Understanding the structure of bin packing problems through principal component analysis

Eunice López-Camacho[a,*], Hugo Terashima-Marín[a,*], Gabriela Ochoa[b], Santiago Enrique Conant-Pablos[a]

[a] *Tecnológico de Monterrey, Av. E. Garza Sada 2501, Monterrey, NL, 64849, Mexico. Telephone: +52 818158-2045. Fax number: +52 818328-4379.*
[b] *Computing Science and Mathematics, University of Stirling, Scotland, UK.*

Abstract

This paper uses a knowledge discovery method, Principal Component Analysis (PCA), to gain a deeper understanding of the structure of bin packing problems and how this relates to the performance of heuristic approaches to solve them. The study considers six heuristics and their combination through an evolutionary hyper-heuristic framework. A wide set of problem instances is considered, including one-dimensional and two-dimensional regular and irregular problems. A number of problem features is considered, which is reduced to the subset of nine features that more strongly relate with heuristic performance. PCA is used to further reduce the dimensionality of the instance features and produce 2D maps. The performance of the heuristics and hyper-heuristics is then super-imposed into these maps to visually reveal relationships between problem features and heuristic behavior. Our analysis indicates that some instances are clearly harder to solve than others for all the studied heuristics and hyper-heuristics. The PCA maps give a valuable indication of the combination of features characterizing easy and hard to solve instances. We found indeed correlations between instance features and heuristic performance. The so-called DJD heuristics are able to best solve a large proportion of instances, but simpler and faster heuristics can outperform them in some cases. In particular when solving 1D instances with low number of pieces, and, more surprisingly, when solving some difficult 2D instances with small areas with low variability. This analysis can be generalized to other problem domains where a set of features characterize instances and several problem solving heuristics are available.

Keywords: heuristics, hyper-heuristics, bin packing problem, principal component analysis, algorithm selection, knowledge discovery

1. Introduction

The problem of finding an arrangement of pieces to cut or pack inside larger objects is known as the cutting and packing problem. This NP-hard problem is not only of academic interest; numerous applications of its many variants can be found in practice. The one-dimensional (1D) and two-dimensional (2D) bin packing problems (BPP) are particular cases of the cutting and packing problem, which consists of finding an arrangement of items inside identical bins or objects such that the number of objects required to contain all pieces is minimum. For the 2D BPP, the case of rectangular pieces is the most widely studied. However, the irregular case is seen in a number of industries where parts with irregular shapes are cut from rectangular materials. For example, in the shipbuilding industry, plate parts with free-form shapes for use in the inner frameworks of ships are cut from rectangular steel plates. Also, in the apparel industry, parts of clothes and shoes are cut from fabric or leather (Okano, 2002). Other direct applications include the optimization

*Corresponding author

Email addresses: eunice.lopez@itesm.mx (Eunice López-Camacho), terashima@itesm.mx (Hugo Terashima-Marín), gabriela.ochoa@cs.stir.ac.uk (Gabriela Ochoa), sconant@itesm.mx (Santiago Enrique Conant-Pablos) of layouts within the wood, plastics, glass, textile, metalware and paper industries (Burke et al., 2006; Yu et al., 2009). In these industries, small improvements of the arrangement can result in a large saving of material (Hu-yao and Yuan-jun, 2006).

Within the field of machine learning, the term *meta-learning* has been associated with the idea of, given a new dataset, automatically selecting the best learning algorithm for the problem at hand. Meta-learning ideas have traditionally been applied to learning algorithms to solve classification problems, where the goal is to relate performance of algorithms to characteristics or measures of classification datasets. In Smith-Miles (2008a) a framework is presented for the generalization of algorithm selection and meta-learning ideas to algorithms focused on other tasks such as sorting, forecasting, constraint satisfaction and optimization. Meta-learning ideas have been only little explored within optimization, although several approaches can be found in the related area of constraint satisfaction (Leyton-Brown et al., 2002). In Smith-Miles (2008b), meta-learning ideas are used for modeling the relationship between instance characteristics and algorithm performance for the quadratic assignment problem. The study considered a set of 28 problem instances and three metaheuristic algorithms. Both unsupervised and supervised neural network models were used to learn the relationships in the meta-dataset and automate the algorithm selection process. The unsupervised model, self-organization maps (Kohonen et al., 2001), was used to select the best algorithm by creating visual explorations of the performance of different algorithms under various conditions describing the complexity of the problem instances. Given the limited size of the data, this is a preliminary study, but it demonstrates the relevance of meta-learning ideas to the optimization domain. In a later study, Smith-Miles et al. (2009) found correlations between problem features and the effectiveness of scheduling heuristics using a large collection of instances in a production scheduling problem. Other authors have followed up this type of study in optimization (Kanda et al., 2011).

This paper proposes using Principal Component Analysis (PCA) for visualizing n-dimensional feature data related to bin packing problems. The goal is to improve our understanding of the problem structure and its relationship with heuristic performance. PCA is a mathematical algorithm that reduces the dimensionality of the data while retaining most of the variation in the dataset. It accomplishes this reduction by converting a set of observations into a set of values of uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has as high a variance as possible (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (uncorrelated with) the preceding components. Observations can then be plotted, making it possible to visually assess similarities and differences between observations and determine whether observations can be grouped (Ringner, 2008). In this study, PCA graphs are used to visually reveal the problem feature combinations that are mostly related with an improved heuristic performance. The ultimate goal is to inform the design of effective heuristics and hyper-heuristics (heuristic combination rules) for bin-packing.

A large testbed is employed with 1417 instances including 1D and 2D bin packing problems (rectangular, convex and non-convex). Several types of bin packing problems are considered in order to enhance the generality of the analysis. There is a recent trend to produce more generally applicable methodologies to solve combinatorial optimization problems. For example, Ochoa et al (2012a) proposed a software framework *HyFlex* (Hyper-heuristic Flexible framework) for developing cross-domain search methodologies. The framework features a common software interface for dealing with different combinatorial optimization problems, and provides the algorithm components that are problem specific. In this way, the algorithm designer does not require a detailed knowledge of the problem domains, and thus can concentrate his/her efforts on designing adaptive general-purpose optimization (Burke et al., 2010a; Ochoa et al, 2012b). Six problem domains are currently implemented: boolean satisfiability, one dimensional bin packing, permutation flow shop, personnel scheduling, vehicle routing and traveling salesman problems (Ochoa and Hyde, 2011). In another study, Burke et al. (2012) proposed a general packing methodology that includes 1D, 2D (orthogonal) and 3D (orthogonal) bin packing and knapsack packing. They present a genetic programming system to automatically generate a good quality heuristic for each instance among the different problems considered.

Section 3 describes both the set of packing heuristics and the evolutionary hyper-heuristic framework employed. Section 4 describes the PCA technique. The instances testbed and the set of features computed for each instance are described in Section 5. The mapping of all instances through PCA, the core of our analysis, is presented in Section 6.

2. The bin packing problem

The cutting and packing problem is among the earliest problems in the literature of operations research. Wäscher et al. (2007) suggested a complete problem typology which is an extension of an earlier typology (Dychoff, 1990). This paper considers the following three problems in Wäscher et al. typology: (a) the 1D single bin size bin packing problem, (b) the 2D regular single bin size bin packing problem, and (c) the 2D irregular single bin size bin packing problem. In the 1D BPP, there is an unlimited supply of bins, each with capacity $c > 0$. A set of n items (each one of size $s_i < c$) is to be packed into the bins. The task is to minimize the total number of bins used. In the 2D BPP, there is a set $L = (a_1, a_2, \ldots, a_n)$ of pieces to cut or pack and an infinite set of identical rectangular larger elements (called *objects*). The problem consists of finding an arrangement of *pieces* inside the *objects* such that the number of objects required to contain all pieces is minimum. A feasible solution is an arrangement of pieces without overlaps and with no piece outside the object limits. A *problem instance* or *instance* $I = (L, x_0, y_0)$ consists of a list of elements L and object dimensions x_0 and y_0. The term 2D regular BPP is mainly used when all pieces are rectangular although circles and other regular shapes could also fall under this term (Wäscher et al., 2007). Otherwise, the problem is called 2D irregular BPP. This study considers the offline BPP, in which the list of pieces to be packed is static and given in advance.

Cheng et al. (1994) and Lodi et al. (2002) reviewed the literature on 1D and 2D bin packing problems, respectively. Cheng et al. (1994) categorized the research related to the cutting and packing problem in more than 400 books and articles. A variety of approaches have been applied to tackle the cutting and packing problem in general, and the BPP in particular. For many real-world problems, an exhaustive search for solutions is not a practical proposition. Hence, many heuristic approaches have been adopted. Heuristic approaches for the 2D bin packing problem present at least two phases: first, the selection of the next piece to be placed and the corresponding object to place it; and second, the actual placement of the selected piece in a position inside the object. Some approaches consider a third phase as a local search mechanism. For the 1D BPP, the second phase (placement procedure) is not necessary. Hyper-heuristic approaches for the BPP have recently been developed (Burke et al., 2007b; Terashima-Marín et al., 2010; López-Camacho et al., 2011; Burke et al., 2012).

