processes

MDPI

Article

A Software Toolbox for Realistic Dataset Generation for Testing Online and Offline 3D Bin Packing Algorithms

Luis Ribeiro *,† **and Anan Ashrabi Ananno** †

Department of Management and Engineering, Linköping University, 58183 Linköping, Sweden; anan.ashrabi.ananno@liu.se

* Correspondence: luis.ribeiro@liu.se

† These authors contributed equally to this work.

Abstract: Packing products into a pallet or other medium is an unavoidable activity for producing companies. In many cases, packing is based on operator experience and training using packing patterns that have worked before. Automated packing, on the other hand, requires a systematic procedure for devising packing solutions. In the scientific literature, this problem is known as 3D bin packing (3DBP) and many authors have proposed exact and heuristic solutions for many variations of the problem. There is, however, a lack of datasets that can be used to test and validate such solutions. Many of the available datasets use randomly generated products with extremely limited connection to real practice. Furthermore, they contain a reduced number of product configurations and ignore that packing relates to customers' orders, which have specific relative mixes of products. This paper proposes a software toolbox for generating arbitrarily large datasets for 3DBPP based on real industry data. The toolbox was developed in connection with the analysis of a real dataset from the food and beverages sector, which enabled the creation of several synthetic datasets. The toolbox and the synthetic datasets are publicly available and can be used to generate additional data for testing and validating 3DBP solutions. The industry is increasingly becoming data dependent and driven. The ability to generate good quality synthetic data to support the development of solutions to real industry problems is of extreme importance. This work is a step in that direction in a domain where open data are scarce.

Keywords: 3DBPP; toolbox; dataset; problem instance; online 3DBPP; offline 3DBPP

Citation: Ribeiro, L.; Ananno, A.A. A Software Toolbox for Realistic Dataset Generation for Testing Online and Offline 3D Bin Packing Algorithms. *Processes* **2023**, *11*, 1909. https://doi.org/10.3390/pr11071909

Received: 7 June 2023
Revised: 19 June 2023
Accepted: 23 June 2023
Published: 25 June 2023

1. Introduction

Packing is an activity that is inherent to any production company. Depending on the activity sector, the products to be packed will vary along many different dimensions, for example, geometry, weight, load capacity, composition, etc. Depending on the heterogeneity of an order with respect to the types and quantities of products to be packed, the solutions for the packing problem can be straightforward or complex but feasible or directly infeasible. This has motivated many companies to use specialized human operators when packing highly heterogeneous orders. Other companies have normalized packaging sizes so that their dimensions correlate in a way that facilitates packing. General and deterministic solutions for the 3D bin packing problem (3DBPP) are not known. The problem is known to be NP hard in its simplest form, which disregards practical industrial constraints, such as load limits, container stability, incompatibility of items, etc.

The availability of realistic datasets is of paramount importance for the scientific community to be able to provide industry and practitioners with high-quality solutions for 3DBPP. However, there is a scarcity of such datasets online, and many of the available datasets are not necessarily grounded in real industry data. In this context, it is also important to note that today's industrial systems are increasingly reliant on data for their operations. However, in many cases, real data can only be created at an extremely low

pace. Alternatively, certain systems create vast amounts of data, but access to said data is normally restricted. This renders providing data-based solutions for complex industrial problems a real challenge. With this in mind, this paper attempts to support the research community by providing a methodology, a set of digital tools and two exemplary datasets to be used for developing solutions for the 3DBPP.

Considering the obvious economic and industrial impact of 3DBPP, over the years, numerous exact and approximation algorithms have been developed to provide an exact or approximation solution [1]. During the development process, researchers opt to benchmark the performance of their current 3DBPP solutions with those of the existing state-of-the-art algorithms [2]. However, to the best of the authors' knowledge, there are no standard datasets or problem sets that are universally accepted for benchmarking and testing 3DBPP algorithms' performance.

Instead, in the literature, there are a myriad of different datasets (see Tables 1 and 2), some of which are specifically designed for special kinds of 3DBPP (e.g., open dimension problem, and uniform bin packing) [3–5]. Whether or not a current 3DBPP algorithm is state of the art cannot always be evaluated solely based on its performance on virtually generated data, which does not reflect the trends and practical requirements of an industry. As the state-of-the-art literature uses non-standard problem sets to test their solutions, the underlying numerical experiments might not be comparable to others [2].

This paper reviews and characterizes 19 datasets that are referenced in the literature both for the offline (8 datasets) and online (11 datasets) variations of the 3DBPP. A summary of these findings can be found in Tables 1 and 2. Online datasets are fundamentally different from offline datasets due to how knowledge is transferred to the 3DBPP algorithm. In offline 3DBPP, the knowledge of the entire dataset is available to the algorithm at the beginning of the optimization process [6]. The online 3DBPP algorithms cannot access the knowledge of the entire dataset immediately, but instead, each data point is fed, one at a time, to the algorithm in an arbitrary sequence [7]. It is therefore, important for an ideal dataset to support both modes of operation.

