Optimising Bin Packing in Transportation Logistics: Heuristic Approaches and Instance Space Analysis

R.G. Rakotonirainy[1*] & F.Netshitungulu[1]

ARTICLE INFO

Article details
Presented at the 35th annual conference of the Southern African Institute for Industrial Engineering, held from 20 to 22 October 2025 in Cape Town, South Africa

Available online 8 Dec 2025

Contact details
* Corresponding author
georgina.rakotonirainy@uct.ac.za

Author affiliations
1 Department of Statistical Sciences, University of Cape Town, South Africa

ORCID® identifiers
R.G. Rakotonirainy
https://orcid.org/0000-0002-1672-2695

F.Netshitungulu
https://orcid.org/0009-0001-3508-748X

DOI
http://dx.doi.org//10.7166/36-3-3329

ABSTRACT

Transportation management plays a vital role in supply chain logistics, emphasizing the efficient movement and storage of goods to meet demand in a cost-effective way. In many inter-firm networks, goods are produced using various strategies, then consolidated into appropriate bins by third-party providers, and stored at different locations before reaching the final customers. This study investigates the bin-packing process in the context of transportation logistics, which serves as a key decision support tool for logistics managers who are responsible for optimising the packing and transportation of goods. Heuristic bin-packing algorithms are proposed to improve existing upper bound solutions in the literature. In addition, instance space analysis is applied to assess the effectiveness of these algorithms, focusing on how different problem features influence their performance. The results show that the selected features can explain the complexity of packing instances, providing insights into the strengths and weaknesses of the algorithms.

OPSOMMING

Vervoerbestuur speel 'n belangrike rol in voorsieningskettinglogistiek, met die klem op die doeltreffende beweging en berging van goedere om op 'n koste-effektiewe manier aan die vraag te voldoen. In baie interfirma-netwerke word goedere vervaardig deur verskeie strategieë te gebruik, dan deur derdeparty-verskaffers in toepaslike houers gekonsolideer en op verskillende plekke gestoor voordat dit die finale kliënte bereik. Hierdie studie ondersoek die houerverpakkingsproses in die konteks van vervoerlogistiek, wat dien as 'n belangrike besluitnemingsondersteuningsinstrument vir logistieke bestuurders wat verantwoordelik is vir die optimalisering van die verpakking en vervoer van goedere. Heuristiese houerverpakkingsalgoritmes word voorgestel om bestaande boonste grensoplossings in die literatuur te verbeter. Daarbenewens word instansieruimte-analise toegepas om die doeltreffendheid van hierdie algoritmes te bepaal, met die fokus op hoe verskillende probleemkenmerke hul prestasie beïnvloed. Die resultate toon dat die gekose kenmerke die kompleksiteit van verpakkingsinstansies kan verklaar, wat insigte bied in die sterk- en swakpunte van die algoritmes.

1. INTRODUCTION

Global industrial production, logistics, and supply chains are increasingly driven by a strategic focus on core competencies, the outsourcing of manufacturing to enhance supply chain value, and the optimisation of returns on capital investments, all while offering comprehensive solutions to target customers. In this context, the effective use of materials and transportation capacity is critical to maintaining a competitive advantage. A key area of focus in academic research has been the exploration of packing problems, which examine the efficient packing of items (e.g., customer goods) into larger containers (e.g., trucks, shipping containers) while considering practical constraints (e.g., capacity limits) and optimising key factors (e.g., minimising costs or maximising space use). These problems arise in many real-world applications, such as determining how to load delivery vans efficiently that operate in congested urban environments or allocating cargo in aircraft based on volume and weight constraints. Both scenarios require careful consideration of item priorities, container capacities, and operational costs – key aspects that are captured in formal packing problem models. Consequently, packing problems have garnered significant attention from the scientific community, resulting in a substantial body of literature [1, 2].

This paper investigates a specific variant of the bin-packing problem, known as the generalised bin-packing problem (GBPP) [15]. As the name suggests, the GBPP is a generalisation of the classic bin-packing problem, incorporating a set of compulsory and non-compulsory items, along with a set of bins. The goal of the GBPP is to allocate all compulsory items, as well as some non-compulsory items, into the bins in such a way that costs are minimised. The non-compulsory items offer additional profit if they are packed into the bins; however, no cost is incurred if they are not allocated. Therefore, the cost minimised in the GBPP is the difference between the cost of the bins used and the profit generated from the non-compulsory items that are packed. This problem is particularly relevant to logistics, where changes in supply chain processes, driven by the implementation of green logistics, have led researchers and practitioners to redefine their operations. Green logistics has an impact on the structures and systems in the transport, distribution, and storage of goods. In this regard, the GBPP takes into account bin costs (e.g., costs associated with packaging materials used for goods) and item profits (e.g., profits from goods to be packed into boxes), while considering restrictions related to bin availability and their heterogeneity in cost and volume. The GBPP also provides innovative insights into airfreight transportation, where items are loaded based on volume. It captures the fundamental trade-off between shipping costs and item profits – a trade-off that exists in all transportation settings.

Although the literature on the GBPP remains limited owing to its recent introduction, several solutions have been proposed to address such problems, given that the foundations of GBPP solutions are derived from the well-established bin-packing problem. These solutions have been mainly either exact methods or heuristic approaches [15, 16]. Exact methods, based on mathematical formulations, aim to find the optimal solution; however, as problem complexity increases, these methods often become computationally expensive or impractical. On the other hand, heuristic methods provide near-optimal solutions within a reasonable timeframe, making them more suitable for large-scale problem settings. For effective decision support, particularly in logistics and transportation planning, bin-packing methods must deliver high-quality solutions rapidly. The first major contribution of this paper is the proposal of heuristic algorithms that improve on existing approaches, offering enhanced solution quality.