3. Heuristics and Hyper-heuristics

This section describes both the base heuristics and the hyper-heuristic model used in this article.

3.1. Set of heuristics

The 1D and 2D bin packing problems share the same selection heuristics. Heuristics are rules to select the next piece to be placed, and the corresponding object to place it. The following six heuristics were considered:

1. **First Fit Decreasing (FFD)**. Considers the open objects in the order they were initially opened, and places the largest piece in the first object where it fits. If the piece does not fit in any open object, a new object is opened to place it.
2. **Filler**. Sorts the pieces in order of decreasing area and packs as many pieces as possible within the open object. When no single piece can be placed in the open object, a new object is opened to continue packing the pieces from largest to smallest.
3. **Best Fit Decreasing (BFD)**. Sorts the unplaced pieces in order of decreasing area and places the next piece in the open object where it best fits, that is, in the object that leaves minimum waste. If the piece does not fit in any open object, a new object is opened to place it.

4. **Djang and Finch with initial fullness of 1/4 (DJD$_{1/4}$).** Places pieces in an object, taking pieces by decreasing size, until the object is at least one-forth full. Then, it initializes $w = 0$, a variable indicating the allowed waste, and looks for combinations of one, two, or three pieces producing a waste up to w. If any combination fails, it increases w by one twentieth of the object.
5. **Djang and Finch with initial fullness of 1/3 (DJD$_{1/3}$).** Same as DJD$_{1/4}$ when the object is full until 1/3 before trying combinations of pieces.
6. **Djang and Finch with initial fullness of 1/2 (DJD$_{1/2}$).** Same as DJD$_{1/4}$ when the object is full until 1/2 before trying combinations of pieces.

The first three heuristics place exactly 1 piece in each application. However, DJD$_{1/4}$, DJD$_{1/3}$ and DJD$_{1/2}$ may place 1, 2 or 3 pieces at a time. Since DJD heuristics try several combinations before placing up to 3 pieces, these heuristics are more time consuming. For the 2D BPP, a placement heuristic is also needed to find a solution. The heuristic called **Constructive Approach with Maximum Adjacency** was employed for finding the actual placement of the selected piece in a position inside the object for all the 2D instances. This heuristic is partially based on the approach suggested by Uday et al. (2001) and adapted by Terashima-Marín et al. (2010). It explores several possible positions and the position with the largest adjacency (i.e., the common boundary between its perimeter and the placed pieces and the object edges) is selected as the position of the new piece. This heuristic was chosen because of its good performance (López-Camacho et al., 2013). We employed only single-pass constructive heuristics for the offline BPP.

3.2. The hyper-heuristic method

A hyper-heuristic is a search method or learning mechanism for selecting or generating heuristics to solve computational search problems (Burke et al., 2010b). A hyper-heuristic can be regarded as a high-level approach that, given a particular problem instance and a number of low-level heuristics, selects and applies an appropriate low-level (single) heuristic at each decision point (Ross, 2005). López-Camacho et al. (2011) proposed a method that produces general hyper-heuristics for 1D and 2D BPP instances. A hyper-heuristic in this context is a condition-action rule that relates each possible instance state (condition) with a single heuristic to be applied (action). Once a hyper-heuristic is developed, it is able to solve any 1D or 2D instance without further parameter tuning. The proposed method (López-Camacho et al., 2011) is based on a genetic algorithm (GA) that evolves combinations of condition-action rules after going through a learning process which includes training and testing phases (the evolutionary model is described by Terashima-Marín et al. (2010)). Hyper-heuristics are sometimes described as *heuristics that search over a space of heuristics.* The GA in that hyper-heuristic approach itself searches a space of particular ways of combining heuristics. Thus, the reader might regard either the chromosomes themselves, or the whole search process itself, as being hyper-heuristics. In this paper we choose to refer to the chromosomes as being hyper-heuristics.

The structure or characterization of 1D and 2D problem instances can be summarized by several features in a numerical vector. This vector represents the condition according to which the hyper-heuristic chooses the heuristic to apply. The numerical representation is applied to both *unsolved* instances (where no piece has been placed yet) and *partially solved* instances (where some pieces have already been placed). Therefore, for a given problem instance, the value of the numerical representation is computed in every intermediate state until it is completely solved.

Each chromosome in the GA is composed of a series of *blocks* (see Figure 1). Each block contains one representative vector R and also an associated choice of heuristic (López-Camacho et al., 2011). The numerical vector R represents an instance state. The label is the last number, which identifies a single heuristic from a predefined heuristic repository. A chromosome consists of a number of points in a simplified state space, each point being labeled with a single heuristic. A chromosome solves a problem instance as follows: given an instance and having computed its numerical representation P, find the closest block R in the chromosome (with Euclidean distance) and apply the single heuristic recorded on the label. This will place one or several items and will produce a new problem-state representation P'. The process is repeated until all pieces are placed and a complete solution has been constructed. The GA's task is to find a chromosome (a hyper-heuristic) that is capable of obtaining good solutions for a wide variety of problems.

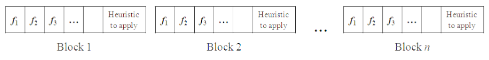
f_1 f_2 f_3 … Heuristic to apply
Block 1
f_1 f_2 f_3 … Heuristic to apply
Block 2
…
f_1 f_2 f_3 … Heuristic to apply
Block n

Figure 1: A chromosome is a set of blocks.

The problem instances are divided into training and testing sets. The GA is used with the training set until a termination criterion is met and a general hyper-heuristic has been evolved. All instances in the testing set are then solved with this hyper-heuristic. The quality of a solution, produced by either a single heuristic or a hyper-heuristic, is based on the percentage of usage for each object and is given by:

$$Q = \frac{\sum_{i=1}^{N} U_i^2}{N} \quad . \tag{1}$$

where N is the total number of objects used and U_i is the fractional utilization for each object i. Note that $0 < Q \leq 1$. The result of the best single heuristic for each instance is stored (BSH). The fitness function of a chromosome is the average difference between the solution qualities obtained by the chromosome with respect to the quality of the best single heuristic for every particular instance:

$$f = \frac{\sum_{k=1}^{m} (Q_k - BSH_k)}{m} \tag{2}$$

where BSH_k is the best quality solution obtained by a single heuristic for the k-th assigned instance, Q_k is the quality solution obtained by the hyper-heuristic for the k-th assigned instance and m is the number of instances solved so far. f is to be maximized. BSH_k and Q_k are computed using Equation 1.

4. Principal component analysis (PCA)

PCA is a useful multivariate statistical technique that has found application in fields such as face recognition and image compression. It is a common technique for finding patterns in high dimensional data (Smith, 2002). The general idea behind PCA has been rediscovered and renamed several times. For example, it is called the Karhunen Loeve method in electrical engineering, empirical orthogonal functions in geophysical areas, proper orthogonal decomposition in applied mathematics, and factor analysis in many other fields (Zhao et al., 2004). Moreover, PCA is often used as a clustering technique (Luss and d'Aspremont, 2007; Anzanello and Fogliatto, 2011).

PCA identifies new variables, called the principal components, which are linear combinations of the original variables. For visualization purposes, the first two (or three) components are usually chosen as new axis for plotting all observations. However, as much information will typically be lost in two or three-dimensional visualizations, it is important to systematically try different combinations of components. Each component can be interpreted as the direction which maximizes the variance of the observations when projected onto the component. As the principal components are uncorrelated, they may represent different aspects of the observations. The computation of the principal components for a dataset is based on linear algebra operations. If data are standardized (with zero average and standard deviation of one unit), the principal components are normalized eigenvectors of the covariance matrix of the instances and ordered according to how much of the variation present in the data they contain.

As a brief example, let us consider a 2-variable dataset plotted in Figure 2, in which the four larger points are special observations in some way. The vector showing the first principal component (PC1) goes through the cloud of points in the direction where the points are most spread. The second and last principal component (PC2) is orthogonal to the first. The projection of the observations over PC1 is shown in Figure 2b. The closeness of the four larger points is partially preserved in this dimensionality reduction from two dimensions to one. For 3D data, the plane through the data where the points are most spread is built by the first two principal components. This is the plane that minimizes the sum of squares of the orthogonal distances from all points to the plane. A two-dimensional visualization of the 3D data is the orthogonal projection of the points over the plane.

