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],

quadratic assignment [128], and job shop scheduling [131]), and have achieved excellent accuracy in predicting the best algorithm. The success of this framework depends critically upon the chosen features, where one must ensure that the relationship to algorithm performance can be learned. This paper describes the efforts to date on the identification and construction of suitable hardness-revealing problem features: problem-independent features based on search space and landscape analysis in Section 3, and problem-specific features that have been constructed for a wide range of combinatorial optimization problems in Section 4. We have chosen broad classes of combinatorial optimization problems, providing new generalized formulations in some cases, that form the core of other variations and many real-world problems. These formulations allow us to unify the literature and highlight where features from one problem can be adapted to study another related problem. Promising directions for future research are discussed in Section 5, and conclusions are drawn in Section 6.

2. Algorithm selection

As early as 1976, Rice [110] 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 a collection of problem instances. There are four essential components of the model:

- the problem space $\mathcal{P}$ represents the 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 the set (portfolio) of all considered algorithms for tackling the problem;
- the performance space $\mathcal{Y}$ represents the mapping of each algorithm result for a given problem instance to a vector of performance metrics (e.g. running time, solution quality, etc.).

In addition, 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 metric $\|y\|$ for $y(\alpha,x) \in \mathcal{Y}$. The collection of data describing $\{\mathcal{P},\mathcal{A},\mathcal{Y},\mathcal{F}\}$ is known as the *meta-data*. Rice's framework for algorithm selection is summarized in Fig. 1.

There have been many studies in the broad area of algorithm performance prediction, which is strongly related to algorithm selection in the sense that supervised learning or regression models are used to predict the performance ranking of a set of algorithms, given a set of features of the instances. In the artificial intelligence (AI) community, most of the relevant studies [71,84,85] have focused on constraint satisfaction problems like SAT ($\mathcal{P}$, in Rice's notation), using solvers like DPLL, CPLEX or heuristics ($\mathcal{A}$), and building a regression model (S) to use the features of the problem structure ($\mathcal{F}$) to predict the run-time performance of the algorithms ($\mathcal{Y}$). In recent years these studies have extended into the algorithm portfolio approach [155] and a focus on dynamic selection of algorithm components in real-time [119,133]. In the machine learning community, research in the field of meta-learning [2,26,146] (learning about learning) has focused on classification problems ($\mathcal{P}$), solved using typical machine learning classifiers such as decision trees, neural networks, or support vector machines ($\mathcal{A}$), where supervised learning methods (S) have been used to learn the relationship between the statistical and information theoretic measures of the instances ($\mathcal{F}$) and the accuracy ($\mathcal{Y}$) of the classifier algorithms. In the operations research community, there has been more focus on studying search space characteristics using landscape analysis metrics [18,77,94,132,134]. The developments in hyper-heuristics [28], which use a higher-level heuristic to select amongst simple heuristic search algorithms, can also be discussed within Rice's framework since the performance of simple heuristics can be seen as features ($\mathcal{F}$) as an alternative to calculating intrinsic characteristics of the problem instances. Clearly, Rice's framework is a useful approach to unifying the cross-disciplinary literature. Our survey paper [127] has discussed the developments in algorithm selection across a variety of disciplines, using Rice's notation as a unifying framework, through which ideas for cross-fertilization can be explored.

It is interesting to note though that Rice's framework provides no advice about the mapping from problem instance space $\mathcal{P}$ to the feature space $\mathcal{F}$, which is the focus of this paper. Rice and co-authors went on to use the approach to automatically select algorithms for solving partial differential [149], numerical integration problems [107] and scientific software packages [111], but they acknowledged "the way problem features affect methods is complex and algorithm selection might depend in an unstable way on the features actually used" [107]. While the construction of suitable features cannot be incorporated readily into Rice's abstract model, largely due to the problem-specific nature of the feature construction process (to be discussed in Section 4), we acknowledge here the criticality of the task of constructing suitable candidate features that adequately measure the relative difficulty of the instances. Once suitable candidate features have been constructed, well-studied feature selection methods (see for example [104,140,144,156]) can be employed to determine an optimal subset of the features. We refer the interested reader to [63], an excellent survey paper on feature selection. We now focus on the construction of suitable candidate features via problem-independent metrics (in Section 3) before turning to the construction of problem-specific features (in Section 4).

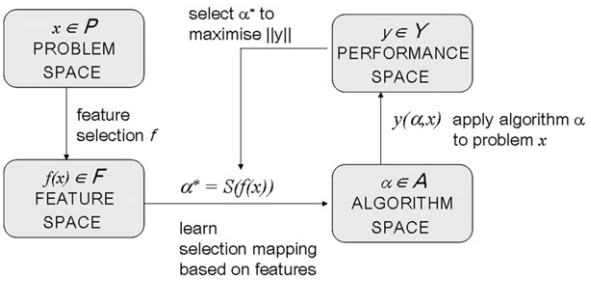

Fig. 1. The algorithm selection problem.

3. Problem independent feature construction

One successful approach to characterize the degree of difficulty of an optimization problem has been to consider the search space and its properties. Fitness landscape analysis [108,121] is the term used to characterize the search space. We start by reviewing the definition of a fitness landscape, and then we review metrics to characterize its properties.

A fitness landscape is a tuple composed of a set of solutions Ω, a fitness function Φ, which assigns a numeric value to each solution, and a neighborhood N_k defined over the set Ω, which is given by a distance metric of size k. More precisely, the fitness

landscape L is defined as

$$L = (\Omega, \Phi, N_k) \tag{1}$$

where

$$\Phi : \Omega \rightarrow \mathbb{R} \tag{2}$$

$$N_k(u) = \{v \in \Omega : d(u,v) \leq k\} \tag{3}$$

with the distance metric $d(u,v)$ defined as the number of applications of an elementary operator (such as flipping binary variables) to transform one solution $u \in \Omega$ to another $v \in \Omega$. The fitness landscape L can be viewed equivalently as a graph with Ω as the vertices and the edges included if they connect vertices within the neighborhood N_k [136]. We note that the landscape for a problem is not defined until the solution set Ω is known, so it can only be useful for gaining insights into algorithm design (our second question), and not for automated algorithm selection (our first question).

Now that the fitness landscape has been defined, we describe metrics that have been developed to characterize it. One feature of a fitness landscape is its *ruggedness*, measured by calculating the autocorrelation of the time series recording the solution quality generated by a random walk process. The autocorrelation function

$$\rho(s) \approx \frac{1}{\sigma_\Phi^2 (m-s)} \sum_{t=1}^{m-s} (\Phi(u_t) - \overline{\Phi})(\Phi(u_{t+s}) - \overline{\Phi}) \tag{4}$$

of a time series $\{\Phi(u_t)\}$ defines the correlation of two points s steps away along a random walk of length m through the fitness landscape (σ_Φ^2 denotes the variance of the fitness values). The landscape is considered to be rugged if the autocorrelation function is low. A hierarchy of combinatorial optimization problems has been proposed [5] in relation to the ruggedness of their associated landscape, defined by the neighborhood and cost function; it supports well the relative performance of local search algorithms found in the literature.

Another common feature used to characterize fitness landscapes is the fitness distance correlation (FDC) [75] which tries to measure the difficulty of problem instances by determining the correlation between the fitness value and distance to the nearest optimum in the search space. The FDC can be estimated by

$$\rho(\Phi, d) \approx \frac{1}{|\Omega| \sigma_\Phi \sigma_d} \sum_{u \in \Omega} (\Phi(u) - \overline{\Phi})(d(u) - \overline{d}) = \frac{\mathrm{cov}(\Phi, d)}{\sigma_\Phi \sigma_d} \tag{5}$$

with $d(u)$ being the minimum distance from a solution (local optimum) to a globally optimal solution. A value of $\rho = -1$ shows that the fitness and the distance to the optimum are perfectly negatively correlated; thus, for problems where the goal is maximization of the fitness function, the search is relatively easy. A FDC of $\rho = 1$ provides a more challenging task for a local search algorithm seeking to maximize fitness. Several studies have been conducted to show that the FDC can be misleading for certain types of functions, and that scatter plots of fitness against distance can reveal the structure of the landscape in these circumstances [105].

Other landscape analysis features include the number and distribution of local minima [151], the structure of the basins of attraction [10], the degree of randomness in the location of the optima [87], the density of the states in the landscape [114], and the Kolmogorov complexity of the landscape which measures the degree of structure (as opposed to randomness) that algorithms may be able to exploit [23].

Many of these metrics have been used to characterize a variety of combinatorial optimization problems, including the traveling salesman problem [132], job shop scheduling [18], quadratic assignment problems [95], and knapsack problems [136], to name just a few examples. Extensions of landscape analysis to multi-objective functions has also been extensively studied [44,77]. Most of these studies have supported the view that some of these features are useful predictors of problem difficulty (except some types of functions like ridge functions [105]). Certainly, minimization problems with low FDC and short autocorrelation length seem to be hard for all local search methods in general, but "despite extensive analyses of the fitness landscapes of numerous combinatorial optimization problems, the link between problem difficulty and fitness landscape structure is poorly understood" [18]. Considering in addition the need for these features to be able to explain variation in algorithm performance, relying solely on the use of landscape features to discriminate between the performance of different algorithms is unlikely to be very revealing.

The final set of problem-independent features that can be constructed to characterize a search space or landscape are based on the concept of *landmarking* [101]. Here, metrics gathered from the performance of simple and quick algorithms (such as gradient descent) are used to characterize the relative difficulty of the problem instances, as a proxy measure and as an alternative to a computationally expensive feature calculation or exploration of the entire landscape. The use of landmarks as features within an algorithm selection framework is similar to the hyper-heuristics approach [28] whereby a higher-level heuristic is used to determine the switch between lower-level simple heuristics selected from a portfolio, without any problem-specific knowledge included as features.

If our goal is to obtain a good set of features within a time constrained setting, as required for automated algorithm selection, then it makes sense to combine landmarking with problem-specific features constructed with knowledge of what makes problem instances challenging. If our goal extends beyond automated algorithm selection to the (time-unconstrained) development of genuine insights to inform algorithm design, then the problem-independent landscape analysis metrics, combined with problem-specific features, may provide the richest set of features from which to explore the constituents of problem difficulty. This is a hypothesis that we leave for a future paper.

4. Problem specific feature construction

In this section we present some common combinatorial optimization problems, and review some of the features that have been constructed to characterize problem difficulty. We focus on broad classes of combinatorial optimization problems, which form the core of many practical optimization problems: assignment, traveling salesman, and knapsack problems, bin-packing, graph coloring, and timetabling. Throughout this section, we adopt a uniform style for formulating the problems which, in many cases, results in a weak formulation. It should be noted that many formulations are possible for each of these problems, and much effort has been spent by many researchers over several decades to improve integer programming representations [139] (for example, by redefining constraint sets with tighter linear relaxations). It is not our goal here to give a state-of-the-art review of integer programming formulations for combinatorial optimization problems, but rather to summarize these problems in terms of a common notation; it will be helpful for a coherent discussion of problem-specific features.

