



Homogeneous grouping of non-prime steel products for online auctions: a case study

Borja Ena¹ · Alberto Gomez² · Borja Ponte² · Paolo Priore² · Diego Diaz¹

Accepted: 15 March 2022 / Published online: 5 April 2022
© The Author(s) 2022

Abstract

Not all products meet customers' quality expectations after the steelmaking process. Some of them, labelled as 'non-prime' products, are sold in a periodic online auction. These products need to be grouped into the smallest feasible number of bundles as homogeneous as possible, as this increases the attractiveness of the bundles and hence their selling prices. This results in a highly complex optimisation problem, also conditioned by other requirements, with large economic implications. It may be interpreted as a variant of the well-known bin packing problem. In this article, we formalise it mathematically by studying the real problem faced by a multinational in the steel industry. We also propose a structured, three-stage solution procedure: (i) initial division of the products according to their characteristics; (ii) cluster analysis; and (iii) allocation of products to bundles via optimisation methods. In the last stage, we implement three heuristic algorithms: FIFO, greedy, and distance-based. Building on previous works, we develop 80 test instances, which we use to compare the heuristics. We observe that the greedy algorithm generally outperforms its competitors; however, the distance-based one proves to be more appropriate for large sets of products. Last, we apply the proposed solution procedure to real-world datasets and discuss the benefits obtained by the organisation.

Keywords Bin packing problem · Distance-based heuristics · FIFO algorithm · Greedy algorithm · Online auctions · Steel industry

1 Introduction

This work considers an important optimisation problem faced by many steel companies. This concerns the grouping of 'non-prime' steel products, i.e. those not meeting the quality requirements of prime orders for any reason (e.g. technical defects) that are then unassigned to the specific order, into bundles that are later sold through an auction mechanism. The grouping

Paolo Priore—Deceased.

✉ Borja Ponte
ponteborja@uniovi.es

¹ ArcelorMittal, Global R&D Asturias, Aviles, Spain

² Department of Business Administration, University of Oviedo, Gijón, Spain

needs to consider several product characteristics and operational restrictions. Well-designed bundles are more attractive in the eyes of bidders (potential customers); thus, the problem has considerable effects on the selling prices. In large, multinational steel companies, the auction generally takes place on a periodic basis with extensive amounts of non-prime products from many origins. This makes the optimisation problem computationally very demanding. All in all, we highlight that the grouping decision is a complex optimisation problem that is central to the auction process, and providing an effective solution strategy would arguably result in substantial benefits.

From an algorithmic perspective, this optimisation problem may be interpreted as a variant of the *bin packing problem* (BPP) (e.g. Coffman et al., 2013; Taylor et al., 2017; Yao, 1980), where a set of products of different volumes have to be allocated into the smallest possible number of bins of a given capacity. In terms of computational complexity, this is an NP-hard problem (Berlińska, 2020; Garey & Johnson, 1979). Therefore, finding optimal solutions is frequently impractical and approximate algorithms are commonly used, which are often able to provide high-quality solutions in reasonably low computational times.

In this article, we address the grouping problem of non-prime products by focusing on a specific case study in the steel sector. We examine the relevant characteristics of the problem under study, highlighting the distinctive features in comparison with the general BPP. We argue that it represents a new BPP model that has not been studied in prior literature. Then, we discuss different methodological approaches to solve this problem, from which we suggest a three-stage solution procedure. Our structured approach integrates a variety of methods, through which we deal with the complexity of the problem. The third stage, which optimises the allocation of products to bundles, incorporates three heuristic algorithms. We compare their performance, in terms of solution quality and computation time, through different experiments.

The remaining of this article is structured as follows. Section 2 describes in detail the specific problem under consideration, emphasising its economic relevance. We highlight the constraints that must be satisfied and the additional requirements that good solutions should meet. Sections 3 and 4 review the BPP and clustering streams of literature, respectively, which provide the theoretical background for our study. Section 5 formalises the optimisation problem mathematically. Section 6 develops our solution strategy, providing in-depth information on its structure and the different algorithms. Section 7 describes the 80 test instances developed for our study, which are built on existing datasets. Section 8 shows and discusses the results obtained by our solution procedure in these instances. Section 9 analyses the application of our solution strategy to the real-world problem under study. Finally, Sect. 10 concludes and suggests interesting avenues for future research.

2 Problem description

The steel industry is a pillar of the global economy. Steel is a fundamental component of many other prominent sectors, including construction, automotive, transportation, energy, and food. This is thanks to its excellent mechanical and structural properties along with a relatively low production cost. In addition, this material is environmentally friendly, as it can be reused almost infinitely and is 100% recyclable (Johnson et al., 2008). Furthermore, it is highly available due to iron being a very abundant metal; it makes up more than 5% of the Earth's crust and most of its core, with existing iron ore mines all over the world.

Steels can be classified according to their chemical composition and/or physical properties, with over 2000 steel grades existing nowadays (Thelning, 2013). Broadly speaking, steel products can be categorised into flat and long products. In the production of high-quality steel products, generally flat, some outputs do not satisfy customer requirements due to different motives. These products, which are referred to as non-prime, cannot be sold to their original customer, thus being unassigned to the prime order.

In this work, we consider the problem faced by a large steel company in their operations in Europe, which could be extrapolated to other enterprises in this industry. This company first offers non-prime products to affiliated companies and regular customers, often in the local regions of their plants. Unsold non-prime products go to a weekly online auction, where around 20 plants from all over Europe participate together with about 300 regular bidders. Approximately 500,000 tons of flat steel products are sold yearly through this mechanism. This corresponds to roughly 10,000 tons per week. This means that several thousands of non-prime products with very different weights are auctioned on a weekly basis. Products are only removed from the auction list to be reused in the manufacturing processes if they have received no offers after several weeks.

Due to the high volume of products, the auction is a fundamental part of the sales process. Indeed, it has a considerable impact on the financial performance of the organisation. Non-prime products are not auctioned individually, but they are grouped into bundles, which need to be carefully designed prior to the auction. All bundles are auctioned at the same time, with the auction being held in the form of a first-price sealed-bid auction with a reserve price. The bundles, with different capacities (which depend on the plant that offers the bundle), have to be as homogeneous as possible. This makes them more attractive to bidders, which increases the selling price, and hence the profits of the company. This emphasises the importance of the grouping problem. Also, large sets of non-prime products are often more attractive for bidders and small sets are generally expensive to deliver, so the number of bundles should be minimised for the sake of profitability. In this fashion, a minimum weight requirement is defined for each bundle.

Each non-prime product is defined by a set of parameters that refer to its characteristics and properties. Due to reasons of different nature, all the products in the same bundle need to have some parameters in common, which will be referred to as ‘global parameters’. These are: manufacturing plant, location, category, family, and coating sides. For the other relevant parameters, which are named ‘local parameters’, there may be some discrepancies within the same bundle, but these should be minimised to keep the bundles as homogeneous as possible. The local parameters are: subfamily, steel grade, oiling, weight, width, thickness, and coating thickness (on one or both sides). Not all customers prioritise the same parameters, which must be considered in the solution procedure.

A key aspect to mention is the limited time available to design and prepare the bundles for the auction. This involves all the necessary tasks from updating the list of non-prime products to uploading the bundles’ information to the online platform. In between, our solution strategy will need to be applied to the grouping problem, and the execution time should be reasonable. Therefore, the algorithms employed do not only need to be accurate but also time-efficient.

Finally, we summarise the main features and criteria of the grouping problem under consideration:

- Non-prime products have to be grouped into bundles, considering a set of global parameters that all the products in each bundle must have in common and a set of local parameters that determine the degree of homogeneity of the bundle, which needs to be maximised.

- The degree of homogeneity needs to be defined in a flexible manner as a function of the local parameters, given that the parameters that should be prioritised may vary across bundles.
- The number of bundles and the number of unassigned non-prime products should be minimised simultaneously. In doing so, the weight of each bundle needs to be as close as possible to its capacity, which depends on the plant where the non-prime products were manufactured. Also, the weight of each bundle must be compliant with a minimum requirement.
- Despite the computational complexity of the optimisation problem (up to around 5000 products per auction), the algorithm must be able to provide a good solution in a reasonable amount of time.

3 Theoretical background (I): the bin packing problem

The grouping problem that we formulated in the previous section may be encapsulated within the theoretical framework defined by the BPP. There are many variants of this well-known, industrially relevant problem in the operations management discipline. Coffman et al. (2013) categorised them into four main classes: (i) dual versions; (ii) variations on bin sizes; (iii) variations on item packing; and (iv) additional conditions.

First, dual versions address different objectives, such as the maximisation of the total number of items that are packed in a fixed number of bins (Azar et al., 2002) and the maximisation of the number of bins that are used for packing all the items with each bin weighing at least a predefined value (Coffman et al., 1987). Second, variations on bin sizes cover different models of variable-sized bins (e.g. Kang & Park, 2003; Liu et al., 2021) and problems with resource augmentation or bin stretching (e.g. Csirik & Woeginger, 1998; Dhabbi et al., 2021). Third, variations on item packing comprise a wide range of variants, such as the dynamic BPP (Coffman et al., 1983), selfish BPP (Bild, 2006), BPP with rejection (Dósa & He, 2006), and BPP with fragile objects (Chan et al., 2007). Fourth, additional conditions include item-size restrictions (Gutin et al., 2006), item types (Adler et al., 2002), constrained cardinality (Kellerer & Pferschy, 1999), constrained distances (Beaumont et al., 2010), partial orders (Wee & Magazine, 1982), conflicts (Epstein & Levin, 2008), and compatible categories (Santos et al., 2019), among others.

Finally, we note that some authors developed generalisations of the BPP. For instance, we refer to the multi-dimensional BPP, in which items need to be packed according to more than one dimension, such as width and height (e.g. Dahmani et al., 2015; Polyakovskiy & M'Hallah, 2021), and the generalised BPP, which is characterised by multiple items and bins attributes (e.g. Baldi & Bruglieri, 2017; Baldi et al., 2019).

None of these variants matches exactly the requirements and captures all the complexities of our case study. Rather, our grouping problem combines characteristics of different BPP variants. We highlight three:

- *Conflicts*. Products that have differences in the global parameters cannot be grouped together.
- *Constrained distances*. Homogeneity in the items of each bin (a bundle of non-prime steel products) is rewarded. The distance between items is measured through the differences in the local parameters.
- *Minimum weight requirements*. Each bin needs to be compliant with a minimum weight established.