Both online and offline datasets were analyzed in light of several parameters: application, number of instances in the dataset, the presence of specific or re-configurable features in the instances' items, the distribution of the different item/package sizes (when applicable), and the heterogeneity of the items/packages as a function of their dimensions. Table 1 summarizes the analysis for offline datasets.

One of the still most widely used datasets for offline 3DBPP problem was developed by Bischoff and Ratcliff [8,9] and is commonly denoted in the literature as BR0–BR15 as a means to specifically refer to each of the 16 instances of the dataset. All instances are defined for single container applications and contain 100 items each. Single container means, in this context, that the different instances are meant to be packed in only one container with fixed dimensions. The dataset also contains information regarding the allowed orientations for each box within an instance. It is not possible to configure the distribution of box types (i.e., dimensions) on the problem instances. The heterogeneity of the instances with respect to the relative dimensions of the items ranges from weakly to strongly heterogeneous.

Other datasets specifically target multiple container applications [10–13]. In these scenarios, the container size may be fixed or variable, but the items to be packed are bounded by the volumes of the containers. Examples of multi-container datasets include [13–15]. In [13], the authors discuss a dataset with 47 instances, each of which has between 47 and 180 items. The distribution of item types is fixed and restricted to between two and five possible types. There are no specific constraints with respect to the possible orientations of the packages in the instances. The instances are weakly heterogeneous. The multi-container dataset discussed in [14] contains 385 instances, and each has between 1 and 10 items. Users can control the heterogeneity of package types by varying the distribution among nine different types, but the orientation of certain packages is predefined [16,17]. The data result in strongly heterogeneous instances. The dataset discussed in [15], with an example application in [10,12,15] also targets multi-container applications. The 16 available instances of this dataset consist of between 70 and 175 packages

that are combinations of 2 to 6 package types without any specific orientation constraints. The dataset is weakly heterogeneous.

Table 1. Comparison of different test problem data sets for different types of offline bin packing problems.

Dataset Source	Current Application	Number of Instances	Additional Constraints	Distribution of Box Type	Instance Heterogeneity
Bischoff and Ratcliff (BR1–BR7) [8]	Single container	7 instances each with 100 items	Specific orientations are allowed	Pre-defined, fixed	Weakly heterogeneous
Davies and Bischoff (BR0, BR8-BR15) [9]	Single container	9 instances each with 100 items	Specific orientations are allowed	Pre-defined, fixed	Strongly heterogeneous
Ivancic et al. [13]	Multiple identical containers	47 instances each has between 47 and 180 boxes, 2–5 box types	None	Fixed	Weakly heterogeneous
Martello et al. [14]	Multiple identical containers	385 instances each has between 1 and 10 boxes, 9 box types	Specific orientations are allowed	User-defined	Strongly heterogeneous
Mohanty et al. [15]	Multiple weakly/strongly heterogeneous containers	16 instances each has between 70 and 175 boxes, 2 and 6 box types	None	Fixed	Weakly heterogeneous
Bortfeldt and Gehring [18]	Single container ODP	10 instances each has between 127 and 140 boxes, 3 and 50 box types	Specific orientations are allowed	Uniform	Strongly/weakly heterogeneous
Bortfeldt and Mack [4]	Single/multiple ODP	10 instances each with 1000 boxes, 3–50 box types	Specific orientations are allowed	Uniform	Weakly heterogeneous
Bosch Group (mentioned in [19])	Single/Multiple container ODP	14 instances each has between 90 and 1145 boxes, 5 and 29 box types	Weight limit for vertical stability	None	Strongly/weakly heterogeneous

Open dimension problem (ODP) is a class of 3DBPP, where items need to be packed into single/multiple containers with one or more variable dimensions such that the container volume is minimized. Bortfeldt and Gehring et al. created a dataset with 10 instances capable of benchmarking algorithms for single container ODP [3,18]. Each instance of this dataset consists of between 127 and 140 boxes, distributed uniformly among 3–50 item types. As such, this dataset contains both strongly and weakly heterogeneous product types. Consequently, Bortfeldt and Mack et al. modified the previous dataset (Bortfeldt and Gehring et al.) with 1000 boxes per instance. This dataset is designed to test solutions for multi-container ODP. However, the modified dataset still uses 10 instances with a uniform distribution of boxes among 3–50 package types. This essentially populates the dataset with weakly heterogeneous products [3,4]. Both of these ODP datasets provide information on the allowed orientations for each box.