4.1. Assignment problems

Assignment problems are generally concerned with assigning labels to objects, whether those objects are people or machines

assigned to tasks, facilities being assigned locations, etc. Depending on the costs of the assignments, the objective function may be linear or quadratic, and the constraints may include budget or capacity constraints. We present here a generalized formulation of the assignment problem between N objects and M labels, from which most commonly studied variations are special cases. We express all families of assignment problem in this generalized form for convenience of discussing the literature only. Special cases with related formulations are not necessarily related in complexity. Furthermore, for some special cases there are tighter formulations than the most generic one.

Let:

- $x_{i,j}=1$ if object i is assigned label j, and 0 otherwise;
- $D=(d_{j,l})$, the $M\times M$ distance matrix between labels;
- $V=(v_{i,k})$, the $N\times N$ volume of flow between objects;
- $W=(w_{i,j})$, the $N\times M$ weight matrix for each assignment;
- $C=(c_{i,j})$, the $N\times M$ matrix of fixed assignment costs;
- χ_i, the capacity of each object;
- s_1,s_2, artificial slack variables, included to generalize the formulation for either equality or inequality constraints.

Then the generalized quadratic assignment problem (GQAP) can be expressed as

$$\text{minimize} \quad \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{N}\sum_{l=1}^{M} x_{i,j}d_{j,l}v_{i,k}x_{k,l}+\sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j}x_{i,j} \tag{6}$$

$$\text{subject to} \quad \sum_{i} x_{i,j}+s_{1,j}=1 \quad \forall j\in\{1,\ldots,M\} \tag{7}$$

$$\sum_{j} w_{i,j}x_{i,j}+s_{2,i}=\chi_i \quad \forall i\in\{1,\ldots,N\} \tag{8}$$

$$x_{i,j}\in\{0,1\} \quad \forall (i,j)\in\{1,\ldots,N\}\times\{1,\ldots,M\} \tag{9}$$

$$s_{1,j}\geq 0,\ s_{2,i}\geq 0 \quad \forall (i,j)\in\{1,\ldots,N\}\times\{1,\ldots,M\} \tag{10}$$

Cario et al. [29] considered the impact of the parameters of the linear generalized assignment problem (GAP) (with no quadratic cost term, $D=0$) in Eq. (6), and $s_1=0$ in Eq. (7) on the performance of two solvers and four heuristics. The problem instances were characterized by a set of features including the problem size N (with $N=M$), the magnitude of the cost coefficients $c_{i,j}$, the tightness of the constraints, as well as the correlation structure between the objective function parameters $c_{i,j}$, and the constraint coefficients $w_{i,j}$ and the capacity requirements χ_i in Eq. (8). Their conclusions stated that the correlation between the objective function costs and the capacity constraint coefficients can have a significant impact on the ability of algorithms to solve the GAP. Specifically, the branch-and-bound solvers LINDO and CPLEX performed poorly on problem instances with strongly negatively correlated coefficients, perhaps due to the difficulty they experience pruning the tree when the choice is between variables with high cost and low weights versus low costs and high weights. The heuristics (mostly greedy) performed better with increasing absolute values of the correlation feature. They also considered the gap between the objective function of the linear programming relaxation of the GAP (relaxing the integrality constraint given by Eq. (9)) and the known optimal 0–1 solution as a feature to characterize the problem difficulty. The heuristics performed better on problem instances where this gap was small, while the exact solvers were largely unaffected by this feature. Correlation structure is clearly a relevant problem-specific feature, and has been shown to be revealing of algorithm performance on other problems involving similar constraints, such as knapsack problems [68,109].

If we set $s_1=\mathbf{0}$,$s_2=\mathbf{0}$,$W=\mathbf{1}$, and $\chi=\mathbf{1}$ so that Eqs. (7) and (8) become the standard permutation constraints:

$$\sum_{i} x_{i,j}=1 \quad \forall j\in\{1,\ldots,M\} \tag{11}$$

$$\sum_{j} x_{i,j}=1 \quad \forall \in\{1,\ldots,N\} \tag{12}$$

then we arrive at the quadratic assignment problem (QAP). Unlike in the case of the linear assignment problem, in the QAP we must explicitly keep the integrality constraint Eq. (9). The QAP is concerned with assigning a set of N facilities to a set of N (again with $N=M$) locations in such a way as to minimize the total cost of the volume of flow between facilities. In some variants the fixed cost component C is also kept. Lawler [82] proposed a more general version of the QAP, with a four-dimensional matrix $\overline{C}$ replacing matrices D, V and C.

The QAP is one of the most well-studied NP-hard optimization problems. Despite the breakthrough of solving the well-known instance NUG30 in a week using a massively parallel branch-and-bound code [7], proving optimality is still not practical for general QAPs significantly larger than $N=30$, although a variety of good lower bounds exist [8,27,103,106]. Commercial solvers like Gurobi and CPLEX are often able to produce good solutions on careful QAP formulations, but are only able to prove optimality if some of the bounds mentioned are incorporated into the search, since the LP relaxations of the pure models tend to be weak. Even for meta-heuristics researchers the QAP offers a great challenge, despite the fact that there are natural formulations of the QAP based on permutations of short strings – every permutation being feasible – a structure that in theory is especially amenable to meta-heuristic searches.

The difficulty of the problem varies considerably depending on the nature of the objective function, captured in the matrices D, V and C. There has been much research over many years focused on measuring the characteristics of the search space for QAPs. The earliest feature was the dominance approach of Vollmann and Buffa [147] which measures the extent to which the matrices demonstrate dominant patterns (e.g. a large volume of flow between a small number of facilities corresponds to a high flow dominance, and likewise, large distances between a small number of facilities corresponds to a high distance dominance). The dominance feature is calculated as the standard deviation of the matrix entries divided by their average value. Presence of high dominance of either D or C, can indicate that there is significant variation in the costs of the optimal solution and the worst solution, and the landscape of the problem is more rugged. Additional features of the QAP problem structure include the problem size N as well as the sparsity of the volume and distance matrices, defined as N_0/N^2 where N_0 is the number of zero entries in the matrix. Analytic expressions for the mean and variance of the QAP objective function across the search space have also been used as a feature to characterize instance difficulty [90] and were shown to account for variations in algorithm performance on some key benchmark problems not explained by dominance metrics alone. Additional features used in numerous studies of algorithm performance include the problem-independent landscape analysis metrics such as ruggedness [6,95,134].

For these studies, the generation of test problem instances displaying a range of characteristics is critical, and the QAP is relatively well served in this regard. The well-studied benchmark collection of QAP instances, QAPLIB, contains about 130 instances that have been grouped into four classes [135], based on the methods for generating D and V: (i) unstructured, randomly generated instances (entries in matrices D and V generated at random according to a uniform distribution), (ii) grid-based

distance matrix (distance matrix D based on Manhattan distance of grid locations), (iii) real-life instances (generated from real practical applications of the QAP, characterized by flow matrices V containing frequent zero entries), and (iv) real-life like instances (larger artificial datasets with similar distributions to class iii). Other instances have been designed to be hard [45]. The most systematic and comprehensive set of QAP instances has been generated by Stutzle and Fernandes [134], who generated six different classes of QAP instances of different sizes ranging from $N=50$ to 500, resulting in a total of 644 new benchmark instances.

4.2. Traveling salesman problems

The traveling salesman problem (TSP) involves finding the minimal cost tour visiting each of N cities exactly once and returning to the starting city. The travel cost between city i and city j is notated as $c_{i,j}$ and asymmetry of the travel cost matrix C ($c_{i,j} \neq c_{j,i}$) renames the problem to the asymmetric traveling salesman problem (ATSP) [74]. This problem variant is more general and challenging; it describes also certain scheduling problems. Another important feature of (A)TSP instances is whether or not the costs in C satisfy the triangle inequality [100].

Let $x_{i,j}=1$ if city i is followed by city j in the tour, and 0 otherwise. Then the ATSP can be expressed as

$$\text{minimize} \quad \sum_{i=1}^{N}\sum_{j=1}^{N} c_{i,j}x_{i,j} \tag{13}$$

$$\text{subject to} \quad \sum_{i\in S}\sum_{j\in S} x_{i,j} \leq |S|-1 \quad \forall S\neq\{0\},\ S\subset\{1,2,\ldots,N\} \tag{14}$$

$$\text{and Eqs. (9), (11),(12)} \tag{15}$$

This formulation of the ATSP resembles that of the assignment problem, with the additional subtour elimination constraint (14). An alternative formulation, which does not require the subtour elimination constraint, involves redefining the binary variable $x_{i,j}$ to be 1 if city i is visited at stop j in the tour, and 0 otherwise. The permutation constraints given by Eqs. (9), (11), (12) remain, but Eq. (14) can be dropped if this quadratic objective function is used instead of the linear one; and the (now quadratic) objective function is

$$\text{minimize} \quad \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} x_{i,j}c_{i,k}(x_{k,j+1}+x_{k,j-1}) \tag{16}$$

The quadratic formulation, while avoiding the subtour problems, creates a non-convex quadratic objective function with many local minima, and has been used primarily within the neural network community due to the internal dynamics of the Hopfield neural network naturally minimizing quadratic energy functions [125]. For most other communities, the integer linear programming formulation, often strengthened by dynamic constraints produced during the traversal of the branch-and-bound tree, or simpler formulations based on defining a permutation vector of cities, have been used.

The difficulty of this problem has been well-studied for many years. Properties of the cost matrix C naturally govern the difficulty. For example, if all of the inter-city costs were identical, the problem is extremely easy to solve and there are $(n-1)!$ equally minimal cost solutions. Christofides' [33] polynomial time approximation algorithm showed that ATSP instances with costs satisfying the triangle inequality were much easier to solve that those where the triangle inequality did not hold, and the proof of this was demonstrated soon after Papadimitriou and Steiglitz [100].

By the early 1990s, the AI community had started to explore the question of whether all NP-complete problems could be characterized as easy or hard depending on some critical parameter embedded within the problem. Cheeseman et al. [30] conjectured that this was the case, that phase transitions exist for all NP-complete problems including the TSP, and contain at least one critical control parameter around which the most difficult problem instances are clustered. The degree to which the triangle inequality is satisfied is a strong candidate for such a parameter, as are several metrics of variance within the cost matrix C. The value of these metrics has been demonstrated for both exact approaches [30] and heuristics such as ant colony optimization [112]. But are there other parameters that can be constructed from C that could demonstrate such a phase transition from easy to hard?