Our solution strategy, which we present in Sect. 6, deals with conflicts by splitting the input data in accordance with the global parameters. Thereby, P sub-problems emerge. These are solved via algorithms that need to consider both the constrained distances and minimum weight requirements. Next, we briefly review the literature on algorithms for BPPs.

3.1 Algorithms for bin packing problems

Algorithms for BPPs can be broadly classified into online and offline (Coffman et al., 2013). The former assign items to bins as items appear. Hence, the bin for each item is selected without knowledge of the following items. Traditional online algorithms include: next-fit (Johnson, 1973), first-fit (Johnson, 1973), worst-fit (Johnson, 1974a), refined first-fit (Yao, 1980), best-fit (Falkenauer & Delchambre, 1992), and bounded-space (Csirik & Johnson, 2001). Nonetheless, this area is in continuous development, and many other online algorithms have been proposed to solve different variants of the BPP, such as the recent articles by Verma et al. (2020), Balogh et al. (2020), and Epstein and Mualem (2021).

In contrast, offline algorithms have full information about the list of items and use it to make the overall allocation. The most popular offline algorithms are those with presorting (i.e. reordering algorithms), such as the next-fit decreasing (Baker & Coffman, 1981), first-fit decreasing (FFD) (Baker, 1985), refined first-fit decreasing (Yao, 1980), and best two-fit (Friesen & Langston, 1991). Their time complexity is at least $O(n \cdot \log n)$. Other algorithms propose solutions faster, without presorting, such as the Group-X-Fit Grouped (Johnson, 1974b) and H_7 (Békési et al., 2000). More advanced methods have been recently applied to solve BPPs. For instance, Abdel-Basset et al. (2018) proposed a whale optimisation algorithm; Abdul-Minaam et al. (2020) developed an adaptive fitness-dependent optimiser with swarm intelligence; and Munien et al. (2020) implemented two hybrid metaheuristics (hybrid cuckoo search genetic algorithm and mutated firefly algorithms). These prior works focused on one-dimensional BPPs, while others developed metaheuristics for two-dimensional packing problems, such as Grandcolas and Pain-Barre (2021).

In between both extremes, semi-online algorithms do not have information about the complete list of items (like in offline ones) but have more information than pure online ones. In this sense, repacking items is sometimes allowed. This leads to different algorithms, such as the buffered next-fit (Galambos, 1985), which uses two open bins, and REP_3 (Galambos & Woeginger, 1993), with three open bins. In other cases, the algorithm is allowed to look ahead to later items, such as in the revised warehouse (Grove, 1995).

In the following subsection, we focus on algorithms that have been used for solving BPPs with conflicts, in which items in conflict cannot be assigned to the same bin, due to the similarities with our problem noted above.

3.2 Methodological approaches for bin packing problems with conflicts

The BPP with conflicts is an NP-hard optimisation problem that can be interpreted as a combination of the BPP and the vertex colouring problem (e.g. Diaby, 2010). Different methods have been used in the literature to approach this problem, which are briefly discussed below.

- *Asymptotic approximation scheme.* This refers to a family of algorithms where, for all $\varepsilon > 0$, there is an algorithm of performance ratio at most $1 + \varepsilon$, where the running time is polynomial in n (Epstein & Levin, 2020). Given a set of items $V = \{1, \dots, n\}$ and a conflict

graph $G = (V, E)$, Jansen (1999) proposed this scheme for a BPP with conflicts restricted to d -inductive graphs with constant d .

- *Modified FFD algorithm.* Gendreau et al. (2004) adapted the conventional FFD algorithm to consider the conflicts. Specifically, in this algorithm (H1), the items are assigned to the first bin in which there is enough capacity without incurring conflicts with the already assigned items.
- *Graph colouring algorithms.* Gendreau et al. (2004) developed three heuristics (H2, H3, H4) that make use of a conflict graph G . They colour the vertices of V through the DSatur heuristic (Brélaž, 1979). H2 creates sets of mutually non-conflicting items and applies the FFD algorithm to each set. H3 uses the same rationale after removing the less conflictive items, which are later assigned to bins with the modified FFD algorithm. H4 is based on the iterated use of the FFD algorithm.
- *Clique-based algorithms.* Gendreau et al. (2004) proposed two heuristics (H5, H6) based on cliques determined through Johnson's (1974a) greedy heuristic. H5 uses a clique of non-conflicting items, which are then assigned to bins. H6 uses a clique of conflicting items, which are considered for computing cliques of non-conflicting items. This inspired Maiza and Radjef (2011) in their MSS-based heuristic, which converts the BPP with conflicts to a set of sub-problems without conflicts.
- *Greedy-approximation algorithms.* Beaumont et al. (2010) showed that the BPP with constrained distances could be transformed into a BPP with conflicts. Their solution is based on the algorithm developed by Epstein and Levin (2008), using an FFD algorithm to fill the bins.
- *Adapted minimum bin slack (MBS) heuristic.* Maiza and Radjef (2011) extended the MBS algorithm developed by Gupta and Ho (1999) for BPPs to the case with conflicts. This heuristic executes the MBS procedure with a compatibility test of the current item with those already considered.
- *Branch-and-price algorithm.* Elhedhli et al. (2011) solved the BPP with conflicts by means of this algorithm. First, they employ a branching rule to match the conflicting constraints. Later, they create maximal clique valid inequalities according to these constraints. Similar approaches were used by other authors, such as Sadykov and Vanderberck (2013).
- *Iterated local search (ILS) metaheuristic.* To solve the BPP with conflicts, Capua et al. (2018) developed an ILS, with several classes of local and large neighbourhoods for solution improvement.
- *Sequential maximum degree packing heuristic.* Ekici (2021) proposed this algorithm for BPPs with conflicts and item fragmentation. The key idea of this approach is to start the allocation with the items with the highest number of conflicts, as these are the most problematic ones.

All these algorithms used for BPPs, and particularly BPPs with conflicts, are heuristics or metaheuristics that look for near-optimal solutions due to their complexity, as discussed by prior works (e.g. Asta et al., 2016; Fernandez et al., 2013; López-Camacho et al., 2013). In our case, the complexity may be even higher because of the interactions of conflicts with the other characteristics of the grouping problem of non-prime steel products (restricted distances, minimum weight) and the size of the problem (number of products, set of parameters, different weights, etc.). Also, the solution needs to be provided in a short period of time.

4 Theoretical background (II): clustering

In the light of previous discussions, cluster analysis (Duran & Odell, 1974; Farnè & Vouldis, 2021) emerges as an interesting approach for grouping similar non-prime products and separating those very different in an attempt to reduce computational requirements. This section briefly reviews the main set of techniques that are used for clustering, based on the categorisation by Xu and Wunsch (2005).

- *Hierarchical clustering* algorithms structure data in a hierarchical manner according to a proximity matrix (Kou & Lou, 2012). They may be divided into agglomerative algorithms, such as BIRCH and CURE, and divisive algorithms, such as MONA and DIANA (Kaufman & Rousseeuw, 2005).
- *Squared error-based clustering* assigns the objects into non-hierarchical clusters by using the sum of squared error criterion function. The most popular algorithm is the traditional k -means algorithm (Morissette & Chartier, 2013), but many other examples can be found; see e.g. Yu et al. (2018).
- *Combinatorial clustering* defines the problem via an objective function that aims to allocate the objects into clusters by optimising a criterion function (Kim et al., 2017). As this is computationally demanding, metaheuristics are generally used to achieve high-quality solutions (Levin, 2015).
- *Graph-based clustering* uses graph-theoretical concepts and properties to create hierarchical or non-hierarchical clusters. There are different clustering approaches based on graphs, including spectral clustering (Hendrickson & Leland, 1995), dynamic modelling (Karypis et al., 1999), and density peaks (Xu et al., 2021). We also highlight the work by Kawaji et al. (2004), who developed an algorithm for the clustering of a large set of proteins that finds distantly-related proteins. Vertices of the graph denote proteins, and edges denote their similarity. The graph is partitioned repetitively by removing edges with small weights (dissimilar proteins), achieving promising results.
- *Fuzzy clustering* algorithms may assign an object to several clusters simultaneously, with different degrees of membership (Sakawa, 2013), unlike the previous approaches. The most popular fuzzy clustering algorithm is the Fuzzy c -means algorithm; see Pantula et al. (2020).
- *Other clustering approaches* include mixture densities (Chacón, 2019), neural networks (Du, 2010), and kernel-based clustering (Piciarelli et al., 2013). Also, several interesting approaches have been recently proposed for clustering, such as self-organising features maps (Li et al., 2020a), adaptive hyper-spheres (Li et al., 2020b), and dual iterative local search (González-Almagro et al., 2020).

We conclude our review by highlighting that graph-based clustering facilitates the representation of the problem and allows for a rapid generation of clusters with similar characteristics. Therefore, it provides an interesting approach from which to address the grouping problem under consideration. In Sect. 6, we describe how clustering fits within our solution strategy, together with the other elements.

5 Mathematical formalisation of the problem

In line with the description of the problem in Sect. 2 and the discussion of relevant background in Sects. 3 and 4, we now formalise the grouping problem of non-prime steel products mathematically.

The restriction on the global parameters (i.e. they need to be equal for all products in a bundle) will be dealt with by dividing the initial set of products into P subsets, thus resulting in P sub-problems. Each of them can be interpreted as a BPP with restricted distances, which emerge as homogeneity in the local parameters is rewarded, and minimum weight requirements. In this sense, it is important to note that the goal of the problem is not only to minimise the bundles used, and the number of non-prime steel products unassigned, but also to make the bundles as homogeneous as possible. We call it the ‘homogeneous bin packing problem with minimum weight requirements’ (HBPPMWR).

5.1 Notation

In the mathematical formulation of the problem, we use the following indices:

- i refers to items (non-prime steel products), $i = 1, 2, \dots, n$, where n is the number of items,
- h refers to bins (bundles), $h = 1, 2, \dots, m$, where m is the number of bins;

the following decision variables:

- y_h is a binary variable, with $y_h = 1$ indicating that bin h is used in the proposed assignment, and $y_h = 0$ otherwise (note: y_h results in a row vector of m binary variables),
- x_{ih} is a binary variable, with $x_{ih} = 1$ indicating that item i is assigned to batch h , and $x_{ih} = 0$ otherwise (note: x_{ih} results in a matrix of $n \times m$ binary variables);

and the following parameters:

- w_i is a set (row vector) of n binary variables, with w_i denoting the weight of item i ,
- C is the capacity of the bins,
- w_{min} is the minimum allowed weight of the bins,
- d_{max} is the maximum allowed distance between any pair of items in a bin.