Although these virtually generated datasets can demonstrate an algorithm's effectiveness for improving objective functions over time, they have some drawbacks. The limited number of built-in constraints (e.g., orientation and weight limit constraint) that an algorithm has to satisfy while in operation does not reflect the realistic packing constraints (e.g., customer positioning constraints, loading priority and stacking constraint) [2]. Additionally, the products and pallet sizes do not conform to any industry standard. The majority of the datasets are static and do not allow user modification or scaling. To overcome some of these limitations, in 2023, Chen et al. benchmarked their 3DBPP algorithm on a real-world-based dataset provided by the Bosch Group [19]. All 14 instances of this dataset are generated

based on 5–29 real-world scenarios and product information. The number of items per instance can range between 90 and 1145, allowing a good balance between strongly and weakly heterogeneous instances.

So, in general, offline datasets tend to contain a fixed set of problem instances of different degrees of heterogeneity. With the exception of the work reported in [19], which used industry data, the distribution of package types and dimensions appears to have been tuned by the authors based on their own experiences and benchmarking needs. Another characteristic of the analyzed datasets is that their usage has been reported in other works, but they do not appear to be freely available as an online resource, with the exception of BR1-BR7 and the work of Ivancic et al., which are available at http://people.brunel.ac.uk/ mastjjb/jeb/orlib/thpackinfo.html, accessed on 24 May 2023.

Unlike offline datasets, online datasets tend to rely on generating functions or processes that create packages to be fed progressively to the 3DBPP algorithms under testing. Online problems are usually solved under unbounded conditions, where an unlimited number of containers are available for packing items (see Table 2). However, in the real world, due to logistical reasons, a limited number of containers are available. Commonly used online datasets are based on variable-sized datasets, random sequences or random sampling, which neglects the limited availability of containers [6]. Recently, Zhao et al. generated a dataset with 64 instances addressing this issue [20,21]. A large body of research uses the guillotine cut method to generate the problem instances [22–24]. Since, in this method, the optimal solution for each problem instance is known, these datasets are widely used to test the packing efficiency of online algorithms. However, they are also prone to randomness and do not resemble the geometric dimensions of real-world products [6]. As such, machine learning models, a quite popular way of addressing online problems, trained on virtual datasets can become quite impractical in real industrial applications.

In [25,26] an event-driven method for creating packages is proposed that attempts to emulate real-world online packing scenarios. The idea is to guarantee that the generated items change, progressively or abruptly, over time according to a certain probability of change. The goal is to enable algorithms to be tested dynamically against changing inputs. Another attempt to mimic real-world conditions is reported in [27], where a generator function samples from a real package distribution from a specific company. Another approach to realistic scenario creation is reported in [28], where packages are sampled from seven real package sizes used by DHL.

In order to test solutions for online bin packing problems with uniform distribution, specific problem sets are proposed by Asta et al. and Ender et al. [29,30]. The generator functions for these datasets were used by several other studies for benchmarking state-of-the-art 3DBPP online algorithms [5,29,31].

Real 3DBP problems are inherently multi-objective optimization problems. However, according to Ali et al., a limited number of studies have used a multi-objective optimization approach for solving 3DBPPs [6]. Therefore, most datasets are designed with single-objective problems in mind, and they are either tuned to evaluate the maximization of volume utilization or weight. Even in instances where more package information is provided, most of the data are still drawn from one or several fictitious distributions. An obvious problem is that, in some cases, the weights of the items are not proportional to their volumes [2]. The other important pitfall of the existing datasets is that products are not randomly ordered. There is a certain coherence in real product orders, which is then implicitly reflected in the relative dimensions of products that are usually packed together and affect the heterogeneity of the packing problem.

In order to overcome the drawbacks of the existing datasets, this study developed a software toolbox to generate synthetic datasets based on real-world product information for testing online and offline 3DBPP algorithms. The toolbox was designed based on an analysis of order information from the food and beverages industry. Together with the toolbox, two datasets of different sizes are also distributed. These datasets serve a dual function: they can be used directly and they can be used with the toolbox to create additional data.

The generated data can be served to an algorithm all at once (offline mode) or product by product using a generator function (online mode). The preliminary analysis of the results suggests that the toolbox generates datasets that capture with good fidelity: the distribution and composition of the orders (as a function of product types and their relative quantities), the dimension and weights of the packages (which are real-world data) and that hold these properties irrespective of the size of the generated dataset.

Table 2. Comparison of different test problem data sets for different types of online bin packing problems.