Some researchers have shown for the Euclidean TSP in the plane that phase transitions exist for the TSP decision problem [54,138], which seeks to determine a binary (yes/no) response to the question, does a tour of length less than l exist? If the N cities are distributed randomly within a square of area A, then the decision problem becomes extremely difficult for instances with $(l/\sqrt{NA})\approx 0.75$) [54]. Below this phase transition boundary, instances are over-constrained and the decision problem is easy (there is not likely to be a tour of length less than l), and above this boundary the instances are under-constrained and the decision problem is also easy (there is likely to be a tour of length less than l). The decision problem is only difficult around the boundary provided by the critical phase transition parameter $(l/\sqrt{NA})\approx 0.75$). Returning to the optimization version of the general ATSP, Zhang and colleagues have examined the distribution of costs (distances) and shown that the number of distinct distance values affects algorithm performance [158], and that phase transitions exist controlled by the fraction of distinct distances [157]. The critical control parameter β given by the relation $(\beta-2)\log_{10}(N)=2.1$ was determined by numerical experiments of heuristics based on $N\leq 1500$ cities.

Similar to the concept of landmarking [101] and hyper-heuristics [28], the difficulty of the ATSP can also be characterized by the cost of the assignment problem solution when the subtour elimination constraints (14) are relaxed. Studies have shown that the difference between the costs of the ATSP and the relaxed assignment problem is influenced by the number of zero costs (distances) in the matrix C [49]. Landscape metrics have also been calculated for the ATSP and TSP [132], and have shown that the landscape is highly correlated and can be well understood in terms of the mean and variance of the costs, the value of N, as well as the average number of exchanges of edges (switching placement of cities in the tour) permitted by various algorithms. These landscape metrics require a thorough search of the solution space; as such they are not useful for automated algorithm performance prediction. Metrics related to the size of the backbone [76] also fall into this category. A backbone variable has fixed values amongst all optimal solutions, and if its value is changed it becomes impossible to reach an optimal solution. A larger backbone corresponds to a highly constrained, more difficult problem. So the fraction of the variables that comprise the backbone correlates well with problem difficulty, but this fraction cannot readily be calculated until all optimal solutions have been found.

The final contribution towards characterizing the difficulty of TSP instances comes from those who have been seeking to generate hard instances. van Hemert [142] has used genetic algorithms to evolve TSP instances that are difficult for the Lin–Kernighan algorithm [86] and its variants to solve. This is not done by studying structural properties of hard instances, and then generating instances that exhibit those properties, but by

using the performance of the Lin–Kernighan algorithm as a proxy for instance difficulty, which becomes the fitness function for an evolutionary algorithm to evolve instances that maximize their difficulty (for that algorithm). The statistical properties of the generated instances can then be studied to glean some insight into the properties that these instances share, and what distinguishes them from randomly generated instances. This approach [141] has shown the importance of the cluster distribution of the cities, and the location and distribution of outliers. The ratio of the number of clusters to the number of cities was demonstrated experimentally ($N \leq 200$) to create an easy–hard–easy phase transition, with instance difficulty maximized when the ratio is in the range [0.1,0.11]. In this region, the average number of steps required for the Lin–Kernighan algorithm to reach a "good solution" was 5.9 times greater than that required for randomly generated instances [141]. By such an analysis of evolved hard instances, one can extract ideal instance features for automated algorithm selection, as shown recently by Smith-Miles and van Hemert in a series of studies of two variations of the Lin–Kernighan algorithm [130,126].

4.3. Knapsack problems

A generalization of the knapsack problem (GKP) involves placing up to k_i copies of N kinds of items into M knapsacks. Each item of kind i is worth c_i and measures $w_{i,p}$ along dimension $p \in 1,\ldots,P$. Each knapsack j has capacity $\chi_{j,p}$ along dimension p. The goal is to maximize the value in the knapsacks while respecting all the capacities.

Let $x_{i,j}$ be the number of items of kind i included in knapsack j. Then the GKP can be expressed as

$$\text{maximize} \quad \sum_{i=1}^{N}\sum_{j=1}^{M} c_i x_{i,j} \tag{17}$$

$$\text{subject to} \quad \sum_i w_{i,p}x_{i,j} \leq \chi_{j,p} \quad \forall (j,p) \in \{1,\ldots,M\} \times \{1,\ldots,P\} \tag{18}$$

$$\sum_j x_{i,j} \leq k_i \quad \forall i = 1,\ldots,N \tag{19}$$

$$x_{i,j} \in \mathbb{Z} \quad \forall (i,j) \in \{1,\ldots,N\} \times \{1,\ldots,M\} \tag{20}$$

By far the most studied case of the GKP is the traditional Knapsack problem from Held and Karp when $M = P = 1$ and $k = \mathbf{1}$ (0–1 Knapsack problem). In another important case, commonly referred to as the multi-dimensional knapsack (MKP), $M=1$ and $k = \mathbf{1}$ while $P > 1$. Of special interest is the case where $P=2$ (known as 2KP or bi-dimensional knapsack). The difficulty of several variants of the MKP has been characterized [68,109] by using a set of features related to the correlations between the instance parameters (weights, profits, and capacities) and the constraint slackness ratio for each knapsack j defined as $\chi_j / \sum w_i$. Based on this ratio alone, instances are classified as tightly constrained (ratio around 0.3) or loosely constrained (ratio around 0.7). An examination of the commonly used MKP benchmark instances [16] utilizing these metrics revealed that they are rather similar in terms of correlations and slackness ratios, and the new set of instances generated by Hill and Reilly [68] controlled the characteristics of the instances to enable their relative difficulty for various algorithmic approaches to be explored. Cho et al. [32] utilized these features to demonstrate that both enumerative methods (like dynamic programming) and heuristics (like greedy search) were affected by variation in the constraint characteristics and correlations with item values. They examined the effect of instance features on the performance of three greedy heuristics, developing rules based on observations rather than a predictive model. Instances were classified into various types based on their features, and rules developed to determine which heuristic should be used. In addition, a heuristic was developed with a new gradient function for the greedy search, based on knowledge gleaned from exploration of the meta-data. This new heuristic improved the solution quality in 69% of unseen test instances. Clearly, this is a case of meta-data being used for both automated algorithm selection (with the mapping S based on inspection of the relationships in the data rather than any machine learning process) as well as the creation of insights to assist with the development of new and improved algorithms.

The difficulty of the 0–1 Knapsack problem with only one knapsack, ($M = 1$, $P = 1$,$k = \mathbf{1}$) and capacity χ, has been well-studied for many decades. Chvatal [36] described in 1980 a class of problems that are hard for recursive algorithms such as dynamic programming, with run-times growing exponentially and bounded from below by $2^{N/10}$. These problems have $c_i = w_i$ randomly selected from the range $[1,10^{N/2}]$, and the capacity $\chi = \sum w_i/2$. Balas and Zemel [11] showed that instances with $\lambda_i = c_i/w_i$ ratios close to each other for all i are indeed difficult for enumeration methods, but not for their new heuristic. They proposed a new measure of difficulty to characterize instances, based on the gap between the linear programming solution (relaxing the constraint (20)) and the optimal solution to the 0–1 problem, notated as δ. If $\delta < 1$ the instance is deemed to be easy, but otherwise, the degree of difficulty is measured as

$$\Delta = \frac{2\delta}{\max_i |c_i - w_i\lambda^*|} \tag{21}$$

where λ^* is the critical ratio of item values to weights that provides a unique solution to the pair of inequalities:

$$S_1(\lambda) < \chi \leq S_2(\lambda) \tag{22}$$

where

$$S_1(\lambda) = \sum_{i|\lambda_i > \lambda} w_i \quad \text{and} \quad S_2(\lambda) = S_1(\lambda) + \sum_{i|\lambda_i = \lambda} w_i \tag{23}$$

Clearly this metric cannot be used unless the optimal solution is known. Chung et al. [34] described a set of hard instances for the unbounded general knapsack problem as those with $w_i = w+i-1$ and $c_i = w_i + z$ for all i, with the instances fully parameterized by the set $\{w,N,z,\chi\}$. They noted that such problems have $\delta > 1$ and usually have a narrow range of λ_i ratios making them challenging, consistent with the observations of others [11,36]. They observed that if the capacity χ is a multiple of w (the weight of the first item), then the problem is trivial. Other observations included that if the difference between an item's weight and value is zero ($z=0$) then the instance is easy to solve for branch-and-bound algorithms [34]. For instances with a near constant difference between c_i and w_i values though, regardless of whether this difference (z) is positive or negative, the instances are hard to solve for branch-and-bound algorithms, at least for certain values of χ that are not multiples of w.

The relationship between c_i, w_i, and the range $[1,R]$ from which they are randomly generated has been the focus of several 0–1 Knapsack investigations [68,81,92,102]. Pisinger [102] generated several classes of instances, from strongly correlated to weakly correlated, and increased the data range R with a view to challenging the run-time of a dynamic programming algorithm. Weakly correlated instances, with w_i randomly generated from $[1,R]$ and c_i randomly generated from the range $[w_i - R/10, w_i + R/10]$ were easier than strongly correlated instances, and are believed to be more realistic of real-world knapsack problems [55,102]. They are also more stable as R increases. For strongly correlated instances with $c_i = w_i + R/10$ the value of δ used by Balas and Zemel [11] is

large, and the instances are ill-conditioned and hard. As the range R is increased the instances tend to become even more challenging, but correlation alone does not explain the performance of algorithms, and the skewness of the distribution of c_i values in feasible solutions has also been used to explain the relationship between correlation structure and run-time performance [55].

The goal of using such metrics to learn the relationship between instance characteristics and algorithm performance has been tackled by Cho et al. [32] for the MKP, as discussed above, and also by Hall and Posner [65] for the 0–1 Knapsack problem. While Cho et al. focused only on the correlation and constraint slackness metrics, Hall and Posner considered a wide range of features to characterize instance difficulty, some of which are based on the intrinsic properties of the instance, and others requiring knowledge of the optimal solutions. Their feature set included: N, the coefficients of variation of the item values and the weights, the knapsack size χ, the ratio of χ/N, the proportion of dominant item pairs (i,j) for which $c_i \geq c_j$ and $w_i \leq w_j$, and a number of metrics related to the gap between the upper bound provided by the linear programming relaxation and the lower bound from the first feasible solution, the proportion of the variables that are backbone variables in both the relaxed and binary solutions, and the Balas–Zemel difficulty metric Δ provided in Eq. (21). Their meta-data (see Section 2), generated according to their previously published guidelines for such experimental studies [64], included the performance of two algorithms: dynamic programming and a branch and search algorithm [65]. A regression model was used to learn the relationship between the feature set and algorithm performance, producing $R^2 = 97.7\%$ for predicting the run-time of dynamic programming, but only $R^2 = 30\%$ for predicting the performance of the branch and search algorithm. This was despite utilizing much domain knowledge of the conditions under which the branch and search algorithm performs well or poorly when devising suitable features. Feature subset selection was not discussed however, and it is likely that the optimal subset of features to predict algorithm performance is algorithm-dependent.

