Volume 38(2), pp. 147-175
http://orion.journals.ac.za

ORiON
ISSN 0259–191X (print)
ISSN 2224–0004 (online)

Towards a framework for predicting packing algorithm performance across instance space

RG Rakotonirainy*

Received: 7 July 2022; Revised: 31 October 2022; Accepted: 31 October 2022

Abstract

Finding the conditions under which packing algorithms succeed or fail with respect to a set of test instances is crucial for understanding their strengths and weaknesses, and for automated packing algorithm selection. This paper tackles the important task of objective packing algorithm selection. A framework for understanding the relationship between critical features of packing problem instances and the performance of packing algorithms is proposed. The framework can be used to predict algorithm performance on previously unseen instances with high accuracy and can be applied to find predictions in other instances of cutting and packing problems. It can also be applied to determine the relative strengths and weaknesses of each algorithm within the instance space. The effectiveness of the framework is demonstrated using the two-dimensional strip packing problem as a case study.

Key words: Packing problems, metaheuristics, machine learning.

1 Introduction

Over the past few years, the development of fast and effective packing algorithms — mainly employing heuristic and metaheuristic techniques — has been the major concern of most packing-related research due to the complexity and combinatorial nature of the problem. Various experimental studies have been reported in the literature demonstrating the effectiveness of newly developed algorithms proposed by the authors in comparison with previously published approaches [11, 33, 44]. The popular use of benchmark libraries of packing instances (*e.g.* the repository of ESICUP [14]) helps to standardise the comparisons and performance evaluations of those algorithms.

*Department of Statistical Sciences, University of Cape Town, Rondebosch 7701, South Africa *e-mail:* rosephine.rakotonirainy@uct.ac.za

http://dx.doi.org/10.5784/38-2-741

Assessment of packing algorithm performance is however difficult when the conclusions depend on the chosen test instances. It has been documented that there is a lack of diversity in the benchmark instances employed in the above-mentioned comparative studies and there is a risk that algorithms are developed to perform well on these instances without understanding the effect of benchmark characteristics on the performance of the algorithms [2, 32]. As cautioned by Rakotonirainy [32], there is a need to consider the aspects of the test problems employed during comparative algorithmic studies in order to avoid biased conclusions with respect to the relative performance of algorithms. Moreover, a description of the conditions under which an algorithm can be expected to succeed or fail is rarely included in the study [32]. Furthermore, none of the available methods for selecting the most appropriate algorithm to solve a particular instance in the literature have also been applied in the context of cutting and packing problems.

Yet for the advancement of the field, it is essential to address such research gap. As raised by Smith-Miles and Lopes [40], the true value of a comparative algorithmic study lies in its ability to answer the following questions: "Which algorithm in a (broad) portfolio is likely to be best for a relevant set of problem instances?", "On which type of instances does an algorithm outperform its competitors?", "For which types of problem instances can we expect a given algorithm in a portfolio to perform well, and why?", and "How can we describe those instances"? Answers to these questions first lead to understanding of the conditions under which a particular packing algorithm can be expected to succeed or fail with respect to the features of the benchmark instances, and help in developing improved algorithm design. Addressing these questions also hold the key for uncovering relationships between characteristics of problem instances and algorithm performance, and have implications for effective packing algorithm selection model capable of predicting the algorithm from a given portfolio that is likely to be best for a given (unseen) instance.

A small body of literature exists addressing these questions in the cutting and packing field. In a rare attempt at predicting bin packing algorithm performance predictors, Perez *et al.* [31] proposed a methodology that model the relationship between algorithm performance and characteristics of bin packing problem instances using machine learning techniques. In [40], Smith-Miles and Lopes proposed a methodology for adequately characterising the features of a problem instance and showed how such features can be defined and measured for various optimisation problems including the bin packing problems. They suggested that the methodology could be applied to the task of algorithm selection.

A methodology capable of identifying the strengths and weaknesses of algorithms as well as their relative power with respect to instance space was proposed by Smith-Miles *et al.* [38]. Based on a set of problem instances with various properties, they applied data mining methods to measure algorithm footprint — the boundary in instance space where an algorithm can be expected to perform well, and relate this boundary to properties of instances to infer the relative performance of algorithms across all instances. In the same vein, Smith-Miles *et al.* [41] explored the ideas of footprints of algorithms in the context of graph coloring and demonstrated the use of data mining to reveal the performance of algorithms, their strengths and weaknesses, with respect to the instances space.

Recently, Rakotonirainy [34] developed a methodology for the characterisation of algorithm performance and its application to algorithm selection in the context of strip packing

problems. The proposed methodology consists of two phases: The training phase, during which a set of training test instances is solved with a representative sample of packing algorithms and machine learning techniques are applied to learn the relationship between the algorithm performance and the problem instance characteristics, and the prediction phase, whereby the relationship learned during the training phase is applied to select the best performing algorithm for a given new instance.

While this previous research has generated an initial methodology, it has raised a number of questions that need to addressed for a more comprehensive tool to be developed: How to determine the sufficiency and diversity of the set of test instances? How to select the appropriate features that best represent the problem instances effectively? How to accurately identify the best algorithm for a given instance? How to determine the boundary of which an algorithm is expected to perform well based on limited observations? How to reveal the strengths and weaknesses of a set of algorithms with respect to the characteristics of test instances?

This paper extends the framework proposed by Rakotonirainy [34] with the aim of addressing the aforementioned questions. The methodology adopted in this paper involves a broader agenda: The concern is not only on the identification of the best performing algorithm for a given instance, but also to reveal insights into the relative power of the selected algorithms that were not apparent by considering performance averaged across all instances. The main objective is to propose a data mining-based framework capable of modelling the relationship between instance characteristics and algorithm performance, and for an automated packing algorithm selection. The framework is developed such that groups of problem instances with similar characteristics, and for which an algorithm had a better performance than the others, are learned into formal classifiers, which are predictors that model the relationship between problem characteristics and algorithm performance. Thereafter, the learned classifier can be used to predict the best algorithm to solve new problem instances. Finally, the relative algorithm power is analysed, and insights are drawn from the analysis to explain algorithm strength or weakness by inspecting the distribution of instance characteristics across the instance space.

The methodology is applied to the well-known two-dimensional strip packing problem (2D-SPP). In such problem, the objective is to pack a set of rectangular items orthogonally in a non-overlapping manner into a single, rectangular object of fixed width but unlimited height, such that the resulting height of the packed items is a minimum. This problem finds a wide range of applications, and is typically encountered in the wood, glass and paper industries. Large scale computational studies involving the assessment of the relative performance of a variety of strip packing algorithms across a collection of diverse classes of benchmark instances are investigated. The proposed methodology can be extended to other cutting and packing problems, such as the bin packing problem (minimise number of used bins), the knapsack problem (maximise the total value in the knapsack), and the cutting stock problem (minimise the amount of scrap).

The remainder of this paper is structured as follows. Section 2 is devoted to a brief description of the proposed framework. The main components of the methodology are presented in Section 3. The detailed steps of the methodology when applied to the two-dimensional strip packing problem are then described in Section 4. Discussion of the

results obtained follows in Section 5, and a conclusion along with future research directions is provided in Section 6.

2 Framework: Packing algorithm selection

Recently, Rakotonirainy [34] proposed a framework for automated strip packing algorithm selection, which models the relationship between packing instance characteristics and algorithm performance in an attempt to predict the best algorithms to solve a set of unseen problem instances. The model involves seven essential steps:

- The instance generation is the first step aimed at selecting the most suitable packing instances which help in discovering the limits and behaviour of algorithm performance. The generated instances should be diverse enough that the instance distribution seen in both the training and testing sets are similar;
- The feature selection step consists of selecting the most appropriate features in respect of which to measure the influence of problem characteristics on the algorithm performance;
- The characteristics measurement entails calculation of characteristic values of each instance generated in the first step based on the selected features of step 2;
- The performance evaluation step concerns the generation of feasible solutions of the problem instances by means of packing solution techniques;
- The instance clustering involves a cluster analysis whereby the benchmark instances of step 1 are grouped into different classes of test problems based on their underlying features calculated from step 3;
- In the classification step, the identified grouping in step 5 is learned into formal classifiers, which are predictors that model the relationship between problem characteristics and algorithm performance;
- The relationship learned during the above steps is used to predict the best algorithm to solve a new given instance in the prediction step.

In this paper, the above framework is utilised and extended to consider a broader scope: not only the winning algorithm will be identified but the strengths and weaknesses of algorithms within the instance space will also be defined. Figure 1 presents the proposed framework. This new framework pictures the two phases in the framework of Rakotonirainy [34] with an additional phase, *feedback phase*, aimed at improving the prediction and selection processes performed during the two previous phases. This last phase also enables the analysis of algorithm power, thus facilitates the identification of the relative performance of algorithms across all instances.