Dataset Source	Type of Bin Packing Problem	Distinct Product Types	Additional Constraints	Distribution of Box Type	Instance Heterogeneity
Zhao et al. [21]	Single Container	64	Order dependence, vertical stability	Pre-defined	Strongly/weakly heterogeneous
Variable-sized item (VSI) [32]	Single Container	Variable	Specific orientations are allowed	Random	Strongly/weakly heterogeneous
Discreet dataset [33]	Multiple identical containers	3 different settings for unlimited instances	Specific orientations are allowed, robot manipulation constraint	Pre-defined	Strongly heterogeneous
Guillotine cut [22–24]	Single/multiple containers	Variable	Order dependence, specific orientations are allowed	Random	Strongly/weakly heterogeneous
Event simulation based [25,26]	Single/multiple containers	20–1000	Specific orientations are allowed	Pre-defined, fixed	Weakly heterogeneous
Random sequence based [34–36]	Multiple containers	Variable	None	Random/uniform	Strongly/weakly heterogeneous
Random sampling based [37]	Multiple containers	100	Full support constraint	Random	Strongly/weakly heterogeneous
Large-scale real world transaction order dataset (LRTOD) [27]	Single/multiple containers	-	Specific orientations are allowed	None	Strongly/weakly heterogeneous
DHL dataset [28]	Single container	7	Specific orientations are allowed	Random	Weakly heterogeneous
Asta et al. [29]	Uniform bin packing	100	None	Pre-defined, uniform	Weakly heterogeneous
Ender et al. [30]	Uniform bin packing	Variable	None	User-defined, uniform	Weakly heterogeneous

The structure of the paper is as follows: Section 2 describes the methodological approach for developing and using the software toolbox. Section 3 presents the results related to the functionality and accuracy of the toolbox. Sections 4 and 5 discuss the capabilities of the toolbox in light of the results and summarize the findings, respectively.

2. Materials and Methods

As mentioned before, the purpose of this paper is to document a software toolbox that can be used to generate arbitrarily large datasets that mimic real-world orders in product diversity, quantity and properties. Such synthetically generated orders correlate moderately with the orders placed to a relatively large facility in the food and beverages domain. The toolbox requires a base representative dataset, which is used along a specific workflow towards generating new datasets. The workflow is, therefore, applicable to other datasets containing the same data fields. This allows researchers to create and provide their own datasets based on other industry examples and domains.

A base dataset, that would be compatible with the toolbox, is organized as follows. The dataset must be provided as CSV file. The header of the CSV file should have the following categories:

- Order—The identifier of the order. The value should be an integer number. All the rows with the same order identifier are part of the same order.
- Product—An order includes several products. Product is the product identifier. A product identifier should not repeat within an order (i.e., data rows with the same order may not have repeating values for the same Product). The value of the product should be an integer number.
- Quantity—Represents the amount of each product in an order. It should be an integer number.
- Length—Is an integer representing the length of one individual product in millimeters.
- Width—Is an integer representing the width of one individual product in millimeters.
- Height—Is an integer representing the height of one individual product in millimeters.
- Weight—Is a floating point number representing the weight of one individual product in kilograms.

Floating point numbers should use "." as the decimal separator. An example of a valid input set of values for a CSV file is detailed in Table 3.

Table 3. Example of valid values in the input CSV file.

Order	Product	Quantity (Units of Product)	Length (mm)	Width (mm)	Height (mm)	Weight (kg)
92839	30367	2	300	160	216	8.953
92839	66212	1	385	160	215	10.727
92839	67805	36	225	225	180	5.186
92839	53347	1	244	164	274	5.811
92839	17310	10	160	160	111	1.781
33920	42882	4	244	164	274	5.811
33920	30920	3	225	225	180	5.19

The values in Table 3 are interpreted as follows. There are two orders in the sample (orders 92839 and 33920). Order 33920 has two product types on it (42882 and 30920). There are four items of product 42882 and three items of 30920. The corresponding characteristics for one individual item are then detailed. For example, items of product type 42882 have length 244 mm, width 164 mm, height 274 mm and weight 5.811 kg.

With a valid base dataset, the toolbox will apply the following workflow:

1. Load the dataset.
2. Cluster the data.
3. Calculate the cluster distribution (i.e., what is the frequency of orders belonging to each cluster in the dataset).
4. Sample the clusters and generate representative data points from each sampled cluster.

The procedure above, also documented in Figure 1, was used to generate two synthetic datasets based on a real industry dataset. The original dataset included around 3000 orders from a product assortment of about 200 different product types. The two generated datasets can be used directly to test and validate existing algorithms, but they can also be used to generate new data. Dataset1000.CSV contains 1000 orders and Dataset10000.CSV contains 10,000 orders. Both the toolbox and the datasets are available in https://github.com/luferi/3DBPP, accessed on 22 June 2023.

In order to properly evaluate the results, it is important to clarify steps 2 to 4 of the procedure detailed before. There are many different possible strategies for data clustering. The analysis of the original dataset was a determinant in deciding on the best approach. Figure 2 shows a plot of the orders on the original dataset as a function of the number of items in each order as well as the number of different products in them. It also shows an

entropy measurement of the order. The entropy of an order was calculated in the following way: the dataset was grouped by order and product, and the quantity values were summed for each group. The Shannon index was then calculated and normalized for each order. The function *entropy* from the *python scipy.stats._entropy* package was used for calculating the index. The normalization of values between 0 and 1 was carried out by subtracting the minimum entropy value from the calculated index and then dividing by the difference between the maximum and minimum entropy values. The higher the entropy, the higher the uncertainty in the distribution of the number of items per product type. Figure 2 therefore shows that the composition of the order is not straightforwardly predictable. This fact has been confirmed by the company providing the dataset.