4.4. Bin-packing problems

Bin-packing is closely related to the MKP. We use the same decision variable $x_{i,j}$, but the objective changes; it is now to minimize the number of knapsacks M used to pack all items, rather than maximizing the value from the subset of items packed into a fixed number M of knapsacks. To count the number of knapsacks used, we define a new variable $y_j \in \{0,1\}$ that is 1 if the jth knapsack (bin) is used. Choosing M large enough, we define the generalized bin-packing problem (GBPP) to be

$$\text{minimize} \quad \sum_{j=1}^{M} y_j \tag{24}$$

$$\text{subject to} \quad \sum_i w_{i,j} x_{i,j} \leq \chi_{j,p} y_j, \quad \forall (j,p) \in \{1,\ldots,M\} \times \{1,\ldots,P\} \tag{25}$$

$$\sum_j x_{i,j} = k_i, \quad \forall i \in \{1,\ldots,N\} \tag{26}$$

and Eq. (20)

In the classical bin-packing problem (BPP) we have $P=1$, $k=\mathbf{1}$, and $\chi_{j,k} = \chi$ (identical capacities for all knapsacks). There is not much to parametrize BPP instances apart from the weight vector and the capacity. Several researchers have investigated the effect of the range of the weights [48,122], showing that a smaller range makes the problem more difficult for common algorithms such as the first-fit-decreasing algorithm [38]. Suppose the weights are generated from the range $[v_L \cdot \chi, v_U \cdot \chi]$ so that v_L and v_U correspond to the lower and upper bounds on the weights respectively, relative to the bin capacity χ. Instances generated with $v_L = 1/4$ and $v_U = 1/2$ are considered to be special cases, known as "triplets", since a well-filled bin must contain one heavy and two small items. These instances are difficult since a solution that deviates only slightly from this strategy is likely to lead to much wastage of capacity, and will be far from optimal [122]. Thus the backbone is quite large for triplet instances. Other ranges also create difficult instances when the average weight $\overline{w} \simeq 1/3$. Instances with average weights close to $1/n$ for $n \geq 3$ have also been shown to be difficult [48].

The analysis conducted by Gent [52] suggests another useful set of measures could be constructed based on the proportion of item pairs whose weights sum to the capacity, since this provides some proxy measure of constrainedness, and the likelihood of have two large items or a triplet solution. The genetic algorithm of Ross et al. [117] selects one of four heuristics to switch to during run-time, but this method does not utilize any metrics about the relative difficulty of the instance. There seems to be much room for additional analysis of hard instances of bin-packing, particularly examining statistical properties of the weight vector and dominance metrics.

4.5. Graph problems

Many combinatorial optimization problems can be formulated as optimization problems on graphs. In many other cases, concrete relationships from specific domains can be expressed using graphs, and the features of those graphs provide insight into algorithmic behavior.

A graph G is defined as $G=(V,E)$ comprising a set of N vertices or nodes V, and a set of edges E connecting pairs of vertices. The adjacency matrix A of the graph G is an $N \times N$ matrix whose elements $A_{i,j}$ are 1 if there is an edge from vertex i to vertex j, otherwise 0. Common optimization problems on graphs include finding subgraphs, coloring problems, routing and network flow problems, and covering problems, but a review of graph problems is not the purpose of this section and excellent and comprehensive descriptions of the wide variety of optimization problems based on graph structures can be found elsewhere [20,25,40,62]. In this section we will focus on a few key optimization problems arising from graphs, and discuss how properties of graphs have been used to characterize the relative difficulty of instances of these problems.

Regardless of the optimization problem, a graph can readily be characterized by a number of simple features such as its size N, whether its edges are directed or undirected, the degree of connectivity (edge density), and statistics of the vertex degrees. The density of a graph is the ratio of the number of edges M of the graph to the number of possible edges in the complete graph, and is given by

$$\rho(G) = \frac{2M}{N(N-1)} \tag{27}$$

For specific optimization problems on graphs, we may be interested in measuring certain properties. Spectral analysis of the adjacency matrix A and the Laplacian matrix $L=D-A$, where D is a diagonal matrix containing the degree of each vertex as its entries, can be used to provide information about the properties of the graph [35], with certain eigenvalues revealing information about partitioning characteristics for example [60]. The connectivity of the graph can also be measured, not just by its density, but by its resistance to becoming disconnected after removal of edges. If removal of a single edge causes the graph to become

disconnected, then that edge is called a bridge, and the number of bridges in a graph can be measured [99]. The connectivity of a graph can be summarized by Cheeger's constant or isoperimetric number [98], given by

$$h_G = \min_{S \subseteq V : S \neq \emptyset} \frac{|E(S,\overline{S})|}{\min(|S|,|\overline{S}|)} \tag{28}$$

with $h_G = 0$ corresponding to a disconnected graph, and $h_G > 0$ showing the degree of connectedness.

The density metric can be modified to consider the minimum number of edges needing to be removed or added to achieve certain properties (such as being connected, planar, bipartite, etc.) and the conditional density can then be measured [152]. If a graph does not possess a certain property, it is sometimes useful to know how far off it is from having the property. A graph G is said to be ε-far from satisfying a property if εN^2 edges need to be added and/or deleted to G in order to convert it into a graph which satisfies the property. Certain algorithms may rely on assumptions about graph properties, and it is useful to explore how the presence of these properties, or the distance from these properties, contributes to algorithm performance. For example, genetic algorithms are premised on the building blocks hypothesis [56,148] that assumes that if two good disjoint sub-solutions are suitably combined then a better solution can be obtained. Establishing whether the building block hypothesis holds for a given graph optimization problem and an instance may be no trivial task however.

Certainly algorithms exist for testing various properties of graphs including regularity [78], isomorphism, triangle-freeness, partitionability, and cycles [4]. While these may be useful features to describe an instance, it is important to examine the computational complexity of algorithms required to determine these properties and ensure that they are indeed simple to measure as features for automated algorithm selection. Property testing has attracted much attention in recent years [3,57,58,113], and defines a test to be efficient and therefore "testable" if the number of queries made by the testing algorithm is a constant regardless of the size of the graph. So while there are many features or properties of graphs that may be valuable to shed light on variation in algorithm performance, the concept of testable features is a relevant one for first question about automated algorithm selection. For our second question about understanding the types of instances for which we can expect a given algorithm to perform well, we may be prepared to consider a computationally expensive calculation of features if it will lead to the desired insights.

We describe now some examples of optimization problems on graphs and research that has contributed to knowledge about what makes instances of these problems hard. Much of this research has focused on showing the existence of phase transitions [30,67]. The first problem we consider is *graph coloring*, where the goal is to color each vertex of the graph in such a way that no two vertices connected by an edge are of the same color. Variations of this problem include finding the minimum number of colors needed to color the whole graph, or finding the maximum sub-graph that can be colored with k colors. Most research has focused on the Erdös–Rényi model [47] of a random graph $G(N,p)$ where p is the probability of an edge being connected, given by $p = d/(N-1)$, and d is the average degree of the vertices. Random instances of graph coloring have been shown to be hard [145], and the chromatic number required to color a random graph has been shown to be l_d or l_d+1 where l_d is the smallest integer l such that $d < 2l(\ln(l))$ [1]. Suppose however that the number of colors available, k, is less that the chromatic number? The k-colorability problem has been shown to undergo a phase transition when the average degree of the vertices is around $d = 2k\ln(k) - 1 + o(1)$ [80]. Phase transitions can also be controlled by the edge connectivity parameter p. Culberson and Luo [42] explored the difficulty of instances of the k-colorability problem for various combinations of N, k, and p and showed that the hardest instances are those where the optimal solution involves equal distribution of colors across the graph, or where the graphs are equi-partite. Other factors that contribute to hard instances include minimal variation in the degree of the vertices [46]. The location of the phase transition in p has been explored as a function of the size of the graph [37], and the hardest random instances of 3-colorable graphs have been shown by Eiben et al. [46] to be for p in the range $[7/N, 8/N]$, tested on a range of approaches including genetic algorithms.

Another important optimization problem based on a graph is the *minimum vertex cover problem*. This problem involves finding the smallest subset $\hat{V} \subseteq V$ such that each edge of the graph G is incident to at least one vertex in $\hat{V}$. The decision form of this problem is to determine if a vertex cover exists for a given cover of size C. The phase transition for the decision form of the problem with random graphs has been shown to occur at a critical value of C at approximately $0.39N$ for graphs with low connectivity ($p < 1/(\ln(N-1))$), with the critical point for more densely connected graphs being harder to quantify [66,150]. This observation has been further explored by clustering the optimal solutions in the landscape and observing that optimal solutions are collected in a finite number of tight clusters for graphs with low connectivity, while for denser graphs the number of clusters diverges and structure is lost [13].

The *set covering problem* aims to find a minimal number of $k \leq N$ of N defined subsets of the universe set $\{1,\ldots,M\}$, such that the union of the k subsets is the universe set $\{1,\ldots,M\}$. This is equivalent to finding a minimal cover of the incidence matrix which contains information on whether a vertex is connected to an edge [120]. Difficult instances of the set covering problem have been studied by Avis [9], who was motivated by earlier analysis of the difficulty of knapsack problems [36] and took a similar approach to develop insights into why the set covering instances of Fulkerson et al. [50] were hard for branch-and-bound algorithms. These instances are generated from a class known as Steiner triple systems, which have three one's in each row of the incidence matrix, so properties of the incidence matrix are clearly significant for this problem [120].

The *maximum clique problem* aims to find the largest subset of vertices $\hat{V} \subseteq V$ such that each vertex in $\hat{V}$ is connected by an edge to every other vertex in $\hat{V}$. Battiti and Protasi [15] have investigated the factors affecting the run-time performance of several heuristics utilized in a reactive local search algorithm, and found that the run-time tends to be proportional to the number of missing edges, with dense graphs being easier to solve quickly. Bomze [21] has transformed the landscape of the problem by regularization of the adjacency matrix, and showed that this helps to isolate the global minima, so properties of the adjacency matrix are also likely to impact on algorithm performance. The decision form of the maximum clique problem is equivalent to the decision form of the minimum vertex cover problem [22] since a graph $G = (V,E)$ has a vertex cover W with $|W| \leq C$ if and only if $\overline{W}$ is in a clique in the graph $(V,\overline{E})$ with $|\overline{W}| \geq |V| - C$. This equivalency makes the features discussed earlier for the minimum vertex cover problem suitable candidates for the maximum clique problem, and vice-versa.

It is clear that there are many properties of graphs that can be measured or tested, beyond the connectivity and vertex degree measures that are common to most phase transition studies. For automated algorithm selection, promising directions include considering the spectral properties of the incidence and adjacency matrices [35] which has received relatively little attention.