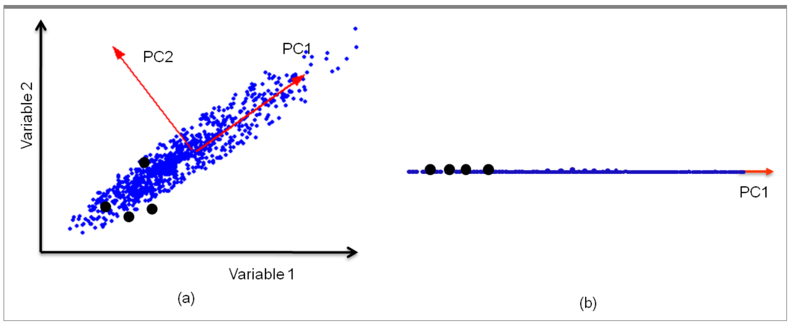

Figure 2: An example of principal component analysis for a dataset of 2 variables. (a) PCA identifies the two orthogonal directions (PC1 and PC2) along which the data have the largest spread. (b) Observations plotted in one dimension using their projections onto PC1.

5. Experimental Setup

This section describes the set of instances and the problem features computed for each instance.

5.1. Testbed instances

Our experimental testbed comprises 1417 instances of different types, which are summarized in Table 1. The one-dimensional problem instances (397 in total) were drawn from the literature as follows. The first eight types in Table 1 (named DB1 and DB2) are taken from Scholl et al. (1997), where we chose one out of every four instances in Scholl's databases 1 and 2. Wäscher instances come from Wäscher and Gau (1996), and the last four types are triplets from Falkenauer (1996) whose optimal solutions have exactly 3 items per bin with zero waste. The testbed includes 540 two-dimensional instances with convex polygonal pieces that were randomly generated by Terashima-Marín et al. (2010). It also contains 30 rectangular instances (type *Conv I* from Table 1). The 480 2D instances containing some non-convex polygons were randomly produced (López-Camacho et al., 2011). The optimum of the 480 instances has zero waste since all objects can be completely filled. The first half of these instances were generated by splitting at least five pieces from each instance from types *Conv A*, *Conv B*, *Conv C*, *Conv F*, *Conv H*, *Conv L*, *Conv M*, *Conv O*, respectively. Convex pieces from these instances were randomly selected and were split in one convex and one non-convex polygon. The other half of the non-convex instances was produced by creating new convex instances and then splitting some of the pieces into non-convex polygons.

The experimental testbed exhibits a variety of feature values. For example, there are instances whose pieces have an average size of 1/30 of the object, while others have huge pieces (averaging almost 2/3 of object size). The optimum number of objects (or best known results) ranges from 2 to 373. Rectangularity represents the proportion between the area of a piece and the area of a horizontal rectangle containing it. Among the 2D instances, the average rectangularity varies between 0.35 and 1.0.

5.2. Meta-data for the bin packing problem

A critical part of the proposed analysis is the identification of suitable features that might explain heuristic performance. The bin packing problem is a source of many possible features. Twenty three numerical features were computed for each instance (see Table 2).

For 1D instances, the area is computed assuming that all items and bins have a fixed width (i.e their width variance is zero), which means that their area is proportional to the height. For 2D instances, the width variance is greater than zero. All 1D items and 2D rectangles have rectangularity of one. According to Wang (1998), the degree of concavity is defined as the concaveness of the **largest** internal angle and it can be computed by $DC = \frac{B}{A}$ (see Figure 3). For 1D items and 2D convex polygons (including rectangles), the degree of concavity is equal to one. The degree of concavity for a concave polygon is more than one.

The convex hull of a given set S of points in the plane, is the smallest convex polygon that contains all of the points of S. The convex hull may be easily visualized by imagining an elastic band stretched open

Table 1: Characteristics of the problem instances.

	Number of instances	Number of pieces	Average piece size	Piece size standard deviation	Average rectangularity	Percentage of right angles	Percentage of orthogonal sides	Average of concavity degree	Average of ratio area / convex hull	Optimal (number of objects)
1D problem instances										
minimum			0.106	0.011						6
average			0.359	0.128						81.4
maximum			0.669	0.322						373
			Average of instances per type							
DB1 n1	45	50	0.485	0.199						26.6
DB1 n2	45	100	0.487	0.202						51.8
DB1 n3	45	200	0.489	0.203						102.7
DB1 n4	45	500	0.488	0.202						254
DB2 n1	30	50	0.199	0.060						10.5
DB2 n2	30	100	0.201	0.062						20.7
DB2 n3	30	200	0.199	0.061						40.2
DB2 n4	30	500	0.198	0.060						99.8
Wäscher	17	57 - 239	0.255	0.062						unknown
Trip60	20	60	0.333	0.077						20
Trip120	20	120	0.333	0.075						40
Trip249	20	249	0.333	0.075						83
Trip501	20	501	0.333	0.074						167
Convex 2D problem instances										
minimum			0.033	0.014	0.35	11	34	1	1	2
average			0.154	0.100	0.68	42	65	1	1	5.94
maximum			0.354	0.280	1	100	100	1	1	15
			Average of instances per type							
Conv A	30	30	0.100	0.069	0.70	42	68	1	1	3
Conv B	30	30	0.333	0.162	0.87	67	84	1	1	10
Conv C	30	36	0.167	0.124	0.68	36	63	1	1	6
Conv D	30	60	0.050	0.036	0.57	23	51	1	1	3
Conv E	30	60	0.050	0.035	0.41	12	38	1	1	3
Conv F	30	30	0.067	0.050	0.59	29	57	1	1	2
Conv G	30	36	0.332	0.156	0.87	67	83	1	1	unknown
Conv H	30	36	0.333	0.158	0.86	67	84	1	1	12
Conv I	30	60	0.053	0.017	1	100	100	1	1	3
Conv J	30	60	0.067	0.034	0.83	68	83	1	1	4
Conv K	30	54	0.154	0.150	0.63	34	60	1	1	6
Conv L	30	30	0.100	0.075	0.51	23	50	1	1	3
Conv M	30	40	0.125	0.102	0.55	28	55	1	1	5
Conv N	30	60	0.033	0.024	0.62	32	60	1	1	2
Conv O	30	28	0.250	0.223	0.57	27	58	1	1	7
Conv P	30	56	0.143	0.173	0.49	17	43	1	1	8
Conv Q	30	60	0.250	0.053	0.89	51	76	1	1	15
Conv R	30	54	0.167	0.153	0.63	36	62	1	1	9
Non-convex 2D problem instances										
minimum			0.044	0.036	0.38	6	27	1.004	0.834	2
average			0.160	0.135	0.59	26	50	1.130	0.930	5.9
maximum			0.333	0.314	0.84	60	74	1.560	0.987	12
			Average of instances per type							
Nconv A	30	35 - 50	0.074	0.062	0.60	28	52	1.12	0.935	3
Nconv B	30	40 - 52	0.214	0.158	0.69	38	58	1.22	0.923	10
Nconv C	30	42 - 60	0.123	0.111	0.59	25	49	1.11	0.939	6
Nconv F	30	35 - 45	0.051	0.045	0.53	20	46	1.10	0.940	2
Nconv H	30	42 - 60	0.245	0.163	0.73	46	64	1.15	0.944	12
Nconv L	30	35 - 45	0.076	0.065	0.47	16	41	1.10	0.941	3
Nconv M	30	45 - 58	0.099	0.092	0.50	20	46	1.07	0.956	5
Nconv O	30	33 - 43	0.186	0.190	0.51	19	46	1.10	0.940	7
Nconv S	30	17 - 20	0.106	0.097	0.45	10	33	1.16	0.918	2
Nconv T	30	30 - 40	0.293	0.239	0.60	26	51	1.24	0.916	10
Nconv U	30	20 - 33	0.197	0.161	0.55	17	44	1.19	0.888	5
Nconv V	30	15 - 18	0.306	0.236	0.62	27	54	1.09	0.936	5
Nconv W	30	24 - 28	0.155	0.097	0.78	53	69	1.12	0.931	4
Nconv X	30	25 - 39	0.097	0.072	0.66	32	53	1.17	0.895	3
Nconv Y	30	40 - 50	0.135	0.129	0.61	25	51	1.09	0.943	6
Nconv Z	30	60	0.200	0.234	0.54	19	45	1.09	0.940	12

Table 2: Total set of 23 numerical features computed for the bin packing instances.