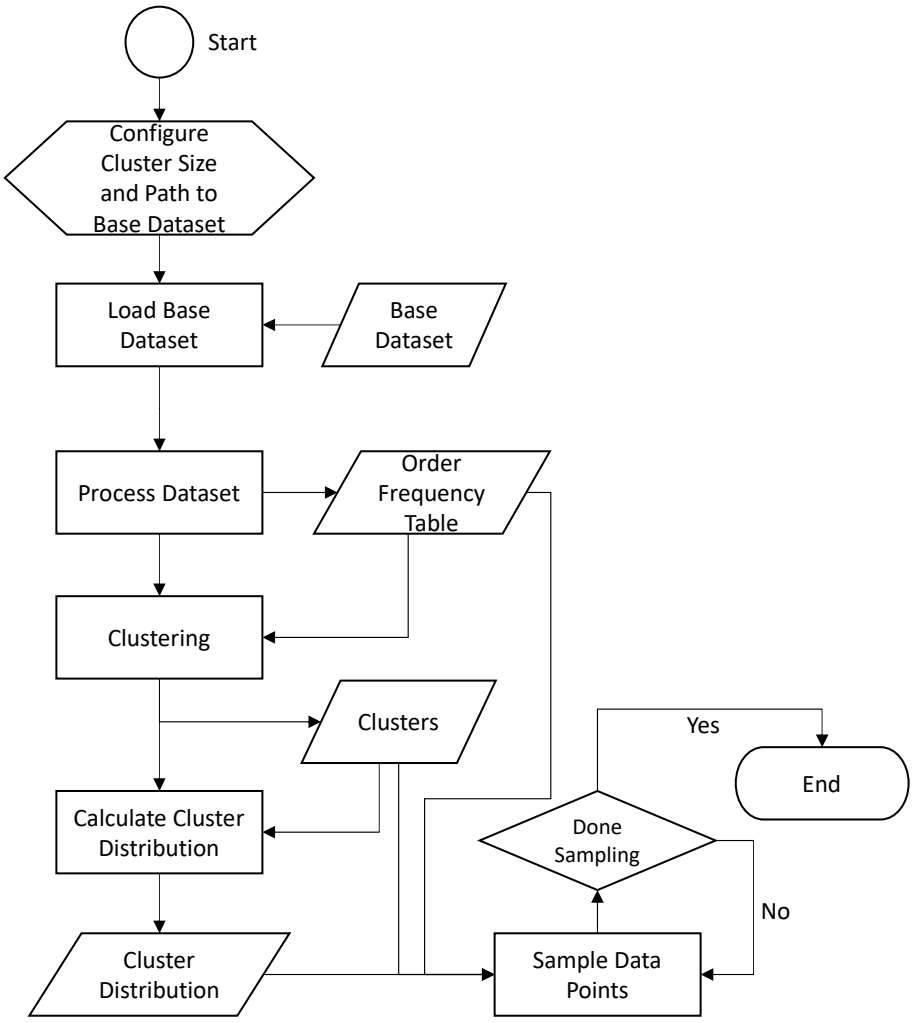


Figure 1. Flowchart diagram for the dataset generation procedure.

This means that orders with many items and orders with fewer items may range from being very homogeneous or very heterogeneous with respect to the variety of product types. With that mentioned, it is also important to note that, generally, orders with a higher number of items will also tend to have a more homogeneous distribution of item quantity for a product type. From a data generation perspective, this means that the number of items alone is not a good predictor of product variety and vice versa.

Another pertinent observation is that there is oftentimes a certain coherence in the product types in an order (i.e., certain types of products are often ordered together). The relative mix of these products varies though. The challenge then becomes capturing such coherence in a reasonable way.

