Computers & Operations Research 63 (2015) 102–113

Contents lists available at ScienceDirect

Computers & Operations Research

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

Generating new test instances by evolving in instance space

Kate Smith-Miles*, Simon Bowly

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

ARTICLE INFO

Available online 9 May 2015

Keywords:
Test instances
Benchmarking
Graph colouring
Instance space
Evolving instances

ABSTRACT

Our confidence in the future performance of any algorithm, including optimization algorithms, depends on how carefully we select test instances so that the generalization of algorithm performance on future instances can be inferred. In recent work, we have established a methodology to generate a 2-d representation of the instance space, comprising a set of known test instances. This instance space shows the similarities and differences between the instances using measurable features or properties, and enables the performance of algorithms to be viewed across the instance space, where generalizations can be inferred. The power of this methodology is the insights that can be generated into algorithm strengths and weaknesses by examining the regions in instance space where strong performance can be expected. The representation of the instance space is dependent on the choice of test instances however. In this paper we present a methodology for generating new test instances with controllable properties, by filling observed gaps in the instance space. This enables the generation of rich new sets of test instances to support better the understanding of algorithm strengths and weaknesses. The methodology is demonstrated on graph colouring as a case study.

1. Introduction

In his seminal paper of 1994, “Needed: an empirical science of algorithms”, Hooker [1] challenged the OR community to augment its theoretical focus on worst-case or average-case analysis of algorithms with a more empirical approach to algorithmic analysis: one that enables better understanding of the likely performance of algorithms on diverse test instances. He identified two main directions: more rigorous experimental design to select test instances intended to expose how the characteristics of the instances affect algorithm performance; and the use of empirical results to suggest hypotheses about algorithm behaviour, creating empirically based theories that can be submitted to rigorous testing. As such, he was proposing a paradigm shift in OR towards an experimental mathematics approach [2].

In a follow-up paper in 1995, “Testing heuristics: we have it all wrong” , Hooker [3] argued that randomly generated test instances lack diversity and rarely resemble real-world instances. He also expressed concern about the usefulness of commonly studied benchmark instances and their intrinsic bias, typically well suited to the first study that chooses to report on them but not necessarily diverse or challenging. The over-tuning of algorithms to a relatively small set of aging instances has major impact on the applicability of those algorithms for real-world deployment, and for our ability to learn about the strengths and weaknesses of algorithms from empirical evidence.

As an illustrative example, consider a commercial university timetabling algorithm that has been found to outperform competitor software packages when tested on Italian university timetabling instances (from the University of Udine instances used in the International Timetabling Competition). An Australian university then purchases the software, and finds that it results in more student clashes in the timetable compared to its previous software. While the Italian and Australian universities are tackling the same optimization problem (university curriculum timetabling), their *instances* of the problem seem to have different enough properties (class sizes, room capacities, number of subjects, etc.) that the algorithm performs well on one class of instance but not another.

Ensuring that an algorithm (and choice of parameters) has been tested thoroughly under all possible conditions that could be encountered in real world deployment is a significant challenge. Exhaustive testing is usually not possible due to the potentially infinite state space. Instead, a handful of test instances are usually the focus of algorithm development efforts, from which inferences about performance on other instances are optimistically made. Obtaining a sufficient number of real-world or real-world-like instances for statistical inference can be difficult, especially since synthetically generated instances of problems often have quite different properties and underlying structure to real-world instances [4–6]. Sampling from the set of possible test instances,

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

http://dx.doi.org/10.1016/j.cor.2015.04.022
0305-0548/© 2015 Elsevier Ltd. All rights reserved.

ensuring that the selected instances are real-world-like, and that no selection bias has been introduced that would affect the conclusions, is a significant challenge affecting algorithmic testing in both industrial and academic environments.

Indeed, poor research practice in academia and deployment disasters in industry can be viewed as stemming from the same problem: inadequate stress-testing of algorithms, and a consequential failure to understand an algorithm's strengths and weaknesses. The research culture often found in academic environments highlights the need for rigorous new methodologies and tools to support better research practice in algorithm testing. The OR literature, for example, is dominated by a methodology in which a new algorithm is introduced and usually claimed to be superior by showing that it outperforms previous approaches on a set of well-studied test instances. However, the weaknesses of the algorithm are rarely reported. Objective assessment of optimization algorithm performance is notoriously difficult [1,3,7], especially when the conclusions depend so heavily on the chosen test instances. The opportunity now exists to challenge and extend these benchmarks, and to generate new test instances that enable strong inferences to be made about algorithm strengths and weaknesses to support objective algorithmic testing [8].