It is important to note that algorithms for the GBPP are typically evaluated on the basis of their average performance across benchmark problem instances. However, as suggested by the “no-free-lunch” theorem [5], no single algorithm performs optimally in every scenario. Even the most effective algorithms may reveal inherent weaknesses, which are often overlooked or inadequately reported. As a result, an algorithm selection approach is necessary to identify the most appropriate algorithm for any given problem instance [10]. Therefore, the second contribution of this work is the application of the instance space analysis (ISA) [13] approach to GBPP algorithms for the first time. This approach provides deeper insights beyond average performance metrics, enabling more precise recommendations for selecting the best algorithm for specific instances. Rather than relying solely on the algorithm with the best overall average performance, we propose selecting the algorithm that is best suited to each individual problem instance.

The remainder of this paper is organised as follows. Section 2 provides a detailed problem description and a review of related work on the GBPP, including an overview of ISA methods in the literature. Section 3 presents a comprehensive description of the developed heuristic algorithms. In Section 4, the performance of these heuristics is analysed with respect to benchmark instances. Section 5 discusses the application of

ISA to the GBPP, including the results and performance evaluation of the proposed heuristic selection framework. Section 6 concludes the paper and offers suggestions for future research.

2. PROBLEM DESCRIPTION AND LITERATURE REVIEW

In recent years, with the substantial growth of e-commerce and the corresponding increase in parcel deliveries in urban areas, last-mile problems have become increasingly complex. As a result, many researchers have focused on finding solutions at various levels. At the operational level, the optimisation of last-mile logistics primarily involves addressing vehicle routing and scheduling problems, which are often complicated by factors such as multiple depots, products, and stringent loading constraints. At the tactical level, effective planning is crucial for evaluating options (e.g., various transportation tenders) and for allocating the appropriate resources (vehicles, operators, products) to the right locations (depots, facilities, distribution centres) for day-to-day operations. In addition to the successful application of packing problems at the operational level, packing problems have increasingly been used as tools to model tactical decisions in transportation and supply chain management. This has led to the emergence of new problem variants, such as the generalised bin-packing problem with bin-dependent item profit (GBPPI) [16], the generalised extensible bin-packing problem with overload cost (GEBPPOC) [20], the multi-period variable cost and size bin-packing problem with assignment cost (MVCBPP-AC) [21], and the generalised bin-packing problem with incompatible categories (GBPPIC) [7]. The problem studied in this paper is the original GBPP of Baldi *et al.* [15], which serves as a decision support tool at the tactical level for last-mile logistics.

In the GBPP, bins are characterised by both their capacity and their cost of use, and are classified by type. Bins of the same type share identical capacity and cost characteristics. The items to be packed can be either compulsory (i.e., mandatory to load) or non-compulsory, and are characterised by their weight and profit. The objective of the GBPP is to load compulsory items and profitable non-compulsory items into appropriate bins in such a way as to minimise the net cost. This is determined by the difference between the total cost of the bins used and the total profit derived from the items that are loaded. In the context of last-mile logistics, the GBPP models a tactical problem in which the company responsible for parcel delivery must decide on the number of transportation companies (represented by bin types) with which to contract, the number of vehicles to be selected from the fleet of the chosen companies (represented by bins), and the assignment of parcels (represented by items) to the selected vehicles for delivery.

The GBPP can be formulated as follows [15]. Let I denote the set of items and v_i and P_i be the volume and profit ot item $i \in I$ respectively. Define I^C and I^{NC} as the subsets of compulsory and non-compulsory items. Let J be the set of bins and T the set of bin types. Define $\sigma(j) = t \in T$ and the type of t of bin $j \in J$. For each bin type $t \in T$, let L_t and U_t be the minimum and the maximum number of bins of that type respectively, which may be used as well as the cost C_t and the volume V_t of the bin. Finally, let $U \leq \sum_{t \in T} U_t$ be the total number of available bins of any type. The mathematical formulation is given by

$$\text{Minimise } \sum_{j \in J} C_j y_j - \sum_{j \in J} \sum_{i \in I} P_i x_{ij} \tag{1}$$

Subject to

$$\sum_{i \in I} v_i \, x_{ij} \leq V_j y_j, \, j \in J, \tag{2}$$

$$\sum_{j \in J} x_{ij} = 1, \, i \in I^C, \tag{3}$$

$$\sum_{j \in J} x_{ij} \leq 1, \, i \in I^{NC}, \tag{4}$$

$$\sum_{j \in J : \sigma(j) = t} y_j \leq U_t, \, \mathrm{t} \in T, \tag{5}$$

$$\sum_{j \in J : \sigma(j) = t} y_j \geq U_t, \, \mathrm{t} \in T, \tag{6}$$

$$\sum_{j \in J} y_j \leq U, \tag{7}$$

$$y_j \in \{0,1\}, j \in J, \tag{8}$$

$$x_{ij} \in \{0,1\}, i \in I, j \in J. \tag{9}$$

In this formulation, items are assigned to bins ($x_{ij} = 1$ if item i is assigned to bin j) to minimise the net cost, defined as the difference between the total cost of the bins used and the profit from non-compulsory items (1). The profit of compulsory items is not considered in the objective function, as it is constant and guaranteed. Constraint (2) ensures that the volume of items in a bin does not exceed the bin's capacity. Constraint (3) mandates that all compulsory items be packed into the bins, while Constraint (4) allows for

a subset of compulsory items to be packed, with others left unpacked. Constraints (5) and (6) limit the number of bins available for each bin type, which may be company-dependent, as the company specifies the number of bins of each type that are available for use. Constraint (7) stipulates that the total number of bins used cannot exceed the total number of bins that are available for use. Finally, constraints (8) and (9) define binary decision variables: the selection of a bin ($y_j = 1$ if bin is selected, 0 otherwise) and the assignment of an item to a bin, ensuring binary values for both.