1. Number of pieces,
2. Mean number of sides for the instance pieces,
3. Variance of the number of sides of all instance pieces,
4. Mean area for the instance pieces (area for each piece is measured as a fraction of the object total area),
5. Variance of the area of all instance pieces,
6. Mean height for the instance pieces (height for each piece is measured as a fraction of the object height with the difference between its maximum and minimum y coordinates),
7. Variance of the height of all instance pieces,
8. Mean width for the instance pieces (width for each piece is measured as a fraction of the object width with the difference between its maximum and minimum x coordinates),
9. Variance of the width of all instance pieces,
10. Mean rectangularity for the instance pieces,
11. Variance of the rectangularity of all instance pieces,
12. Mean ratio (largest side)/(smallest side) for the instance pieces,
13. Variance of the ratio (largest side)/(smallest side) of all instance pieces,
14. Percentage of large pieces (whose area is greater than 1/2 of the object total area),
15. Percentage of small pieces (whose area is less than or equal to 1/4 of the object total area),
16. Percentage of right internal angles (respect to the total angles of all pieces of the instance),
17. Percentage of vertical/horizontal sides (respect to the total sides of all pieces of the instance),
18. Percentage of high rectangularity pieces (items which rectangularity is greater than 0.9),
19. Percentage of low rectangularity pieces (items which rectangularity is less than or equal to 0.5),
20. Percentage of non-convex pieces,
21. Average of the largest internal angle of all instance pieces,
22. Mean of the degree of concavity of the instance pieces (explained below),
23. Average of the proportion (area of piece)/(area of convex hull) for all instance pieces (explained below).

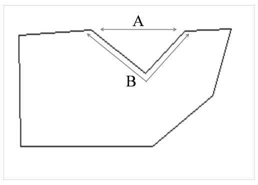

Figure 3: Degree of concavity.

to encompass the given object. When released, it will assume the shape of the required convex hull. The convex hull of a given polygon is defined as the convex hull of all its vertices.

Twenty three is a large number of features. Therefore we consider the data-mining methodology proposed in (López-Camacho et al., 2010) for selecting the most relevant features related to heuristic performance. The methodology requires solving all instances with all heuristics in the repository H, calculating their performances Q (using Equation 1). For a given instance, $\mathbf{q}$ is the performance vector for a set of heuristics H, $\mathbf{q} = \{q_1, q_2, \cdots, q_H\}$. These performance vectors are normalized using their norm: $|\mathbf{q}| = \sqrt{q_1^2 + \ldots + q_H^2}$. The normalized performance $\mathbf{q}*$ is given by:

$$\mathbf{q}^* = \frac{\mathbf{q}}{|\mathbf{q}|} \tag{3}$$

This normalization helps to reduce the effect of *easy* and *hard* instances. As further discussed in the example below, an instance is considered easy to solve if it obtains high performance values, and hard to solve if it obtains performance values.

Heuristics' performance example. Let $\mathbf{q}_a$ and $\mathbf{q}_b$ be the performance vector for two given instances evaluated with $H = 6$ heuristics. Let

$\mathbf{q}_a = \left[\begin{array}{cccccc} 0.906 & 0.906 & 1.000 & 0.918 & 0.918 & 0.922 \end{array}\right]$ with length $|\mathbf{q}_a| = 2.275$, and

$\mathbf{q}_b = \left[\begin{array}{cccccc} 0.604 & 0.614 & 0.613 & 0.616 & 0.619 & 0.617 \end{array}\right]$ with length $|\mathbf{q}_b| = 1.503$.

The third heuristic is the best for instance a as it gets the maximum performance (1.000), meaning that all objects could be filled at 100%. For instance b, the fifth heuristic is the best (performance = 0.619), followed by the sixth heuristic (performance = 0.617). In this example, all six heuristics obtain higher performance in instance a compared to instance b, indicating that instance a is easier to solve than instance b. The vectors of normalized performance are given by:

Table 3: Features selected by a data-mining methodology as the most related to heuristic performance. These features were selected out from a total of 23 characteristics (listed in Table 2).

Feature	Description
1	Number of pieces.
2	Mean of the pieces area.
3	Variance of the area.
4	Mean of the rectangularity.
5	Variance of the rectangularity.
6	Mean of the height.
7	Variance of the width.
8	Percentage of huge pieces (whose area is above 1/2 of the object area).
9	Mean of degree of concavity.

$\mathbf{q}_a^* = \frac{\mathbf{q}_a}{|\mathbf{q}_a|} = \begin{bmatrix} 0.398 & 0.398 & 0.439 & 0.403 & 0.403 & 0.405 \end{bmatrix}$, and
$\mathbf{q}_b^* = \frac{\mathbf{q}_b}{|\mathbf{q}_b|} = \begin{bmatrix} 0.402 & 0.408 & 0.408 & 0.410 & 0.412 & 0.410 \end{bmatrix}$.
Vectors $\mathbf{q}_a^*$ and $\mathbf{q}_b^*$ still indicate the heuristics producing the best and worst results for a particular instance, but without distinguishing which instance gets the higher absolute performance.

The normalized performance vectors are then used for clustering all instances, so that clusters group those instances better solved by the same heuristics, regardless of their difficulty. Additional steps in the feature-reduction methodology require pruning highly correlated features as they carry redundant information, and applying statistical regressions methods to identify the features which better predict the instance clusters (López-Camacho et al., 2010).

Upon applying this methodology (López-Camacho et al., 2010) to all the testbed instances (Table 1) and the 23 features computed (Table 2), nine features were obtained (see Table 3), which are used for the PCA discussed below. The objective of the analysis is to determine whether and how these nine features relate with the performance of both the single heuristics and hyper-heuristics described in Section 3.

6. Results and Analysis

The PCA was conducted using the *R* package (R Development Core Team, 2011), considering 1417 problem instances of different types (Table 1) and 9 variables per instance (Table 3). All variables were standardized (average of zero and standard deviation of 1) to ensure magnitude consistency. The analysis reveals that the first two principal components explain 42.7% and 22.3% of the variance, respectively. That is, 65% of the data variation is maintained by the plot in Figure 4a. The third principal component explains 11% of the dataset variance while the remaining 6 principal components explain jointly the remaining 24%. We select the first two principal components to plot all the dataset (Figure 4a). The three graphs below (Figure 4b.1, 4b.2 and 4b.3) show the location of the three main instance categories (1D, convex 2D and non-convex 2D). The 540 2D convex instances include 30 rectangular cases, which are concentrated in the circle of Figure 4b.2. In these plots, close points represent instances that are similar according to the 9 variables.

Each principal component is a linear combination of the 9 variables. The coefficients (called loadings) for each of the two main principal components are shown in Table 4. Loadings with the largest absolute value provide an interpretation for the new variables PC1 and PC2.

The PC1 and PC2 scores, representing the horizontal and vertical coordinates in Figure 4, are obtained by multiplying the standardized variables by the respective vector of loadings. Let $\mathbf{x}$ be the vector of the nine standardized variables for a given instance, and α_1 and α_2 be the vectors of loadings for PC1 and PC2, respectively. For example, a 2D regular instance has
$\mathbf{x}' = \begin{bmatrix} -0.25 & -1.12 & -0.97 & 1.28 & -1.3 & -0.54 & -0.95 & -0.57 & -0.5 \end{bmatrix}$. Then,

$$\mathbf{x}'\alpha_1 = \begin{bmatrix} -0.25 & -1.12 & \cdots & -0.5 \end{bmatrix} \begin{bmatrix} -0.34 \\ -0.42 \\ \cdots \\ 0.19 \end{bmatrix} = -0.72$$

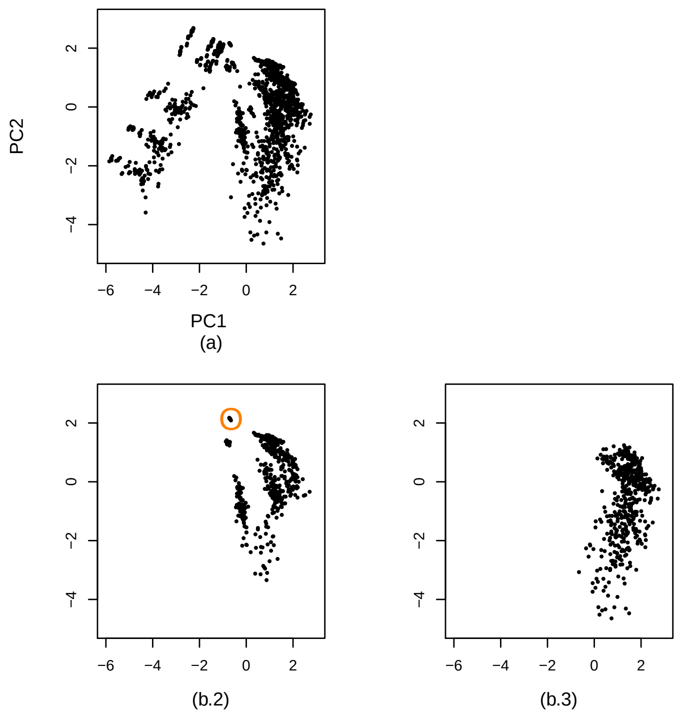

Figure 4: (a) The whole set of 1417 instances plotted along PC1 (horizontal axis) and PC2 (vertical axis). (b.1) 397 1D instances. (b.2) 540 convex 2D instances, including the 30 rectangular instances, which are concentrated in the place indicated by the circle. (b.3) 480 irregular 2D instances (non-convex).

is the instance horizontal coordinate (PC1 score). The vertical coordinate is given by $\mathbf{x}'\alpha_2$.

