Computers & Operations Research 45 (2014) 12–24

Contents lists available at ScienceDirect

Computers & Operations Research

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

Towards objective measures of algorithm performance across instance space

Kate Smith-Miles [a,*], Davaatseren Baatar [a], Brendan Wreford [a], Rhyd Lewis [b]

[a] School of Mathematical Sciences, Monash University, Victoria 3800, Australia
[b] School of Mathematics, Cardiff University, Wales, United Kingdom

ARTICLE INFO

Available online 6 December 2013

Keywords:
Comparative analysis
Heuristics
Graph coloring
Algorithm selection
Performance prediction

ABSTRACT

This paper tackles the difficult but important task of objective algorithm performance assessment for optimization. Rather than reporting average performance of algorithms across a set of chosen instances, which may bias conclusions, we propose a methodology to enable the strengths and weaknesses of different optimization algorithms to be compared across a broader instance space. The results reported in a recent *Computers and Operations Research* paper comparing the performance of graph coloring heuristics are revisited with this new methodology to demonstrate (i) how pockets of the instance space can be found where algorithm performance varies significantly from the average performance of an algorithm; (ii) how the properties of the instances can be used to predict algorithm performance on previously unseen instances with high accuracy; and (iii) how the relative strengths and weaknesses of each algorithm can be visualized and measured objectively.

1. Introduction

Objective assessment of optimization algorithm performance is notoriously difficult [1,2], especially when the conclusions depend so heavily on the chosen test instances of the optimization problem. The popular use of benchmark libraries of instances (e.g. the OR-Library [3]) helps to standardize the testing of algorithms, but may not be sufficient to reveal the true strengths and weaknesses of algorithms. As cautioned by Hooker [1,2] nearly two decades ago, there is a need to be careful about the conclusions that can be drawn beyond the selected instances. It has been documented that there are some optimization problems where the benchmark library instances are not very diverse [4] and there is a danger that algorithms are developed and tuned to perform well on these instances without understanding the performance that can be expected on instances with diverse properties. Furthermore, while the peer-review process usually ensures that standard benchmark instances are used for well-studied problems, for many real-world or more unusual optimization problems there is a lack of benchmark instances, and a tendency for papers to be published that report algorithm performance based only on a small set of instances presented by the authors. Such papers typically are able to demonstrate that the new algorithm proposed by the authors outperforms other previously published approaches (it is difficult to get published otherwise), and the choice of instances cannot be challenged due to the lack of alternative instances.

The No-Free-Lunch (NFL) Theorems [5,6] state that all optimization algorithms have identically distributed performance when objective functions are drawn uniformly at random, and all algorithms have identical mean performance across the set of all optimization problems. Does this idea apply also to different instances of a particular optimization problem, which gives rise to only a subset of possible objective functions? Probably not [7], but it still seems unwise to believe that any one optimization algorithm will always be superior for all possible instances of a given problem. We should expect that any algorithm has weaknesses, and that some instances could be conceived where the algorithm would be less effective than its competitors, or at least instances exist where their competitive advantage disappears. Our current research culture, where negative results are seen as somehow less of a contribution than positive ones, means that the true strengths *and* weaknesses of an optimization algorithm are rarely exposed and reported. Yet for advancement of the field, surely we must find a way to make it easier for researchers to report the strengths and weaknesses of their algorithms. On which types of instances does an algorithm outperform its competitors? Where is it less effective? How can we describe those instances?

Occasionally we find a paper that presents a well-defined class of instances where an algorithm performs well, and reports its failing outside this class (see [8] for a recent example). Such studies assist our understanding of an algorithm and its applicability. Does the class of instances where an algorithm is effective overlap real-world or other interesting instances? Is an algorithm

* Corresponding author. Tel.: +61 3 99053170; fax: +61 3 99054403.
E-mail address: kate.smith-miles@monash.edu (K. Smith-Miles).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.11.015

only effective on instances where its competitors are also effective, or are there some classes where it is uniquely powerful? How do the properties of the instances affect algorithm performance? Until we develop the tools to enable researchers to quickly and easily determine the instances they need to consider to enable the boundary of effective algorithm performance to be described and quantified in terms of the properties of the instances, the objectivity of algorithm performance assessment will always be compromised with sample bias.