4.6. Timetabling problems

Timetabling problems, in their most generic form [43], involve assigning classes and teachers to rooms at certain time periods in such a way that no person is assigned to more than one room at any time, and no room has more than one class and teacher in it at any time. Such a timetable is considered to be feasible (meaning clash-free), but a number of context-specific constraints may need to be considered for the timetable to become operational. Timetabling problems studied in the literature range from school timetabling [137], to university examination timetabling [116], and university course timetabling [118]. Further information about several kinds of timetabling problems and a wealth of research resources can be found associated with the International Timetabling Competition [93]. In order to discuss these variants efficiently, we introduce a common notation to describe them mathematically, although this is far from the most efficient integer programming representation of any of the problems. We define an event as the entity requiring assignment to a room and time period as follows:

Let $x_{i,j,k}=1$ if event i is assigned to period j in room k, and 0 otherwise. Suppose there are N events, P periods, M rooms, K classes, and T teachers. In addition, data about the problem is stored in the following matrices:

$\hat{C}_{c,i}=1$ if class c is included in event i (and 0 otherwise); and $\hat{T}_{t,i}=1$ if teacher t is included in event i (and 0 otherwise).

Then a feasible solution to the timetabling problem (TTP) satisfies the following constraints:

$$\sum_k x_{i,j,k}=1 \quad \forall (i,j)\in\{1,\ldots,N\}\times\{1,\ldots P\} \tag{29}$$

$$\sum_j x_{i,j,k}=1 \quad \forall (i,k)\in\{1,\ldots,N\}\times\{1,\ldots M\} \tag{30}$$

$$\sum_i x_{i,j,k}\hat{C}_{c,i}\le 1 \quad \forall (k,j,c)\in\{1,\ldots,M\}\times\{1,\ldots P\}\times\{1,\ldots,K\} \tag{31}$$

$$\sum_i x_{i,j,k}\hat{T}_{t,i}\le 1 \quad \forall (k,j,t)\in\{1,\ldots,M\}\times\{1,\ldots P\}\times\{1,\ldots,T\} \tag{32}$$

$$x_{i,j,k}\in\{0,1\} \quad \forall (i,j,k)\in\{1,\ldots,N\}\times\{1,\ldots P\}\times\{1,\ldots,M\} \tag{33}$$

These constraints are rather similar to assignment constraints, and the allocation of events to time periods can also be viewed as a form of bin-packing. Such timetabling problems are also well represented as a graph coloring problem, where the vertices of the graph represent the N events, edges are present if two events cannot occur at the same time, and the task is to assign one of p colors (where p is the number of time periods available) to each vertex in such a way that no vertices connected by an edge share the same color. Edges can also be weighted by the severity of the clash. Let n_i be the number of students involved in event i, then it makes sense to also include the product weight $n_i n_j$ on the edges (i,j) as a measure of the severity of the clash.

The difficulty of various timetabling problems has been studied, and depends largely on the additional constraints and objectives that are added to the problem beyond these basic feasibility constraints. For example, studies of classic school timetabling, where an event is defined as a pairing of class and teacher, have shown [137] that the computational complexity of the problem is affected by adding or removing constraints such as block requirements (where 2 lessons are blocked consecutively). Exam timetabling defines an event as an exam paper, and adds hard constraints related to seat capacity for the rooms in addition to avoiding student clashes. A soft constraint that is often added is to avoid students undertaking exams in two consecutive time periods. Adding the seat capacity constraint requires the addition of a knapsack-capacity type of constraint, similar to Eq. (18), and the seat capacity relative to the number of students clearly would have a strong influence over the difficulty of an instance. Ross et al. [116] explored the effect of seat capacity, as well as graph metrics such as edge density, and concluded that edge density alone was no guide to the difficulty experienced by their genetic algorithm. In fact much of the explanation they found for the failure of the GA to solve certain problems had to do with the encoding used by the GA to represent the problem, and the failure to simultaneously tackle the bin-packing problem component, especially for large exams. Gent and Walsh [53] considered exam timetabling with the additional constraint that lecturers could exclude certain exams from being scheduled in specific time slots. They compared the properties of randomly generated instances with real-world instances and found significant differences in the kinds of large scale structures and properties of the resulting graphs. In particular, the real-world examination scheduling problems they considered frequently contained large cliques which made feasible solutions impossible to find when the number of time periods was insufficient to ensure that each of the vertices in the clique was assigned a unique color. Other studies of examination timetabling have shown the importance of the topology of the graph and its influence on difficulty for various algorithms [115], including the existence of phase transitions based on connectivity and homogeneity.

University course timetabling defines an event as a course or unit of study, where each course is associated with a list of students, a set of required features of a room (e.g. data projector, blackboard, etc.), and sometimes the name of the lecturer who will teach the course (although the teaching allocation is often treated as part of the problem). Data is provided about the number of available rooms, their capacities and their features. The hard constraints of the course timetabling problem are to avoid student and room clashes, and to ensure that the room allocated has the required features. Additional soft constraints are often considered including avoiding any student having two consecutive classes, a class in the last period of the day, or only one class on a given day. Several large scale studies [31,118] have compared the performance of a variety of meta-heuristic approaches on this challenging problem. Chiarandini and Stutzle [31] concluded that the hardness of an instance for a given algorithm appears to be an intrinsic property of the instance itself, rather than depending on the algorithm. Instances which were hard for one algorithm tended to be hard for all algorithms. A detailed analysis of run-time distributions, correlated with various properties of the instances was used to explore what makes an instance difficult. Apart from the parameters that determine the size of the instance (the number of events, the number of students, and the maximum number of events per student), they explored the impact of the maximum number of students per event, and the number of rooms available, as well as a ratio of the average number of features per room compared to the number of features available. This last ratio was suggested to behave like a phase transition parameter, with the degrees of freedom considerably increased in the search space if many rooms are well equipped [31]. Rossi-Doria et al. [118] observed that, based on the parameters of the instance generator (but not specially constructed features), it was not possible to predict which meta-heuristic would perform best for a given instance, but they did observe that heuristics that tackle the problem as a two-stage solution, solving the bin-packing problem first, tend to perform better. This approach was taken in Lewis and Paechter [83] who designed a special-purpose genetic algorithm to solve the bin-packing sub-problem first to find a feasible solution, and then to satisfy the soft constraints. Exploring the effectiveness of this approach on various algorithms, and the dependence of the instance difficulty on factors known to cause challenges for bin-packing (as discussed in Section 4.4) was not considered however.

The closest approach to automated algorithm selection for timetabling problems comes from a study of how well

$\mathcal{MAX}-\mathcal{MIN}$ ant systems perform on a set of randomly generated university course timetabling instances [79]. The authors measure hardness as proportional to the solution quality achieved after a fixed running time of the algorithm, and developed a regression model to predict hardness based on a refined set of 8 features (pruned using a cross-validation method from an original candidate set of 27 features). Four of these selected features are based on graph representations of the problem which include only student clashes for one graph, and then student and room clashes for a second graph. The four features selected from the graph properties are the average weighted vertex degree (weighted by the number of student clashes), the average number of students per event, the average vertex degree in the first graph, and the standard deviation of the average vertex degree in the second graph. Other graph properties were considered insignificant for the linear model. Additional selected features included statistical properties of room constraints: the average weighted number of room options per event, the number of events with only one room option, and the standard deviation of the number of events that were suited to a given room. The most important feature according to their model, was the slackness of the instance, defined as the number of rooms multiplied by the number of available time periods. If this quantity exceeds the number of events that need to be timetabled, then the instance contains slackness and is more easily solved, otherwise it is constrained. The full set of candidate features considered in this study is not included in the paper [79], but there is certainly room to consider a wide range of graph properties, and non-linear models, to extend this work and gain insights into hardness prediction. The chromatic number of a graph, which is the number of colors needed to color the graph to find a clash-free solution to the timetabling problem, is often greater than the number of available time periods, and surely provides some valuable information about the difficulty of an instance. While calculating the chromatic number may not be efficient as a feature of the instance for automated algorithm selection, a recent study has shown that the clustering coefficient and edge density combined are useful predictors of chromatic numbers of timetabling conflict graphs [17]. The clustering coefficient is defined as the probability that two neighboring vertices of a given vertex are also neighbors of each other, and may provide another useful feature to characterize instance difficulty for graph based problems like timetabling. Our previous work [129] has shown that features related to the timetabling problem and the graph coloring problem can be combined to partition the instance space into different classes of instances, with strong correlation to the source of the instances (randomly generated versus real-world).

4.7. Constraint satisfaction problems

This paper has focused on optimization problems, which typically involve satisfying a set of constraints while minimizing or maximizing some objective function. There is an enormous body of literature, predominantly from the AI community, focused on constraint satisfaction problems (with no objective function) such as the SAT problem: a generic constraint satisfaction problem with binary variables and arbitrary constraints expressed as clauses. In recent years, many studies have contributed considerable progress towards studying the empirical hardness of instances of such problems. Selman et al. [123] demonstrated the existence of algorithm-independent phase transitions for random 3-SAT problems, and later provided a formal proof [96], showing that when the ratio of the number of clauses to variables is approximately 4.26 the instance becomes hard (i.e. takes longer to solve for all algorithms). Following these findings, much research has been conducted in the AI community to study algorithm performance empirically, particularly focusing on identifying features that correlate with the empirical hardness of problem instances (see for example [1,24,70,84,85,97,124,155]). Much of this work falls well under the automated algorithm selection framework of Rice [110], where regression models are used to predict the (usually run-time) performance of algorithms based on features of the instances. Typical features used to characterize the properties of SAT instances include metrics of the size of instance (number of clauses and variables, and several ratios), statistical features of the variable–clause graph, variable graph, and clause graph, ratios of positive and negative literals in each clause, features based on the LP relaxation (objective value, fraction of binary variables, variable integer slack statistics), search space size estimates based on DPLL algorithm statistics, and often several measures based on local search statistics [97]. Since this paper is focused on optimization problems, rather than constraint satisfaction problems, we refer the reader to the survey paper of Smith-Miles [127] and the references in this section for further details of approaches to measuring the difficulty of SAT and other constraint satisfaction problem in the context of algorithm selection.

5. Discussion and future directions

In our previous work [127], we have provided a unified perspective on the multi-disciplinary literature on algorithm selection utilizing the framework of Rice [110], and shown how the meta-data generated can be used for both automated algorithm selection as well as obtaining insights into *why* certain algorithms perform well on certain classes of problems and not others. The primary challenge to doing this effectively is the quality of the meta-data, and the dependence on choosing suitable features of the instances to reveal relationships to algorithm performance. This paper has focused on how suitable features can be devised for various optimization problems. The definition of whether the features are suitable depends greatly on our goal. If we are restricting our attention to automated algorithm selection, as a form of time-saving to avoid future trial and error approaches to finding the best algorithm for a given instance, then features are only suitable if the time taken to calculate them is significantly less than the time taken to run all of the candidate algorithms. Landmarking approaches [101] have been shown to be useful as features to quickly characterize an optimization problem instance (e.g. by relaxing integrality constraints and taking information from a linear programming solution as a proxy measure of instance difficulty).