2.1. Existing GBPP heuristics

Baldi *et al*. [15] introduced two key heuristic variants based on two basic algorithms: the first fit decreasing (FFD) and best fit decreasing (BFD) algorithms. These variants are referred to as U-FFD and U-BFD in this work, and serve as benchmarks by which to compare the performance of the proposed heuristics. The U-FFD algorithm begins by sorting the items in non-increasing order of volume, creating a sorted item list (SIL), and sorting the bins in increasing order of capacity, creating a sorted bin list (SBL). The algorithm starts by considering used bins, which are initially empty. For compulsory items, the algorithm searches for the first used bin in which the item fits. If no used bins can accommodate the item, a new bin is opened. For non-compulsory items, the algorithm prioritises fitting the item into a used bin. If this is not possible, the algorithm evaluates the profitability of packing the item. A new bin is opened only if the total profit (the item's profit plus the remaining items' profit in the SIL) exceeds the cost of the bin, provided that the combination of items fits in the bin. Items that do not meet this condition are discarded. The U-BFD algorithm shares many similarities with U-FFD, but differs in how bins are selected. Instead of fitting an item into the first available bin, items are packed into the "best" bin, which is the bin that leaves the least residual space after packing. For compulsory items, the best bin is the first bin in the sorted list of bins, while for non-compulsory items the process mirrors that of U-FFD, selecting the best bin based on the same profit considerations.

Baldi *et al*. [15] introduced two additional heuristics, best profitable (BP) and best assignment (BA), both of which are based on the U-BFD algorithm but extend the concept of the "best" bin. The primary difference between BP and BA lies in how new bins are opened. The BP algorithm opens a new bin if the combined profit of an item and any additional items that can fit into the bin exceeds the bin's cost. In contrast, the BA algorithm opens the bin that maximises the profit of the item being packed. Both approaches aim to optimise the packing of items while considering bin-dependent profits, thus addressing the GBPP problem. In addition to the heuristics, Baldi *et al*. [15] proposed two metaheuristics: the greedy adaptive search procedure (GASP) and the model-based metaheuristic (MBM). The GASP algorithm involves generating multiple feasible solutions and using them as starting points for further iterations. In each iteration, a packed item is swapped with an unpacked item if the swap is feasible and profitable, continuing until no further improvements can be made or a set number of iterations is reached. The MBM algorithm simultaneously solves two subproblems, minimising the total cost and maximising item profits, by combining the results, ensuring no duplicates in the solutions.

In 2021, Crainic *et al*. [21] proposed three heuristic algorithms to solve the MVCBPP-AC problem. These algorithms, named HS1, HS2, and HS3, focus on packing items based on their cost contributions and the residual space in bins. The HS1 algorithm prioritises items with the highest cost-to-residual capacity ratio; the HS2 algorithm selects bins based on the lowest cost-to-capacity ratio; and the HS3 approach is similar to that of HS2, but adds the cost of items to the bin cost when selecting bins. To address the GEBPPOC, Ding *et al*. [20] introduced two heuristic algorithms: ant colony optimisation and particle swarm optimisation. These algorithms modify traditional solution representations and operators, adjusting methods such as the construction of the initial solution and setting termination criteria to solve the GEBPPOC efficiently and to deliver high-quality results. Finally, to solve the GBPPIC, Rodrigues *et al*. [7] proposed the adaptive large neighbourhood search metaheuristic. This algorithm effectively addresses the issue of incompatible item categories while minimising the number of vehicles required.

2.2. Instance space analysis

Assessing algorithm performance objectively is inherently difficult. As Hooker [9] points out, an algorithm may perform well on a specific set of data instances, but there is no guarantee that it will yield similar results when applied to different instances. Liu *et al*. [11] also argue that the average performance of an algorithm, based on potentially unrepresentative problem instances, cannot be relied upon as a sole indicator of its effectiveness. In response to such biases, the ISA was proposed by Smith-Miles *et al*. [13] to provide a more comprehensive approach to evaluation. The underlying idea of ISA is to project a given

complex problem into a two-dimensional instance space in which algorithms can be tested and compared, offering valuable insights into their performance.

The ISA approach maps problem instances on to a two-dimensional space, referred to as the "instance space", where each problem instance is represented as a distinct point. This mapping is achieved by calculating specific attributes of the problem instances, known as "features". The goal is to select features that accurately capture the complexity and difficulty of each instance for the algorithms being tested. By projecting problem instances into this space, it becomes possible to visualise how different instances relate to one another and where algorithms excel or face difficulties. This process extends the understanding of algorithm performance beyond merely reporting average-case results.

The ISA framework, introduced by Smith-Miles *et al.* [13], builds on the foundational concept of the algorithm selection problem (ASP), first proposed by Rice in 1976 [10]. Rice's ASP aimed to establish a mapping between algorithms and a set of performance metrics, thereby making it easier to choose the most appropriate algorithm for a given problem instance. The ISA refines this concept further by mapping each algorithm's performance in the problem instance space in a way that optimises the selection process. Moreover, the ISA framework enables the extraction of the algorithm's "footprint", which represents its behaviour in different problem instances and performance measures, as illustrated in Figure 1. More details of the component of the framework may be found in the original paper [10].