Finally, the variable u refers to the number of items that have not been allocated to any bin in a proposed assignment, and d_h refers to a variable that includes the maximum distance between items within batch h for the proposed assignment. Thus, u and d_h depend on the decision variables y_h and x_{ih} .

5.2 Optimisation problem

Mathematically, the optimisation problem for a specific set of products with the same global parameters can be expressed as follows:

$$\min J = z_y \sum_{h=1}^m y_h + z_d \sum_{h=1}^m d_h + z_u u \tag{1}$$

$$\text{s.t. } \sum_{h=1}^m x_{ih} \leq 1 \quad \forall i \tag{2}$$

$$\sum_{i=1}^n w_i x_{ih} \leq C y_h \quad \forall h \tag{3}$$

$$\sum_{i=1}^n w_i x_{ih} \geq w_{min} y_h \quad \forall h \tag{4}$$

$$d_h \leq d_{max} \quad \forall h \tag{5}$$

$$y_h \in \{0, 1\} \quad \forall h \tag{6}$$

$$x_{ih} \in \{0, 1\} \quad \forall i, h \tag{7}$$

Equation (1) defines the objective function. This is a cost function that attempts to minimise the number of bins used, the heterogeneity of these bins, and the number of unassigned items. In this equation, z_y , z_d , and z_u (such that $z_y + z_d + z_u = 1$) is the weight given to each criterion, depending on the prioritisation strategy. Constraint (2) ensures that each item is assigned to at most one bin. Constraint (3) makes sure that the total weight of each bin does not exceed its capacity. Constraint (4) ensures that the minimum weight is achieved by the sum of items in the bin. Constraint (5) ensures that the degree of homogeneity of all bundles fulfils the requirements. Last, constraints (6) and (7) ensure that the variables y_h and x_{ih} have binary values.

6 Solution procedure

Given the complexity of the problem under study, our solution strategy for the HBPPMWR aims to achieve near-optimal solutions in a time-efficient manner. Figure 1 provides an overview of the procedure, which is composed of three main stages:

- (1) *Initial division of the products according to the global parameters.*
- (2) *Cluster analysis of the products based on similarity in the local parameters.*
- (3) *Optimisation-based allocation of non-prime steel products to bundles.*

This three-stage solution procedure fits well with the nature of the industrial problem under consideration. It allows the users to easily fine-tune the model for each specific sub-problem

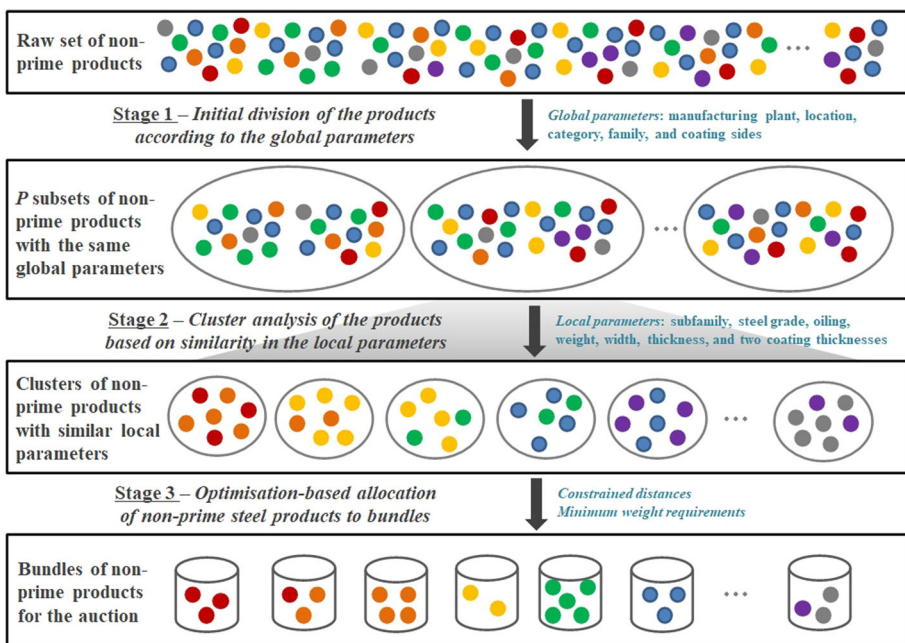


Fig. 1 Schematic representation of the solution strategy

($w_i, C, w_{min}, d_{max}, z_y, z_d$, and z_u differ in the various subsets due to the different resources and requirements of each plant and family of products, among others). This procedure also adds transparency to the allocation problem, facilitating that managers understand the generation of the final bundles of non-prime steel products. Each of the stages that characterise the proposed solution strategy is developed in depth below.

6.1 Initial division

After receiving the data with the non-prime products of the different plants, the overall dataset is divided into P subsets, such that all the products in each one have the same global parameters. This stage ensures that this fundamental restriction will be satisfied in all the bundles of the final solution. We thus deal with conflicts by dividing the overall problem into smaller ones without conflicts, which is in line with some of the approaches described in Sect. 3. This stage also reduces considerably the computational complexity faced by the clustering and optimisation algorithms in the following phases of the solution procedure.

6.2 Cluster analysis

In each of the P subsets with the same global parameters, clustering is applied to guide the optimisation algorithm towards more homogeneous and time-efficient solutions. Clustering is performed according to the local parameters, whose distance should be minimised in each bundle. Following the discussion in Sect. 4, we adopt a graph-based approach. Each item is represented by a node, which is linked to the other items in the same subset by means of undirected edges characterised by their distances d_{ij} (where i and j represent the nodes under consideration). Next, we describe how such distances, which measure the similarity between products, are computed.

6.2.1 Definition of distance

We consider an eight-dimensional space, where the dimensions refer to the eight local parameters of our homogeneous grouping problem (i.e. subfamily, steel grade, oiling, weight, width, thickness, and two coating thicknesses).

Then, we use a weighted Euclidean distance to measure the difference between two items. On the one hand, we selected the Euclidean distance, instead of other alternatives, as it is a common and recommended practice for measuring the distance between items when parameters of different nature are involved, as in our case (Kou et al., 2014; Xu & Wunsch, 2005). On the other hand, weighing the differences in the various parameters allows us to define the degree of homogeneity in a flexible manner, which is a requirement of our real-world problem, as discussed before. To define the importance of homogeneity in each parameter, we use weights, γ_t , with $t = \{1, 2, \dots, 8\}$. These weights can be configured between 0.1 and 10; therefore, they cover differences of importance of up to two orders of magnitude. These weights have been rescaled through geometric means, thus resulting in the normalised weights $\gamma'_t = \gamma_t / \left(\sqrt[8]{\prod_{q=1}^8 \gamma_q} \right)$.

Finally, we note that we have normalised the five numerical parameters, namely, weight, width, thickness, and the two coating thicknesses. Also, we have transformed the three categorical parameters, that is, subfamily, steel grade, and oiling, into a numerical (and

normalised) format. We explain how we have normalised and transformed the different parameters in the following subsection.

Taking all the above into consideration, the distance between items i and j , d_{ij} , can be obtained via

$$d_{ij} = \sqrt{\sum_{t=1}^8 \gamma'_t (\varphi'_{t,i} - \varphi'_{t,j})^2},$$

where $\varphi'_{t,i}$ and $\varphi'_{t,j}$ refer to the normalised values of parameter t for items i and j , respectively.

6.2.2 Normalisation and transformation

Normalising the numerical parameters is essential to ensure the robustness of the distance measurements. In this sense, we have rescaled the values of the parameters into a range of $[0,1]$. Specifically, we have used

$$\varphi'_{t,i} = \frac{\varphi_{t,i} - \varphi_{t,\min}}{\varphi_{t,\max} - \varphi_{t,\min}},$$

where $\varphi_{t,i}$ denotes the actual value of parameter t for item i , and $\varphi_{t,\max}$ and $\varphi_{t,\min}$ denote the maximum and minimum values, respectively, of parameter t in the subset under consideration.

For the categorical parameters, it was necessary to transform their distances into a numerical format, as discussed before. To this end, we have assigned the distance values, $\varphi'_{t,i} - \varphi'_{t,j}$, ranging between 0 and 1 based on the evaluations of experts in the different processes. They considered the degree of similarities between each pair of categorical levels for each parameter.

By way of example, we focus on the parameter ‘subfamily’. In this case, the experts agreed to define three levels of distances: 0 (no distance), 0.1 (low), and 1 (high). For the organic coating (OC) product family, there are eight subfamilies, characterised by the following labels: OCR, OCH, OAS, OAZ, OZ, OZA, OZE, and OZO. The ideal is to bundle together products of the same subfamily ($\varphi'_{t,i} - \varphi'_{t,j} = 0$). Nonetheless, products of subfamilies OCR and OAH can also be included in the same bundle at a low cost ($\varphi'_{t,i} - \varphi'_{t,j} = 0.1$). The same happens with products of subfamilies OAS, OAZ, OZ, OZA, OZE, and OZO ($\varphi'_{t,i} - \varphi'_{t,j} = 0.1$). However, combining products of both groups in the same bundle (e.g. OCR and OAS, or OAH and OZ) is not desirable, as this would reduce the attractiveness of bundles to potential customers ($\varphi'_{t,i} - \varphi'_{t,j} = 1$). This information is summarised in Table 1.

Although we do not include all tables here for the sake of brevity, we clarify that the same methodological approach has been followed for the rest of the product families, as well as for the other two categorical parameters. In the case of the steel grade, we have also used three levels of distances with the aim of promoting the bundling of products with the same ($\varphi'_{t,i} - \varphi'_{t,j} = 0$) or similar steel grades ($\varphi'_{t,i} - \varphi'_{t,j} = 0.1$) rather than those with highly different steel grades ($\varphi'_{t,i} - \varphi'_{t,j} = 1$). In contrast, in the case of oiling, we only use two values: $\varphi'_{t,i} - \varphi'_{t,j} = 1$ if one product went through oiling but the other did not; and $\varphi'_{t,i} - \varphi'_{t,j} = 0$ if both or none of them went through this oxidation prevention process.

