Computers & Operations Research 39 (2012) 875–889

Contents lists available at ScienceDirect

Computers & Operations Research

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

Review

Measuring instance difficulty for combinatorial optimization problems

Kate Smith-Miles *, Leo Lopes

School of Mathematical Sciences, Monash University, Victoria 3800, Australia

ARTICLE INFO

Available online 12 July 2011

Keywords:
Algorithm selection
Combinatorial optimization
Hardness prediction
Instance difficulty
Landscape analysis
Phase transition
Traveling salesman problem
Assignment problem
Knapsack problem
Bin-packing
Graph coloring
Timetabling

ABSTRACT

Discovering the conditions under which an optimization algorithm or search heuristic will succeed or fail is critical for understanding the strengths and weaknesses of different algorithms, and for automated algorithm selection. Large scale experimental studies – studying the performance of a variety of optimization algorithms across a large collection of diverse problem instances – provide the resources to derive these conditions. Data mining techniques can be used to learn the relationships between the critical features of the instances and the performance of algorithms. This paper discusses how we can adequately characterize the features of a problem instance that have impact on difficulty in terms of algorithmic performance, and how such features can be defined and measured for various optimization problems. We provide a comprehensive survey of the research field with a focus on six combinatorial optimization problems: assignment, traveling salesman, and knapsack problems, bin-packing, graph coloring, and timetabling. For these problems – which are important abstractions of many real-world problems – we review hardness-revealing features as developed over decades of research, and we discuss the suitability of more problem-independent landscape metrics. We discuss how the features developed for one problem may be transferred to study related problems exhibiting similar structures.

Contents

1. Introduction

For many decades, researchers have been developing ever-more sophisticated algorithms for solving hard optimization problems. These algorithms include mathematical programming approaches, constraint programming, and many heuristics including meta-heuristics and nature-inspired heuristics. Experimental

* Corresponding author.
E-mail addresses: kate.smith-miles@monash.edu (K. Smith-Miles), leo.lopes@monash.edu (L. Lopes).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.07.006

studies have been conducted to determine which algorithms perform best, usually based on publicly available collections of benchmark datasets. The conclusions from these comparisons are often not insightful [69], limited by the scale of the studies which typically restrict either the type or quantity of benchmark problem instances used, or consider only a small number of algorithms [12]. The no-free-lunch (NFL) theorems [153] tell us that there does not exist a single algorithm that can be expected to outperform all other algorithms on all problem instances. If a study demonstrates the superiority of one algorithm over a set of other algorithms, then one may claim that there are probably untested problem instances where we could expect this algorithm to be outperformed. A description of the conditions under which an algorithm can be expected to succeed or fail is rarely included in the study [41].

The true value of good experimental studies lies in their ability to answer two key questions. The first question is: *which algorithm in a* (*broad*) *portfolio is likely to be best for a relevant set of problem instances*? Useful are studies where diverse algorithms are compared across *enough* instances (making statistical conclusions valid), with the *types* of instances matched to the interests of the study (e.g. real-world instances, or intentionally challenging instances [88]). A good experimental study can uncover relationships between features of instances and algorithmic performance. The outcome can be an automated algorithm selection model predicting the algorithm from a given portfolio that is likely to be best for a given instance. The second question that can potentially be addressed by good experimental studies is a more general and far-reaching one: *for which types of problem instances can we expect a given algorithm in a portfolio to perform well*, *and why*? Answers to this second question hold the key to understanding the strengths and weaknesses of algorithms, and have implications for improved algorithm design.