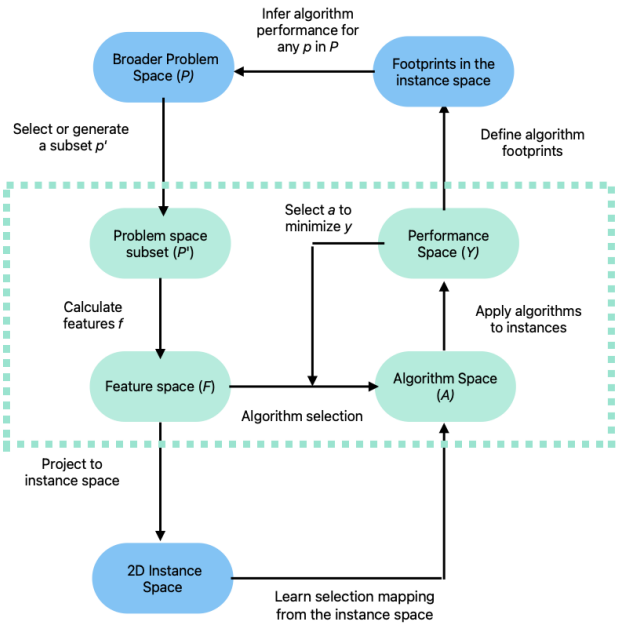

Figure 1: A representation of the ISA framework by Smith-Miles *et al.* [13]

The ISA methodology has been applied to a variety of packing problems since its introduction. In 2020, Liu *et al.* [11] applied ISA to one- and two-dimensional (1D and 2D) bin-packing problems. Using instances from the literature and newly generated instances, they evaluated several algorithms, including variants of the FFD and BFD, along with the Djang and Finch (DJD) algorithm for the 1D case. The results revealed that the FFD variant performed well on 81.5% of instances, while the DJD algorithm performed well on 79%, with DJD showing a higher precision (91.4%) than FFD (88.2%). This suggested that DJD would be the preferred algorithm in regions where both performed well. For the 2D case, algorithms such as hybrid first-fit, guillotine, maximal rectangles (MAXREACTS), and skyline were tested. MAXREACTS performed particularly well, achieving good results on 96.4% of instances, suggesting that the instances were not sufficiently difficult for it.

In 2021, Smith-Miles *et al.* [12] applied ISA to the 0-1 knapsack problem (KP), building on Pisinger's work, which questioned the difficulty of instances for KP problems. Their goal was to develop harder test instances that could help to create more advanced algorithms. They evaluated three algorithms: EXPKNAP, MINKNAP, and COMBO. The ISA framework provided valuable insights into the regions where these

algorithms performed well or poorly, helping to identify the harder instances that challenge current algorithms. COMBO emerged as the best-performing algorithm in the instance space, while MINKNAP performed well in a small subset of instances. The study not only answered Pisinger's question, "Where are the hard knapsack instances?", but also highlighted the importance of generating harder instances for future algorithm development.

In 2023, Jooken *et al*. [8] advanced the use of ISA for the 0-1 KP by focusing on identifying features that make instances hard to solve for state-of-the-art solvers. They used machine learning models to predict the empirical hardness of instances, aiming to compile features that would help to create more robust algorithms. This study contributed new insights into the empirical hardness of instances, which had been lacking in previous literature because of the predominance of easy-to-solve instances. Their findings showed that the harder instances, derived from their machine learning model, posed significant difficulties, as evidenced by the extended computational times required by COMBO compared with other algorithms.

Most recently, Liu *et al*. [11] applied ISA to the 2DBPP to compare mixed integer programming models, including those by Chen [4], Pisinger [6], Blum [3], and Castro [18]. The models were compared on the basis of the time required to solve various instances. Chen's model performed the best, achieving 48.3% good performance, while Pisinger's model performed the worst at 19.5%. The study demonstrated that the precision of Blum's model increased with new instances, becoming the best-performing model in certain regions. Overall, the ISA proved effective in providing valuable insights into where each algorithm excels or struggles in the instance space. This research also highlighted the importance of having a diverse set of instances to span the instance space, providing deeper insights into algorithm performance.

3. IMPROVED GBPP HEURISTICS

In this work, three algorithms are proposed, respectively referred to as GRASPm, LNSm, and GRASP LNS. These algorithms are inspired by heuristics that are in the literature. A detailed description of each algorithm is provided in this section.

3.1. The GRASPm algorithm

The GRASPm algorithm, as presented in Algorithm 1, generates a packing solution by applying a local search method designed to refine the current solution iteratively. The fundamental principle of GRASPm is that, whenever a local search produces an improved solution, the algorithm updates the current solution to this enhanced configuration. The search process continues until no further improvements can be achieved, or until the algorithm reaches its predetermined maximum number of iterations. Initially, the GRASPm generates multiple solutions based on a variant of the U-FFD algorithm, with each solution incorporating different sorting strategies for items and bins. As a result, the initial solutions produced by GRASPm show variability, as the sorting strategy has a significant impact on the way in which the U-FFD algorithm allocates items to bins. By integrating these diverse starting points, the GRASPm algorithm conducts a local search to explore neighbouring solutions, as shown in Algorithm 2, making potential adjustments to the bin assignments in an attempt to optimise the solution. The algorithm ends either when no additional improvements are found, or when the maximum iteration limit is reached.