The key factors and prerequisites for tackling the algorithm selection problem include the availability of a large set of problem instances with various properties, the extraction of

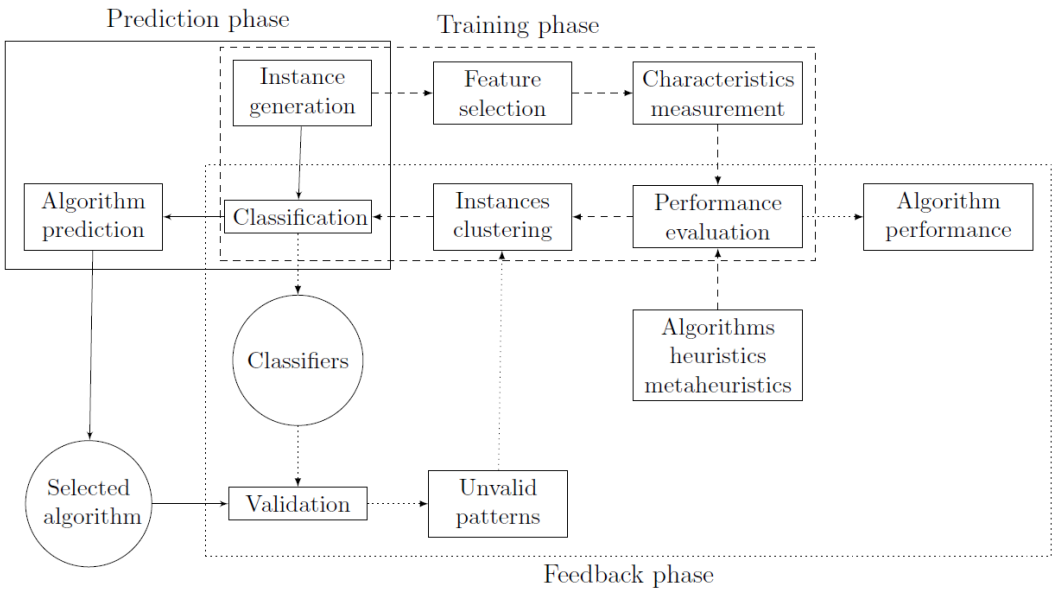


Figure 1: A framework for effective packing algorithm selection.

relevant features to characterise the properties of the instances, the existence of a large number of algorithms for solving problem instances, and performance metrics to evaluate algorithm performance [43]. The choice of instances, and their diversity, play an important role in learning the boundaries of algorithm performance, and also in determining the limits and behaviour of the problem. It is vital to take steps through careful instance generation mechanisms to produce instances that are diverse and well separated. Measuring such diversity could be achieved by visualizing the instances in common space and verifying that they are spatially diverse and are discriminating of algorithm performance, *i.e.* the instances are not equally easy or equally hard for all algorithms.

The diversity of the instances depends on how their characteristics or features are selected and measured. Features must be chosen so that the varying complexities of the problem instances are exposed, any known structural properties of the problems are captured, and any known advantages and limitations of the different algorithms are related to the features. The task of feature selection is critical and is successful if the relative difficulty of the instances for different algorithms can be adequately measured. The methodology presented in this paper provides a feature selection method that creates a useful feature space.

Much can be learnt form the instance and feature spaces. The instance space can be visualised, by means of dimension reduction methods, to determine whether the diversity requirement is achieved by the choice of instances and the feature selection. The performance of algorithms across the instance space can also be visualised to confirm the degree to which the selected instances are discriminating of algorithm performance. More precisely, the diversity of instances and discriminatory algorithm performance in the instance space consist of criteria which enable the selection of an optimal set of features and set of instances. Moreover, machine learning techniques can be used to predict algorithm performance with respect to the feature vector describing an instance, and out-of-sample

testing can be employed to validate predictive power. An approach is presented in this paper to identify the regions in instance space where there is sufficient evidence that good performance can be expected from an algorithm.

It is from within this framework that the following methodology is proposed to develop the computational resources for the cutting and packing community. It is noted that all the components of the framework and the methodology are generic and applicable to a wide variety of cutting and packing problems.

3 Methodology

The proposed methodology consists of three stages:

1 The training stage — a process whereby instances are selected, their features are calculated, and a set of optimal features is generated to model the relationship between instance characteristics and algorithm performance and also to create a high-dimensional summary of the instances in feature space that achieve good separation of the easy and hard instances.

2 The prediction stage — using the feature and instance spaces, machine learning techniques are used to classify the group of instances or the areas where an algorithm is predicted to perform well or poorly, and to identify which algorithm is recommended for which areas of the instance space.

3 The feedback stage — the accuracy of the prediction model is evaluated and constructive recommendation is provided with regards to the performance of an algorithm on previously unseen instances. The relative strength and weakness of each algorithm can also be measured objectively by inspecting the distribution of features across the instance space and conclusion can be drawn about relative algorithm power.

Details of this methodology are now presented before a case study is presented in the next section to illustrate the methodology.

3.1 Training stage

This stage entails the generation of a suitable instance space, which is a complicated interplay between selecting diverse instances, measuring the right features that correlate with instance difficulty, and selecting a set of optimal features that produces the best separation of instances across all algorithms.

3.1.1 Instance generation

It is apparent that a large collection of problem instances is required when adopting the proposed methodology. A large set of instances must be selected, which can be achieved by collecting benchmark problem instances documented in the literature, and if not sufficient by generating new instances from existing benchmark generators. The choice of

instances, and their diversity, play an important role in learning the boundaries of algorithm performance, and also in determining the limits and behaviour of the problem. It is vital to take steps to ensure that the generated instances are as broad as possible and well separated. The key is to include a variety of characteristics of the problem and to avoid biased information.

For packing problems, a large collection of instances exist (*e.g.* in the repository of ESICUP [14]). Often these instances are very similar, which does not provide sufficient diversity with respect to measurable features [26]; hence additional instances may be required to be generated to extend the boundaries of the instance space. A greater variation of the problem characteristics may be achieved by using problem generators' capability to generate a large number of problem instances with different parameters under controllable aspects. An example of a packing problem generator, which will be used later in the case study, is the 2DCPackGen of Silva *et al.* [37].

It should be noted that measuring the diversity of the instances across the instance space requires two considerations: *instance dissimilarity* — the instance should span across a reasonable region in the instance space, which could be measured by the average distance to the centroid of the instances, and *algorithmic discrimination* — the instances should elicit the behaviours and relative performance of each algorithm across the instance space, which could be measured by the relative difference between the worse and best performance metrics averaged across all instances. The process of generating diverse instances is thus iterative.

3.1.2 Feature selection

The feature selection process consists of selecting the most appropriate features in respect of which to cluster the benchmark data instances and to measure the influence of problem characteristics on the algorithm performance. Features must be chosen so that the varying complexities of the problem instances are exposed, any known structural properties of the problems are captured, and any known advantages and limitations of the different algorithms are related to features. Generally, feature selection consists of two-steps: Firstly the determination of all metrics which likely measure the goodness of a set of features and then utilisation of search strategies to find the subset that maximises the goodness metrics. In this methodology, the goodness of a set of features is defined based on the extent to which instances elicit performance of algorithms in the instance space with respect to the set of features. That is, for a candidate set of features, the goodness is measured by how well a machine learning method can discriminate between easy and hard instances and also the relative performance of algorithms in the feature space.

A wide range of feature selection methods have been proposed in the literature, including supervised feature selection approaches and principal component analysis (PCA) for dimensionality reduction (see [17] for a comprehensive review). Any of these methods could be used to select appropriate features in the context of packing problems.

3.1.3 Performance evaluation

Feasible solutions of the problem instances are calculated by means of packing solution techniques. Various approaches have been proposed in the literature for solving cutting and packing problems. These approaches may be classified into the classes of exact methods, heuristic approaches and metaheuristic techniques [32]. Exact methods are typically based on a mathematical programming modelling approach and find a best packing solution, but are slow and may hence only be used to solve small problem instances. Heuristic and metaheuritsic techniques, on the other hand, are approximate solution approaches that attempt to provide near-optimal solutions in minimal time. They are more practical and provide solutions to large problem instances within reasonable time frames.

In this paper, the problem instances are solved using a representative sample of metaheuristic algorithms from the literature. The effectiveness of these algorithms are evaluated by means of a standard performance measure utilised in the packing field. There exist a variety of performance evaluation methods in the literature but the most commonly used computational measure for evaluating the performance of a packing algorithm is the relative difference between the packing height returned by the algorithm and the height of an optimal solution of the problem [21]. This measure is often expressed in terms of a percentage gap or ratio. An alternative performance measure is the packing time efficiency or the time required by the algorithm to find a packing solution for the problem [32]. This computation time may be measured by time tracking during the execution of the algorithm.

3.2 Prediction stage

The instance space can be used for predicting algorithm performance. For the performance prediction task, standard machine learning approaches may be used that use a subset of the instances to learn the relationship between the instance features and the label assigned to each algorithm for each instance to indicate how well the algorithm performed. The benchmark instances can also be grouped into different classes of test instances based on the set of features and the relationship between the instance features and the label assigned to each cluster is learned to elicit the algorithm performance.