These questions have been raised by various research communities. In the meta-heuristics community statements such as "currently there is still a strong lack of understanding of how exactly the relative performance of different meta-heuristics depends on instance characteristics" [134] have highlighted the need to measure key characteristics of optimization problems and explore their relationship with algorithm behavior. In the artificial intelligence community, a similar concept has lead to the development of algorithm portfolios, whereby knowledge of the relationship between instance characteristics and algorithm performance based on training data is used to build a regression model to predict which algorithm is likely to perform best for a new problem instance [59,85]. This approach of selecting the likely best algorithm from a portfolio after gaining knowledge into that relationship has been most successful, winning the 2007 SAT (constraint satisfaction) competition [155]. There have also been extensions of these ideas beyond static or off-line algorithm selection to reactive search [14] and racing algorithms [19,91], where knowledge of the characteristics or features of the search space is exploited to fine-tune or re-select an algorithm during run-time [51,119,133].

A key challenge with all of these approaches is to adequately characterize the problem instance search space by devising suitable measures. In order for any useful knowledge to be learned from modeling the relationships between problem instance characteristics and algorithm performance we need to ensure that we are measuring features of the problem instances that are revealing of the relative hardness of each problem instance as well as revealing of the strengths and weaknesses of the various algorithms.

So how can we determine if an optimization problem, or an instance, is hard or challenging for a particular algorithm? And what are the characteristics or features of the instance that present this challenge? The most straightforward features of an optimization problem instance are those that are defined by the sub-class of the instance: features like the number of variables and constraints, whether the matrices storing instance parameters are symmetric, etc. There are numerous candidate features that can be derived by computational feature extraction processes applied to instance parameters that often serve well as proxy measures for instance difficulty. We note here the distinction between the definition of a feature, and the suitable measurement of that feature.

Measuring hardness of an instance for a particular algorithm is typically done by comparing the optimization precision reached after a certain number of iterations compared to other algorithms, and/or by comparing the number of iterations taken to reach the best solution compared to other algorithms [154]. More sophisticated measures of hardness of a problem for a particular algorithm include measuring the fraction of the search space that corresponds to a better solution than the algorithm was able to find [89]. These performance metrics may enable us to determine if an algorithm struggles with a problem instance or solves it easily, and have been used to demonstrate that there are indeed classes of problems that are intrinsically harder than others for different algorithms [89]. However, they do not help us to explain why this might be the case.

To understand the challenging features or properties of the problem instance, there have been numerous efforts to characterize the objective function and search space, identifying challenges such as isolation, deception, multi-modality, and features such as the size of basins of attraction [10,61,87,154], as well as landscape metrics (reviewed in Section 3) based on analysis of autocorrelation structures and number and distributions of local minima [108,121]. Obviously these features can only be measured after an extensive analysis of the landscape, and are not suitable as inputs to a performance prediction model that seeks to answer our first question about which algorithm is likely to perform best for a given instance. They are useful for our second question—for gathering insights into the relationship between the structure of the problem and the performance of algorithms for the purposes of algorithm design and explaining performance. As a preliminary step for automated algorithm selection though, we need to ensure that the set of features used to characterize problem instances are quickly measurable.

Despite the importance of this key task, very little focus has been given in the literature as to how to construct features for characterizing a set of problem instances as a preliminary step for algorithm selection and performance modeling. As early as 1976, Rice posed the algorithm selection problem [110], defined as learning a mapping from feature space to algorithm performance space, and acknowledged the importance of selecting the right features to characterize the hardness of problem instances. For any optimization problem there is a variety of problem-specific metrics that could be used to expose the relative hardness of problem instances, as recent studies on phase transitions have shown [1,143]. In addition, we may wish to include metrics to expose the relative strengths (and weaknesses) of algorithms in the portfolio. Further, rules of guidance may be appropriate, in order to select the candidate features that are likely to be most useful for studying the difficulty of a given optimization problem.

This paper aims to provide a starting point for answering the critical question: how do we devise a suitable set of hardness-revealing features and/or metrics for an optimization problem? We tackle this question by first revisiting the framework for algorithm selection of Rice [110], presented in Section 2. We have recently used this framework to tackle our first question focused on automated algorithm selection for a number of optimization problems (traveling salesman [126,130], timetabling [129],