As indicated in Table 4, variables *variance of rectangularity* (0.42) and *variance of width* (0.34) have a positive projection onto the first component. In consequence, the largest positive values of PC1 refer to instances with high variability of shapes and pieces width. *Variance of width* measures variability only for 2D instances, as 1D instances have a constant value for this metric (zero). Variables *number of pieces* (-0.34), *mean area* (-0.42), *mean rectangularity* (-0.44) and *percentage of huge pieces* (-0.36) have a negative projection onto the first component. In consequence, instances with many items and large regular pieces have negative PC1 values. PC1, then, almost perfectly separates the 1D instances (plotted in the left side, see Figure 4b.1) from the 2D instances (plotted in the right side, see Figures 4b.2 and 4b.3).

On the second principal component, PC2, variable *variance of area* (-0.63) has the highest negative projection. This variable measures variability in item sizes for both 1D and 2D instances. Since the loading is negative (-0.63), the greater the variability the lower the PC2 score. Therefore, instances with huge variety of items sizes tend to be plotted lower in Figure 4a. Variables *mean area* (-0.34), *variance of*

Table 4: Loadings for the two main principal components of the data. Features 1 through 9 are those referred in Table 3. Figures with largest absolute values are in bold font.

Feature	1 Number of pieces	2 Mean area	3 Variance of area	4 Mean rec- tangularity	5 Variance of rectangularity	6 Mean height	7 Variance of width	8 % of huge pieces	9 Concavity degree
PC1	**-0.34**	**-0.42**	-0.15	**-0.44**	**0.42**	0.18	**0.34**	**-0.36**	0.19
PC2	0.15	**-0.34**	**-0.63**	0.17	-0.13	-0.13	**-0.42**	**-0.42**	-0.22

width (−0.42) and *percentage of huge pieces* (−0.42) also have a negative projection onto PC2. Therefore, instances with the largest items tend to be plotted lower in the plots of Figure 4. All 2D regular instances have high positive values for PC2, thus, they are plotted in the upper part of the plot (inside the circle in Figure 4b.2).

6.1. Distribution of features across the PCA map

The nine plots in Figure 5 illustrate the distribution of the selected nine features' values on the PCA map, respectively. The darker the color the higher the feature value. Each section of the plot represents a different combination of features. For example, the coloring patterns across the plots in Figure 5 indicate that the bottom-left portion contain instances with high number of pieces, high mean area, high variance of area, high mean rectangularity, and low variance of rectangularity.

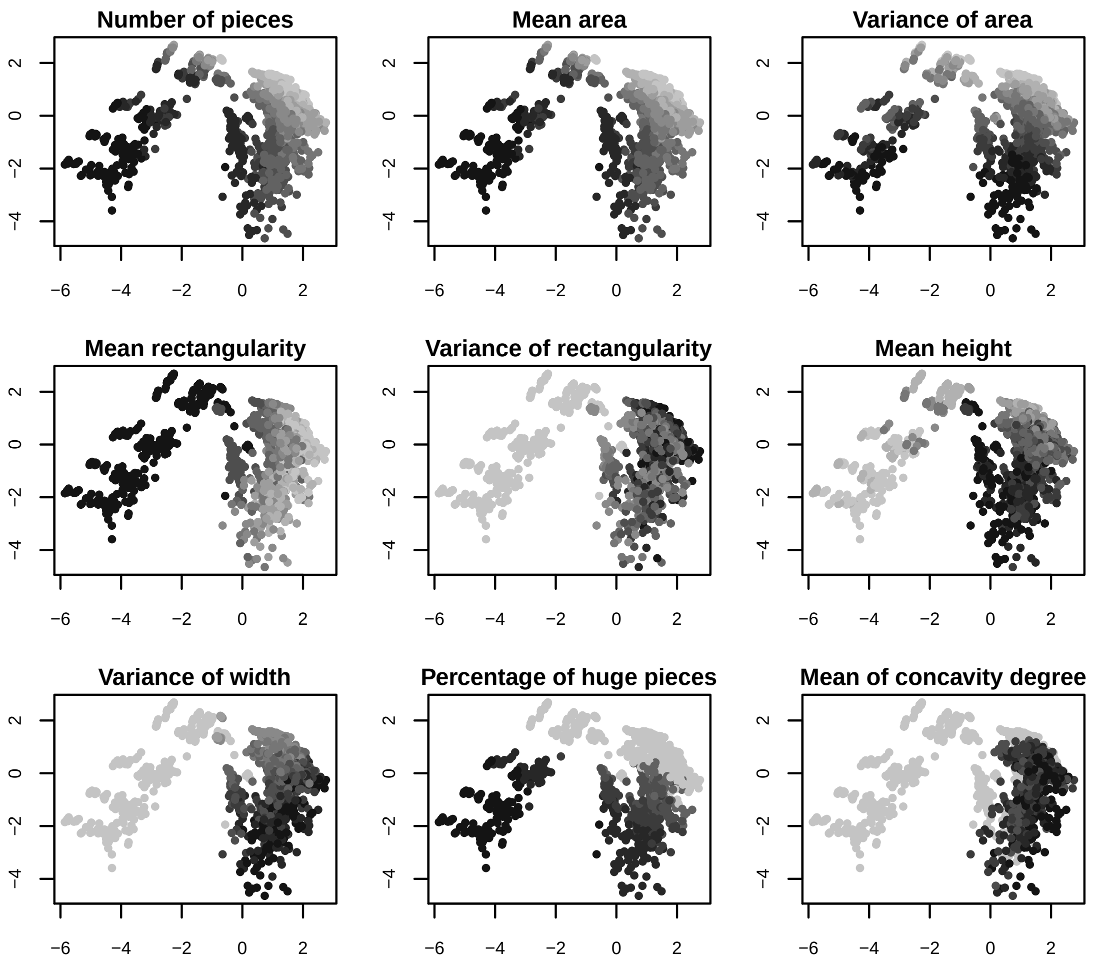

Figure 5: Distribution of feature values across the PCA map for the nine selected instance features (the darker the color the higher the value). The horizontal axis represents PC1 scores, while the vertical axis PC2 scores.

6.2. Distribution of heuristic performance across the PCA map

The six plots in Figure 6 illustrate the distribution of the six heuristics performance values (calculated with Equation 1) on the PCA map, respectively. Again, the darker the color, the higher the performance value and thus the easier to solve the instance. The color pattern is very similar across the six plots, indicating that instance difficulty is almost independent from the heuristic used to solve it. The left portion of the plots, which represents the 1D instances, contain the higher proportion of darker (easy to solve) instances. The hardest to solve instances (lighter dots) are located at the top-right portion of the plots. These instances have high PC1 and PC2 scores. According to Figure 5, this portion of the plots represents instances with low number of pieces which are small in size, with reduced variance of area, low value but high variance of rectangularity, minimum percentage of huge pieces and high average of concavity degree. In other words, the hardest to solve instances for the six heuristics studied are those with a low number of small and highly irregular shapes. More specialized heuristics should be considered when solving this type of instances.

Figure 7 is analogous to Figure 6 but reports the **normalized** performances (calculated with Equation 3). Darker points in this case correspond to instances that are better solved by a particular heuristic compared to the rest. Different color patterns can now be found across the six subplots. For example, heuristics $\text{DJD}_{1/4}$ and $\text{DJD}_{1/3}$ show low performance for instances in the bottom-left portion of the map (very light dots), when compared with the performance of the other four heuristics. This suggests that instances with low PC1 and PC2 scores are not especially suited for heuristics $\text{DJD}_{1/4}$ and $\text{DJD}_{1/3}$. According to Figure 5, low PC1 and PC2 values characterize instances with many large and regular pieces from the complete dataset. In general, these instances are better solved by simpler and faster heuristics such as FFD, Filler, BFD and even $\text{DJD}_{1/2}$ (which is faster than $\text{DJD}_{1/4}$ and $\text{DJD}_{1/3}$).