3.2.1 Instance clustering and classification

Instance clustering entails a cluster analysis whereby the benchmark instances are grouped into different classes of test problems based on their underlying features. Each group is comprised of instances with similar characteristics, and for which an algorithm had a better performance than the others according to the performance evaluation. Typical clustering analysis involves clustering algorithm design and clustering output assessment. Clustering algorithm design encompasses the selection of a proximity or distance measure, and the choice of an appropriate clustering algorithm for subsequent use. An abundance of clustering algorithms has been proposed in the literature for solving different types of clustering problems in a variety of different fields [6, 46]. There is, however, no clustering algorithm that is generally applicable to all types of clustering problems. It is, therefore,

important to investigate the problem at hand properly in order to select an appropriate clustering method.

Clustering output assessment refers to the process of evaluating the clustering results derived from the selected and employed algorithms for validation purposes. Usually, different clustering techniques result in different clusters, and even for the same algorithm, different input parameters typically lead to different cluster results [25]. Effective evaluation or testing criteria are therefore required for the assessment of the performance of the algorithms considered.

During the classification process, the identified grouping is learned into formal classifiers, which are predictors that model the relationship between problem characteristics and algorithm performance. It should be noted that the inclusion of a clustering process before the classification step is meant that the meaningful characteristics that best describe the data instances are identified and that the relationship between the identified features and algorithm performance is accurately explored. The process of directly classifying the instances based on their characteristics and determining the consequential best performing algorithms will also be discussed in this paper.

Standard machine learning methodologies that use a subset of the instances generated (the training set) to learn the relationship between the instance features and the label assigned to each algorithm (thereof the corresponding cluster) for each instance can be employed during this task. Machine learning classification methods such as decision tree classifiers, Naive Bayes classifiers or support vector machines, can be used accordingly.

3.2.2 Algorithm performance prediction

The relationship learned during the classification process is used to predict the best algorithm to solve a new given instance during the prediction phase. For a new problem instance, which can be generated using the instance generation method of the training stage, its critical characteristic values are measured based on the selected features. Based on these characteristics, the learned classifiers from the classification process are employed to determine the best algorithm for the instance. If a clustering analysis is performed before classification, the learned classifiers are employed to determine the cluster into which the instance belongs to and the algorithm associated to this cluster is the expected best algorithm for the instance.

3.3 Feedback stage

The objective during this stage is to provide feedback on the selection system conducted during the prediction stage and to maintain continuous training. This stage is also used to analyse algorithm power or to measure and reveal the relative strength and weakness of each algorithm across the instance space.

3.3.1 Validation

During this validation process, the selected algorithm obtained from the prediction phase is compared against the real best one to assess the accuracy of the prediction model. More

precisely, for each new characterised instance, whose best algorithm was selected during the prediction phase, the real best algorithm is evaluated. If the prediction is wrong, that is, the real best algorithm does not match the predicted algorithm, or if the average accuracy is out of a specified threshold, then the classifiers are rebuilt using the old and new problem instances; otherwise the new instance is stored and the process ends.

3.3.2 Analysis of algorithm power

With a well-defined description of "good" algorithm performance — an example is to define an algorithm performance to be "good" if it achieves a solution that lies within 1% of the known optimal solution after a fixed run time — each algorithm can be assigned a label of "good" or "bad" and the boundary in instance space between good and bad performance for each algorithm can be visualised. The region in instance space where an algorithm is expected to perform well is called the *footprint* [39]. There exist a number of methods that can be used to measure the relative size of an algorithm footprint. An example is the ratio of the area of the convex hull created by the points where good performance of the algorithm was observed to the area of the hull covering all instances proposed by Smith-Miles [39]. According to this method, the area of the convex hull of a region defined by a set of points $S = \{(x_i, y_i), i = 1, \ldots, n\}$, denoted by $H(S)$, is given by

$$Area(H(S)) = \frac{1}{2}\sum_{j=1}^{k-1}(x_j y_{j+1} - y_j x_{j+1}) + (x_k y_1 - y_k x_1),$$

with the subset $\{(x_j, y_j), j = 1, \ldots, k-1, k \leq n\}$ defining the extreme points of $H(S)$. The relative size of the footprint of a given algorithm is therefore given by the ratio of the area of the convex hull of the algorithm footprint to the area of the hull covering all instances. This method can be used to measure the footprint of algorithms in the context of packing problems.

The relative size of each algorithm footprint provides information on the algorithm strength across the instance space. By understanding 'where' in the instance space an algorithm performs best helps us to draw a conclusion about the power of an algorithm. This could be achieved by measuring the degree of overlap between an algorithm footprint with the footprint of other algorithms. The final analysis to complete the methodology is to explore the instance space to gain insights into how the features of instances affect algorithm footprints. The distribution of each feature across the instance space can be visualised and conclusions can be drawn about the particular properties of the instances that are found in certain regions of the instance space, including those that define the footprint boundaries.

A case study on the 2D-SPP is considered in the following section to illustrate how this methodology can be applied to achieve effective packing algorithm selection, and to generate new insights into the strengths and weaknesses of packing algorithms.

4 Case study: The 2D-SPP

The methodology described in the previous section is demonstrated here when applied to the 2D-SPP. The process of instance generation is presented in Section 4.1, followed by the task of feature selection in Section 4.2. Performance evaluation using seven state-of-the-art strip packing algorithms is described in Section 4.3. This is then followed by the classification process in Section 4.4. The tasks of algorithm prediction and validation are presented in Section 4.5. The analysis of algorithm power is finally discussed in Section 4.6.

4.1 Instance generation

The use of data mining techniques to predict the packing algorithm requires a large problem instance data set, mainly to accurately represent the limits and behaviour of the problem and the variability that the instances accommodate, and also to consider the influence of aspects and characteristics that affect the algorithm performance.

A total of 1 718 benchmark 2D-SPP instances were identified in the literature which are grouped in two classes. The first class consists of zero-waste problem instances for which the respective optimal solutions are known and do not contain any wasted regions (regions of the strip not occupied by items). This class of benchmark instances comprises nine data sets. The second class consists of non-zero-waste instances for which optimal solutions are not known in some cases. Furthermore, those with known optimal solution involve some wasted regions. This second class of problem instances comprises eleven data sets. The main characteristics of these data sets organised by name, number of problem instances, and source are described in Table 1.

The majority of these instances are, however, very similar and are not diverse enough to cover some important aspects of the 2D-SPP [26]. Therefore, a set of 1 680 problem instances generated by the 2DCPackGen problem generator of Silva *et al.* [37] is also considered in this study. The 2DCPackGen allows the generation of problem instances, using different parameters under controllable aspects, and ensures the reproducibility of the data. Specifically in the 2D-SPP, the parameters that must be defined are: the item minimum and maximum size dimension (within the range [1-100]), the strip minimum and maximum width (within the range [10-1,000]), the minimum and maximum number of different item types (within the range [5-500]), and the minimum and maximum number for item type demand (within the range [1-10]). The problem instance variation is fitted using a beta distribution [16]. Different curve behaviours represent different geometric rectangles and strip shapes.

In order to obtain a variety of problem instances, two different classes were generated using the 2DCPackGen problem generator. The first class contains 560 instances grouped into 16 different sets, generated based on the 16 different characteristics of the size and shape of the rectangles. In each set, the strip width, the number of different item types, and the item type demand were assigned values according to the 7 proposed distribution curves, named as subsets. For each subset, 5 similar problem instances were generated, resulting in a total of 35 problem instances in each set.

Set	Authors & Year	Description	Comment
		Zero-waste problem instances	
J	Jakobs (1996) [23]	2 problems instances	Both optimal solutions known
SCP	Hifi (1998) [18]	25 problem instances	All optimal solutions known
babu	Babu and Babu (1999) [1]	1 problem instance	Optimal solution know
C	Hopper and Turton (2001) [21]	21 problem instances	All optimal solutions known and 14 guillotineable
NT(n), NT(t)	Hopper and Turton (2002) [20]	70 problem instances	All optimal solutions known and 35 guillotineable
N	Burke *et al.* (2004) [9]	12 problem instances	All optimal solutions known and guillotineable
CX	Pinto and Oliveira (2005) [15]	7 problem instances	All optimal solutions known
IY	Imahori and Yagiura (2010) [22]	170 problem instances	All optimal solutions known
		Non-zero-waste problem instances	
cgcut	Christofides and Whitlock (1977) [12]	3 problem instances	1 optimal solution known
beng	Bengtsson (1982) [4]	10 problem instances	6 optimal solutions known
gcut, ngcut	Beasley (1895) [3]	25 problem instances	13 guillotineable (2 of which with optimal solutions known) and 12 non-guillotineable (11 of which with optimal solutions known)
bwmv	Berkey and Wang (1987) [5]	300 problem instances	All optimal solutions unknown
bwmv	Martello and Vigo (1998) [29]	200 problem instances	All optimal solutions unknown
DP	Dagli and Poshyanonda (1997) [13]	1 problem instance	Optimal solution unknown
BK	Burke and Kendall (1999) [8]	1 problem instance	Optimal solution known and guillotineable
SCPL	Hifi (1998) [19]	9 problem instances	Optimal solutions unknown
Nice, Path	Wang and Valenzuela (2001) [42]	480 problem instances	Floating-point data sets and all optimal solutions known
AH	Bortfeldt and Gehring (2006) [7]	360 problem instances	Optimal solutions unknown
Zdf	Leung *et al.* (2011) [27]	16 problem instances	5 optimal solutions known
G	Rakotonirainy (2020) [35]	560 problem instances	All optimal solutions unknown
G(m)	Rakotonirainy (2020) [35]	1 120 problem instances	All optimal solutions unknown

Table 1: The 3 398 benchmark problem instances employed in this study.