6.2.3 Clustering algorithm

In line with the discussion in Sect. 4, we follow a graph-based approach to generate clusters (groups) of non-prime steel products with low distances among them. To this end, we gathered

Table 1 Distances assigned to the parameter ‘subfamily’ for products that belong to the OC family

	OCR	OCH	OAS	OAZ	OZ	OZA	OZE	OZO
OCR	0	0.1	1	1	1	1	1	1
OCH	0.1	0	1	1	1	1	1	1
OAS	1	1	0	0.1	0.1	0.1	0.1	0.1
OAZ	1	1	0.1	0	0.1	0.1	0.1	0.1
OZ	1	1	0.1	0.1	0	0.1	0.1	0.1
OZA	1	1	0.1	0.1	0.1	0	0.1	0.1
OZE	1	1	0.1	0.1	0.1	0.1	0	0.1
OZO	1	1	0.1	0.1	0.1	0.1	0.1	0

ideas from Hendrickson and Leland (1995), who use spectral graphs, Zhou et al. (2016), who implement a weighted summation, and Kawaji et al. (2004), who repeatedly partition the graph by removing edges with low similarities, among others.

The clustering algorithm, which operates in each of the P subsets by means of a specific function (SPLIT), follows the sequence of events described in Fig. 2. In this sense, the operation of the clustering algorithm is based on four steps that are executed as follows:

1. An initial undirected graph is generated, which represents products in the form of nodes. These are linked by edges, characterised by their distance d_{ij} (calculated as explained before). The undirected graph is created such that the distances are lower than a predefined threshold, ϑ , i.e. $d_{ij} \leq \vartheta \forall i, j$.
2. The initial graph is divided into several subgraphs by considering the connected components. In this sense, we generate initial clusters that are formed by relatively similar non-prime steel products.

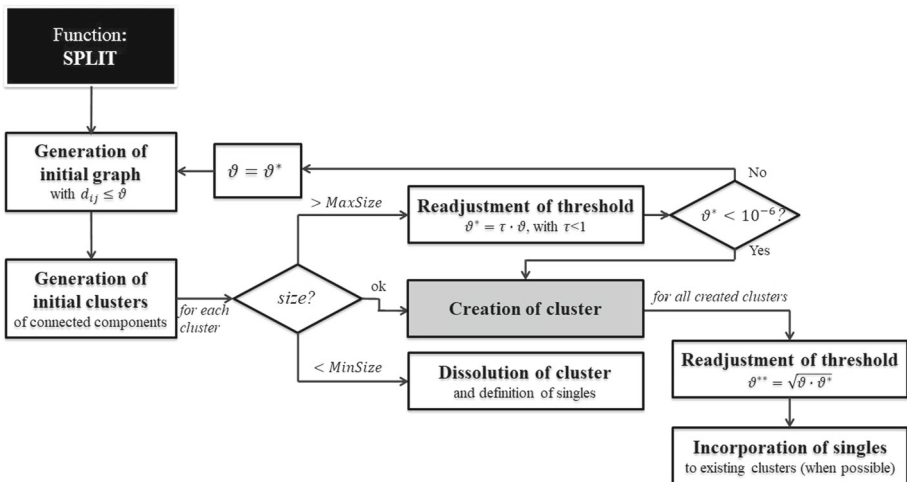


Fig. 2 Schematic representation of the operations of the clustering algorithm

3. The number of items in each initial cluster is compared to a predefined range, $[MinSize, MaxSize]$.
 - a. If the number of items in the initial cluster is within the limits, we create a cluster.
 - b. If it is lower than desired (i.e. $< MinSize$), the initial cluster is dissolved and its products are defined as ‘singles’.
 - c. If it is higher than desired (i.e. $> MaxSize$), the threshold is readjusted as follows: $\vartheta^* = \tau \cdot \vartheta$, where $\tau < 1$ is the step parameter. Then, the sequence starts again for the products in this large cluster. As the new threshold is more restrictive ($\tau < 1 \Rightarrow \vartheta^* < \vartheta$), this cluster will tend to generate several initial clusters of a smaller size. If necessary, this occurs recursively until the threshold becomes lower than 10^{-6} (when this occurs, the cluster is formed).
4. Singles are regrouped into the clusters created, when this is possible. To this end, a new threshold is defined with the geometric mean of the last two thresholds, i.e. $\vartheta^{**} = \sqrt{\vartheta \cdot \vartheta^*}$. Then, we evaluate if ϑ^{**} ($\vartheta^* < \vartheta^{**} < \vartheta$) allows for the incorporation of any of the singles to the new cluster created.

6.3 Allocation of products to bundles

For each cluster, the products need to be allocated to a set of bundles, taking into consideration both their minimum allowed weight and their capacity, which are introduced as inputs, and making sure that they are as homogeneous as possible. In order to provide the best possible solution for the online auction, the algorithm creates many assignments using different strategies, which are described in the following subsection, and finally selects the best one according to a user-defined fitness function (based on Eq. 1).

6.3.1 Algorithm structure

The algorithm, which has to be executed several times before each auction, needs to provide a high-quality allocation of the non-prime steel products in a short amount of time. In line with the description in Sect. 5, it will aim to minimise the number of bundles employed, the differences in the items of each bundle, and the number of unassigned items, according to the weights assigned by the user.

The structure of the algorithm, which has been implemented in a specific function (CREATE_BUNDLES), is summarised in Fig. 3. There are three main phases:

1. *Item sorting*. The products in each cluster first need to be sorted according to a predefined criterion. We consider three methods: random sorting (used $L-2$ times per call to the function, where L is a decision parameter that considers the replications of each heuristic method), largest-first sorting (1 time per call), and smallest-first sorting (1 time per call).
2. *Item allocation*. Now the products are allocated to bundles. To do this, we implement three heuristics: FIFO, greedy, and distance-based. The traditional models have been adapted to accommodate the homogeneity requirements, as we will discuss in the next subsection.
3. *Fitness evaluation*. Once the L solutions of each algorithm have been generated, all the allocations ($3L$) are analysed, and the one that provides the lowest cost, according to Eq. (1), is selected. This allocation is proposed to be used in the online auction.

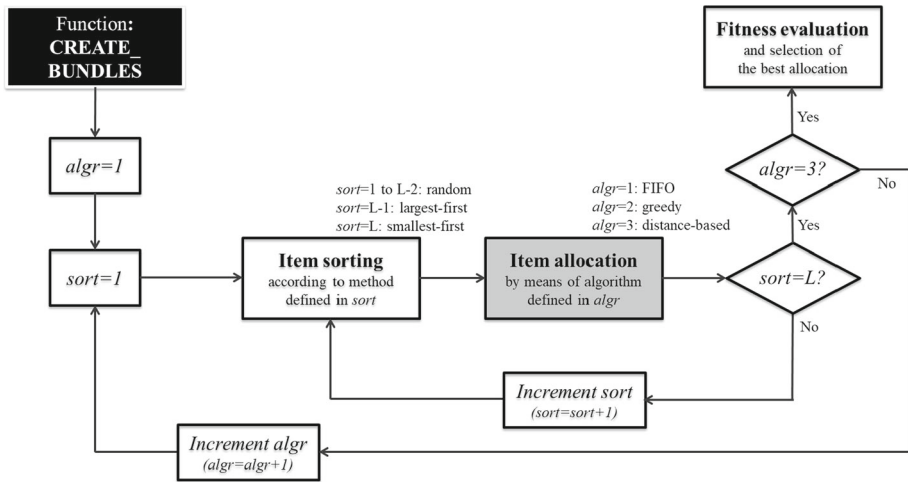


Fig. 3 Schematic representation of the operations of the algorithm for allocating the non-prime products to bundles

6.3.2 Heuristic techniques

The first heuristic algorithm adopts a FIFO (first-in-first-out) approach. It assigns items to bundles by going over the list of items in sequential order. If the following item in the list verifies the maximum distance requirement and there is enough capacity in the bundle, it is assigned to the open bundle. If there is enough capacity but the item does not verify the homogeneity requirement, then the next item is checked. As soon as an item cannot be introduced in a bundle due to capacity restrictions, the bundle is closed and a new one will be opened. Therefore, the time complexity of this algorithm is $O(n)$.

The second heuristic is a greedy algorithm. In this case, if a product cannot be assigned to a bundle because of capacity limitations, the bundle is not closed. Rather, the following products may be assigned to this bundle (if this was possible considering the maximum distance and capacity). Logically, this requires more time. Thus, we have implemented additional checks to avoid consuming unnecessary time. For example, if the smallest-first sorting method is used, and an item cannot be included in a bundle for capacity reasons, the rest of the items will not be considered for that bundle. Similarly, for the largest-first sorting method, when an item cannot be introduced into a bundle due to capacity limitations, the algorithm evaluates if the last (i.e. the smallest) item could be introduced. If not, there is no need to consider the remaining items for the bundle. The time complexity of this algorithm is $O(n^2)$.

Last, the distance-based algorithm makes decisions based on the similarity between the items in the open bundles and the rest of the items. This heuristic works as follows. The first item goes to the first bundle. Then, the closest item to the first one (lowest d_{ij}) is also introduced in this bundle. Subsequently, the item that is closest to those two items is added (specifically, we consider the maximum d_{ij} to those items in the bundle). The process is repeated as long as there is enough capacity. If, at some point, the closest item cannot be introduced for capacity reasons, the second closest item is evaluated, and so on. Finally, once no more items can be accepted (due to capacity and/or maximum distance restrictions), the bundle is closed. Then, the process starts again for the next bundle. The time complexity of the distance-based algorithm is $O(n^3)$.

7 Design of experiments

In this paper, we first evaluate our solution strategy for the HBPPMWR through a new set of test instances. In this section, we describe the procedure followed to generate these instances based on existing datasets.

7.1 Existing sets of test instances

ESICUP, the EURO's Working Group on Cutting and Packing, collects on their website¹ several test instances from different works (e.g. Falkenauer, 1996; Scholl et al., 1997; Schwerin & Wäscher, 1997) that have been widely used in the BPP literature. As discussed by Bai et al. (2012), those test instances generated by Falkenauer (1996) are probably the most commonly employed dataset in the BPP field.

Authors dealing with BPPs with conflicts have also used these instances, adapting them to the new context. Gendreau et al. (2004) selected the first 10 Falkenauer's (1996) uniform instances for different numbers of products, $n = \{100, 250, 500, 1000\}$. These instances consider items with discrete weights that are uniformly distributed within the range [20,100], and the bin capacity is 150. The authors also used the first 10 triplet instances developed by Falkenauer (1996) for $n = \{60, 90, 249, 501\}$, and multiplied the weights by 10 to obtain integer numbers. In this case, the bin capacity was set equal to 1000. They added 10 graphs of random conflicts, characterised by density values within 0 and 0.9, which resulted in 800 test instances.