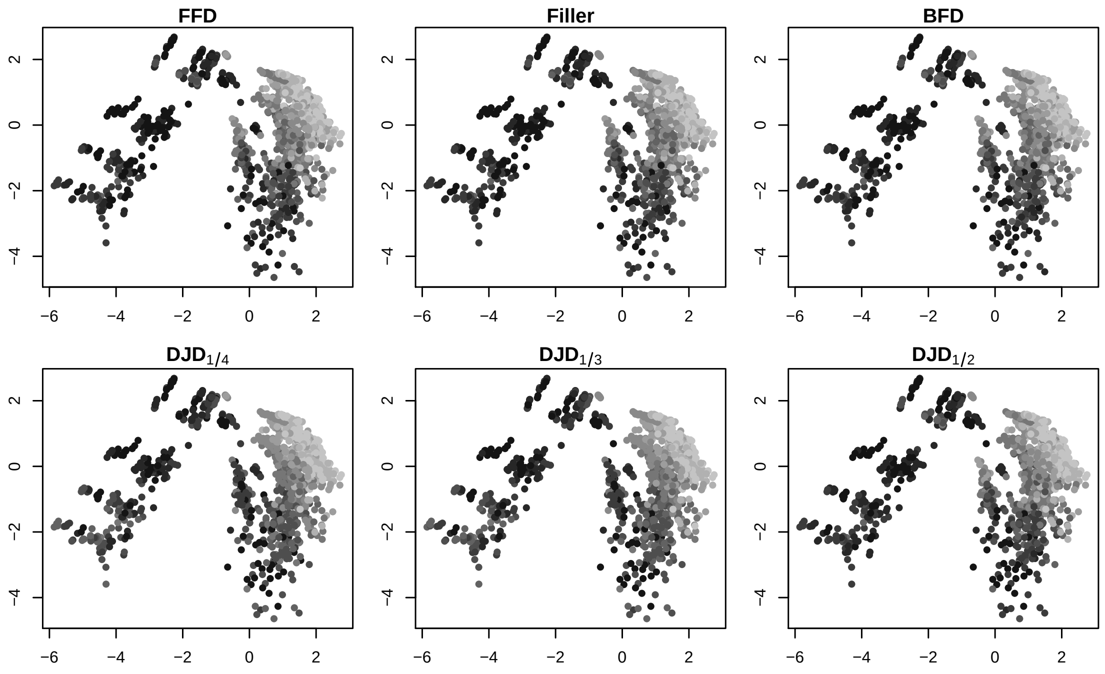

Figure 6: Distribution of performance values across the PCA map for the six heuristics considered (the darker the color the higher the value). The horizontal axis represents PC1 scores, while the vertical axis PC2 scores. Similar color patterns are observed across all heuristics, suggesting that instance difficulty has a high degree of independence from the heuristic used to solve it.

6.3. Normalized performance clustering

All testbed instances were grouped into 8 clusters according to the vectors of normalized performance. The k-means algorithm was used as it minimizes the variance within instances in each cluster, even though the number of observations in each cluster is unbalanced. The number of clusters was chosen according to the Hartigan criteria, described by Chiang and Mirkin (2007). Broadly speaking, instances with the same best and worst performing heuristic appear in the same cluster.

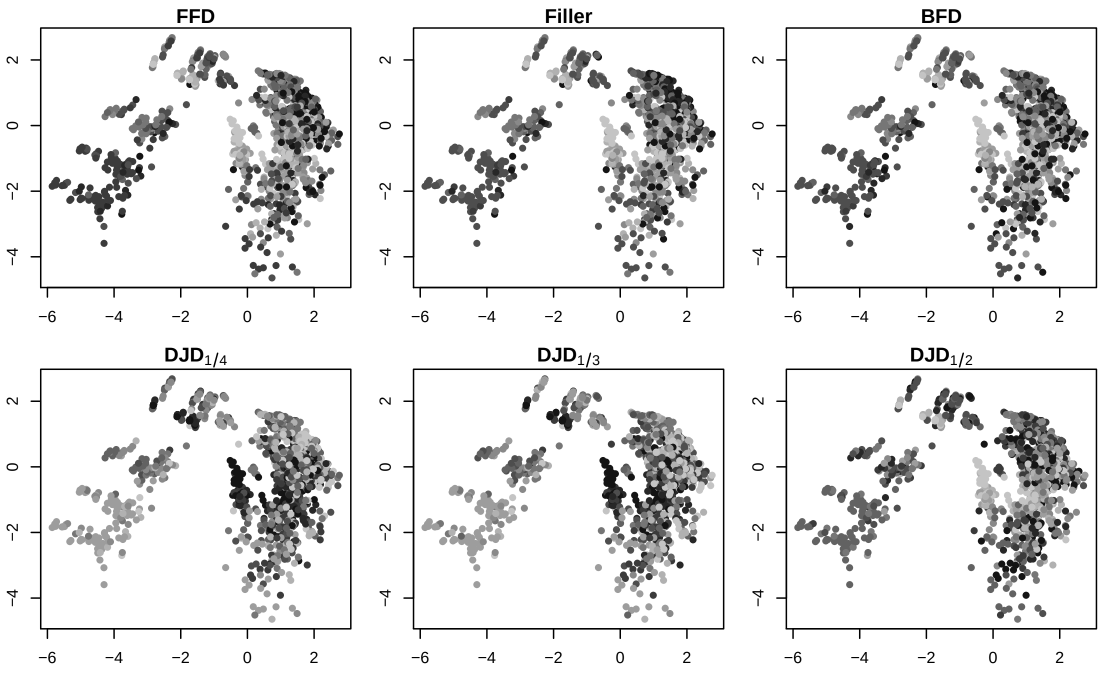

Figure 7: Distribution of *normalized* performance values across the PCA map for the six heuristics considered (the darker the color the higher the value). The horizontal axis represents PC1 scores, while the vertical axis PC2 scores. Different color patterns are now observed across all heuristics, suggesting that some heuristics are better suited to solve some types of instances.

Figure 8 illustrates the 8 instance clusters plotted against the PCA map. The first cluster (top left plot in Figure 8) is the most populated, and the remaining clusters concentrate a particular portion of instances. It is worth emphasizing that clusters were built considering heuristic performance only, while the PCA considered only instance features. However, for most clusters, instances are concentrated in a particular portion of the PCA map. As an example, only the first cluster (top left plot in Figure 8) contains instances belonging to the bottom-left PCA map (Figure 4a). This is a confirmation that instance features are indeed correlated with heuristic performance.

6.4. Identifying the best-performing heuristic

If the number of objects is considered as the heuristic performance metric, our study indicates that in 96% of the instances more than one heuristic will be the best performing. The quality metric expressed by Equation 1 is sometimes preferred (Terashima-Marín et al., 2010; Burke et al., 2007a; Falkenauer and Delchambre, 1992), as it distinguishes solutions with the same number of objects, rewarding solutions with filled or nearly filled objects. This is relevant because empty space concentrated in one or few objects is more likely to be useful later. With this metric, our study still produces more than one best performing heuristic (ties) in 58% of the instances. A more detailed analysis revealed that most of these ties occur among either the three 1-piece heuristics (FFD, Filler or BFD) or the three DJD heuristics. Therefore, we divided the heuristics into two classes: 1-piece and DJD. Figure 9 illustrates the distribution of the best-performing heuristic classes (1-piece in light color, DJD in dark color) on the PCA map. For this analysis, we discarded 16.4% of the instances as they were best solved by both a 1-piece and a DJD heuristic. We found that 15.2% of the instances are best solved by a 1-piece heuristic, while DJD heuristics are the best for the remaining 68.3% cases, indicating the effectiveness of this latter class of heuristics.

Some patterns can be observed in Figure 9. The oval encloses a group of light points, i.e instances which are best solved by 1-piece heuristics. These are 1D instances (see Figure 4b.1). However, not all 1D instances are best solved by 1-piece heuristics (note that the oval covers the bottom-right portion of the 1D

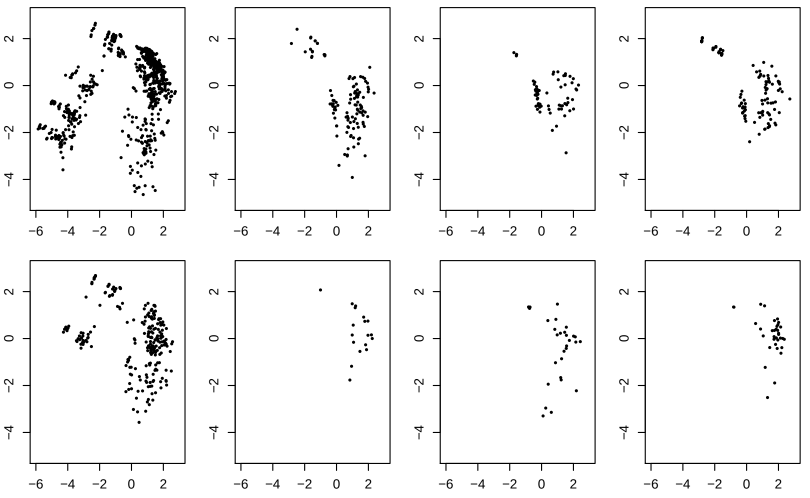

Figure 8: Clustering of instances according to heuristic normalized performance (8 clusters) and their distribution on the PCA map. The horizontal axis represents PC1 scores, while the vertical axis PC2 scores.

instances, which are in the left the PCA map). For a low and intermediate PC2 values, 1D instances with high PC1 scores (compared to other 1D instances) are best solved by 1-piece heuristics. High PC1 values are related to reduced number of items. Therefore, many of the 1D instances with fewer items, compared to other 1D instances, are best solved by 1-piece heuristics. The instances in the oval have an average number of pieces of 107, less than the half of the 1D average, but this number is higher than the overall average for our dataset.