While many studies have measured the properties of particular combinatorial optimization problems, it is also apparent that there is very little borrowing of concepts and ideas for suitable features between problem classes. Many of the problems we have considered in this paper are related to each other in pivotal ways, such as the one-to-one assignment constraints found in QAP, TSP, TTP; the capacity constraints found in GQAP, MKP, BPP, TPP; the relationship between TPP, BPP, and graph coloring, etc. Understanding these relationships helps us to identify opportunities for considering new features to characterize problem difficulty based on the shared properties of the problems. Some of the more revealing features that seem to correlate with instance difficulty (either algorithm-dependent, or as an algorithm-independent phase transition control parameter) include:

- for problems with capacity constraints—slackness metrics, the correlations between capacities and profit coefficients, ratios of coefficients to capacities, the ranges of the coefficients;
- for problems with matrices—the dominance and sparsity of the matrices, the distribution of the matrix entries, spectral properties of the matrices;

- for problems involving distances—the clustering properties of the locations, the degree to which the triangle inequality is satisfied;
- for problems with graph representations—the statistical properties of the graph (vertex degrees and edge density), the size of the cliques, the spectral properties of the adjacency and Laplacian matrices of the graph.

When statistical properties of problem instances are described, they are invariably restricted to the mean and variance of some observation, but additional statistical metrics that consider distributions may create an additional wealth of candidate features from which to learn the relationships to algorithm performance.

For automated algorithm selection for combinatorial optimization problems, only a few studies have utilized the extended methodology of Rice [110] to devise suitable features of problem instances and learn their relationship to algorithm performance, enabling prediction of the algorithm most likely to be best for a new instance. Studies of this nature include Nudelman et al. [97] and Xu et al. [155] for SAT problems, Cho et al. [32] and Hall and Posner [65] for knapsack problems, Leyton-Brown et al. [84] for combinatorial auctions, and Smith-Miles and colleagues for the TSP [130,126], QAP [128], timetabling [129], and job shop scheduling [131]. Most of these studies have used regression models to learn the linear relationship between the features and algorithm performance, with the exception of the studies of Smith-Miles [128–131,126] which have used non-linear modeling (neural networks) and rule-based learners, as well as clustering to infer relationships and visualize the similarities and differences between classes of instances. Clearly there is significantly more potential to apply this approach to a wider range of optimization problems, and to utilize the full repertoire of knowledge discovery processes to explore the relationships that may be hidden in the meta-data.

Further opportunities exist to extend the definition of an algorithm to include a particular instantiation of an algorithm, with unique parameter setting. Thus the same optimization algorithm run with different parameter settings (e.g. simulated annealing with two different cooling schedules) can be viewed as two different algorithms in the portfolio, and we can use this methodology to explore the optimal parameter settings for an algorithm for different classes of problem instances. Parameter optimization [72,73] falls well within the scope of intelligent optimization and algorithm selection frameworks.

For the second question posed by this paper, understanding which instance classes match the strengths and weaknesses of different algorithms, there is much more work to be done. Firstly, our hypothesis that the richest possible set of features will come from a combination of both problem-independent and problem-specific metrics needs to be confirmed or denied. Do generic features provide additional discrimination or explanatory power beyond those provided by problem-specific metrics? Could new generic features be derived that make obsolete problem-specific metrics, and lead the way to a generic approach for all optimization problems? We suspect not, but this issue deserves to be explored in greater depth. Critical to this question is an understanding of when the meta-data is sufficient for the required insights to be revealed by data mining techniques. We will be developing a mathematical definition of sufficiency of the meta-data in future research. Instance generation techniques, and commonly used benchmark test instances, must come under the spotlight to determine if they provide sufficient coverage of the instance space, and we have already begun to explore this issue [88]. We will also need to develop a mechanism for visualizing the strengths and weaknesses of an algorithm in the high-dimensional instance space characterized by relevant features. Recent ideas on this topic have defined an algorithm *footprint* [39] as the boundary in instance space where strong algorithm performance is expected to generalize beyond the training examples. The mathematical definition of this boundary and its visualization in high-dimensional instance space is an exciting challenge for future research.

6. Conclusions

The aim of this paper has been to present a coherent summary of work relevant to measuring the difficulty of instances of combinatorial optimization problems. Such measures are critical for identifying the particular conditions under which an algorithm is likely to perform well. In order to provide a starting point for researchers focused on this task, we have provided a comprehensive review of the numerous studies over many decades that have sought to understand the intrinsic properties that determine the difficulty of a set of problem instances for certain algorithms. While there is much literature that has established easy–hard phase transition control parameters for a particular optimization problem, there is little work that seeks to generalize what has been learned to provide possible measures of instance difficulty for other related problems. It is timely to take stock of what has been learned about specific optimization problems, and how this knowledge can be applied to related problems. In this paper we have presented generalized formulations of classical problems. A wide collection of problem-specific features have been reviewed, and we have discussed the opportunities to adapt these features to other problems exhibiting similar structures. Our generalized formulations are intended to reveal the relationships between specific problem formulations, and to suggest suitable features from related problems that may be predictive of algorithm performance. We have also reviewed a number of problem-independent features of an optimization problem in the form of landscape analysis. Landscape metrics are usually not suitable as features for automated algorithm selection, due to the fact that they require knowledge of all local and global minima of the problem. Nevertheless, they can provide a useful set of features to explore why some algorithms struggle on certain classes of problem instances, and can assist with the goal of insight and improved algorithm design.

The algorithm selection framework of Rice provides a firm foundation upon which future research can develop. The construction of candidate features to measure instance difficulty is an important first step. If the quality of the meta-data is sufficient, particularly in how we generate a diverse enough set of test instances while measuring and selecting the "right features", we will have greater chance of success when answering both questions addressed in this paper: *which algorithm in a broad portfolio is likely to be best for a relevant set of problem instances*? As well as the more ambitious question, *for which types of problem instances can we expect a given algorithm in a portfolio to perform well, and why*?

Acknowledgments

The authors are grateful to the two reviewers and editor for their valuable suggestions which greatly improved the clarity of this paper.

References

[1] Achlioptas D, Naor A, Peres Y. Rigorous location of phase transitions in hard optimization problems. Nature 2005;435(7043):759–64.
[2] Ali S, Smith KA. On learning algorithm selection for classification. Applied Soft Computing Journal 2006;6(2):119–38.
[3] Alon N, Fischer E, Krivelevich M, Szegedy M. Efficient testing of large graphs. Combinatorica 2000;20(4):451–76.
[4] Alon N, Fischer E, Newman I, Shapira A. A combinatorial characterization of the testable graph properties: it's all about regularity. In: Proceedings of the