Muritiba et al. (2010) followed the same procedure as in Gendreau et al. (2004) and generated 800 new test instances,² which were also used by Yuan et al. (2014) and Maiza et al. (2016), among others. Sadykov and Vanderberck (2013) also followed Gendreau et al.'s (2004) procedure to generate a new set of instances for the BPP with conflicts.

7.2 Generating test instances for our problem

To create the test instances for the HBPPMWR, we also start from Falkenauer's (1996) dataset. Specifically, we consider their 80 uniform instances, that is, 20 instances for each n , with $n = \{100, 250, 500, 1000\}$. In our problem, we also need to provide the other local parameters (in addition to the weight, which is used in the original dataset) with values for the different instances. To this end, we have proceeded as follows. We note that the test instances have been developed for the hot-dip galvanised steels (HD) family, which is very representative of the problem under consideration. Also, we highlight that the probabilities and data provided below in brackets for the various local parameters are based on actual information provided by the steel company studied.

- *Subfamily*. We consider two subfamilies within the HD family labelled as Z (90%) and ZM (10%).
- *Steel grade*. We consider the most common (12) grades in the HD family: DX51 (24%), DX52 (1%), DX53 (16%), DX54 (19%), DX56 (6%), HCT590X (4%), HX700LAD (2%), S220GD (12%), S350GD (5%), S390GD (1%), S450GD (4%), and S550GD (6%).
- *Oiling*. We establish a difference between oiled (80%) and unoiled (20%) steel products.

¹ Available at <https://www.euro-online.org/websites/esicup/data-sets/> (accessed on 14/10/2021).

² Available at <http://or.dei.unibo.it/library/bin-packing-problem-conflicts> (accessed on 14/10/2021).

- *Weight*. As in Falkenauer's (1996) dataset, we employ a uniform distribution between 20 and 100.
- *Width*. According to a preliminary analysis (Kolmogorov–Smirnov goodness-of-fit P -value: 0.105), we use a generalised Pareto distribution with $k = -1.4106$, $\sigma = 1, 651.9$, and $\mu = 838.65$.
- *Thickness*. According to a preliminary analysis (Kolmogorov–Smirnov goodness-of-fit P -value: 0.187), we use a Kumaraswamy distribution with $\alpha_1 = 0.44024$ and $\alpha_2 = 1.5529$, adjusted to the interval [0.56,6].
- *Coating thicknesses*. According to a preliminary analysis (Kolmogorov–Smirnov goodness-of-fit P -value: 0.162 for side 1, and 0.175 for side 2), we use Dagum distributions with the following parameters: $k = 1.2080$, $\alpha = 6.8828$, $\beta = 90.655$, and $\gamma = -31.139$ for side 1; and $k = 1, 1793$, $\alpha = 7.2491$, $\beta = 94.253$, and $\gamma = -33.785$ for side 2.

The set of 80 instances is available upon request. In the following numerical study, we will assume that all the instances have the same values for the global parameters.

8 Numerical results

We now apply the solution strategy proposed in Sect. 6 to the 80 test instances generated as described in Sect. 7. In the tests reported here, we use the following configuration of the parameters of the system:

- *Prioritisation strategy*. We give the same importance to the three criteria, $z_y = z_d = z_u = 1/3$.
- *Capacity*. We employ $C = 150$, as in the original Falkenauer's (1996) dataset.
- *Minimum weight requirement*. We define $w_{min} = 2C/3 = 100$.
- *Maximum distance allowed*. We employ $d_{max} = 1$.
- *Weight of local parameters*. We use $\gamma_t = 1\forall t$.

In relation to the configuration of the clustering algorithm, the interval of products allowed for the creation of clusters is defined by $MinSize = 5$ and $MaxSize = 100$. In addition, the initial threshold is defined by $\vartheta = d_{max} = 1$, and the threshold adjustment (step parameter) is set to $\tau = 0.8$. Nonetheless, we note that we do not focus on the cluster analysis in this section; rather, we compare the performance of the algorithms.

Regarding the configuration of the algorithm for allocating non-prime products to bundles, we use $L = 200$ replications. This is based on a preliminary analysis, in which this value has proven to provide a good trade-off between the quality of the allocation and the computation time required.

We also clarify that, while the function CREATE_BUNDLES selects the most appropriate assignment (that with the lowest J) in its final phase, it stores the best solution and the computation time required by each heuristic. This facilitates the comparison of the effectiveness and the efficiency of the different algorithms. Moreover, due to the randomness in the performance of our solution strategy, we have carried out 10 runs with each of the 80 test instances. In this sense, the results we report are the average of these 10 runs.

In line with our objective function, we analyse the quality of the solutions by looking at the number of bundles used, the unassigned non-prime steel products, and the heterogeneity of the bundles (i.e. measured as the maximum distance between the items of a bundle). We also evaluate the mean computation time.

Table 2 Distribution of the best heuristic algorithm for different numbers of products

Algorithm	Number of products (n)				Overall
	120	250	500	1000	
FIFO	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Greedy	13 (65%)	19 (95%)	13 (65%)	5 (25%)	50 (62.5%)
Distance-based	7 (35%)	1 (5%)	7 (35%)	15 (75%)	30 (37.5%)

Table 3 Mean computation time (in seconds) of each heuristic algorithm for different numbers of products

Algorithm	Number of products (n)				Overall
	120	250	500	1000	
FIFO	1.15	1.16	3.65	13.34	4.83
Greedy	1.32	2.21	6.07	20.06	7.42
Distance-based	1.92	4.07	10.14	30.45	11.64

8.1 Overall performance vs computation time

First, we consider the cost function J . Table 2 shows the heuristic algorithm that provided (on average) the lowest value of J in the 80 test instances (20 instances for each n , with $n = \{100, 250, 500, 1000\}$). Note that the FIFO algorithm did not provide the best solution for any of the 80 instances. The greedy algorithm achieved the best result in 50 instances, a 62.5%, while the distance-based algorithm provided the best result in the remaining 30 instances, a 37.5%. From this perspective, we may conclude that the greedy algorithm generally outperforms its competitors. Nonetheless, it can be highlighted that the distance-based algorithm offers better performance for large datasets (namely, for $n = 1000$).

Table 3 reports the mean time (in seconds) spent by the optimisation algorithm in the different tests. As expected, we can observe that the FIFO algorithm is the fastest one. Interestingly, the greedy heuristics requires less time than the distance-based algorithm. This order applies to the four scenarios defined by different numbers of products. Note that the time difference between the algorithms grows as n increases. By combining the insights from Tables 2 and 3, we can conclude that the additional time that the distance-based algorithm requires over the greedy algorithm may only be justified for high numbers of products.

8.2 Number of bundles and unassigned items

To better understand the different performances of the three algorithms, we now look at two components of the cost function that are highly interrelated, i.e. the number of bundles used in the allocation and the number of unassigned items.

Figures 4 and 5 show the mean value of these metrics and the sum of both for the 20 test instances with $n = 250$ and $n = 1000$, respectively. Detailed inspection of these graphs reveals that the distance-based algorithm is the heuristic that generally uses the lowest number of bundles (see top-left bar diagrams). However, in most cases, the greedy algorithm is able



Fig. 4 Comparison of bundles generated (top left), unassigned items (top right), and sum (bottom) for 250 products



Fig. 5 Comparison of bundles generated (top left), unassigned items (top right), and sum (bottom) for 1000 products

to provide the lowest number of unassigned items (see top-right diagrams). When these two perspectives are considered simultaneously, both algorithms provide a relatively similar performance (see bottom line charts). Nonetheless, the sum of bundles used and unassigned items is more often lower for the greedy heuristic than for the distance-based heuristic. Looking at the FIFO algorithm, we observe that it is clearly outperformed by its competitors.

Figures 6 and 7, in Appendix, represent the same information for $n = 120$ and $n = 500$, respectively. Their analysis leads to the same general conclusion: the distance-based algorithm generates allocations with fewer bundles, but the greedy algorithm is able to assign a higher number of products to the bundles. In this way, we highlight that the most suitable algorithm for a specific company would depend on the prioritisation strategy (the distance-based algorithm is more appropriate when minimising bundles is more important; if minimising the unassigned items was the priority, the greedy algorithm would be preferable).

All in all, in 71 out of the 80 instances, a 88.75%, the distance-based algorithm generated the lowest number of bundles. However, this occurs at the expense of leaving a higher number of products unassigned, given that in 69 instances, a 86.25%, it was the greedy algorithm that allocated the most items into the bundles. Considering the sum of both, the greedy algorithm provided the best results in 48 instances, a 60%.

8.3 Homogeneity in the bundles

Finally, we complete the picture by analysing the third component of the cost function. This refers to the homogeneity in the bundles, which is measured by the distance in the local parameters of the non-prime steel products that are included in the same bundle (low distances results in high homogeneity).

Tables 7, 8, 9, 10, in Appendix, provide information about the algorithm that generates the most homogeneous solution in the 80 test instances. In 59 instances, a 73.75%, the greedy algorithm outperformed its competitors, while in the remaining 21 instances, a 26.25%, the most homogeneous result was offered by the distance-based algorithm. Nevertheless, it is interesting to note that the number of instances in which the greedy algorithm outperforms the distance-based algorithm decreases as n grows. Specifically, for $n = \{100, 250, 500, 1000\}$, the greedy algorithm was the one that provided the most homogeneous solution in 20, 19, 15, and 5 instances, respectively. That is, the distance-based algorithm tends to produce more homogeneous allocations than the greedy algorithm for high values of n .

In this sense, we conclude that the fact that the distance-based algorithm is the most appropriate option for large datasets can be explained from the perspective of the homogeneity of the solutions that it generates (rather than by the sum of the number of bundles and unassigned items). This can also be observed from the analysis of Table 10, which reports the best algorithm from the viewpoint of the different criteria.

9 Real-world application

We now apply the solution strategy developed to the specific problem under study. From this perspective, we address its usefulness in real-world environments, providing a complementary lens to the previous analysis. We use a dataset of non-prime products provided by the organisation for one of their online auctions. This contains all the necessary information about the global and local parameters for 2771 steel products. It should also be noted that

in this case the capacity of the different bundles is $C = 25$ tons, and the minimum weight accepted per bundle is $w_{min} = 3C/5 = 15$ tons.

To preserve confidentiality, we use the same values for the weights of the local parameters as in the previous experiments (i.e. $\gamma_t = 1\forall t$) as well the same weights for the different criteria ($z_y = z_d = z_u = 1/3$), but we note that in the real-world use of the system these parameters need to be dynamically adjusted by the experts in agreement with the management team. Regarding the maximum distance allowed, we use two different levels, $d_{max} = 1$ and $d_{max} = 1.5$, to better understand the impact of this controllable parameter.