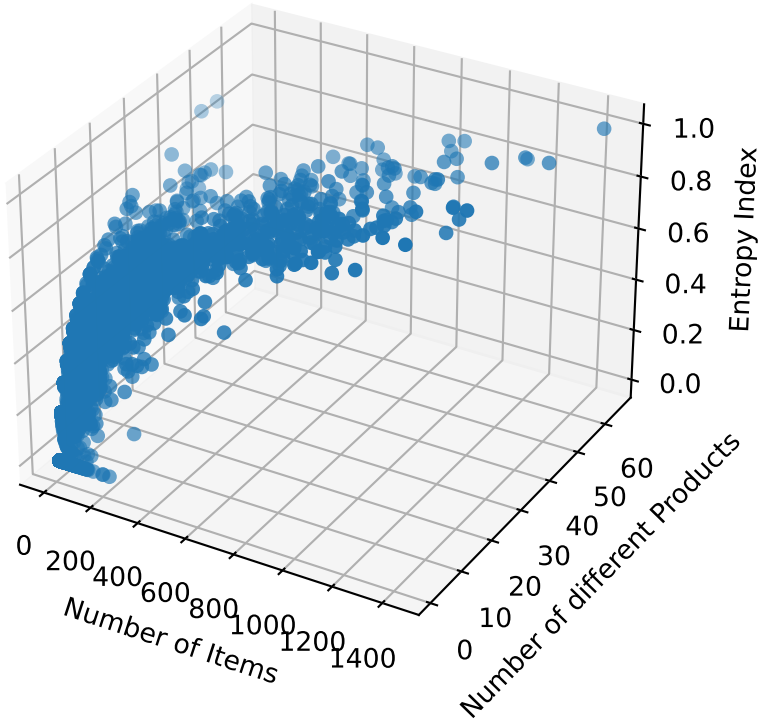


Figure 2. Order composition and entropy on the original dataset.

The procedure used for clustering consists of the following:

1. Generate a frequency table with the product counts per order.
2. Evaluate the cosine similarity between the different orders of the frequency table.
3. Interpret the cosine similarity matrix as an adjacency matrix in an undirected graph.
4. Cluster the data by using a network community detection algorithm.

The cosine similarity measurement evaluates the angle between two data vectors. One row in the frequency table can be interpreted as a data vector with a dimension equal to the total number of products in the assortment. Measuring cosine similarity has, in this case, the advantage that it does not depend on the magnitude of the vectors. It is, therefore, a good way to capture the coherence of the orders as a function of the product types on them but not necessarily their quantities. The cosine similarity measurement generates an n-by-n matrix, where n is the number of orders in the sample. The matrix then details how similar one order is to all the other orders.

One takes advantage of that matrix by interpreting it as the adjacency matrix of a graph, or a network, and then analyzing it using the Louvain community detection algorithm. Large communities are re-evaluated. The threshold for what defines a large community is configurable. In the original dataset analysis, that number was experimentally and empirically restricted to 100 orders per community/cluster.

To generate a set of representative data points, the cluster distribution is calculated. This is achieved by counting how many orders belong to each cluster. Sampling from a cluster is then performed probabilistically considering the frequency of orders in each cluster. One of the reasons why the size of the clusters must be controlled is that if their size is disproportionate, the resolution on the nature of the order when sampling will be lost.

When a cluster is selected, the next step is to evaluate the distribution of product types in that cluster and sample from that distribution. However, when sampling from the clusters, only one cluster is selected. When sampling from the product types, a slightly different approach is considered. Product types are n-times randomly selected, and their likelihood of selection is proportional to their occurrence in the cluster (i.e., products whose type occurs more frequently in the cluster are more likely to be selected). Note that this sampling procedure does not take into account the number of items of the same type in

the cluster. The reasons for that are that the quantity is not an indicator of the frequency of the order (as Figure 2 shows) and quite a few products ordered in a low quantity actually take a lot of volume, or otherwise are already a large aggregated product. To clarify with a practical example, it is possible to order 20 bottles of juice (quantity = 20), or a half-pallet of juice (quantity = 1). The latter obviously contains a lot more volume and units of bottles of juice but exists in the dataset as a single unit.

After the types of products are sampled, then the quantity of each product type is sampled from the quantity distribution for that type. This final step guarantees that examples such as the one mentioned before for the bottles of juice are preserved in the generated data. That is important because it directly affects the composition of the orders and their packing characteristics.

There are two final pertinent observations about the generated dataset and the process for generating them.

The first is that the provided datasets do not make any assumption about the bin size. For reference, the base dataset used to create the published datasets considered standard euro pallets with length 1200 mm, width 800 mm and a maximum palletizing height of 1400 mm. However, companies may change their bin format while keeping package sizes constant and vice versa

The second is that the provided datasets do not include any information about what constitutes an optimal palletization sequence and positioning for the products in the orders. The reason for this is manifold. Different companies will use specific parameters for evaluating the quality of a pallet. Classic scientific metrics, such as the minimization of envelope volume, minimization of the number of bins, minimization of bin volume, etc., actually have relatively low importance for many practical applications. Other constraints, such as products of the same type being packed together, expensive products being packed in the center, the creation of interlocking layers that improve the stability of the bin/pallet when being displaced, the need to include additional load carriers between certain products, etc., are much more important in real-world applications. The goal with the provided datasets is that whoever uses them must clearly contextualize the application area and indicate which palletization quality metrics were considered. Any order can be palletized/packed, but quality indicators will vary greatly. Dataset users will subsequently establish the benchmark for their specific application scenario.