The second class contains 1 120 instances also grouped into 16 different sets as in the first class. In each set, the strip width, the number of different item types, and the item type demand were varied according to the 7 proposed distribution curves, representing 7 different combinations. For each combination, 10 similar problem instances were generated, resulting in a total of 70 instances in each set. Descriptions of these data instances are given in Table 1 and they can be downloaded from the repository for problem instances published online by Rakotonirainy [35].

The 1 718 data sets from the literature together with the first class (560 newly generated) instances are employed as training data, while the second class (1 120 newly generated benchmark) instances are employed as test data in the classification task.

4.2 Feature selection

The greatest challenge is the derivation of suitable metrics as features to characterise the data sets. Relevant features of the problem parameters need to be identified, and expressions to measure the values of identified critical characteristics must be derived. A methodology based on linear correlations and PCA has been employed by Júnior *et al.* [26] to identify the most significant characteristics for the 2D-SPP benchmark instances. They considered 56 descriptive variables, based on parameters found in the most used packing problem generators, and conducted an exploratory analysis to determine the most relevant characteristics with respect to a set of frequently used benchmark data sets. Their analysis suggested that the problem can be reduced to 19 characteristics, retaining the largest part of the total variance.

The same analysis was performed in this study. In order to avoid redundancy, 4 descriptive variables were considered, where 2 of the variables have 4 components each, resulting in 10 characteristics. These features were selected based on the parameters and characteristics produced by the most popular problem generators as well as some intrinsic factors of the newly generated problem instances. The four descriptive variables are the *aspect ratio* of each item of an instance, the *area ratio* of any pair of items of an instance, the *heterogeneity ratio*, and the *width ratio*.

- The *aspect ratio* of an item i is to be

$$\rho(i) = \frac{d_{max}(i)}{d_{min}(i)},$$

  where $d_{min}(i)$ and $d_{max}(i)$ denote its length along the smaller side dimension and its length along the larger side dimension, respectively.

  Four components of the aspect ratio variable are considered in this study: The *maximum aspect ratio* of all items of an instance, which is determined by $\rho_{max} = \max\{\rho(1), \ldots, \rho(n)\}$; the *minimum aspect ratio* of all items of an instance, which is given by $\rho_{min} = \min\{\rho(1), \ldots, \rho(n)\}$; the *mean aspect ratio* of all items of an instance, which is given by $\rho_{mean} = \text{mean}\{\rho(1), \ldots, \rho(n)\}$; and the *variance aspect ratio* of all items of an instance, which is given by $\rho_{var} = \text{var}\{\rho(1), \ldots, \rho(n)\}$. The parameter n represents the total number of items involved in the given instance.

- The *area ratio* of a pair of items i, j is given by

$$\gamma(i,j) = \frac{a(i)}{a(j)},$$

where $a(i)$ denotes the area of item i.

Four components of the area ratio variable are also considered. The *maximum area ratio* of all pairs of items of an instance, which is defined as $\gamma_{max} = \max\{\gamma(i,j) \mid i,j = 1,\ldots,n;\ i \neq j\}$; the *minimum area ratio* of all pairs of items of an instance, which is given by $\gamma_{min} = \min\{\gamma(i,j) \mid i,j = 1,\ldots,n;\ i \neq j\}$; the *mean area ratio* of all pairs of items of an instance, which is defined as $\gamma_{mean} = \text{mean}\{\gamma(i,j) \mid i,j = 1,\ldots,n;\ i \neq j\}$; and the *variance area ratio* of all pairs of items of an instance, which is given by $\gamma_{var} = \text{var}\{\gamma(i,j) \mid i,j = 1,\ldots,n;\ i \neq j\}$.

- The *heterogeneity ratio* is given by $\nu = n_t/n$, where n_t denotes the number of distinct types of items in an instance. Two items are of the same type if they have identical smaller and larger side dimensions.

- The *width ratio* is determined by $\delta = W/d_{mean}$, where W denotes the width of the strip, and d_{mean} represents the mean value of all (smaller and larger) items dimensions.

The critical characteristic values of each instance of the 3 398 data sets described in Section 4.1 were calculated using the aforementioned selected features. A scatter plot of the data instances based on four features: the mean aspect ratio, the mean area ratio, the heterogeneity ration, and the width ratio, is shown in Figure 2.

The aspect ratio provides information on the shapes of the items in an instance: Relatively small values of the four components of aspect ratio feature indicate that the respective instance is heavily populated by approximately square-shaped items. The variety in the size of the items in an instance, on the other hand, may be deducted from the area ratio: Large values of the four components of the area ratio feature imply that the corresponding instance is dominated by items of widely varying sizes. Furthermore, the miscellany of items in an instance may be gauged from the heterogeneity ratio: An instance with a value of the heterogeneity ratio feature close to unity indicates that the dimensions of the items involved in that instance are all different (*i.e.* heterogeneous). Finally, the width ratio characterises the mean item width relative to that of the strip: That is, an instance with a relatively large value of the width ratio feature contains a large number of wide items.

4.3 Performance evaluation

The problem instances were solved using seven state-of-the-art strip packing metaheuristics from the literature. The first algorithm is the two-stage intelligent search algorithm (ISA) of Leung *et al.* [28], which combines a local search algorithm with the method of simulated annealing (SA) in an attempt to find feasible packing solutions. The second algorithm is a hybrid technique, where a genetic algorithm is executed in conjunction with the

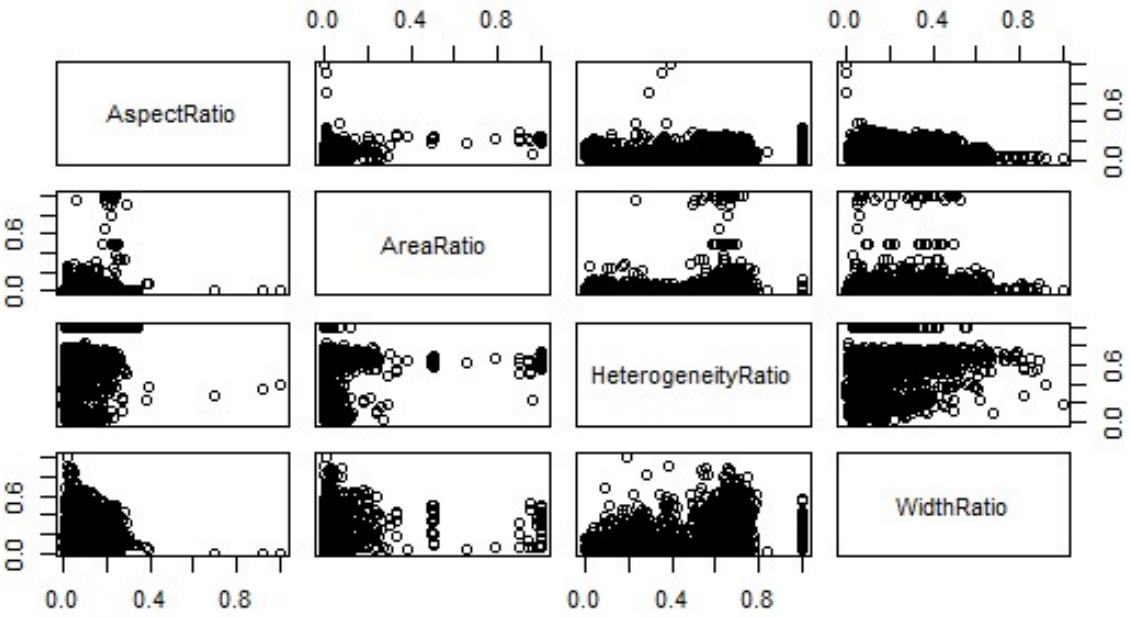


Figure 2: A scatter plot of the data instances of Section 4.1 based on four of the ten features, mainly the mean aspect ratio, the mean area ratio, the heterogeneity ratio, and the width ratio, of Section 4.2. The axes represent the calculated features, standardised to have values between 0 and 1.

constructive heuristic of Leung *et al.* [28]. The simple randomised algorithm (SRA) of Yang *et al.* [47] and the efficient intelligent search algorithm (IA) of Wei *et al.* [45] are also considered, which are both improvements of the ISA algorithm.

The last three algoritms are among the recently proposed strip packing techniques: The improved skyline based heuristic algorithm (ISH) of Wei *et al.* [44], which may be considered as an improved version of the constructive heuristic embedded in the IA algorithm, the CIBA algorithm of Chen and Chen [11], and the modified intelligent search algorithm (IAm) of Rakotonirainy and Van Vuuren [33]. These algorithms have been selected for consideration as they were among the most recently proposed algorithms, and were reported as best algorithms, for solving instances of the 2D-SPP [33].