Another concentration of light points (and thus instances best solved by the 1-piece heuristics) can be observed on the top-right of the plot (enclosed by the rectangle in Figure 9). This portion corresponds to 2D instances, and includes a mixture of convex and non-convex instances. These instances, however, share some features. Specifically, they have small pieces with low variance of area (this can be seen in Figure 5, and was verified by inspection). The non-convex instances in the rectangle have a low concavity degree compared to the rest of non-convex instances. Both convex and non-convex instances on the rectangle tend to have a smaller mean area. According to Figure 6, these instances in the rectangle are among the hardest to solve (the studied heuristics produced the lowest performance). It is therefore, surprising, that many of these instances are best solved by the simplest and fastest 1-piece heuristics (FFD, Filler or BFD).

6.5. Hyper-heuristic performance

Twenty hyper-heuristics were generated with different training and testing sets using the evolutionary framework described in Section 3.2. The best hyper-heuristic for each instance (that solving the instance with the least number of objects) was selected for the analysis. Figure 10 indicates with letters *b* (for best) and *w* (for worse) those instances whose best hyper-heuristic obtained a different performance when compared to the best performing simple heuristic. These cases seem to be concentrated on particular sections of the PCA map. For the 1D, however, the best and worst cases are mixed in these sections. Therefore, this particular analysis seems to identify different hyper-heuristic performances but does not clearly distinguish between solving cases with fewer or more objects. The rectangle in Figure 10 encloses fifteen *b*'s and three *w*'s that corresponds to 1D instances. These have the highest PC1 and PC2 values for 1D instances (see Figure 4b.1), which correlates with instances with both small value and variation of items areas. Therefore, the rectangular region encloses instances with small items of similar sizes. These are likely to be best solved by hyper-heuristics. A closer analysis revealed that 10 of the *b*'s in the rectangle represent instances with 501 items whose optimum has 3 items per bin (type *Trip501* in Table 1). Hyper-heuristics, then, found a better result for half of the *Trip501* instances. The other 1D cases in the rectangle are instances of *DB2*

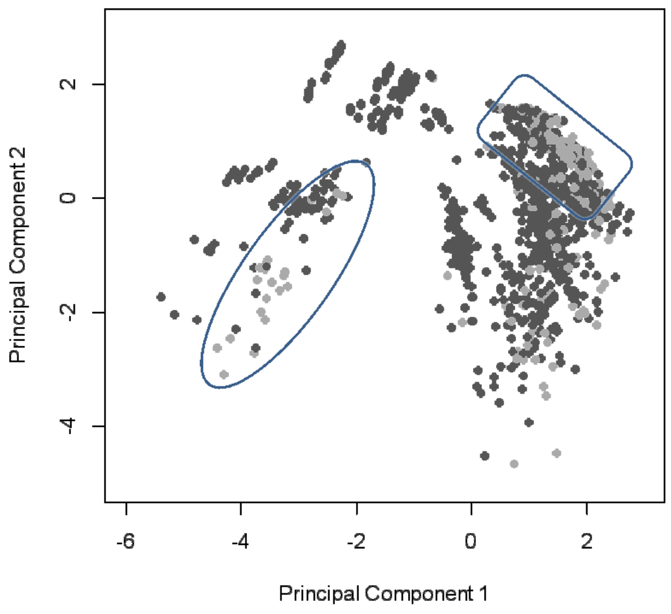

Figure 9: Identifying the best-performing heuristic. Light grey dots represent instances better solved by 1-piece heuristics (FFD, Filler or BFD), while dark dots represent instances better solved by the DJD heuristics.

and *Trip249*, all with smaller items than the 1D average. The b's and w's on the right portion of the plot correspond to 2D convex and non-convex instances. Most of them with PC2 values close to 0 (inside the circle).

For some instances, hyper-heuristics achieve better results than the best single heuristic for that instance. These cases support the use of hyper-heuristics, since for some applications, any reduction in material is extremely valuable. For most instances, the evolved hyper-heuristic produces the same quality than the best single heuristic. This is also beneficial, as the choice of best heuristic varies from instance to instance, and this is not known in advance. Using a hyper-heuristic may be, therefore, preferable than selecting a single heuristic for all problem instances (López-Camacho et al., 2011). After the hyper-heuristic is evolved, the computational cost of applying a generated hyper-heuristic to a problem is lower than the time used in applying all the heuristics and selecting the best result (Terashima-Marín et al., 2010).

7. Conclusions

This analysis constitutes a first step into the multivariate nature of the bin packing problem structure. PCA is applied for the first time to further compress the instance feature space and understand the impact of problem structure on the performance of packing heuristics and hyper-heuristics. A large testbed of instances of different types is considered, which makes the analysis reliable and robust. The dataset contains both one-dimensional and two-dimensional regular and irregular problems; and encompasses a wide range of feature values.

Our analysis indicates that some instances are clearly harder to solve than others for all the studied heuristics and hyper-heuristics. The PCA maps give a valuable indication of the combination of features characterizing easy and hard to solve instances. The hardest instances are those with a small number of both huge and small pieces, a low value but high variance in rectangularity, and high concavity degree (i.e instances with some small and highly irregular shapes). However, more than one heuristic can equally solve most of the instances. The DJD heuristics are able to best solve most of the instances (around 70%), but simpler and faster 1-piece heuristics can outperform them in some cases. In particular, our analysis revealed that DJD heuristics are not well suited for solving 1D instances with low and moderate number

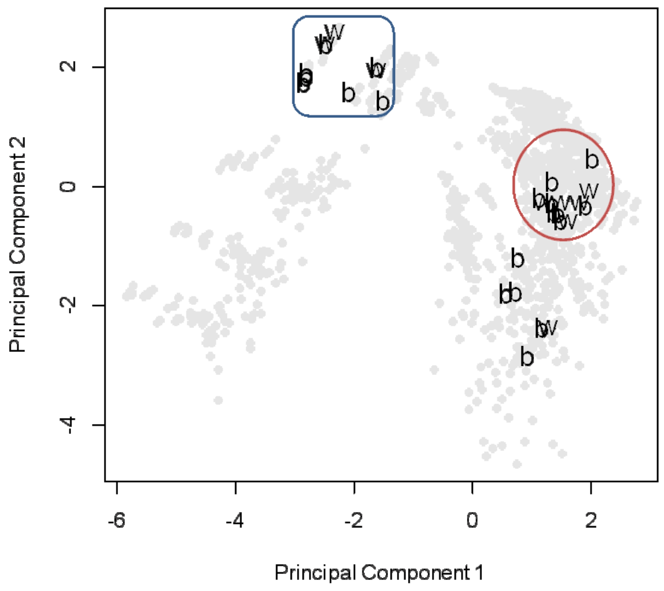

Figure 10: Visualizing the instances that were solved differently by hyper-heuristics. 28 instances were solved with fewer objects by the best hyper-heuristic (**b** letters), while in 9 instances, the best hyper-heuristic could reach the best single heuristic result (**w** letters). For the remaining instances, the best hyper-heuristic obtained the same number of objects than the best performing heuristic.

of pieces. More surprisingly, DJD heuristics are also outperformed by simpler and faster 1-piece heuristics when solving some difficult convex and non-convex 2D instances with small areas with low variability.

The bin packing problem has a complex structure. Our analysis suggests that there are indeed correlations between instance features and heuristic performance. However, few simple rules can be formulated. It is necessary to consider feature combinations and their interactions in order to have a clearer insight into performance prediction. This contrasts with other more structured problems such as constraint satisfaction, where a couple of well selected features (density and tightness) is enough to predict which of two heuristics will be the best (Ortiz-Bayliss et al., 2010).

It is important to carefully choose the set of features to describe the problem structure. They should be able to both characterize problem instances and differentiate algorithm performance (Smith-Miles et al., 2009). Furthermore, features must avoid redundancy and show different aspects of the problem structure. For example, in the bin packing problem, it is possible to have different features related to the size of the items, such as average item area, percentage of small items, etc. Future work will characterize BPP with different sets of features and employ different BPP datasets. This analysis can be generalized to other problem domains where a set of features characterizes instances, and several heuristics are available to solve the problem.

