c).

Given a parametrised algorithm, where each parameter may take different values (settings), a configuration of the algorithm is a unique assignment of values to parameters. When considering a problem to be solved by this parametrised algorithm, the goal of automatic configuration is to find the configuration that minimises a particular cost function over the set of possible instances of the problem. The cost function assigns a value to each configuration when applied to a single problem instance. In the case of stochastic algorithms, this cost measure is a random variable. Since most algorithms and problems of practical interest are sufficiently complex to preclude an analytical approach, the configuration of such algorithms follows an experimental approach (Birattari, 2009; Bartz-Beielstein, 2006).

2.2. Automatic configuration methods

The traditional approach to algorithm configuration consists of ad hoc experiments testing relatively few configurations. The use of experimental design techniques (Coy, Golden, Runger, & Wasil, 2001; Adenso-Díaz & Laguna, 2006) began a trend in which the task of finding the most promising configurations to be tested is performed automatically. The natural evolution of this trend has been to tackle algorithm configuration as an optimisation problem (Nannen & Eiben, 2006; Ansótegui, Sellmann, & Tierney, 2009; Hutter et al., 2009; Bartz-Beielstein, 2006; Bartz-Beielstein, Lasarczyk, & Preuss, 2010; Hutter et al., 2011). It is becoming widely accepted that automatic configuration methods may save substantial human effort during the empirical analysis and design of optimisation algorithms, and, at the same time, lead to better algorithms (Hoos, 2012; Eiben & Smit, 2011; Bartz-Beielstein, 2006; Birattari, 2009).