Since Hooker's concerns were raised two decades ago, some progress has been made, and benchmark datasets have been expanding. Examples include the DIMACS challenge which created new graph colouring benchmarks [4]; efforts to generate more real-world-like instances of well-studied problems [9–12]; new instances that are more challenging for particular algorithms [13,14]; and some new instances with controlled characteristics to support experimental mathematics [15,16]. In the field of graph colouring, Culberson states on his webpage [17] about his graph generators, “my intention is to provide several graph generators that will support empirical research into the characteristics of various colouring algorithms”. Greater awareness now also exists for the importance of rigorous testing of algorithms [18].

It is not always straightforward however to generate new test instances that have the right kind of properties that will (i) challenge algorithms, enabling us to see their strengths and weaknesses and (ii) reflect real-world properties. Certainly, there is no difficulty in generating graphs that have a certain density, or other simple properties that can be easily controlled by an instance generator. But, as an example, it is much more difficult to construct a graph that has a required algebraic connectivity (2nd smallest eigenvalue of the Laplacian matrix of the graph). If we are to construct test instances with controllable properties that are thought to be significant for discriminating between algorithms or reflecting real-world properties, we must have the ability to understand which properties are important and embed their control in the instance generation process.

The design of experiments approach randomly generates instances by varying easily controlled parameters (usually just a subset of a more comprehensive feature space) to create a Latin hypercube design [15]. But more sophisticated ideas exist to enable better control over the difficulty and applicability of the instances. The idea of using a genetic algorithm (GA) to evolve instances with desirable characteristics has been around for a decade, starting with creating instances that are hard or “worst-case” for an algorithm [13,14]. Extending these ideas to create new instances that are easy or hard, and uniquely easy or hard for particular algorithms, new Travelling Salesman Problem instances have been evolved that have helped to learn the strengths and weaknesses of algorithms more effectively than relying on random or benchmark instances alone [19,20].

Another direction for instance generation has been to create instances that are real-world-like. Typically this is done by studying a small set of real-world instances and making small variations to some key parameters [11,12]. We have previously implemented a different approach using machine learning methods to identify differences between real-world instances and those produced by random instance generators, providing feedback on how to modify the generator so that instances become more real-world-like [21]. Additionally, we have imposed the condition that instances must be discriminating of algorithm performance (not uniformly easy or hard for all considered algorithms) so that the instances can be used for learning the unique strengths and weaknesses of algorithms. New timetabling instances were generated using this method, and shown to produce instances that are more similar to real-world instances than those of the seed instance generator [22], while simultaneously eliciting different performance behaviours from competitive algorithms.

Critical to the success of being able to measure how similar a synthetically generated instance is to a real-world instance, or the degree to which it exhibits a certain property, is the idea of representing the instances in a common instance space. Our recent work [8] has developed powerful new methodologies to enable objective assessment of optimization algorithm performance within such an instance space. Given a set of test instances, we can now generate a visual representation of the instance space as a topological mapping of the instance properties, and measure the region of the instance space where an algorithm can be expected to perform well, known as the algorithm footprint [23,24]. We have applied this methodology to assess the power of state-of-the-art optimization algorithms in an objective manner [8,25,26], and shown that existing benchmark instances only populate a small portion of the available instance space [8,21,24], and often do not coincide with the location of real-world instances [21].

It is clear that relying solely on existing benchmark or randomly generated instances limits our ability to understand the strengths and weaknesses of algorithms. What is required now are methods to generate a large number of new test instances for a given problem, with controllable characteristics, to enable more rigorous testing of algorithms. Where are the existing benchmark instances in the instance space, and how diverse are they? How real-world-like are they? How discriminating are they of algorithm performance or do they elicit the same response from all tested algorithms? Where should new test instances be located in the instance space that would provide the most information about the strengths and weaknesses of algorithms? The instance space provides the ideal vehicle to recognize where new test instances are needed and, combined with genetic algorithms, offers a mechanism to generate new test instances that lie at target locations in the instance space. This represents a fundamentally different approach to generating test instances compared to previous methods, which lack the ability to control for some of the sophisticated properties that are critical to explaining algorithm performance.

This paper is the third in a series of papers developing a methodology for providing insights into algorithm strengths and weaknesses. The first paper [27] identified the properties of instances that affect difficulty, for broad classes of combinatorial optimization problems, and provides the starting point for constructing instance spaces for a new problem. The second paper [8] demonstrated how to construct an instance space, and how visualizing and measuring the area of algorithm footprints in the instance space can provide the objective assessment of algorithm strengths and weaknesses we seek. The current paper extends these ideas to propose the use of the instance space to understand where new test instances are required, and a methodology to evolve new instances with controllable properties exploiting the mathematical relationships between the instance properties and their location in the instance space. In Section 2 we define the instance space, using graph colouring as an example, to illustrate how an instance space is constructed. Section 3 then presents the methodology for evolving new test instances within the instance space. A collection of new graph colouring test instances is presented in Section 4, and their diversity as compared to existing benchmarks is discussed. Finally, conclusions are drawn in Section 5, where we also identify opportunities for further research on this important topic.