The relative effectiveness of these algorithms were evaluated according to a performance measure — the relative difference between the packing height returned by an algorithm and the height of an optimal solution or the appropriate lower bound. How to define "similar performance" may involve some tolerance and one can define this specifically by considering the concept that two algorithms would achieve the same result if their performance are within $\epsilon\%$ of each other. In this paper, a set of best algorithms, which is defined in such a way that the performance ratios of any pair of algorithms in the set are equal to almost 1%, was extracted for each sample instance. An example of 5 data instances, each associated with the best performing algorithms, is shown in Table 2.

These seven packing algorithms were coded in Python using Spyder Version Python 3.7. They were run in the same environment on an Intel Core i7-4790 CPU running at 3.60 GHz with 8 GB RAM in the Windows 10 operating system. Further details on the algorithms'

implementations, together with the appropriate parameter fine-tuning, can be found in Rakotonirainy [32].

Problem Instances	Best Algorithms
1985BeasleyJORS11.csv	Hybrid GA
2000HopperT3b.csv	SRA, Hybrid GA
2000HopperN2a.csv	CIBA, ISA, IAm
2001WangNice10.25.csv	ISH, CIBA, IA
2001WangPath2.50.csv	IAm

Table 2: Example of instances with their respective best performing algorithms.

4.4 Classification

An attempt is made to cluster the instances before the classification process. This is meant to identify the meaningful characteristics that best describe the data instances and to explore the relationship between features and algorithm performance.

The cluster analysis performed in this step consists of grouping the 2 278 training data instances described in Section 4.1 into different classes of test problems based on their characteristics identified in Section 4.2. Recently, Rakotonirainy [34] has conducted such analysis in an attempt to cluster the data instances from the literature. In the mentioned study, the clustering process consisted of three steps: The first step involved preparation of the data set based on their selected features. This entailed feature scaling using the normalisation method. The next step consisted of estimating the optimal number of clusters in which to partition the benchmark data by means of a variety of indices. The final step involved evaluation of the performances of different clustering algorithms with respect to a set of validation measures so as to choose the best performing one.

The same clustering process was adopted in this paper in order to generate a sound data clustering output result. The `R` package, *NbClust*, of Charrad *et al.* [10] was employed to estimate an appropriate number of clusters that best partitions the normalised benchmark instances. With a single function call, it computes thirty indices and determines the relevant number of clusters accordingly. A histogram of the distribution of the output is shown in Figure 3. The majority of the indices suggested four as the best number of clusters, and this was utilised in the clustering algorithm of the next step.

The k-means algorithm was selected as the clustering method adopted in this study to cluster the instances into groups, whereby the similarity among members of each group was determined through the characteristic indicators of the instances and the number assigned to the best performing algorithms obtained in the performance evaluation of Section 4.3. The k-means algorithm was performed repetitively until it generates the best clustering results, whereby each cluster was associated a label with respect to the aggregate characteristic values of the instances and a best performing algorithm for it. A summary of the best performing algorithm for each cluster is given in Table 3. It is noted that the ISA, SRA, and IA algorithms were outperformed by the three algorithms reported in this table (or do not differ significantly from their performances for certain instances), and that CIBA and ISH (respectively CIBA and IAm) algorithms performed relatively similarly

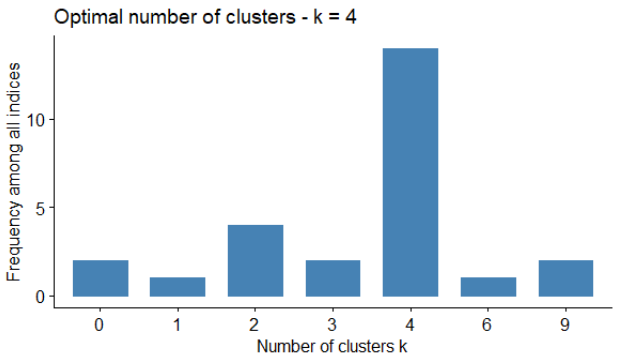


Figure 3: Histogram of the distribution of the results obtained when estimating the number of clusters in which to partition the training data instances of Section 4.1 by means of the *NbClust* function in `R` (Charrad *et al.* [10]). According to the majority rule, the preferred number of clusters is clearly 4.

with respect to the fourth benchmark cluster (respectively third benchmark cluster), so the ISH (respectively IAm) algorithm was selected as the best algorithm for that cluster. This dominance result applies only to the benchmark instances explored in this work.

Cluster	Best Algorithm
Cluster 1	IAm
Cluster 2	Hybrid GA
Cluster 3	IAm
Cluster 4	ISH

Table 3: Summary of the clustering output obtained when performing the clustering analysis. The first cluster contains 520 instances, the second cluster 472 instances, and the third and fourth clusters contain 785 and 501 instances, respectively.

The identified grouping is learned into formal classifiers, which are predictors that model the relationship between problem instance characteristics and algorithm performance. Decision trees can be very powerful tools for modelling this relationship and for elucidating rules that can be used to predict the best performing algorithm for new instances. The decision tree algorithm in [36] was, therefore, employed as a machine learning technique to generate classification rules for this purpose. The decision tree algorithm builds a decision tree from the training data instances, which is then converted to a set of classification rules using the cluster labels as target variables. The rules are ordered by accuracy and are applied in sequence to classify instances in the corresponding group.

To obtain the classification rules, the ten indicators of Section 4.2 were used as independent variables (to generate the corresponding clusters), and the best algorithms associated to each cluster as class variables. The percentage of new correctly classified observations is an indicator of the effectiveness of the classification rules. If these rules are effective on the training data sets, it is expected that they will perform well on new observations with unknown group. The classification analysis was conducted using the Decision Tree Classification package available in Python Scikit-learn and the classifier was trained on the 2 278 data sets of Section 4.1.

In order to optimise the performance of the decision tree classifier, two decision tree methods with two different attribute selection criteria were applied and compared. The first decision tree method employs the "information gain"[1] as a selection criterion, while the second decision tree approach uses the "gini index"[2] criterion. The accuracy of these two methods when applied to the training data sets is shown in Table 4. A pre-pruning was also conducted. This is achieved by controlling the values of parameters and variables defining the classifier. The parameter "maximum depth of the tree", which defines the height or the number of nodes in the tree, was varied in this experimental work. The corresponding accuracy results on the training data sets are also shown in Table 4.

Decision tree methods with	Gini index	Information gain
Maximum depth of the tree = 10	88%	89%
Maximum depth of the tree = 8	68%	71%
Maximum depth of the tree = 5	62%	65%

Table 4: Classification accuracy of the various decision tree methods described in Section 4.4 when applied to the training data sets of Section 4.1.

The accuracies ranged from 89%, for the decision tree classifier using information gain method as a selection criterion and a value equal to 10 for the 'maximum depth of the tree' parameter, down to 62% accuracy for the decision tree classifier with gini index as a selection criterion and a value equal to 5 for the 'maximum depth of the tree' parameter. The method which exhibits the highest accuracy was employed to predict algorithm performance in this work.

4.5 Algorithm prediction and validation

The learned classifier of Section 4.4 was applied to predict the best performing algorithm for each one of the 1 120 test instances of Section 4.1. The decision tree method using information gain method as a selection criterion and a value equal to 10 for the 'maximum depth of the tree' parameter was employed for this purpose, as it provides the highest accuracy results according to Table 4. The output results are given in Table 5. Column 2 of this table contains the real best algorithms for each instance and column 3 corresponds to the algorithm selected by the prediction. As shown in this table, the classifier predicted the correct best performing algorithm with an accuracy of 88%.

To validate the effectiveness of the algorithm performance predictor applied above, the predicted algorithms are compared to the real best algorithms for the test instances. If the predicted algorithm is one of the real best algorithms, the match is counted (equal to 1), as shown in column 4 of Table 5. A 'Match' value equal to 0, for a particular instance, indicates that the classifier fails to predict the correct algorithm for that instance. During the feedback phase, all instances with 'Match' values equal to 0 are combined with the training data sets and the classifiers are rebuilt using this new data set. The prediction phase is repeated to predict the best algorithm for new instances.

[1]Information gain is a statistical property that measures how well a given attribute separates the training data according to their target variables [30].

[2]Gini index is a cost function used to evaluate splits in the data sets. It is calculated by subtracting the squared probabilities of each class from one [30].

Instance	Real best algorithms	Predicted best Algorithm	Match
Inst1-ID1-Class1234.txt	Hybrid GA	IAm	0
Inst2-ID1-Class2345.txt	IA	IAm	0
Inst3-ID1-Class3456.txt	Hybrid GA	Hybrid GA	1
Inst4-ID1-Class4567.txt	SRA, Hybrid GA	Hybrid GA	1
⋮	⋮	⋮	⋮
Inst8-ID1-Class5671.txt	ISA, CIBA	ISH	0
Inst9-ID1-Class6712.txt	Hybrid GA, SRA	Hybrid GA	1
Inst10-ID1-Class7123.txt	Hybrid GA	ISH	0
		Accuracy	88%

Table 5: Classification results for the 1 120 problem instances of Section 4.1 after performing the clustering process. The column 'Match' indicates if the predicted algorithm is correct (its value equal to 1) or not (its value equal to 0).