Algorithm 1: The GRASPm algorithm

```
Input: A list of bins, a list of items, max_iteration
Output: A feasible packing of items into the bins
Solution = multiple starting points
for element in Solution do
    current_objective = objective value of element
    for iteration in range max_iteration do
        generate neighbours using Algorithm 2
        if no feasible neighbours exist then
            break
        else
            if objective value of generated neighbours < current_objective then
                Update current_objective and solution
            else
                break
```

Algorithm 2: Generate neighourhood method

```
Input: Solution
Output: Neighbours to the Solution
1. used_bins = list of bins used in the Solution
2. non_used_bins = list of non used bins in the Solution
3. for bin in used_bins do
4.     for element in non_used_bins do
5.         if items in bin can be placed in element then
6.             if cost of element < cost of bin then
7.                 swap the bins
8.         else
9.             break
```

3.2. The LNSm algorithm

The LNSm algorithm operates similarly to GRASPm, beginning with multiple initial solutions and iteratively attempting to improve them until no further enhancements can be made or the maximum number of iterations is reached. However, as its name indicates, the LNSm algorithm uses a large neighbourhood search strategy. This approach incorporates destroy-and-repair techniques to generate new solutions. That is, beginning with the current solution, the algorithm "destroys" part of it by removing a subset of items, while ensuring that mandatory items, which must be packed, remain intact. The solution is then "repaired" by finding other unassigned items that can be packed into the spaces left by the removed items. The objective of the repair phase is to improve the current solution while maintaining its feasibility.

As detailed in Algorithm 3, the process of generating a large neighbourhood involves the destroy method removing all non-mandatory items, followed by the repair method, which attempts to fill the resulting gaps with unassigned items. If the new solution is superior to the current one, the algorithm updates the solution accordingly. If not, the current solution is retained, and the algorithm proceeds to the next iteration. The final solution is either the best solution found during the iterations, one that cannot be further improved, or the last feasible solution discovered before reaching the maximum number of iterations.

Algorithm 3: Generate large neighbourhood method

```
Input: Solution
Output: Neighbours to the Solution
1. used_bins = list of bins used in the Solution
2. packed_items = list of packed items in the Solution
3. unpacked_items = list of unpacked items in the Solution
4. for bin in used_bins do
5.     for item in packed_items do
6.         if item is compulsory then
7.             leave it in
8.         else
9.             remove the item
10.    for item in unpacked_items do
11.        if item fits in any of the previously used bins then
12.            pack the item into the bin
13.        else
14.            move onto the next item
```

3.3. The GRASP_LNS algorithm

The GRASP_LNS algorithm aims to enhance both the GRASPm and the LNSm algorithms by integrating their respective strengths. In this approach, the initial solution is the one produced by the GRASPm algorithm that is regarded as the best solution that is derived from the GRASPm process. This solution is then fed into the LNSm algorithm to generate the best final solution. The rationale behind this hybrid approach is to provide the LNSm algorithm with a solution that has already been optimised by the GRASPm algorithm. By leveraging this proven solution as the starting point for the LNSm method, the expectation is that the LNSm algorithm will either refine the solution or, at the very least, maintain its high quality throughout the search process.

4. APPRAISAL OF HEURISTIC PERFORMANCE

The results from the proposed heuristics were compared with the benchmark U-FFD and U-BFD algorithms. The benchmark problem instances used in the comparative study are described in Section 4.1, followed by a discussion of the results in Section 4.2. All the algorithms were implemented using Spyder IDE version 5.4.5 with Python version 3.9.14, and RStudio version 2022.07.1 with R version 4.2.0. The experiments were carried out on a system equipped with a 1.8 GHz dual-core Intel core i5 processor, 8 GB of RAM, and running MacOS 12.7.6.

4.1. Benchmark instances

To evaluate and compare the performance of the heuristics, the benchmark datasets generated by Baldi *et al*. [15] was used; these are publicly available, and can be accessed from their repository [17]. Instances containing more than 500 items were excluded owing to the excessively long computational times encountered by the heuristics. A summary of the data is presented in Table 1.

Table 1: Summary of the dataset from [17] used in the comparative study

Classes	No. of instances	No. of items	No. of bin types (capacities)	Item classification
Class 0	240	25, 50,100, 200	3 (100, 120, 150) 5 (60, 80, 100, 120, 150)	All compulsory
Class 1	248	25, 50,100, 200	3 (100, 120, 150) 5 (60, 80, 100, 120, 150)	All non-compulsory
Class 2	247	25, 50,100, 200	3 (100, 120, 150) 5 (60, 80, 100, 120, 150)	All non-compulsory

Class 0 consists of instances containing only compulsory items, with the number of items in each instance being 25, 50, 100, or 200. The profit for each item is fixed at 200. The instances may include either three bin types with capacities of 100, 120, and 150 respectively, or five bin types with capacities of 60, 80, 100, 120, and 150 respectively. The cost of each bin is proportional to its capacity. Classes 1 and 2 reuse the same instances as Class 0, but with all the items being non-compulsory. In these classes, the profits of the items are adjusted to follow a uniform distribution.

4.2. Results of the heuristics

Table 2 presents the average percentage gap for all five algorithms, based on the data instances described in Section 4.1. The percentage gap between the net cost achieved by each algorithm and the lower bound provides a clear indication of the performance of each algorithm in different instance classes. In particular, the proposed algorithms achieved a better reduced percentage gap for the lower bound than the benchmark algorithms. The GRASP_LNS algorithm achieved the lowest percentage gap of 7.9%, making it the best-performing algorithm. The GRASPm followed closely with 8.3%. The LNSm was ranked third, achieving a percentage gap of 8.4%. These general results corroborated the performance of all the algorithms with respect to each class of instances (the last three columns of Table 2). The GRASP_LNS algorithm performed consistently well across all classes, achieving the best results in Class 1 and Class 2 instances and maintaining competitive performance in Class 0. Overall, the three proposed algorithms outperformed the benchmark algorithms.

The exploration of the percentage gap in different classes, especially by dividing the instances into groups based on the number of items, provided valuable insights. The instances were divided into four groups based on the number of items: 25 items, 50 items, 100 items, and 200 items respectively. This segmentation allowed for a detailed comparison of how the algorithms performed as the number of items increased, as shown in Table 3. The improvement in the percentage gap remained consistent across all the algorithms as the number of items increased. This meant that, regardless of the size of the problem (in the number of items), most of the algorithms were still able to outperform the U-FFD and U-BFD benchmarks.