With this procedure in mind, the paper will now concentrate on describing the toolbox and analyzing the results obtained.

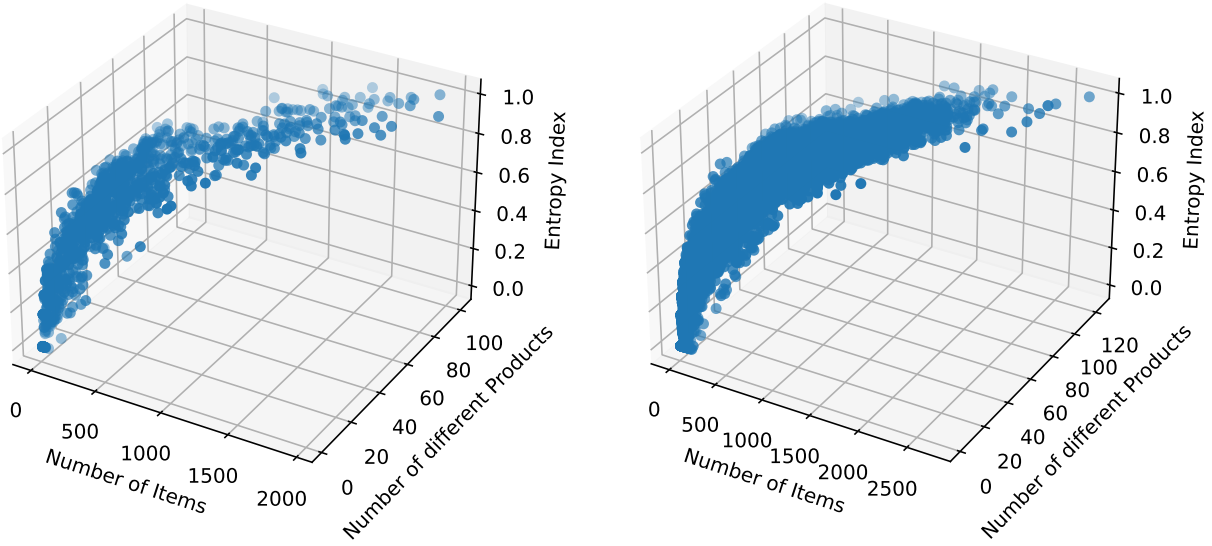
3. Results

The analysis of the created and provided datasets (Dataset1000 and Dataset10000) is a good starting point to evaluate if the toolbox can be successfully used to create realistic datasets. Figure 3 shows the generated orders as a function of the number of items and diversity of product types. The entropy index is also calculated.

Inspection and analysis of Figures 2 and 3a,b show that the generated datasets largely succeed in keeping the profile of the original dataset, both for small samples (Figure 3a) and larger samples (Figure 3b). This means that the toolbox is generating samples that are both high and low volume and high and low variety. The analysis of the entropy index also suggests that the relative quantities of the products are being correctly generated.

There are also a few additional effects to be discussed. The number of items will tend to increase, as well as the number of different products. This is a consequence of the sampling procedure used, where the product types featured in an order are randomly selected from the set of all product types that occur in the cluster and on their frequency of occurrence. This means that there is a probability, even if very low, that all products existing in a cluster would be present in a created order. There is also a probability that products that existed in just a few orders in the cluster, even if they were not the defining products of the cluster, start to show up in more orders. The probability of such an effect

increases with the size of the cluster. These effects are visible in the shapes of Figure 3a,b, which have a smoother envelopethan the original dataset Figure 2.



(**a**) (**b**)

Figure 3. Order composition and entropy on the generated datasets. (**a**) Dataset1000. (**b**) Dataset10000.

The authors evaluated two more effects that result from the usage of the toolboxes and base dataset. The first is the impact of creating datasets with a relatively low number of orders. The second is the effect of creating additional datasets from copies of other datasets. The effects of both are reflected in Figure 4.