38th annual ACM symposium on theory of computing. New York, NY, USA: ACM; 2006. p. 251–60.
[5] Angel E, Zissimopoulos V. On the classification of NP-complete problems in terms of their correlation coefficient. Discrete Applied Mathematics 2000;99(1–3):261–77.
[6] Angel E, Zissimopoulos V. On the hardness of the quadratic assignment problem with metaheuristics. Journal of Heuristics 2002;8(4):399–414.
[7] Anstreicher K, Brixius N, Goux JP, Linderoth J. Solving large quadratic assignment problems on computational grids. Mathematical Programming 2002;91(3):563–88.
[8] Anstreicher KM, Brixius NW. A new bound for the quadratic assignment problem based on convex quadratic programming. Mathematical Programming 2001;89(3):341–57.
[9] Avis D. A note on some computationally difficult set covering problems. Mathematical Programming 1980;18(1):138–45.
[10] Bachelet V. Métaheuristiques parallèles hybrides: application au problème d'affectation quadratique. PhD thesis, Universite des Sciences et Technologies de Lille; 1999.
[11] Balas E, Zemel E. An algorithm for large zero–one knapsack problems. Operations Research 1980:1130–54.
[12] Barr RS, Golden BL, Kelly JP, Resende MGC, Stewart WR. Designing and reporting on computational experiments with heuristic methods. Journal of Heuristics 1995;1(1):9–32.
[13] Barthel W, Hartmann AK. Clustering analysis of the ground-state structure of the vertex-cover problem. Physical Review E 2004;70(6):66120.
[14] Battiti R. Reactive self-search: toward tuning heuristics. In: Modern heuristic search methods; 1996. p. 61–83.
[15] Battiti R, Protasi M. Reactive local search for the maximum clique problem 1. Algorithmica 2001;29(4):610–37.
[16] Beasley JE. OR-library: distributing test problems by electronic mail. Journal of the Operational Research Society 1990:1069–72.
[17] Beyrouthy C, Burke EK, McCollum B, McMullan P, Parkes AJ. Enrollment generators, clustering and chromatic numbers. In: Proceedings of the 7th international conference on the practice and theory of automated timetabling (PATAT 2008), Montreal, Canada; 2008.
[18] Bierwirth C, Mattfeld DC, Watson JP. Landscape regularity and random walks for the job-shop scheduling problem. In: Lecture notes in computer science, vol. 3004; 2004. p. 21–30.
[19] Birattari M, Balaprakash P, Dorigo M. The ACO/F-RACE algorithm for combinatorial optimization under uncertainty. In: Metaheuristics-progress in complex systems optimization. Operations research/computer science interfaces series. Berlin, Germany: Springer Verlag; 2006. p. 189–203.
[20] Bollobas B. Modern graph theory. Springer Verlag; 1998.
[21] Bomze IM. Evolution towards the maximum clique. Journal of Global Optimization 1997;10(2):143–64.
[22] Bomze IM, Budinich M, Pardalos PM, Pelillo M. The maximum clique problem. In: Handbook of combinatorial optimization, vol. 4(1); 1999. p. 1–74.
[23] Borenstein Y, Poli R. Kolmogorov complexity, optimization and hardness. In: IEEE congress on evolutionary computation, 2006. CEC 2006; 2006. p. 112–9.
[24] Boukeas G, Halatsis C, Zissimopoulos V, Stamatopoulos P. Measures of intrinsic hardness for constraint satisfaction problem instances. In: Lecture notes in computer science, vol. 2932; 2004. p. 184–95.
[25] Brandstädt A, Spinrad JP. Graph classes: a survey. Society for Industrial Mathematics; 1999.
[26] Brazdil PB, Soares C, Da Costa JP. Ranking learning algorithms: using IBL and meta-learning on accuracy and time results. Machine Learning 2003;50(3):251–77.
[27] Burer S, Vandenbussche D. Solving lift-and-project relaxations of binary integer programs. SIAM Journal on Optimization 2006;16(3):726–50.
[28] Burke E, Kendall G, Newall J, Hart E, Ross P, Schulenburg S. Hyper-heuristics: an emerging direction in modern search technology. In: International series in operations research and management science; 2003. p. 457–74.
[29] Cario MC, Clifford JJ, Hill RR, Yang I, Yang K, Reilly CH. An investigation of the relationship between problem characteristics and algorithm performance: a case study of the GAP. IIE Transactions 2002;34(3):297–312.
[30] Cheeseman P, Kanefsky B, Taylor WM. Where the really hard problems are. In: Proceedings of the 12th IJCAI; 1991. p. 331–7.
[31] Chiarandini M, Stutzle T. Experimental evaluation of course timetabling algorithms. Technical Report, Technical Report AIDA-02-05, FG Intellektik, TU Darmstadt; 2002.
[32] Cho YK, Moore JT, Hill RR, Reilly CH. Exploiting empirical knowledge for bi-dimensional knapsack problem heuristics. International Journal of Industrial and Systems Engineering 2008;3(5):530–48.
[33] Christofides N. Worst-case analysis of a new heuristic for the traveling salesman problem, Technical Report, Report 388, Graduate School of Industrial Administration, Carnegie Mellon University; 1976.
[34] Chung CS, Hung MS, Rom WO. A hard knapsack problem. Naval Research Logistics 1988;35(1).
[35] Chung FRK. Spectral graph theory. American Mathematical Society; 1997.
[36] Chvatal V. Hard knapsack problems. Operations Research 1980:1402–11.
[37] Clearwater SH, Hogg T. Problem structure heuristics and scaling behavior for genetic algorithms. Artificial Intelligence 1996;81(1–2):327–47.
[38] Coffman Jr EG, Garey MR, Johnson DS. Approximation algorithms for bin packing: a survey. In: Approximation algorithms for NP-hard problems. PWS Publishing Co.; 1996. p. 46–93. ISBN 0534949681.
[39] Corne DW, Reynolds AP. Optimisation and generalisation: footprints in instance space. In: Proceedings of the 11th international conference on parallel problem solving from nature: part I. Springer-Verlag; 2010. p. 22–31. ISBN 3642158439.
[40] Crescenzi P, Kann V. Approximation on the web: a compendium of NP optimization problems. Randomization and Approximation Techniques in Computer Science 1997:111–8.
[41] Culberson JC. On the futility of blind search: an algorithmic view of "no free lunch". Evolutionary Computation 1998;6(2):109–27.
[42] Culberson JC, Luo F. Exploring the k-colorable landscape with iterated greedy. In: Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11–13, 1993; 1996. p. 245.
[43] de Werra D. An introduction to timetabling. European Journal of Operational Research 1985;19(2):151–62.
[44] Deb Jr K. Multi-objective genetic algorithms: problem difficulties and construction of test problems. Evolutionary Computation 1999;7(3):205–30.
[45] Drezner Z, Hahn PM, Taillard ÉD. Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods. Annals of Operations Research 2005;139(1):65–94.
[46] Eiben AE, Van Der Hauw JK, Van Hemert JI. Graph coloring with adaptive evolutionary algorithms. Journal of Heuristics 1998;4(1):25–46.
[47] Erdős P, Rényi A. On random graphs I. Publicationes Mathematicae Debrecen 1959;6:290–7.
[48] Falkenauer E. Tapping the full power of genetic algorithm through suitable representation and local optimization: application to bin packing. Evolutionary Algorithms in Management Applications 1995:167–82.
[49] Frieze A, Sorkin GB. The probabilistic relationship between the assignment and asymmetric traveling salesman problems. In: Proceedings of the 12th annual ACM–SIAM symposium on discrete algorithms. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics; 2001. p. 652–60.
[50] Fulkerson DR, Trotter LE, Nemhouser GL. Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triple systems. Mathematical Programming Study 1974;2:72–84.
[51] Gagliolo M, Schmidhuber J. Learning dynamic algorithm portfolios. Annals of Mathematics and Artificial Intelligence 2006;47(3):295–328.
[52] Gent IP. Heuristic solution of open bin packing problems. Journal of Heuristics 1998;3(4):299–304.
[53] Gent IP, Walsh T. Phase transitions from real computational problems. In: Proceedings of the 8th international symposium on artificial intelligence; 1995.
[54] Gent IP, Walsh T. The TSP phase transition. Artificial Intelligence 1996;88(1–2):349–58.
[55] Ghosh D, Tathagata B, Ghosh D, Tathagata B. Spotting difficult weakly correlated binary knapsack problems. Technical Report, Indian Institute of Management Ahmedabad, (IIMA) Working Papers 2006-01-04; 2006.
[56] Goldberg DE. In: Genetic algorithms in search and optimization; 1989.
[57] Goldreich O. Combinatorial property testing (a survey). In: Randomization methods in algorithm design: DIMACS Workshop, December 12–14, 1997. American Mathematical Society; 1998. p. 45.
[58] Goldreich O, Ron D. Property testing in bounded degree graphs. Algorithmica 2008;32(2):302–43.
[59] Gomes CP, Selman B. Algorithm portfolio design: theory vs. practice. In: Proceedings of UAI-97; 1997. p. 190–7.
[60] Gotsman C. On graph partitioning, spectral analysis, and digital mesh processing. In: Proceedings of the shape modeling international; 2003. p. 165.
[61] Gras R. How efficient are genetic algorithms to solve high epistasis deceptive problems? In: IEEE congress on evolutionary computation, 2008. CEC 2008. (IEEE world congress on computational intelligence); 2008. p. 242–9.
[62] Gross JL, Yellen J. Graph theory and its applications. CRC Press; 2006.
[63] Guyon I, Elisseeff A. An introduction to variable and feature selection. The Journal of Machine Learning Research 2003;3:1157–82.
[64] Hall NG, Posner ME. Generating experimental data for computational testing with machine scheduling applications. Operations Research 2001:854–65.
[65] Hall NG, Posner ME. Performance prediction and preselection for optimization and heuristic solution procedures. Operations Research 2007;55(4):703.
[66] Hartmann AK, Weigt M. Statistical mechanics of the vertex-cover problem. Journal of Physics A—Mathematical and General 2003;36(43):11069–94.
[67] Hartmann AK, Weigt M. Phase transitions in combinatorial optimization problems: basics, algorithms and statistical mechanics. VCH Verlagsgesellschaft Mbh; 2005.
[68] Hill RR, Reilly CH. The effects of coefficient correlation structure in two-dimensional knapsack problems on solution procedure performance. Management Science 2000:302–17.
[69] Hooker JN. Testing heuristics: we have it all wrong. Journal of Heuristics 1995;1(1):33–42.