Table 2: Average percentage gap from the lower bound and the net cost achieved by all algorithms in all data instances

Algorithm	Average % gap	% gap for Class 0	% gap for Class 1	% gap for Class 2
U-FFD	11.9%	6.8%	22.9%	13.4%
U-BFD	11.6%	6.6%	22.4%	13.1%
GRASPm	8.3%	6.2%	13.3%	8.1%
LNSm	8.4%	6.9%	13.7%	8.3%
GRASP_LNS	7.9%	6.2%	12.7%	7.8%

Table 3: Average percentage gap from the lower bound and the net cost achieved by all algorithms with respect to the number of items on data instances

Algorithm	% gap with respect to number of items			
	25	50	100	200
U-FFD	12.7%	13.2%	12.1%	11.3%
U-BFD	12.7%	13.1%	11.9%	10.9%
GRASPm	8.8%	9.3%	8.6%	7.8%
LNSm	9.1%	9.4%	8.7%	8.1%
GRASP_LNS	8.1%	8.8%	8.2%	7.6%

5. APPLICATION OF INSTANCE SPACE ANALYSIS

Despite the advantages of the proposed heuristics, their ability to generalise across different problem domains is often limited. It is also improbable that a single heuristic would outperform all other heuristics in every case. Therefore, an ISA approach was applied to determine the weakness and strength of the considered algorithms against appropriate benchmark test instances. This section contains details of the ISA application to the GBPP problem and the underlying results. The MATILDA platform [14] was used for the implementation of the ISA framework.

5.1. The ISA setting

A new set of instances was generated using the test instances from Baldi *et al.* [15]. The volume of each item was randomly generated within the range of the smallest and largest volumes observed in the original instances, while the profit for each item was randomly assigned between the smallest and the largest profits seen in the other classes. The objective of this class is to maintain the characteristics of the original instances while introducing variability for the ISA framework. All instances in this class contain 50 items, given the high computational costs associated with upper bound calculations and the ISA. Furthermore, each instance in this class consists of a mixture of compulsory and non-compulsory items, with one-third of the items designated as compulsory and the remaining two-thirds as non-compulsory, all selected randomly.

The next step is feature selection, which is a critical step, as it directly affects the model's ability to represent and predict the behaviour of problem instances effectively. The GBPP problem is one-dimensional in nature, represented by a key characteristic, the volume of items. The features are thus designed to emphasise this aspect, and aim to capture various facets of the instances that are relevant to the algorithm's performance. In this work, a total of 11 features have been considered for use within the ISA framework, as outlined in Table 4. These features primarily describe the volumes of items in each instance, with the volumes being normalised for consistency in calculations. Feature 1, for instance, measures the number of items in each instance, which is an important factor, since larger instances require more computational effort to solve effectively. Features 2 through 6 are statistical measures of item volumes, providing insight into the distribution and variability of items in the instance. Features 7 through 11 further characterise the volume distributions and help to differentiate between instances with varying volume profiles.

Table 4: Description of features used for the ISA framework

Feature	Description
n	The number of items in the instance
mean	The average volumes of the items present in the instance
median	The median of the volumes of the items present in the instance
var	The variance of the volumes of the items present in the instance
max	The volume of the largest item in the instance
min	The volume of the smallest item in the instance
large	The proportion of items in the instance that have volume ≤ 1/2 but ≥ 1/3
medium	The proportion of items in the instance that have volume ≤ 1/3 but ≥ 1/4
small	The proportion of items in the instance that have volume ≤ 1/4
tiny	The proportion of items in the instance that have volume ≤ 1/10
mini	The proportion of items in the instance that have volume ≤ 1/20

While these features are designed to capture various characteristics of the instances, the direct correlation between these features and the difficulty of an instance is not immediately clear. Drawing from prior research by Schwerin and Wascher [49], it is hypothesised that instances with a mean volume of around one-third will present more difficulty for the algorithms. This hypothesis is based on the observation that instances with larger volumes are generally easier to solve, as they typically require fewer bins. On the other hand, instances with very small items are often easier to solve as well, as the items can be packed densely into a smaller number of bins [11]. To identify which features are most relevant for predicting algorithm performance, a subset of features is selected, based on correlation analysis, ensuring that only those features that are truly predictive of performance are retained. Furthermore, clustering techniques is used to avoid redundancy and to ensure that the selected features offer complementary information about the difficulty of the instances. This process is conducted automatically in the MATILDA platform. Ultimately, this approach should lead to the identification of an optimal feature set that is capable of reliably predicting algorithm performance in various instances.

For the performance metric, the formula $\frac{BestBound(I)-NetCost(A)}{BestBound(I)}$ is used, where I represents a problem instance and A an algorithm. "Best bound" is the lower bound derived from the original GBPP model formulation by Baldi *et al.* [15], which was obtained by relaxing certain constraints and treating the problem as an aggregate KP. "Net cost" refers to the objective function value returned by all algorithms after packing the items into bins.

5.2. Results of the ISA

The performance distribution of the five algorithms in the instance space is summarised in Table 5. Instances of strong performance are marked by a high proportion of favourable outcomes, while poorer performance is less frequent and appears scattered rather than concentrated in specific regions. This dispersion suggests that no particular class of instances consistently challenges the algorithms. U-FFD and U-BFD perform comparably well, with 74% and 75.8% of instances respectively showing good performance. The GRASPm, LNSm, and GRASP_LNS algorithms show superior performance, with success rates ranging from 81.3% to 93.5%. The GRASP_LNS emerges as the most effective among all the evaluated methods.