Moreover, we configure the clustering and allocation algorithms in a similar manner as the previous study of the instances. That is, the parameters and conditions remain unchanged; specifically: $MinSize = 5$; $MaxSize = 100$, $\vartheta = d_{max}$, $\tau = 0.8$, $L = 200$, and number of runs per algorithm = 10.

Following our solution strategy, the first step requires splitting the steel products according to the global parameters. Then, we have run the clustering algorithm, which provides the results that are shown in Table 4. The algorithm generated 132 clusters with at least 5 items, which together contain 2256 products (out of the 2771 products). The average number of non-prime products per cluster then is 17.1. Also, note that, despite the clusters being allowed to have until 100 items, the largest cluster has 48 items. Indeed, there are only three clusters with more than 40 items, while 38 clusters have less than 10 items.

The allocation algorithm is then applied to assign the products to the final bundles. The (mean) results provided by each heuristic technique in the three main criteria (i.e. number of bundles, unassigned items, and sum of distances in bundles) are displayed in Table 5 for both levels of the maximum distance.

First, we compare the results of the algorithms with $d_{max} = 1$ and $d_{max} = 1.5$. Table 5 provides evidence that when the homogeneity requirements are more demanding (i.e. d_{max} is reduced), the solution procedure provides more uniform bundles (i.e. the mean distance decreases) and the number of bundles decreases. However, these improvements come at the expense of a considerable increase in the number of unassigned items. Note that this holds for the three algorithms. Specifically, when d_{max} decreases from 1.5 to 1, the mean distance decreases more than 0.2 in the three algorithms and the number of bundles used reduces by more than 170 bundles in all cases, but the number of unassigned products increases by at least 700 items.

Table 4 Distribution of non-prime products in clusters

No. of items	No. of clusters	No. of items	No. of clusters	No. of items	No. of clusters	No. of items	No. of clusters
5	12	14	4	23	2	33	2
6	7	15	2	24	6	35	1
8	11	16	4	25	4	38	1
9	8	17	1	26	3	39	1
10	7	19	2	27	3	40	2
11	8	20	6	28	4	42	1
12	5	21	8	30	1	45	1
13	3	22	9	31	2	48	1

Table 5 Performance of the algorithms in the three criteria defined

Algorithm	Max. distance	Criterion		
		No. of bundles	Unassigned items	Mean distance in the bundles
FIFO	$d_{max} = 1$	251	1816	0.59
	$d_{max} = 1.5$	471	1063	0.87
Greedy	$d_{max} = 1$	266	1766	0.60
	$d_{max} = 1.5$	463	1055	0.87
Distance-based	$d_{max} = 1$	265	1652	0.51
	$d_{max} = 1.5$	438	916	0.76

Second, we analyse the performance of the three optimisation algorithms. To this end, we focus on the case with $d_{max} = 1.5$. Table 5 shows that the distance-based algorithm is able to generate an allocation that simultaneously utilises a lower number of bundles (438 versus 463 and 471), it leaves fewer unassigned items (916 versus 1055 and 1063), and it produces more homogeneous bundles (0.76 versus 0.84 and 0.87). The same general findings hold for $d_{max} = 1$, with an interesting exception: in this case, the FIFO algorithm proposes an allocation with fewer bundles than its competitors (251 versus 265 and 266), although this is partially because the number of unassigned items is higher (1816 versus 1652 and 1766). All in all, we conclude that the superiority of the distance-based algorithm is in line with the findings of the previous section, where we observed that, while the greedy algorithm generally provided better results, the distance-based algorithm often emerges as the most appropriate alternative for a high number of products.

At this point, we note that the average total execution time of our solution procedure, including the clustering and the allocation algorithms, has been 244 s. This is a reasonable amount of time, which fits the requirements of the problem under study, and would allow the users to test different configurations of the parameters for each auction. In this sense, the procedure is not only effective but also efficient.

To provide further insights on the behaviour of the algorithms, we finally consider three scenarios, in addition to the base one in which the same weight is given to the three criteria (*scenario 0*, $z_y = z_d = z_u = 1/3$). *Scenario I* assumes that the minimisation of the number of bundles is prioritised ($z_y = 0.6$, $z_d = z_u = 0.2$). *Scenario II* considers that minimising the unassigned items is the priority of the company ($z_u = 0.6$, $z_d = z_y = 0.2$). *Scenario III* models the case in which the homogeneity in the bundles is the most important criterion ($z_d = 0.6$, $z_y = z_u = 0.2$).

Table 6 provides information on the cost function J in the four scenarios (*0*, *I*, *II*, and *III*) in relative terms to the minimum value of J for each value of d_{max} . We can see that the distance-based heuristic offers the best allocation in the four scenarios. This can be easily understood given that, as we discussed before, when the number of items is high, this algorithm tends to generate allocations not only more homogeneous but also with fewer bundles and a lower number of unassigned items. In addition, we observe that the greedy algorithm outperforms the FIFO heuristic in seven out of the eight cases, although both entail a significantly higher cost than the distance-based heuristic. In this regard, it is also interesting to note that the relative difference between this heuristic and its competitors increases as d_{max} grows.

Table 6 Ratio of the metric J of the algorithms to the minimum J in the four scenarios and for both maximum distances

Max. distance	Algorithm	Scenario			
		Scenario 0: equal weights	Scenario I: prioritising the minimisation of bundles	Scenario II: prioritising the allocation of products	Scenario III: prioritising the homogeneity in the bundles
$d_{max} = 1$	FIFO	1.080	1.053	1.092	1.083
	Greedy	1.068	1.055	1.069	1.082
	Distance-based	1	1	1	1
$d_{max} = 1.5$	FIFO	1.152	1.126	1.157	1.174
	Greedy	1.140	1.112	1.146	1.161
	Distance-based	1	1	1	1

10 Concluding remarks

This work has studied the grouping of non-prime products into homogeneous bundles that are later auctioned, a problem that significantly affects the economic performance of many steel producers. The objective is to simultaneously minimise the number of bundles used, the number of unassigned items, and the differences (in a set of parameters) of the products that are included in each bundle. In addition, the allocation problem needs to be solved with a moderate computational effort due to time limitations. We have modelled the problem mathematically as a variant within the family of BPPs that is characterised by the interaction of conflicts, constrained distances, and minimum weight requirements.

To solve the grouping problem of non-prime steel products, we have developed a three-stage solution procedure that employs clustering techniques and optimisation algorithms. Specifically, we have implemented three heuristics; namely, a FIFO, a greedy, and a distance-based algorithm. The value of our solution strategy has been demonstrated both with a sample of test instances and with data from the real-world problem under consideration. It is capable of providing an effective allocation of products to bundles in a reasonable amount of time. We have also observed that in general terms the greedy algorithm outperforms its competitors; however, when the number of items is very high, the distance-based algorithm generally provides better performance. In such cases, this heuristic is able to generate fewer and more homogeneous bundles with fewer unassigned items.

Interesting avenues for research emerge from this work. Studying the effects of the weight of the local parameters and/or the criteria in the objective function would help managers to configure their decision support systems more precisely. We may also include other optimisation algorithms to increase the effectiveness of our solution tool. Nonetheless, this may increase the computation time considerably. Therefore, due to the limited time available, this would motivate us to look for ways to improve the efficiency of our solution procedure. This can be done by delving into the interplays between the sorting methods and the allocation algorithms. In this sense, we also plan to get inspiration from recent developments in different streams of the operational research literature, yet adjacent to the BPP literature, including cluster analysis (e.g. Xu et al., 2021), conflict management (e.g. Ficker et al., 2021), multi-objective optimisation (e.g. Denstad et al., 2021), and multi-criteria decision making (e.g. Kou et al., 2020).

Acknowledgements Borja Ena, Alberto Gómez, Borja Ponte, and Diego Díaz would like to express their sincerest gratitude and recognition to their co-author, Prof. Paolo Priore, who passed away in March 2022. They also express this gratitude and recognition on behalf of his colleagues, who also would like to acknowledge his contributions to their lives and careers. The authors also greatly appreciate the data provided by the participating organisation. Finally, the authors would like to thank the anonymous referees for their detailed and constructive feedback that led to several improvements of this paper.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix

See Figs. 6, 7 and Tables 7, 8, 9, 10.



Fig. 6 Comparison of bundles generated (top left), unassigned items (top right), and sum (bottom) for 120 products

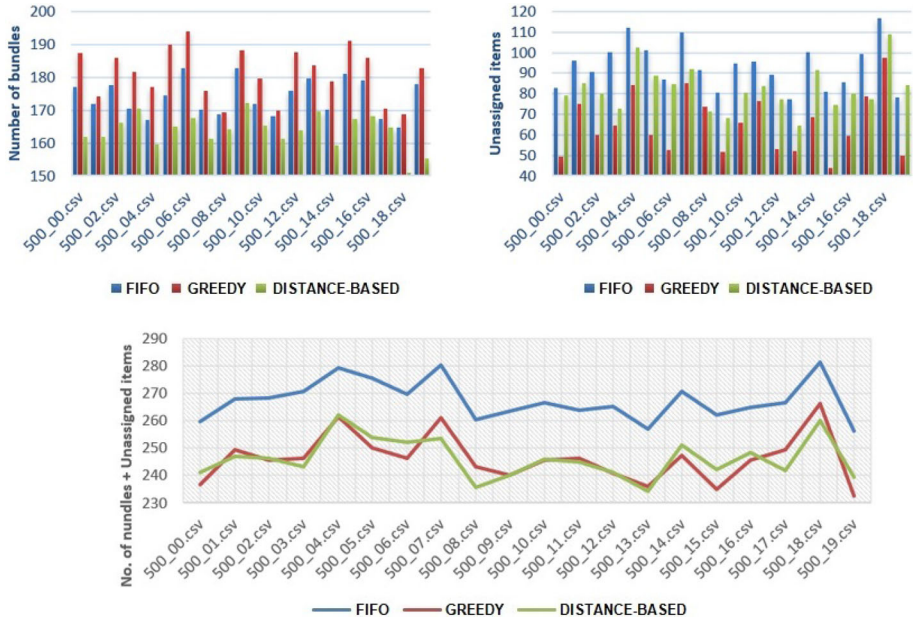


Fig. 7 Comparison of bundles generated (top left), unassigned items (top right), and sum (bottom) for 500 products

Table 7 Algorithm that provided the best allocation for different criteria when $n = 120$