[70] Hoos HH, Stutzle T. Towards a characterisation of the behaviour of stochastic local search algorithms for SAT. Artificial Intelligence 1999;112(1–2): 213–32.
[71] Horvitz E, Ruan Y, Gomes C, Kautz H, Selman B, Chickering M. A Bayesian approach to tackling hard computational problems. In: Proceedings the 17th conference on uncertainty in artificial intelligence (UAI-2001), vol. 216; 2001.
[72] Hutter F, Hoos HH, Leyton-Brown K, Murphy KP. An experimental investigation of model-based parameter optimisation: SPO and beyond. In: Proceedings of the 11th annual conference on genetic and evolutionary computation. ACM; 2009. p. 271–8.
[73] Hutter F, Hoos HH, Leyton-Brown K, Stuetzle T. ParamILS: an automatic algorithm configuration framework. Journal of Artificial Intelligence Research 2009;36(1):267–306.
[74] Johnson D, Gutin G, McGeoch L, Yeo A, Zhang W, Zverovitch A. Experimental analysis of heuristics for the ATSP. The traveling salesman problem and its variations.2004. p. 445–87.
[75] Jones T, Forrest S. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Proceedings of the 6th international conference on genetic algorithms; 1995.
[76] Kilby P, Slaney J, Walsh T. The backbone of the travelling sales person. In: International joint conference on artificial intelligence, vol. 19; 2005. p. 175.
[77] Knowles J, Corne D. Towards landscape analyses to inform the design of a hybrid local search for the multiobjective quadratic assignment problem. Soft computing systems: design, management and applications, vol. 12.2002. p. 271–9.
[78] Komlos J, Simonovits M. Szemerédi's regularity lemma and its applications in graph theory. Technical Report, Center for Discrete Mathematics & Theoretical Computer Science; 1995.
[79] Kostuch P, Socha K. Hardness prediction for the university course timetabling problem. In: Lecture notes in computer science, vol. 3004; 2004. p. 135–44.
[80] Krząkała F, Pagnani A, Weigt M. Threshold values stability analysis and high-q asymptotics for the coloring problem on random graphs. Physical Review E 2004;70(4):46705.
[81] Kulanoot A. Algorithms for some hard knapsack problems. PhD thesis, Curtin University of Technology; 2000.
[82] Lawler EL. The quadratic assignment problem. Management Science 1963:586–99.
[83] Lewis R, Paechter B. Application of the grouping genetic algorithm to university course timetabling. In: Lecture notes in computer science, vol. 3448; 2005. p. 144–53.
[84] Leyton-Brown K, Nudelman E, Shoham Y. Learning the empirical hardness of optimization problems: the case of combinatorial auctions. In: Lecture notes in computer science, vol. 2470; 2002. p. 556–72.
[85] Leyton-Brown K, Nudelman E, Andrew G, McFadden J, Shoham Y. A portfolio approach to algorithm selection. In: International joint conference on artificial intelligence, vol. 18; 2003. p. 1542–3.
[86] Lin S, Kernighan BW. An efficient heuristic algorithm for the traveling salesman problem. Operations Research 1973;21(2).
[87] Locatelli M, Wood GR. Objective function features providing barriers to rapid global optimization. Journal of Global Optimization 2005;31(4): 549–65.
[88] Lopes L, Smith-Miles KA. Generating applicable synthetic instances for branch problems. Operations Research, under review.
[89] Macready WG, Wolpert DH. What makes an optimization problem hard. Complexity 1996;5:40–6.
[90] Maia de Abreu NM, Netto POB, Querido TM, Gouvea EF. Classes of quadratic assignment problem instances: isomorphism and difficulty measure using a statistical approach. Discrete Applied Mathematics 2002;124(1–3):103–16.
[91] Maron O, Moore AW. The racing algorithm: model selection for lazy learners. Artificial Intelligence Review 1997;11(1):193–225.
[92] Martello S, Pisinger D, Toth P. Dynamic programming and strong bounds for the 0–1 knapsack problem. Management Science 1999:414–24.
[93] McCollum B, Schaerf A, Paechter B, McMullan P, Lewis R, Parkes AJ, et al. Setting the research agenda in automated timetabling: the second international timetabling competition. INFORMS Journal on Computing 2010;22(1):120–30.
[94] Merz P. Advanced fitness landscape analysis and the performance of memetic algorithms. Evolutionary Computation 2004;12(3):303–25.
[95] Merz P, Freisleben B. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. IEEE Transactions on Evolutionary Computation 2000;4(4):337–52.
[96] Monasson R, Zecchina R, Kirkpatrick S, Selman B, Troyansky L. Determining computational complexity from characteristic'phase transitions. Nature 1999;400(6740):133–7. ISSN 0028-0836.
[97] Nudelman E, Leyton-Brown K, Hoos HH, Devkar A, Shoham Y. Understanding random SAT: beyond the clauses-to-variables ratio. In: Lecture notes in computer science, vol. 3258; 2004. p. 438–52.
[98] Oshikiri G. Cheeger constant and connectivity of graphs. Interdisciplinary Information Sciences 2002;8(2):147–50.
[99] Ou J. Edge cuts leaving components of order at least m. Discrete Mathematics 2005;305(1–3):365–71.
[100] Papadimitriou CH, Steiglitz K. Some examples of difficult traveling salesman problems. Operations Research 1978:434–43.
[101] Pfahringer B, Bensusan H, Giraud-Carrier CG. Meta-learning by landmarking various learning algorithms. In: Proceedings of the 17th international conference on machine learning table of contents. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 2000. p. 743–50.
[102] Pisinger D. Where are the hard knapsack problems? Computers and Operations Research 2005;32(9):2271–84.
[103] Povh J, Rendl F. Compositive and semidefinite relaxations of the quadratic assignment problem. Discrete Optimization 2009;6(3):231–41.
[104] Pudil P, Novovičová J, Kittler J. Floating search methods in feature selection. Pattern Recognition Letters 1994;15(11):1119–25. ISSN 0167-8655.
[105] Quick RJ, Rayward-Smith VJ, Smith GD. Fitness distance correlation and ridge functions. In: Lecture notes in computer science; 1998. p. 77–86.
[106] Ramakrishnan KG, Resende M, Ramachandran B, Pekny JF. Tight QAP bounds via linear programming. Series on Applied Mathematics 2002;14:297–304.
[107] Ramakrishnan N, Rice JR, Houstis EN. GAUSS: an online algorithm selection system for numerical quadrature. Advances in Engineering Software 2002;33(1):27–36.
[108] Reeves CR. Landscapes, operators and heuristic search. Annals of Operations Research 1999;86:473–90.
[109] Reilly CH. Synthetic optimization problem generation: show us the correlations. INFORMS Journal on Computing 2009;21(3):458–67. ISSN 1526-5528. doi: http://dx.doi.org/10.1287/ijoc.1090.0330.
[110] Rice JR. The algorithm selection problem. Advances in Computers 1976: 65–118.
[111] Rice JR. Methodology for the algorithm selection problem. In: Performance evaluation of numerical software: proceedings of the IFIP tc 2.5 working conference on performance evaluation of numerical software, vol. 301. North-Holland; 1979.
[112] Ridge E, Kudenko D. An analysis of problem difficulty for a class of optimisation heuristics. In: Lecture notes in computer science, vol. 4446; 2007. p. 198.
[113] Ron D. Property testing. Combinatorial Optimization 2001;9(2):597–643.
[114] Rose H, Ebeling W, Asselmeyer T. The density of states—a measure of the difficulty of optimisation problems. In: Parallel problem solving from nature—PPSN IV: international conference on evolutionary computation, the 4th international conference on parallel problem solving from nature, Berlin, Germany, September 22–26, 1996: proceedings. Springer Verlag; 1996. p. 208.
[115] Ross P, Corne D, Terashima-Marín H. The phase-transition niche for evolutionary algorithms in timetabling. In: Practice and theory of automated timetabling: first international conference, Edinburgh, UK, August 29–Septmber 1, 1995: selected papers. Springer Verlag; 1996. p. 309.
[116] Ross P, Hart E, Corne D. Some observations about GA-based exam timetabling. In: Lecture notes in computer science, vol. 1408; 1997. p. 115–29.
[117] Ross P, Marin-Blazquez JG, Schulenburg S, Hart E. Learning a procedure that can solve hard bin-packing problems: a new ga-based approach to hyper-heuristics. In: Lecture notes in computer science; 2003. p. 1295–306.
[118] Rossi-Doria O, Sampels M, Birattari M, Chiarandini M, Dorigo M, Gambardella LM, et al. A comparison of the performance of different metaheuristics on the timetabling problem. In: Lecture notes in computer science; 2003. p. 329–54.
[119] Samulowitz H, Memisevic R. Learning to solve QBF. In: Proceedings of the national conference on artificial intelligence, vol. 22. Menlo Park, CA, Cambridge, MA, London: AAAI Press, MIT Press; 1999. p. 255. [2007].
[120] Sassano A. On the facial structure of the set covering polytope. Mathematical Programming 1989;44(1):181–202.
[121] Schiavinotto T, Stützle T. A review of metrics on permutations for search landscape analysis. Computers & Operations Research 2007;34(10):3143–53. ISSN 0305-0548.
[122] Schwerin P, Wascher G. The bin-packing problem: a problem generator and some numerical experiments with FFD packing and MTP. International Transactions in Operational Research 1997;4(5–6):377–89.
[123] Selman B, Mitchell DG, Levesque HJ. Generating hard satisfiability problems. Artificial Intelligence 1996;81(1–2):17–29.
[124] Slaney J, Walsh T. Backbones in optimization and approximation. In: International joint conference on artificial intelligence, vol. 17; 2001. p. 254–9.
[125] Smith K. An argument for abandoning the travelling salesman problem as aneural-network benchmark. IEEE Transactions on Neural Networks 1996;7(6):1542–4.
[126] Smith-Miles KA, van Hemert J. Discovering the suitability of optimisation algorithms by learning from evolved instances. Annals of Mathematics and Artificial Intelligence, doi: 10.1007/s10472-011-9230-5; published online 19th April 2011.
[127] Smith-Miles KA. Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Computing Surveys 2008;41(1).
[128] Smith-Miles KA. Towards insightful algorithm selection for optimisation using meta-learning concepts. In: IEEE international joint conference on neural networks; 2008. p. 4118–24.
[129] Smith-Miles KA, Lopes L. Generalising algorithm performance in instance space: a timetabling case study. In: Lecture notes in computer science, vol. 6683; 2011. p. 524–39.
[130] Smith-Miles KA, van Hemert J. Understanding TSP difficulty by learning from evolved instances. In: Lecture notes in computer science, vol. 6073; 2010. p. 266–80.
[131] Smith-Miles KA, James RJW, Giffin JW, Tu Y. Understanding the relationship between scheduling problem structure and heuristic performance using

knowledge discovery. In: Lecture notes in computer science, vol. 5851; 2009. p. 89–103.
[132] Stadler PF, Schnabl W. The landscape of the traveling salesman problem. Physics Letters A 1992;161(4):337–44.
[133] Streeter M, Golovin D, Smith SF. Combining multiple heuristics online. In: Proceedings of the national conference on artificial intelligence, vol. 22. Menlo Park, CA, Cambridge, MA, London: AAAI Press, MIT Press; 1999. p. 1197. [2007].
[134] Stutzle T, Fernandes S. New benchmark instances for the QAP and the experimental analysis of algorithms. In: Lecture notes in computer science, vol. 3004; 2004. p. 199–209.
[135] Taillard ED. Comparison of iterative searches for the quadratic assignment problem. Location Science 1995;3(2):87–105.
[136] Tavares J, Pereira FB, Costa E. Multidimensional knapsack problem: a fitness landscape analysis. IEEE Transactions on Systems, Man, and Cybernetics, Part B 2008;38(3):604–16.
[137] Ten Eikelder HMM, Willemen RJ. Some complexity aspects of secondary school timetabling problems. In: Proceedings of the 3rd international conference on practice and theory of automated timetabling (PATAT 2000), Lecture Notes in Computer Sciences, vol. 2079; 2000. p. 18–29.
[138] Thiebaux S, Slaney J, Kilby P. Estimating the hardness of optimisation. In: ECAI; 2000. p. 123–30.
[139] Trick M. Formulations and reformulations in integer programming. In: Lecture notes in computer science, vol. 3524; 2005. p. 366–79.
[140] Tsymbal A, Pechenizkiy M, Cunningham P. Diversity in ensemble feature selection. Technical Report TCD-CS-2003-44, The University of Dublin; 2003.
[141] van Hemert JI. Property analysis of symmetric travelling salesman problem instances acquired through evolution. In: Proceedings of the European conference on evolutionary computation in combinatorial optimization (EvoCop 2005). Springer; 2005.
[142] van Hemert JI. Evolving combinatorial problem instances that are difficult to solve. Evolutionary Computation 2006;14(4):433–62.
[143] van Hemert JI, Urquhart NB. Phase transition properties of clustered travelling salesman problem instances generated with evolutionary computation. In: Lecture notes in computer science, vol. 3242; 2004. p. 151–60.
[144] Vasconcelos N. Feature selection by maximum marginal diversity: optimality and implications for visual recognition. In: 2003 IEEE computer society conference on computer vision and pattern recognition, 2003. Proceedings, vol. 1; 2003.
[145] Venkatesan R, Levin L. Random instances of a graph coloring problem are hard. In: Proceedings of the 20th annual ACM symposium on theory of computing. New York, NY, USA: ACM; 1988. p. 217–22.
[146] Vilalta R, Drissi Y. A perspective view and survey of meta-learning. Artificial Intelligence Review 2002;18(2):77–95.
[147] Vollmann TE, Buffa ES. The facilities layout problem in perspective. Management Science 1966:450–68.
[148] Watson RA, Hornby GS, Pollack JB. Modeling building-block interdependency. In: Lecture notes in computer science; 1998. p. 97–108.
[149] Weerawarana S, Houstis EN, Rice JR, Joshi A, Houstis CE. Pythia: a knowledge-based system to select scientific algorithms. ACM Transactions on Mathematical Software 1996;22(4):447–68. ISSN 0098-3500.
[150] Weigt M, Hartmann AK. Number of guards needed by a museum: a phase transition in vertex covering of random graphs. Physical Review Letters 2000;84(26):6118–21.
[151] Weinberger ED. Local properties of Kauffman's NK model: a tunably rugged energy landscape. Physical Review A 1991;44(10):6399–413.
[152] White DR, Harary F. The cohesiveness of blocks in social networks: node connectivity and conditional density. Sociological Methodology 2001: 305–59.
[153] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1997;1(1):67–82.
[154] Xin B, Chen J, Pan F. Problem difficulty analysis for particle swarm optimization: deception and modality. In: Proceedings of the first ACM/SIGEVO summit on genetic and evolutionary computation; 2009. p. 623–30.
[155] Xu L, Hutter F, Hoos HH, Leyton-Brown K. SATzilla-07: the design and analysis of an algorithm portfolio for SAT. In: Lecture notes in computer science, vol. 4741; 2007. p. 712.
[156] Yusta SC. Different metaheuristic strategies to solve the feature selection problem. Pattern Recognition Letters 2009;30(5):525–34. ISSN 0167-8655.
[157] Zhang W. Phase transitions and backbones of the asymmetric traveling salesman problem. Journal of Artificial Intelligence Research 2004;21: 471–97.
[158] Zhang W, Korf RE. A study of complexity transitions on the asymmetric traveling salesman problem. Artificial Intelligence 1996;81(1–2):223–39.