Of course, it is natural to wonder how heavily the classification outputs depend on the clustering results. How likely is it that a given instance really belongs to cluster i ($i = 1, \ldots, 4$), and if algorithm A is assigned to be the best algorithm for problem instances in cluster i ($i = 1, \ldots, 4$) how likely that decision is wrong? Obviously the result depends on the clustering outputs. One could expect a different result by directly classifying the instances based on their characteristics and their best performing algorithms. This process was investigated in this study and the resulting outputs are reported in Table 6. As shown in this table, the classifier predicted the correct best performing algorithm with accuracy of 90%.

Instance	Real best algorithms	Predicted best Algorithm	Match
Inst1-ID5-Class1234.txt	IAm	IAm	1
Inst2-ID5-Class2345.txt	IA, ISH, CIBA	CIBA	1
Inst3-ID5-Class3456.txt	Hybrid GA	Hybrid GA	1
Inst4-ID5-Class4567.txt	SRA, IAm	Hybrid GA	0
⋮	⋮	⋮	⋮
Inst8-ID5-Class5671.txt	ISA, Hybrid GA	IAm	0
Inst9-ID5-Class6712.txt	Hybrid GA, SRA	Hybrid GA	1
Inst10-ID5-Class7123.txt	Hybrid GA, ISH	ISH	0
		Accuracy	90%

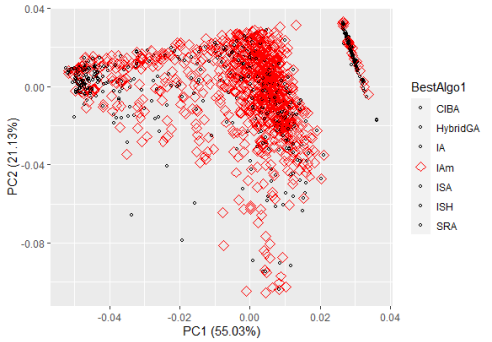
Table 6: Direct classification results for the 1 120 problem instances of Section 4.1, without performing the clustering process. The column 'Match' indicates if the predicted algorithm is correct (its value equal to 1) or not (its value equal to 0).

4.6 Analysis of algorithm power

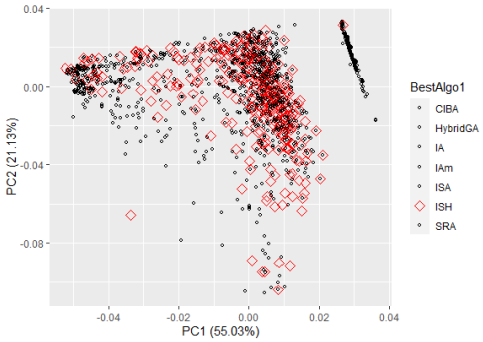
By evaluating the performance of each of the seven metaheuristics described in Section 4.3 with respect to the 3 398 generated instances, one can assign each algorithm a label

of "good" or "bad" on the instance space. For this case study, the goodness is arbitrarily defined as the algorithm achieving a packing solution that lies within 1% of the known optimal solution. In the following sections, the footprints as well as the strengths and weaknesses of the various algorithms, based on the aforementioned goodness, are presented.

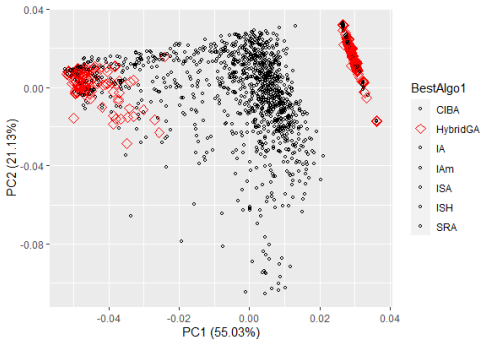
4.6.1 Algorithm footprints



(a) IAm footprint



(b) ISH footprint



(c) Hybrid GA footprint

Figure 4: Algorithm footprints with (red) diamond instances showing good performance. Black circle instances are not within the algorithm footprint. The axes, labelled PC1 and PC2, indicate respectively the first and second principal components obtained from applying a principal component analysis to the data and represent a projection that best spreads the data.

Figure 4 shows the footprint of each of the three best algorithms (IAm, Hybrid GA, and ISH), with (red) diamond instances showing good performance. The area of each footprint can be measured using the method described in Section 3.3.2, as shown in the second column of Table 7. Clearly, the algorithm with the largest footprint is the IAm algorithm, since there are only small regions of the instance space where it is not "good". It is interesting to note that the IAm is not best everywhere and it could lose its competitive advantage if the definition of "good" performance is relaxed. The third column of Table 7 shows the relative size of the algorithm footprints if one relaxes the definition of good performance to be any algorithm that achieves a solution that lies within 5% of the known optimal solution. Larger footprints are obtained for each algorithm and one can see that

the performance of the ISH and Hybrid GA algorithms are not substantially different from the performance of the IAm algorithm.

Algorithm	Area (goodness = 1%)	Area (goodness = 5%)
IAm	53.1%	57.2%
Hybrid GA	21.7%	50.1%
ISH	23.9%	51.8%

Table 7: Relative size of algorithm footprints expressed as a percentage of the total area of the instance space.

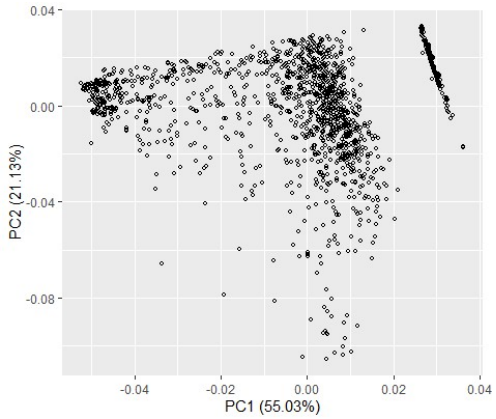
4.6.2 Insights into algorithm strengths and weaknesses

Following all the above analysis, a natural question to ask is how the characteristics of the instances may contribute to good performance of a given algorithm. That is which features and what characteristics are found in certain regions of the instance space where algorithm performance is good. One can answer this question by exploring the instance space and developing a good understanding of where the unique strengths and weaknesses of each algorithm lie. The first step is to visualize where each algorithm offers a unique advantage where others struggle and vice versa. If an algorithm is found to be good where many other algorithms are not good, then it is useful information to assess the relative power of algorithms across the instance space. Once such insights are gained, the next step is to locate the regions in the instance space where an algorithm has a unique strength and determine the underlying conditions with respect to the features of the instance space.

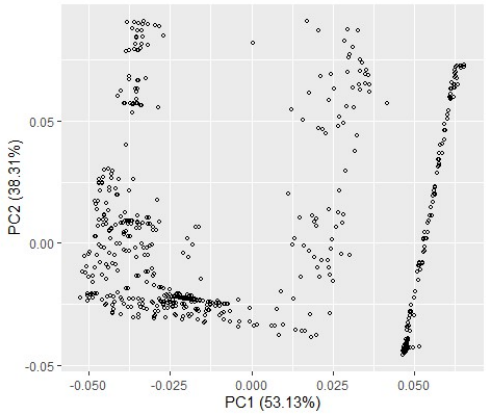
Algorithm	Number of instances	Percentage (%)
IAm	437	64.83%
Hybrid GA	138	20.47%
ISH	59	8.75%
CIBA	40	5.93%
IA, ISA, SRA	0	0%
Total	674	29.59%

Table 8: Number and percentage of instances that are uniquely solved by each of the algorithm considered in this case study.

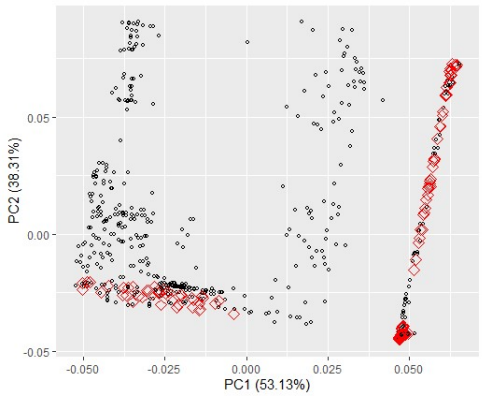
Figures (5a)-(5b) show the location of the instances that are easily solved by many algorithms and the instances that are more challenging since only one algorithm attains the best result. Among the hard instances, it is important to know which algorithm provides unique advantage over others, and this is shown in Figures (**??**)-(**??**). Interestingly, only two algorithms exhibit consistent regions where they are uniquely best: The IAm and Hybrid GA algorithms. These two methods combine local search strategies with global operators that allow much larger changes to be made to a solution. The ISH and CIBA algorithms are sometimes uniquely best but the type of instances that they are best are not well located in the instance space. The remaining algorithms (IA, ISA, SRA) do not differ substantially or are outperformed by the four aforementioned algorithms in terms of performance with respect to the benchmark instances.