Acknowledgments

This research was supported in part by ITESM under the Research Project PRY-075 and the CONACYT Project under grant 99695.

Anzanello, M. J., Fogliatto, F. S., 2011. Selecting the best clustering variables for grouping mass-customized products involving workers' learning. International Journal of Production Economics 130, 268–276.

Burke, E. K., Curtois, T., Hyde, M. R., Kendall, G., Ochoa, G., Petrovic, S., Rodríguez, J. A. V., Gendreau, M., 2010a. Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms. In: IEEE Congress on Evolutionary Computation. pp. 1–8.

Burke, E. K., Hellier, R. S. R., Kendall, G., Whitwell, G., 2006. A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. Operations Research 54 (3), 587–601.

Burke, E. K., Hellier, R. S. R., Kendall, G., Whitwell, G., 2007a. Complete and robust no-fit polygon generation for the irregular stock cutting problem. European Journal of Operational Research 179 (1), 27–49.

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Özcan, E., Woodward, J., 2010b. A classification of hyper-heuristic approaches. Vol. 146 of International Series in Operations Research & Management Science. Springer US, pp. 449–468.

Burke, E. K., Hyde, M. R., Kendall, G., Woodward, J., 2012. Automating the packing heuristic design process with genetic programming. Evolutionary Computation 20 (1), 63–89.

Burke, E. K., Woodward, J., Hyde, M. R., Kendall, G., 2007b. Automatic heuristic generation with genetic programming: Evolving a jack-of-all-trades or a master of one. In: Genetic and Evolutionary Computation Conference, GECCO'07. pp. 7–11.

Cheng, C. H., Feiring, B. R., Cheng, T. C. E., 1994. The cutting stock problem - a survey. International Journal of Production Economics 36 (3), 291–305.

Chiang, M., Mirkin, B., 2007. Experiments for the number of clusters in k-means. In: Neves, J., Santos, M. F., Machado, J. M. (Eds), Progress in Artificial Intelligence. Vol. 4874 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, Ch. 33, pp. 395–405.

Dychoff, H., 1990. A typology of cutting and packing problems. European Journal of Operational Research 44, 145–159.

Falkenauer, E., 1996. A Hybrid Grouping Genetic Algorithm for Bin Packing. Journal of Heuristics 2 (1), 5–30.

Falkenauer, E., Delchambre, A., 1992. A genetic algorithm for bin packing and line balancing. In: Proceedings of the 1992 IEEE International Conference on Robotics and Automation. pp. 1186–1192.

Hu-yao, L., Yuan-jun, H., 2006. NFP-based nesting algorithm for irregular shapes. In: Symposium on Applied Computing. ACM Press, New York, NY, USA, pp. 963–967.

Kanda, J., Carvalho, A., Hruschka, E., Soares, C., 2011. Selection of algorithms to solve traveling salesman problems using meta-learning. International Journal of Hybrid Intelligent Systems 8, 117–128.

Kohonen, T., Schroeder, M. R., Huang, T. S. (Eds), 2001. Self-Organizing Maps, 3rd Edition. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Leyton-Brown, K., Nudelman, E., Shoham, Y., 2002. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In: Van Hentenryck, P. (Ed.), Principles and Practice of Constraint Programming - CP 2002. Vol. 2470 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 91–100.

Lodi, A., Martello, S., Monaci, M., 2002. Two-dimensional packing problems: A survey. European Journal of Operational Research. 141 (2), 241–252.

López-Camacho, E., Ochoa, G., Terashima-Marín, H., Burke, E.K., 2013. An Effective Heuristic for the Two-dimensional Irregular Bin Packing Problem. Annals of Operations Research, doi: 10.1007/s10479-013-1341-4.

López-Camacho, E., Terashima-Marín, H., Ross, P., 2010. Defining a problem-state representation with data mining within a hyper-heuristic model which solves 2D irregular bin packing problems. In: Kuri-Morales, Á. F., Simari, G. R. (Eds), Advances in Artificial Intelligence IBERAMIA. Vol. 6433 of Lecture Notes in Computer Science. Springer, pp. 204–213.

López-Camacho, E., Terashima-Marín, H., Ross, P., 2011. A hyper-heuristic for solving one and two-dimensional bin packing problems. In: 13th annual conference companion on Genetic and evolutionary computation. GECCO '11. ACM, New York, NY, USA, pp. 257–258.

Luss, R., d'Aspremont, A., 2007. Clustering and feature selection using sparse principal component analysis. CoRR abs/0707.0701.

Ochoa, G., Hyde, M. (2011) The Cross-domain Heuristic Search Challenge (CHeSC 2011). Website, `http://www.asap.cs.nott.ac.uk/chesc2011/`

Ochoa, G., Hyde, M., Curtois, T., Vázquez-Rodríguez, J.A., Walker, J., Gendreau, M., Kendall, G., Parkes, A.J., Petrovic, S., Burke, E.K. (2012a) Hyflex: a benchmark framework for cross-domain heuristic search. In: Proceedings of the 12th European conference on Evolutionary Computation in Combinatorial Optimization, EvoCOP'12, Springer-Verlag, Lecture Notes in Computer Science, Vol. 7245, pp 136–147

Ochoa G, Walker J, Hyde M, Curtois T (2012b) Adaptive evolutionary algorithms and extensions to the hyflex hyper-heuristic framework. In: Parallel Problem Solving from Nature - PPSN XII, Springer, Lecture Notes in Computer Science, Vol. 7492, pp 418–427

Okano, H., 2002. A scanline-based algorithm for the 2D free-form bin packing problem. Journal of the Operations Research Society of Japan 45 (2), 145–161.

Ortiz-Bayliss, J. C., Özcan, E., Parkes, A. J., Terashima-Marín, H., 2010. Mapping the performance of heuristics for constraint satisfaction. In: IEEE Congress on Evolutionary Computation'10. pp. 1–8.

R Development Core Team, 2011. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0. http://www.R-project.org/.

Ringner, M., 2008. What is principal component analysis? Nature Biotechnology 26 (3), 303–304.

Ross, P., 2005. Hyper-heuristics. In: Burke, E. K., Kendall, G. (Eds), Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. Springer, New York, pp. 529–556.

Scholl, A., Klein, R., Jürgens, C., 1997. Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Computers & Operations Research 24, 627–645.

Smith, L. I., 2002. A tutorial on principal components analysis. Tech. rep., Cornell University, USA.

Smith-Miles, K. A., 2008a. Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Computing Surveys 41, 6:1 – 6:25.

Smith-Miles, K. A., 2008b. Towards insightful algorithm selection for optimisation using meta-learning concepts. In: IEEE World Congress on Computational Intelligence. IEEE International Joint Conference on Neural Networks. IJCNN. IEEE, pp. 4118–4124.

Smith-Miles, K. A., James, R. J., Giffin, J. W., Tu, Y., 2009. In: Stützle, T. (Ed.), Learning and Intelligent Optimization. Springer-Verlag, Berlin, Heidelberg, Ch. A knowledge discovery approach to understanding relationships between scheduling problem structure and heuristic performance, pp. 89–103.

Terashima-Marín, H., Ross, P., Farías-Zárate, C. J., López-Camacho, E., Valenzuela-Rendón, M., 2010. Generalized hyper-heuristics for solving 2D regular and irregular packing problems. Annals of Operations Research 179, 369–392.

Uday, A., Goodman, E. D., Debnath, A. A., 2001. Nesting of irregular shapes using feature matching and parallel genetic algorithms. In: Goodman, E. D. (Ed.), Genetic and Evolutionary Computation Conference. Late Breaking Papers. pp. 429–434.

Wang, W. X., 1998. Binary image segmentation of aggregates based on polygonal approximation and classification of concavities. Pattern Recognition 31 (10), 1503–1524.

Wäscher, G., Gau, T., 1996. Heuristics for the integer one-dimensional cutting stock problem: A computational study. OR Spectrum 18, 131–144.

Wäscher, G., Hausner, H., Schumann, H., 2007. An improved typology of cutting and packing problems. European Journal of Operational Research. Forthcoming Special Issue on Cutting, Packing and Related Problems 183 (3), 1109–1130.

Yu, M., Lin, T., Hung, C., 2009. Active-set sequential quadratic programming method with compact neighbourhood algorithm for the multi-polygon mass production cutting-stock problem with rotatable polygons. International Journal of Production Economics 121, 148–161.

Zhao, X., Marron, J. S., Wells, M. T., 2004. The functional data analysis view of longitudinal data. Statistica Sinica 14 (3), 789–808.