Table 5: Performance distribution of each algorithm in respect of its percentage good performance, accuracy, precision, and recall for the data instances

Algorithm	% good performance	Accuracy	Precision	Recall
U-FFD	74%	59.3%	83.2%	56.4%
U-BFD	75.8%	60.7%	76%	70.4%
GRASPm	81.3%	44.9%	91.4%	42.2%
LNSm	91.8%	58.1%	93.4%	58.5%
GRASP_LNS	93.5%	93.5%	93.5%	100%

The performance of the different algorithms is explored in the instance space, and the SVM selector[1] embedded in MATILDA is used to recommend the most suitable algorithm for each region. As shown in Figure 2, the GRASPm algorithms are expected to perform well in the bottom-left region, which correspond to instances with low "min" values. That is, instances in this region contain small items that are easier to handle during a packing process. The LNSm algorithm also shows good performance in the bottom-left region; however, its performance spans a broader area in the instance space. This suggests that the LNSm algorithm is more versatile for various instance types.

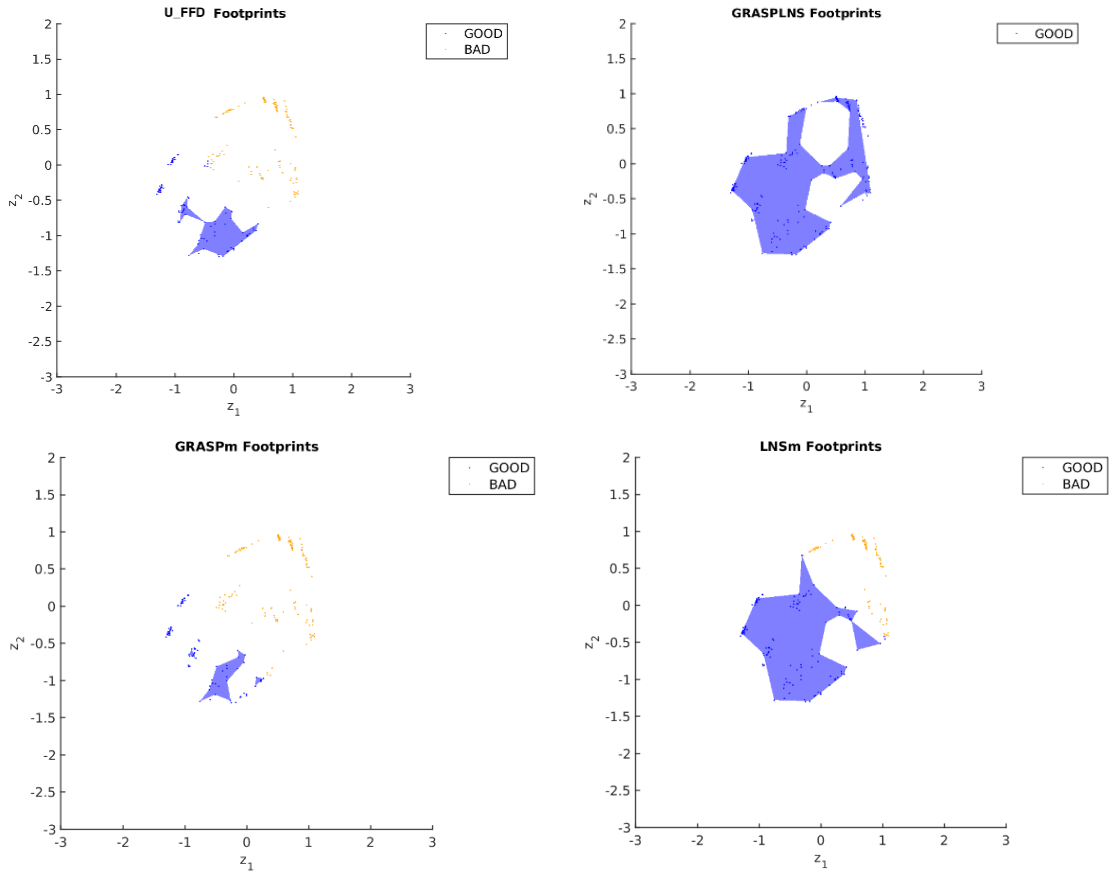

Figure 2: Performance predictions for the four algorithms in the instance space, with regions of expected strong performance highlighted in blue. The U_BFD footprint has been omitted, as its performance closely resembles that of the U_FFD footprint.

The GRASP_LNS algorithm demonstrates wide applicability, performing well in almost the entire instance space. This characteristic makes it the most flexible and consistently high-performing algorithm. The results suggest that the GRASP_LNS algorithm is the best choice for use in all regions in the instance space. This aligns with the previous observation that the algorithm performs well almost universally, showing high accuracy[2], precision[3], and recall[4] (see Table 5).

[1] The SVM tuning or support vector machine is a built-in function in MATILDA used for classification and regression tasks [14].
[2] Accuracy is the measure of instances that the SVM model correctly classified [14].
[3] The precision is a measure of how precise the SVM prediction is – thus, the number of true positives over true positives and false positives [14].
[4] Recall is the number of true positives over the number of true positives and false negatives [14].

6. CONCLUSION

This paper investigated the GBPP, which has significant applications in logistics. To address this problem, three enhanced heuristics were proposed, developed in response to a gap in the literature, such that few methods effectively challenged the widely used U-FFD and U-BFD algorithms. The newly proposed algorithms were designed to improve solution quality by integrating distinct strategies that build upon and refine the basic heuristics. Experimental results showed that the proposed algorithms achieved lower percentage gaps than the theoretical lower bound, compared with the benchmark methods. Among them, the GRASP_LNS algorithm consistently delivered strong performance in all instance classes. These findings represent a meaningful advancement in the state of the art for solving the GBPP.