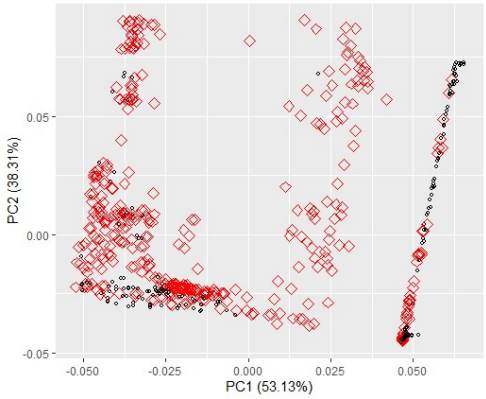
(a) Easy instances



(b) Hard instances



(c) Instances uniquely solved by Hybrid GA



(d) Instances uniquely solved by IAm

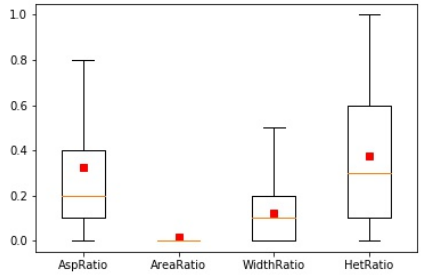
Figure 5: (a) Instances that are easily solved, where multiple algorithms achieve good performance. (b) Instances that are hard to solve (solved uniquely by one algorithm for a goodness equal to 1%). (c) (Red) diamond instances that are uniquely solved by the Hybrid GA algorithm. (d) (Red) diamond instances that are uniquely solved by the IAm algorithm. The axes, labelled PC1 and PC2, indicate respectively the first and second principal components obtained from applying a principal component analysis to the data and represent a projection that best spreads the data.

Table 8 shows the relative unique strengths of each algorithm, focusing on how many instances are uniquely solved well by each algorithm. For 70.41% of the instance space there is no uniquely winning algorithm, but for the 674 discriminating instances, it is the IAm algorithm which uniquely perform best in 64.83% of the instances. The Hybrid GA algorithm is uniquely best for 20.47% of the discriminating instances. The ISH algorithm is uniquely good for 8.75%, while the CIBA algorithm is uniquely good for 5.93% of the discriminating instances.

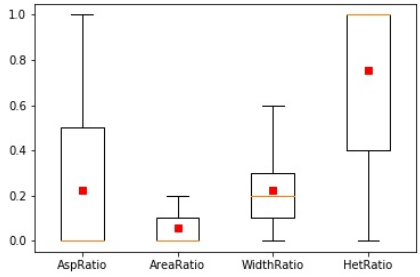
In addition, considering the unique footprints shown in Figure **??**, it is possible to gain insight into what region in the instance space can inform about the conditions under which an algorithm has a unique strength. To achieve this, one can inspect the distribution of the four main features (maximum aspect ratio, maximum area ratio, width ratio, and

heterogeneity ratio) across the instance space, looking in particular for criteria about why the IAm algorithm does not perform well on the majority of instances defined by the thin right-most edge of the instance space, and why the Hybrid GA algorithm seems to struggle in the large portion of the hard instance space.

Figure 6 shows that these interesting regions of unique strength of the two algorithms correspond to lower values of the three features (maximum aspect ratio, maximum area ratio, and width ratio) defining the instances, and higher value of the heterogeneity ratio feature. In particular, the Hybrid GA algorithm appears to outperform all algorithms (including the IAm algorithm), when the benchmark instances are dominated by square items (the maximum aspect ratio assumes values in a small range) and mainly heterogeneous (the corresponding mean heterogeneity ratio is close to 1 as shown in Figure 6 (b)); while the IAm algorithm is the most effective technique to solve instances that contain items which are relatively small (the width feature varies over a small range of value) and of equal size (the respective maximum area ratio assumes values in a small range as shown in Figure 6 (a)).



(a) Distribution of the hard instances' features that are uniquely solved by the IAm algorithm



(b) Distribution of the hard instances' features that are uniquely solved by the Hybrid GA algorithm

Figure 6: Distribution of the four selected features (maximum aspect ratio, maximum area ratio, width ratio, heterogeneity ratio) across the hard instance space uniquely solved by the IAm and Hybrid GA algorithms.

5 Discussion

The algorithm prediction result presented in the previous section suggests that the methodology adopted in this paper achieves an 88% accuracy in correctly predicting the best performing algorithm for a new strip packing instance. This could be considered as a highly accurate prediction of algorithm performance on the basis of packing instances grouping according to their underlying features. The systematic selection of the features that characterise the problem instances was crucial for obtaining this result accuracy. Ten different characteristics, extracted from various parameters and factors used in popular packing generators, were considered to group the benchmark instances into four categories in this work. These chosen features have proven to be sufficient for discovering the properties of the benchmark instances.

The result shows a 3% decrease in the prediction accuracy compared to the result obtained by the recent work of Rakotonirainy [34]. The analysis conducted in this paper, in fact, includes a much broader set of instances than those used by Rakotonirainy, and that the instance space subsequently created was a good representation of the strip packing benchmark problems. In addition, in the latter study, four features were considered to group the benchmark instances into different categories, resulting in a 91% accuracy in correctly identifying the best algorithm for a new instance. It is noted that the test instances employed in [34] are those from the literature which exhibit approximately similar characteristics as of the training instances. In contrast, a new set of instances was considered as test instances in this study which exhibits different features as of the training sets and ten features were considered instead of four.

Furthermore, it is noted that the decision to include a clustering process before the classification step has meant that the meaningful characteristics that best describe the data instances are identified and that the relationship between the identified features and algorithm performance is accurately explored. Each cluster was assigned one best performing algorithm and it was learned into classifiers for predicting the best algorithm for a new given instance. An improved result was obtained when the clustering process was omitted, that is the classification process was applied directly to the instances based on their underlying characteristics and their best performing algorithms. The result shows a 2% increase in the prediction accuracy, indicating that a more robust classification of the problem instances is obtained.

Besides, the majority of the instances incorrectly classified are tied results. In fact, it was assumed a definition of a best algorithm as an algorithm with performance ratio relatively 1% higher than others and that ties are solved by randomly choosing one of the algorithms. The latter case was applied to the best algorithm assigned to instances of clusters 3 and 4: The IAm algorithm (respectively the ISH algorithm), of which its performance did not differ significantly with the CIBA algorithm (respectively the CIBA algorithm), was selected as the best algorithm for these clusters in this work. In reality, however, the assigned algorithm may be best for some proportion (possibly, not for all) of problem instances in a cluster and that it may perform well for a specific instance of other clusters. An example is the classification result of (Inst1-ID1-Class5671.txt) instance, whereby the real best algorithm of the instance is actually the CIBA algorithm, while the classification model predicts the ISH algorithm as its best algorithm. A more robust condition is probably needed to avoid such biased prediction and to improve on the performance accuracy.

The methodology presented in this paper was also able to provide new insights into the packing algorithm power, mainly the identification of strengths and weaknesses of the various algorithms with respect to the instance features defining the instance space. The results indicated that the size of an algorithm footprint depends on the definition of goodness. Moreover, 29.59% of the overall instances are hard instances which are uniquely solved by one algorithm for a goodness equal to 1%, indicating that the instance space generated in this study is a relatively good representation of the set of all strip packing benchmark instances that are typically studied in the literature.

While this paper focused on the application of the methodology to the strip packing problem, the proposed framework can be applied to any type of packing problem and can also be easily adopted for automated algorithm selection in other combinatorial optimisation problems. The adaptation of the methodology to other problems presents no difficulties aside from the challenge of selecting and deriving relevant features to characterise the problem instances, which might require a careful check.

6 Conclusion

An improved methodology for the characterisation of packing algorithm performance and its application to algorithm selection was introduced in this paper. By building upon the framework of Rakotonirainy [34], a pathway to develop the tools required to model the relationship between features of packing problem instances and the performance of packing algorithms was proposed. This model was applied to predict best performing algorithms for unseen instances with high accuracy. The model was also employed to identify the regions of instance space where algorithms have unique strengths and weaknesses and to generate insights of where the unique power of each algorithm lies within footprints based on the instance features. This paper addressed some questions in the methodology developed by Rakotonirainy, and demonstrated the usefulness of the approach by revisiting and extending the case study of Rakotonirainy [34].

It was shown, through a case study of strip packing problems, that data mining techniques like clustering analysis and decision trees can be employed to explore the high-dimensional feature space of the problem instances, to cluster the test problems into different classes of instances according to their features, and to learn the clusters into classifiers for an effective automated packing algorithm selection. The result obtained suggested a 90% (respectively 88%) accuracy in predicting the best performing algorithms on a new set of packing problem instances based on their characteristics and best peforming algorithms (respectively by performing a clustering analysis prior the classification process).