Instance	Criterion		
	No. of bundles + unassigned items	Homogeneity	Cost function J
<i>120_00.csv</i>	Distance-based	Greedy	Greedy
<i>120_01.csv</i>	Distance-based	Greedy	Distance-based
<i>120_02.csv</i>	Distance-based	Greedy	Distance-based
<i>120_03.csv</i>	Greedy	Greedy	Greedy
<i>120_04.csv</i>	Greedy	Greedy	Greedy
<i>120_05.csv</i>	Distance-based	Greedy	Distance-based
<i>120_06.csv</i>	Distance-based	Greedy	Distance-based
<i>120_07.csv</i>	Distance-based	Greedy	Distance-based
<i>120_08.csv</i>	Greedy	Greedy	Greedy
<i>120_09.csv</i>	Distance-based	Greedy	Greedy
<i>120_10.csv</i>	Greedy	Greedy	Greedy
<i>120_11.csv</i>	Distance-based	Greedy	Greedy
<i>120_12.csv</i>	Distance-based	Greedy	Distance-based
<i>120_13.csv</i>	Greedy	Greedy	Greedy
<i>120_14.csv</i>	Distance-based	Greedy	Greedy
<i>120_15.csv</i>	Distance-based	Greedy	Distance-based
<i>120_16.csv</i>	Greedy	Greedy	Greedy
<i>120_17.csv</i>	Greedy	Greedy	Greedy
<i>120_18.csv</i>	Distance-based	Greedy	Greedy
<i>120_19.csv</i>	Distance-based	Greedy	Greedy

Table 8 Algorithm that provided the best Allocation for different criteria when $n = 250$

Instance	Criterion		
	No. of bundles + unassigned items	Homogeneity	Cost function J
250_00.csv	Greedy	Greedy	Greedy
250_01.csv	Distance-based	Greedy	Greedy
250_02.csv	Greedy	Greedy	Greedy
250_03.csv	Distance-based	Greedy	Greedy
250_04.csv	Distance-based	Greedy	Greedy
250_05.csv	Distance-based	Greedy	Greedy
250_06.csv	Distance-based	Greedy	Greedy
250_07.csv	Greedy	Greedy	Greedy
250_08.csv	Greedy	Greedy	Greedy
250_09.csv	Greedy	Greedy	Greedy
250_10.csv	Greedy	Greedy	Greedy
250_11.csv	Greedy	Greedy	Greedy
250_12.csv	Greedy	Greedy	Greedy
250_13.csv	Greedy	Greedy	Greedy
250_14.csv	Distance-based	Distance-based	Distance-based
250_15.csv	Greedy	Greedy	Greedy
250_16.csv	Greedy	Greedy	Greedy
250_17.csv	Greedy	Greedy	Greedy
250_18.csv	Greedy	Greedy	Greedy
250_19.csv	Greedy	Greedy	Greedy

Table 9 Algorithm that provided the best allocation for different criteria when $n = 500$

Instance	Criterion		
	No. of bundles + unassigned items	Homogeneity	Cost function J
500_00.csv	Greedy	Greedy	Greedy
500_01.csv	Distance-based	Greedy	Greedy
500_02.csv	Greedy	Distance-based	Distance-based
500_03.csv	Distance-based	Greedy	Greedy
500_04.csv	Greedy	Greedy	Greedy
500_05.csv	Greedy	Greedy	Greedy
500_06.csv	Greedy	Greedy	Greedy
500_07.csv	Distance-based	Distance-based	Distance-based
500_08.csv	Distance-based	Greedy	Distance-based
500_09.csv	Greedy	Greedy	Greedy
500_10.csv	Greedy	Distance-based	Distance-based
500_11.csv	Distance-based	Greedy	Greedy
500_12.csv	Greedy	Distance-based	Distance-based
500_13.csv	Distance-based	Greedy	Greedy
500_14.csv	Greedy	Greedy	Greedy
500_15.csv	Greedy	Greedy	Greedy
500_16.csv	Greedy	Greedy	Greedy
500_17.csv	Distance-based	Distance-based	Distance-based
500_18.csv	Distance-based	Greedy	Distance-based
500_19.csv	Greedy	Greedy	Greedy

Table 10 Algorithm that provided the best allocation for different criteria when $n = 1000$

Instance	Criterion		
	No. of bundles + unassigned items	Homogeneity	Cost function J
1000_00.csv	Greedy	Distance-based	Distance-based
1000_01.csv	Greedy	Distance-based	Distance-based
1000_02.csv	Greedy	Distance-based	Distance-based
1000_03.csv	Greedy	Greedy	Greedy
1000_04.csv	Distance-based	Greedy	Greedy
1000_05.csv	Greedy	Greedy	Greedy
1000_06.csv	Distance-based	Distance-based	Distance-based
1000_07.csv	Greedy	Distance-based	Distance-based
1000_08.csv	Greedy	Distance-based	Distance-based
1000_09.csv	Greedy	Distance-based	Distance-based
1000_10.csv	Greedy	Distance-based	Distance-based
1000_11.csv	Distance-based	Distance-based	Distance-based
1000_12.csv	Distance-based	Distance-based	Distance-based
1000_13.csv	Greedy	Greedy	Greedy
1000_14.csv	Greedy	Distance-based	Distance-based
1000_15.csv	greedy	Distance-based	Distance-based
1000_16.csv	Greedy	Greedy	Greedy
1000_17.csv	Greedy	Distance-based	Distance-based
1000_18.csv	Distance-based	Distance-based	Distance-based
1000_19.csv	Greedy	Distance-based	Distance-based

References

- Abdel-Basset, M., Manogaran, G., Abdel-Fatah, L., & Mirjalili, S. (2018). An improved nature inspired meta-heuristic algorithm for 1-D bin packing problems. *Personal and Ubiquitous Computing*, 22(5), 1117–1132.
- Abdul-Minaam, D. S., Al-Mutairi, W. M. E. S., Awad, M. A., & El-Ashmawi, W. H. (2020). An adaptive fitness-dependent optimizer for the one-dimensional bin packing problem. *IEEE Access*, 8, 97959–97974.
- Adler, M., Gibbons, P. B., & Matias, Y. (2002). Scheduling space-sharing for internet advertising. *Journal of Scheduling*, 5(2), 103–119.
- Asta, S., Özcan, E., & Parkes, A. J. (2016). Champ: Creating heuristics via many parameters for online bin packing. *Expert Systems with Applications*, 63, 208–221.
- Azar, Y., Boyar, J., Favrholt, L. M., Larsen, K. S., Nielsen, M. N., & Epstein, L. (2002). Fair versus unrestricted bin packing. *Algorithmica*, 34(2), 181–196.
- Bai, R., Blazewicz, J., Burke, E. K., Kendall, G., & McCollum, B. (2012). A simulated annealing hyper-heuristic methodology for flexible decision support. *4OR*, 10(1), 43–66.
- Baker, B. S. (1985). A new proof for the first-fit decreasing bin-packing algorithm. *Journal of Algorithms*, 6(1), 49–70.
- Baker, B. S., & Coffman, E. G., Jr. (1981). A tight asymptotic bound for next-fit-decreasing bin-packing. *SIAM. Journal on Algebraic Discrete Methods*, 2(2), 147–152.
- Baldi, M. M., & Bruglieri, M. (2017). On the generalized bin packing problem. *International Transactions in Operational Research*, 24(3), 425–438.
- Baldi, M. M., Manerba, D., Perboli, G., & Tadei, R. (2019). A generalized bin packing problem for parcel delivery in last-mile logistics. *European Journal of Operational Research*, 274(3), 990–999.

- Balogh, J., Békési, J., Dósa, G., Epstein, L., & Levin, A. (2020). Online bin packing with cardinality constraints resolved. *Journal of Computer and System Sciences*, 112, 34–49.
- Beaumont, O., Bonichon, N., & Larchevêque, H. (2010). *Bin packing under distance constraint*. Technical Report, Université de Bordeaux, Laboratoire Bordelais de Recherche en Informatique, Bordeaux.
- Békési, J., Galambos, G., & Kellerer, H. (2000). A $5/4$ linear time bin packing algorithm. *Journal of Computer and System Sciences*, 60(1), 145–160.
- Berlínska, J. (2020). Heuristics for scheduling data gathering with limited base station memory. *Annals of Operations Research*, 285(1–2), 149–159.
- Bilò, V. (2006). On the packing of selfish items. In *Proceedings of the 20th IEEE international parallel & distributed processing symposium* (pp. 1–9).
- Brélaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, 22(4), 251–256.
- Capua, R., Frota, Y., Ochi, L. S., & Vidal, T. (2018). A study on exponential-size neighborhoods for the bin packing problem with conflicts. *Journal of Heuristics*, 24(4), 667–695.
- Chacón, J. E. (2019). Mixture model modal clustering. *Advances in Data Analysis and Classification*, 13(2), 379–404.
- Chan, W. T., Chin, F. Y. L., Ye, D., Zhang, G., & Zhang, Y. (2007). Online bin packing of fragile objects with application in cellular networks. *Journal of Combinatorial Optimization*, 14(4), 427–435.
- Coffman E. G., Csirik, J., Galambos, G., Martello, S., & Vigo, D. (2013). Bin packing approximation algorithms: survey and classification. In *Handbook of combinatorial optimization* (pp. 455–531), Springer.
- Coffman, E. G., Garey, M. R., & Johnson, D. S. (1983). Dynamic bin packing. *SIAM Journal on Computing*, 12(2), 227–258.
- Coffman, E. G., Garey, M. R., & Johnson, D. S. (1987). Bin packing with divisible item sizes. *Journal of Complexity*, 3(4), 406–428.
- Csirik, J., & Johnson, D. S. (2001). Bounded space on-line bin packing: Best is better than first. *Algorithmica*, 31(2), 115–138.
- Csirik, J., & Woeginger, G. J. (1998). On-line packing and covering problems. In *Online algorithms* (pp. 147–177). Springer.
- Dahmani, N., Krichen, S., & Ghazouani, D. (2015). A variable neighborhood descent approach for the two-dimensional bin packing problem. *Electronic Notes in Discrete Mathematics*, 47, 117–124.
- Denstad, A., Ulsund, E., Christiansen, M., Hvattum, L. M., & Tirado, G. (2021). Multi-objective optimization for a strategic ATM network redesign problem. *Annals of Operations Research*, 296(1), 7–33.
- Dhahbi, S., Berrima, M., & Al-Yarimi, F. A. (2021). Load balancing in cloud computing using worst-fit bin-stretching. *Cluster Computing*. <https://doi.org/10.1007/s10075-021-03302-7>
- Diaby, M. (2010). Linear programming formulation of the vertex colouring problem. *International Journal of Mathematics in Operational Research*, 2(3), 259–289.
- Dósa, G., & He, Y. (2006). Bin packing problems with rejection penalties and their dual problems. *Information and Computation*, 204(5), 795–815.
- Du, K. L. (2010). Clustering: A neural network approach. *Neural Networks*, 23(1), 89–107.
- Duran, B. S., & Odell, P. L. (1974). *Cluster analysis: A survey*. Springer.
- Ekici, A. (2021). Bin packing problem with conflicts and item fragmentation. *Computers & Operations Research*, 126, 105–113.
- Elhedhli, S., Li, L., Gzara, M., & Naoum-Sawaya, J. (2011). A branch-and-price algorithm for the bin packing problem with conflicts. *INFORMS Journal on Computing*, 23(3), 404–415.
- Epstein, L., & Muallem, L. (2021). Online bin packing of squares and cubes. *arXiv*. <https://arxiv.org/abs/2105.08763>
- Epstein, L., & Levin, A. (2008). On bin packing with conflicts. *SIAM Journal on Optimization*, 19(3), 1270–1298.
- Epstein, L., & Levin, A. (2020). A note on a variant of the online open end bin packing problem. *Operations Research Letters*, 48(6), 844–849.
- 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).
- Farnè, M., & Vouldis, A. T. (2021). Banks' business models in the euro area: A cluster analysis in high dimensions. *Annals of Operations Research*. <https://doi.org/10.1007/s10479-021-04045-9>
- Fernandez, A., Gil, C., Baños, R., & Montoya, M. G. (2013). A parallel multi-objective algorithm for two-dimensional bin packing with rotations and load balancing. *Expert Systems with Applications*, 40(13), 5169–5180.
- Ficker, A. M., Spieksma, F. C., & Woeginger, G. J. (2021). The transportation problem with conflicts. *Annals of Operations Research*, 298(1), 207–227.