Recently, we have been developing the components of such a methodology [9]. Instances are represented as points in a high-dimensional feature space, with features chosen intentionally to tease out the similarities and differences between instance classes. For many broad classes of optimization problems, a rich set of features have already been identified that can be used to summarize the properties of instances affecting instance difficulty (see [10] for a survey of suitable features). Representing all available instances of an optimization problem in a single space in this manner can often reveal inadequacies in the diversity of the test instances. We can observe for some problems that benchmark instances appear to be structurally similar to randomly generated instances, eliciting similar performance from algorithms, and are not well designed for testing the strengths and weaknesses of algorithms. We have previously proposed the use of evolutionary algorithms to intentionally construct instances that are easy or hard for specific algorithms [11], thereby guaranteeing diversity of the instance set. Once we have sufficient instances covering most regions of the high-dimensional feature space, we need to be able to superimpose algorithm performance in this space and visualize the boundaries of good performance. Using dimensional reduction techniques such as Principal Component Analysis, we have previously proposed projecting all instances to a two-dimensional "*instance space*" [9] where we can visualize the region where an algorithm can be expected to perform well based on generalization of its observable performance on the test instances. We call this region the *algorithm footprint* in instance space, and the relative size and uniqueness of an algorithm's footprint can be used as an objective measure of algorithm power. Inspection of the distribution of individual features across the instance space can also be used to generate new insights into how the properties of instances affect algorithm performance, and machine learning techniques can be employed in the feature space (or instance space) to predict algorithm performance on unseen instances [12]. Over the last few years we have applied components of this broad methodology to a series of optimization problems including the Travelling Salesman Problem [9,11,13], Job-Shop Scheduling [14], Quadratic Assignment Problem [15], Graph Coloring [12,16], and Timetabling Problems [17,18].

While our previous research has generated an initial methodology, it has raised a number of questions that need to be addressed for a more comprehensive tool to be developed: How should we select the right features to represent the instance space most effectively? How can we determine the sufficiency and diversity of the set of instances? Can we more accurately predict algorithm performance in the high-dimensional feature space or the projected two-dimensional space? How should we determine the boundary of where we expect an algorithm to perform well based on limited observations? How can we reveal the strengths and weaknesses of a portfolio of algorithms, as well as their unique strengths and weaknesses within the portfolio.

This paper extends the methodology that has been under development for the last few years by addressing these last remaining questions. We demonstrate the use of the methodology by applying it to some computational results reported recently for an extensive comparison of graph coloring heuristics [19]. This case study reveals insights into the relative powers of the chosen optimization algorithms that were not apparent by considering performance averaged across all chosen instances.

The remainder of this paper is as follows: in Section 2 we present the framework upon which our methodology rests – the Algorithm Selection Problem [20] – which considers the relationships between the instance set, features, algorithms, and performance metrics. The detailed steps of the methodology are then described in Section 3, after proposing solutions to the questions raised above. In Section 4, we present a graph coloring case study based on the computational experiments of Lewis et al. [19] and discuss the new insights that the methodology has generated. Our conclusions are presented in Section 5, along with suggestions for use of the methodology and future research directions.

2. Framework: the algorithm selection problem

In 1976, Rice [20] proposed a framework for the Algorithm Selection Problem (ASP), which seeks to predict which algorithm from a portfolio is likely to perform best based on measurable features of problem instances. While Rice's focus was not on optimization algorithms, instead applying this approach to predict the performance of partial differential equation solvers [21,22], the framework is one that is readily generalizable to other domains (see the survey paper by Smith-Miles [23] for a review). There are four essential components of the model:

- the problem space $\mathcal{P}$ represents a possibly infinitely sized set of instances of a problem;
- the feature space $\mathcal{F}$ contains measurable characteristics of the instances generated by a computational feature extraction process applied to $\mathcal{P}$;
- the algorithm space $\mathcal{A}$ is a set (portfolio) of algorithms available to solve the problem;
- the performance space $\mathcal{Y}$ represents the mapping of each algorithm to a set of performance metrics.

For performance prediction, we need to find a mechanism for generating the mapping from feature space to algorithm space. The Algorithm Selection Problem can be formally stated as for a given problem instance $x \in \mathcal{P}$, with feature vector $f(x) \in \mathcal{F}$, find the selection mapping $S(f(x))$ into algorithm space $\mathcal{A}$, such that the selected algorithm $\alpha \in \mathcal{A}$ maximizes the performance mapping $y(\alpha, x) \in \mathcal{Y}$. The collection of data describing $\{\mathcal{P}, \mathcal{F}, \mathcal{A}, \mathcal{Y}\}$ is known as the *meta-data*. Analysis of the meta-data, in particular using statistical and machine learning techniques to learn the mapping S, between features of instances and the performance of algorithms, has been used effectively in algorithm portfolio approaches [14,15,17,24–26] to predict the algorithm likely to perform best for unseen instances.

In this research, we utilize the framework of Rice, but extend it to consider our broader agenda: we are not simply concerned with identifying the winning algorithm, but how to use the meta-data to identify the strengths and weaknesses of algorithms, and to visualize and quantify the relative power of algorithms. While our focus is on optimization algorithms, we will retain the generic nature of the framework, and highlight where domain-specific considerations apply. Fig. 1 presents the framework for our proposed methodology.

Generating and analyzing the meta-data $\{\mathcal{P}, \mathcal{F}, \mathcal{A}, \mathcal{Y}\}$ is central to the framework, as shown in the shaded central box describing Rice's Algorithm Selection Problem framework, but we extend the framework in each direction in order to address our broader objectives.

We start by acknowledging that Rice's problem space $\mathcal{P}$ should really be considered as the set of all possible instances of a problem, and not just the subset of instances $\mathcal{I} \subset \mathcal{P}$ for which we have computational results. We therefore must consider how