The next steps for this research include application of the methodology to other packing problems, and make them available to the operations research community via a user-friendly computerised decision support system tool. Such software should take as input a new packing problem instance. After reading in the given problem instance, it should apply the framework — compute the relevant characteristics, assign the instance to a relevant cluster, and generate the best performing algorithm, and report the result as output to the user accordingly. The user may also choose to solve the problem directly by means of an appropriate algorithm, and compare the outputs.

References

[1] Babu AR & Babu NR, 1999, *Effective nesting of rectangular parts in multiple rectangular sheets using genetic and heuristic algorithms*, International Journal of Production Research, **37(7)**, pp. 1625–1643.

[2] Barr RS, Golden BL, Kelly JP, Resende Mauricio GC & Stewart WR, 1995, *Designing and reporting on computational experiments with heuristic methods*, Journal of Heuristics, **1(1)**, pp. 9–32.

[3] Beasley J, 1985, *Algorithms for unconstrained two-dimensional guillotine cutting*, Journal of the Operational Research Society, **36(4)**, pp. 297–306.

[4] Bengtsson B, 1982, *Packing rectangular pieces — a heuristic approach*, The Computer Journal, **25(3)**, pp. 353–357.

[5] Berkey J & Wang P, 1987, *Two-dimensional finite bin-packing algorithms*, Journal of the Operational Research Society, **38(5)**, pp. 423–429.

[6] Berkhin P, 2006, *A survey of clustering data mining techniques*, pp. 25–71 in Kogan J, Nicholas C & Teboulle M (Eds.), *Grouping multidimensional data*, Springer, Berlin (Heidelberg).

[7] Bortfeldt A & Gehring H, 2006, *New Large benchmark instances for the two-dimensional strip packing problem with rectangular pieces*, Proceedings of the 39^{th} Annual Hawaii International Conference on System Sciences, Kauai (HI), p. 30b.

[8] Burke E & Kendall G, 1999, *Applying simulated annealing and the no fit polygon to the nesting problem*, in Foo N (Eds.), *Advanced Topics in Artificial Intelligence*, Springer, Berlin (Heidelberg).

[9] Burke E, Kendall G & Whitwell G, 2004, *A new placement heuristic for the orthogonal stock-cutting problem*, Operations Research, **52(4)**, pp. 655–671.

[10] Charrad M, Ghazzali N, Boiteau V, Niknafs A & Charrad MM, 2014, *Package NbClust*, Journal of Statistical Software, **61**, pp. 1–36.

[11] Chen Z & Chen J, 2018, *An effective corner increment-based algorithm for the two-dimensional strip packing problem*, IEEE Access, **6**, pp. 72906–72924.

[12] Christofides N & Whitlock C, 1977, *An algorithm for two-dimensional cutting problems*, Operations Research, **25(1)**, pp. 30–44.

[13] Dagli C & Poshyanonda P, 1997, *New approaches to nesting rectangular patterns*, Operations Research, **8(3)**, pp. 177–190.

[14] ESICUP, 2015, *Datasets 2D-Rectangular*, [Online], [Cited: 7 July 2022], Available from http://www.euro-online.org/websites/esicup/data-sets/.

[15] Ferreira EP & Oliveira JF, 2005, *Algorithm based on graphs for the non-guillotinable two-dimensional packing problem*, Proceedings of the 2^{nd} ESICUP Meeting, Southampton, no page.

[16] Gupta AK & Nadarajah S, 2004, *Handbook of Beta Distribution and Its Applications*, 1^{st} Edition, CRC Press, Boca Raton (FL).

[17] Guyon I & Elisseeff A, 1974, *An introduction to variable and feature selection*, Journal of Machine Learning Research, **3**, pp. 1157–1182.

[18] Hifi M, 1998, *Exact algorithms for the guillotine strip cutting/packing problem*, Computers and Operations Research, **25(11)**, pp. 925–940.

[19] Hifi M, 1999, *The strip cutting and packing problem: Incremental substrip algorithms-based heuristics*, Pesquisa Operacional, **19(2)**, pp. 169–188.

[20] Hopper E & Turton B, 2002, *Problem generators for rectangular packing problems*, Studia Informatica Universalis, **2(1)**, pp. 123–136.

[21] Hopper E & Turton B, 2001, *An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem*, European Journal of Operational Research, **128(1)**, pp. 34–57.

[22] Imahori S & Yagiura M, 2010, *The best-fit heuristic for the rectangular strip packing problem: An efficient implementation and the worst-case approximation ratio*, Computers and Operations Research, **37(2)**, pp. 325–333.

[23] Jakobs S, 1996, *On genetic algorithms for the packing of polygons*, European Journal of Operational Research, **88(1)**, pp. 165–181.

[24] Jain A & Dubes RC, 1988, *Algorithms for Clustering Data*, Prentice-Hall, River (NJ).

[25] Jain A, Murty MN & Flynn PJ, 1999, *Data clustering: A review*, ACM Computing Surveys, **31(3)**, pp. 264–323.

[26] Júnior AN, Silva E, Gomes AM, Soares C & Oliveira JF, 2019, *Data mining based framework to assess solution quality for the rectangular 2D strip-packing problem*, Expert Systems with Applications, **118**, pp. 365–380.

[27] Leung S & Zhang D, 2011, *A fast layer-based heuristic for non-guillotine strip packing*, Expert Systems with Applications, **38(10)**, pp. 13032–13042.

[28] Leung S, Zhang D & Sim KM, 2011, *A two-stage intelligent search algorithm for the two-dimensional strip packing problem*, European Journal of Operational Research, **215(1)**, pp. 57–69.

[29] Martello S & Vigo D, 1998, *Exact solution of the two-dimensional finite bin packing problem*, Management Science, **44(3)**, pp. 388–399.

[30] Nagesh SC, 2019, *Decision Tree Algorithm – Explained*, [Online], [Cited: 7 July 2022], Available from http://towardsdatascience.com/decision-tree-algorithm-explained-83beb6e78ef4.

[31] Perez J, Frausto J, Cruz L & Fraire H, 2004, *A machine learning approach for modeling algorithm performance predictors* Proceedings of the 1^{st} International Conference on Modeling Decisions for Artificial Intelligence, Barcelona, pp. 70–80.

[32] Rakotonirainy RG, 2018, *Metaheuristic solution of the two-dimensional strip packing problem*, PhD Thesis, University of Stellenbosch, Stellenbosch.

[33] Rakotonirainy RG & Van Vuuren JH, 2020, *Improved metaheuristic for the two-dimensional strip packing problem*, Applied Soft Computing, **92**, no page.

[34] Rakotonirainy RG, 2020, *A machine learning approach for automated strip packing algorithm selection*, ORiON, **36(2)**, pp. 73–88.

[35] Rakotonirainy RG, 2020, *Library of problem instances*, [Online], [Cited: 7 July 2022], Available from https://github.com/Rgeorgina/Library-of-problem-instances.

[36] Rokach L & Maimon OZ, 2008, *Data Mining with Decision Trees: Theory and Applications*, World Scientific, Singapore.

[37] Silva E, Oliveira JF & Wäscher G, 2014, *2DCPackGen: a problem generator for two-dimensional rectangular cutting and packing problems*, European Journal of Operational Research, **237(3)**, pp. 846–856.

[38] Smith-Miles K, Baatar D, Wreford B & Lewis R, 2014, *Towards objective measures of algorithm performance across instance space*, Computers and Operations Research, **45**, pp. 12–24.

[39] Smith-Miles K & Tan L, 2012, *Measuring algorithm footprints in instance space*, Proceedings of the 2012 IEEE Congress on Evolutionary Computation, Brisbane, pp. 1–8.

[40] Smith-Miles K & Lopes L, 2012, *Measuring instance difficulty for combinatorial optimization problems*, Computers and Operations Research, **39(5)**, pp. 875–889.

[41] Smith-Miles K, Wreford B, Lopes L & Insani N, 2013, *Predicting metaheuristic performance on graph coloring problems using data mining*, Talbi EG. (Eds.), Hybrid metaheuristics, SCI, Heidelberg.

[42] Valenzuela CL & Wang PY, 2001, *Heuristics for large strip packing problems with guillotine patterns: An empirical study*, Proceedings of the 4^{th} Metaheuristics International Conference, Porto, pp. 417–421.

[43] Smith-Miles K, 2009, *Cross-disciplinary perspectives on meta-learning for algorithm selection*, ACM Computing Surveys, **41(1)**, pp. 1–25.

[44] Wei L, Hu Q, Leung S & Zhang N, 2017, *An improved skyline based heuristic for the 2D strip packing problem and its efficient implementation*, Computers and Operations Research, **80**, pp. 113–127.

[45] Wei L, Qin H, Cheang B & Xu X, 2016, *An efficient intelligent search algorithm for the two-dimensional rectangular strip packing problem*, International Transactions in Operational Research, **23(1-2)**, pp. 65–92.

[46] Xu R & Wunsch D, 2005, *Survey of clustering algorithms*, IEEE Transactions on Neural Networks, **16(3))**, pp. 645–678.

[47] Yang S, Han S & He W, 2013, *A simple randomized algorithm for two-dimensional strip packing*, Computers and Operations Research, **40(1)**, pp. 1–8.