- Friesen, D. K., & Langston, M. A. (1991). Analysis of a compound bin packing algorithm. *SIAM Journal on Discrete Mathematics*, 4(1), 61–79.
- Galambos, G. (1985). *A new heuristic for the classical bin-packing problem*. Technical Report, Institute fuer Mathematik, Augsburg.
- Galambos, G., & Woeginger, G. J. (1993). Repacking helps in bounded space on-line bin-packing. *Computing*, 49(4), 329–338.
- Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. Freeman.
- Gendreau, M., Laporte, G., & Semet, F. (2004). Heuristics and lower bounds for the bin packing problem with conflicts. *Computers & Operations Research*, 31(3), 347–358.
- González-Almagro, G., Luengo, J., Cano, J. R., & García, S. (2020). DILS: Constrained clustering through dual iterative local search. *Computers & Operations Research*, 121, 104979.
- Grandcolas, S., & Pain-Barre, C. (2021). A hybrid metaheuristic for the two-dimensional strip packing problem. *Annals of Operations Research*. <https://doi.org/10.1007/s10479-021-04226-6>
- Grove, E. F. (1995). Online bin packing with lookahead. In *SODA'95: Proceedings of the sixth annual ACM-SIAM symposium on discrete algorithms* (pp. 430–436).
- Gupta, J. N., & Ho, J. C. (1999). A new heuristic algorithm for the one-dimensional bin-packing problem. *Production Planning & Control*, 10(6), 598–603.
- Gutin, G., Jensen, T., & Yeo, A. (2006). On-line bin packing with two item sizes. *Algorithmic Operations Research*, 1(2), 72–78.
- Hendrickson, B., & Leland, R. (1995). An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing*, 16(2), 452–469.
- Jansen, K. (1999). An approximation scheme for bin packing with conflicts. *Journal of Combinatorial Optimization*, 3(4), 363–377.
- Johnson, D. S. (1973). *Near-optimal bin packing algorithms*. Doctoral dissertation, Massachusetts Institute of Technology, Cambridge.
- Johnson, D. S. (1974a). Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3), 256–278.
- Johnson, D. S. (1974b). Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8(3), 272–314.
- Johnson, J., Reck, B. K., Wang, T., & Graedel, T. E. (2008). The energy benefit of stainless steel recycling. *Energy Policy*, 36(1), 181–192.
- Kang, J., & Park, S. (2003). Algorithms for the variable sized bin packing problem. *European Journal of Operational Research*, 147(2), 365–372.
- Karypis, G., Han, E. H., & Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8), 68–75.
- Kaufman, L., & Rousseeuw, P. J. (2005). *Finding groups in data: An introduction to cluster analysis*. Wiley.
- Kawaji, H., Takenaka, Y., & Matsuda, H. (2004). Graph-based clustering for finding distant relationships in a large set of protein sequences. *Bioinformatics*, 20(2), 243–252.
- Kellerer, H., & Pferschy, U. (1999). Cardinality constrained bin-packing problems. *Annals of Operations Research*, 92, 335–348.
- Kim, J., Lee, W., Song, J. J., & Lee, S. B. (2017). Optimized combinatorial clustering for stochastic processes. *Cluster Computing*, 20(2), 1135–1148.
- Kou, G., & Lou, C. (2012). Multiple factor hierarchical clustering algorithm for large scale web page and search engine clickstream data. *Annals of Operations Research*, 197(1), 123–134.
- Kou, G., Peng, Y., & Wang, G. (2014). Evaluation of clustering algorithms for financial risk analysis using MCDM methods. *Information Sciences*, 275, 1–12.
- Kou, G., Yang, P., Peng, Y., Xiao, F., Chen, Y., & Alsaadi, F. E. (2020). Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods. *Applied Soft Computing*, 86, 105836.
- Levin, M. S. (2015). Combinatorial clustering: Literature review, methods, examples. *Journal of Communications Technology and Electronics*, 60(12), 1403–1428.
- Li, T., Kou, G., & Peng, Y. (2020a). Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods. *Information Systems*, 91, 101494.
- Li, T., Kou, G., Peng, Y., & Shi, Y. (2020b). Classifying with adaptive hyper-spheres: An incremental classifier based on competitive learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(4), 1218–1229.
- Liu, Q., Cheng, H., Tian, T., Wang, Y., Leng, J., Zhao, R., Zhang, H., & Wei, L. (2021). Algorithms for the variable-sized bin packing problem with time windows. *Computers & Industrial Engineering*, 155, 101715.

- 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*, 206(1), 241–264.
- Maiza, M., & Radjef, M. S. (2011). Heuristics for solving the bin-packing problem with conflicts. *Applied Mathematical Sciences*, 5(35), 1739–1752.
- Maiza, M., Radjef, M. S., & Sais, L. (2016). Lower bounds for efficient packing problems in heterogeneous bins with constraint conflicts. *Intelligent Mathematics II: Applied Mathematics and Approximation Theory* (pp. 263–270). Springer.
- Morissette, L., & Chartier, S. (2013). The k-means clustering technique: General considerations and implementation in Mathematica. *Tutorials in Quantitative Methods for Psychology*, 9(1), 15–24.
- Munien, C., Mahabeer, S., Dzitiro, E., Singh, S., Zungu, S., & Ezugwu, A. E. S. (2020). Metaheuristic approaches for one-dimensional bin packing problem: A comparative performance study. *IEEE Access*, 8, 227438–227465.
- Muritiba, A. E. F., Iori, M., Malaguti, E., & Toth, P. (2010). Algorithms for the bin packing problem with conflicts. *INFORMS Journal on Computing*, 22(3), 401–415.
- Pantula, P. D., Miriyala, S. S., & Mitra, K. (2020). An evolutionary neuro-fuzzy C-means clustering technique. *Engineering Applications of Artificial Intelligence*, 89, 103435.
- Piciarelli, C., Micheloni, C., & Foresti, G. L. (2013). Kernel-based clustering. *Electronics Letters*, 49(2), 113–114.
- Polyakovskiy, S., & M'Hallah, R. (2021). Just-in-time two-dimensional bin packing. *Omega*, 102, 102311.
- Sadykov, R., & Vanderbeck, F. (2013). Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS Journal on Computing*, 25(2), 244–255.
- Sakawa, M. (2013). *Fuzzy sets and interactive multiobjective optimization*. Springer.
- Santos, L. F. M., Iwayama, R. S., Cavalcanti, L. B., Turi, L. M., de Souza Morais, F. E., Mormilho, G., & Cunha, C. B. (2019). A variable neighborhood search algorithm for the bin packing problem with compatible categories. *Expert Systems with Applications*, 124, 209–225.
- 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(7), 627–645.
- Schwerin, P., & Wäscher, G. (1997). The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP. *International Transactions in Operational Research*, 4(5–6), 377–389.
- Taylor, G. S., Chan, Y., & Rasool, G. (2017). A three-dimensional bin-packing model: Exact multicriteria solution and computational complexity. *Annals of Operations Research*, 251(1–2), 397–427.
- Thelning, K. E. (2013). *Steel and its heat treatment*. Butterworth-Heinemann.
- Verma, R., Singhal, A., Khadilkar, H., Basumatary, A., Nayak, S., Sing., H. V., Kumar, S., & Sinha, R. (2020). A generalized reinforcement learning algorithm for online 3D bin-packing. *arXiv*. <https://arxiv.org/abs/2007.00463>
- Wee, T. S., & Magazine, M. J. (1982). Assembly line balancing as generalized bin packing. *Operations Research Letters*, 1(2), 56–58.
- Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3), 645–678.
- Xu, X., Ding, S., Wang, Y., Wang, L., & Jia, W. (2021). A fast density peaks clustering algorithm with sparse search. *Information Sciences*, 554, 61–83.
- Yao, A. C. C. (1980). New algorithms for bin packing. *Journal of the ACM*, 27(2), 207–227.
- Yu, S. S., Chu, S. W., Wang, C. M., Chan, Y. K., & Chang, T. C. (2018). Two improved k-means algorithms. *Applied Soft Computing*, 68, 747–755.
- Yuan, Y., Li, Y. J., & Wang, Y. Q. (2014). An improved ACO algorithm for the bin packing problem with conflicts based on graph coloring model. In *2014 international conference on management science & engineering—21th annual conference proceedings* (pp. 3–9).
- Zhou, L., Hu, X., Ngai, E. C. H., Zhao, H., Wang, S., Wei, J., & Leung, V. C. (2016). A dynamic graph-based scheduling and interference coordination approach in heterogeneous cellular networks. *IEEE Transactions on Vehicular Technology*, 65(5), 3735–3748.