A second key contribution of this work lies in the evaluation approach. Beyond traditional average-based performance metrics, algorithm behaviour was analysed using predictions in different regions of the instance space. This enabled a more nuanced understanding of the conditions under which each algorithm performs well or underperforms. The ISA framework was used for this purpose, revealing that GRASP_LNS achieved the best overall average performance. Moreover, the framework predicted that the LNSm algorithm would outperform others in a broader portion of the instance space, offering valuable insight for informed algorithm selection.

In future research, extending the proposed heuristics to accommodate real-world constraints commonly found in logistics, such as time windows, multi-dimensional bin packing, or stochastic demands, could enhance their applicability in practical settings. In addition, investigating the integration of the heuristics into decision support systems or logistics software platforms would be a valuable step towards bridging the gap between theoretical development and real-world implementation. Finally, the current experimental setup used a limited set of instances, which, while effective for baseline comparisons, may not have fully captured the complexity of real-world applications. Generating more representative and varied instances, especially those that reflect practical constraints and scenarios, would significantly enhance the applicability and robustness of the findings. A broader and richer instance space would also deepen the insights gained from the ISA, potentially uncovering new patterns in algorithm behaviour and allowing for more accurate performance predictions across diverse problem settings.

REFERENCES

[1] **A. Sara, A. G. Ramos, M. A. Carravilla, and J. F. Oliveira**, "On-line three-dimensional packing problems: A review of off-line and on-line solution approaches," *Computers & Industrial Engineering*, vol. 168, 108122, 2022.

[2] **A. Sara, A. G. Ramos, M. A. Carravilla, and J. F. Oliveira**, "Heuristics for online three-dimensional packing problems and algorithm selection framework for semi-online with full look-ahead," *Applied Soft Computing*, vol. 151, 111168, 2024.

[3] **C. Blum, V. Schmid, and L. Baumgartner**, "On solving the oriented two-dimensional bin packing problem under free guillotine cutting: Exploiting the power of probabilistic solution construction," *arXiv preprint arXiv:1209.0880*, 2012. [Online]. Available: https://arxiv.org/pdf/1209.0880.

[4] **C. Chen, S. Lee, and Q. S. Shen**, "An analytical model for the container loading problem," *European Journal of Operational Research*, vol. 80, no. 1, pp. 68-76, 1995.

[5] **D. H. Wolpert, and W. G. Macready**, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67-82, 1997.

[6] **D. Pisinger, and M. Sigurd**, "Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem," *INFORMS Journal on Computing*, vol. 19, no. 1, pp. 36-51, 2007.

[7] **G. de Abreu Rodrigues, C. B. Cunha, L. G. Neto, and J. G. V. Vieira**, "An adaptive large neighborhood search metaheuristic for the generalized bin packing problem with incompatible categories," *Computers & Industrial Engineering*, vol. 185, 109586, 2023.

[8] **J. Jooken, P. Leyman, and P. de Causmaecker**, "Features for the 0-1 knapsack problem based on inclusionwise maximal solutions," *European Journal of Operational Research*, vol. 311, no. 1, pp. 36-55, 2023.

[9] **J. N. Hooker**, "Needed: An empirical science of algorithms," *Operations Research*, vol. 42, no. 2, pp. 201-212, 1994.

[10] **J. R. Rice**, "The algorithm selection problem," *Advances in Computers*, vol. 15, pp. 65-118, 1976.

[11] **K. Liu**, "Using instance space analysis to study the bin packing problem," Master's dissertation, University of Melbourne, Melbourne, Australia, 2020.
[12] **K. Smith-Miles, J. Christiansen, and M. A. Muñoz**, "Revisiting *Where are the hard knapsack problems?* via instance space analysis," *Computers & Operations Research,* vol. 128, 105184, 2021.
[13] **K. Smith-Miles, and L. Lopes**, "Measuring instance difficulty for combinatorial optimization problems," *Computers & Operations Research,* vol. 39, no. 5, pp. 875-889, 2012.
[14] **MATILDA**, "Melbourne Algorithm Test Instance Library with Data Analytics," University of Melbourne, 2018. [Online]. Available: https://matilda.unimelb.edu.au/matilda/ [Accessed January 2024].
[15] **M. M. Baldi, T. G. Crainic, G. Perboli, and R. Tadei**, "The generalized bin packing problem," *Transportation Research Part E: Logistics and Transportation Review,* vol. 48, no. 6, pp. 1205-1220, 2012.
[16] **M. M. Baldi, D. Manerba, G. Perboli, and R. Tadei**, "A generalized bin packing problem for parcel delivery in last-mile logistics," *European Journal of Operational Research,* vol. 274, no. 3, pp. 990-999, 2019.
[17] **OR Group Polito**, "Data instances repository," December 2016. [Online]. Available: https://bitbucket.org/ORGroup/workspace/repositories/ [Accessed January 2024].
[18] **P. M. Castro, and J. F. Oliveira**, "Scheduling inspired models for two-dimensional packing problems," *European Journal of Operational Research,* vol. 215, no. 1, pp. 45-56, 2011.
[19] **P. Schwerin, and G. Wäscher**, "The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP," *International Transactions in Operational Research,* vol. 4, no. 5-6, pp. 377-389, 1997.
[20] **R. Ding, B. Deng, and W. Li**, "Meta-heuristic algorithms for the generalized extensible bin packing problem with overload cost," *IEEE Access,* vol. 10, pp. 124858-124873, 2022.
[21] **T. G. Crainic, F. D. Fomeni, and W. Rei**, "Multi-period bin packing model and effective constructive heuristics for corridor-based logistics capacity planning," *Computers & Operations Research,* vol. 132, 105308, 2021.