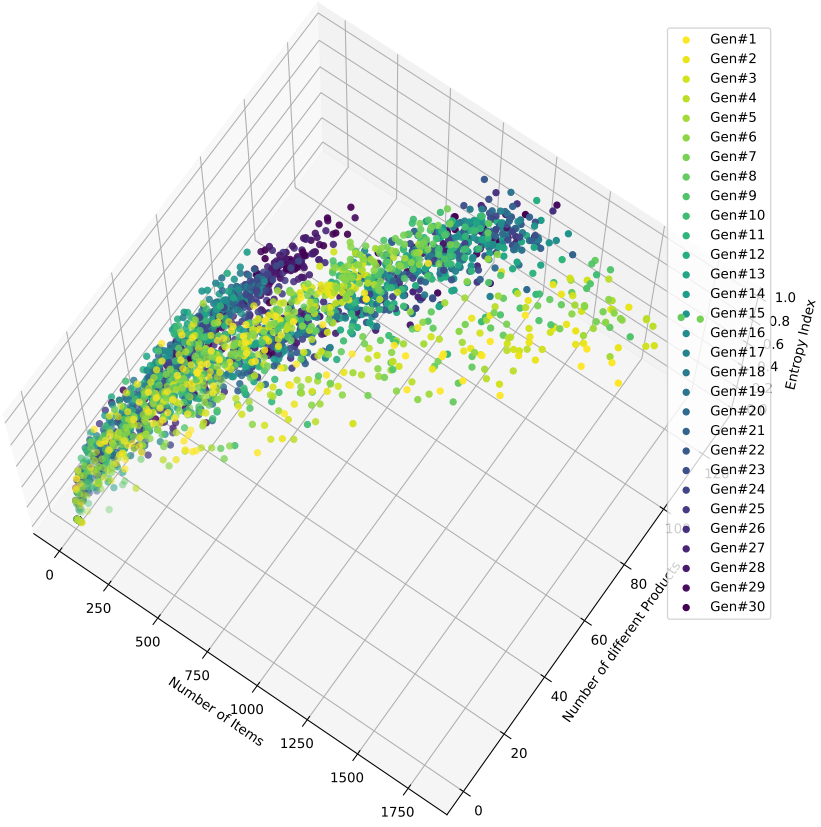


Figure 4. Copies of small datasets.

Figure 4 shows the effects of generating datasets from successive copies. In this case, the Dataset1000 is first used to create a new dataset with 100 orders on it. This new dataset is represented as Gen1. That dataset is then used to generate dataset Gen2. The procedure is repeated until Gen30. The objective here is to analyze how smaller datasets and successive copies lose resolution with respect to the original dataset.

The observations suggest that the generating procedure does not lose products or their relative distribution for samples of size 100. Comparing Figure 4 with Figure 3a shows that the ranges in number of items and number of different products are kept. However, generating datasets from successive copies of smaller datasets has, after a few iterations of copies, effects on the profile of the orders. Figure 4 shows that the latest copies tend to have fewer items and less variety of products.

These results will, in the next section, be used as a basis to discuss the toolbox and its usage.

4. Discussion

The paper now concentrates on the discussion of the software toolbox and the definition of adequate use cases considering the configurable parameters of the toolbox. Before discussing the toolbox, it is worth mentioning that the generated datasets (Dataset1000 and Dataset10000) are immediately available for usage and do not require the toolbox. The toolbox provides additional facility functions to generate additional data points or datasets with specific characteristics that are not found in the existing datasets. The toolbox also supports the analysis of entirely new datasets from other domains as long as the datasets conform to the format described in Section 2.

The toolbox exists as a set of functions codified in the Python programming language. The main function signatures and their role are defined in Table 4

Table 4. Brief description of the toolbox API.

Function Signature	Description
main_workflow(input_file_path, num_samples, output_file_path)	This function executes the main workflow described in Section 2 and in Figure 1. It takes as input the file path to the base dataset as input, the number of order samples to be generated and the path to the output file where the newly created dataset will be stored.
cosine_similarity_community_clustering(freq_table, threshold=100)	This function takes as input a frequency table of the orders in the original dataset and clusters the dataset by applying the Louvain community detection algorithm (this corresponds to the clustering process in Figure 1). The threshold input regulates the maximum size of a community. Communities above that size are recursively re-evaluated into smaller communities. The function returns a dictionary of all the communities identified.
cluster_distribution(communities_flat, total_data_points)	This function computes the distribution of orders per cluster, based on the frequency table and total amount of data points in the sample (this corresponds to the calculate cluster distribution process in Figure 1).
generate_representative_data_point(b_dict)	This function generates a representative data point from a preselected cluster. The cluster information needs to be provided in the form of a dictionary that can be retrieved by calling the function **get_b_dict** (see code repository) that will convert a cluster to the appropriate format.
sample_generator(freq_table, communities_flat, cluster_dist)	This function provides a generator for continuously sampling point from a dataset on demand.

Table 4 is not meant to be an exhaustive description of the API, which can be found in https://github.com/luferi/3DBPP, accessed on 22 June 2023. The idea is to discuss how the

API looks and can be used to generate data. Furthermore, Table 4 complements the diagram in Figure 1 with the purpose of helping prospective users navigate the toolbox code.

In the previous context, calling the main_work_flow function results in a complete dataset being created. That function iteratively calls the functions that sample the clusters according to the computed distribution and subsequently generate a representative data point. With this in mind, it is possible to create an entire dataset. However, other calling patterns are also possible. An example of using the toolbox function as a Python generator is provided in the distributed code. Such code samples make use of the function **sample_generator** described in Table 4.

So, from a toolbox perspective, two modes of operation are possible: the generation of complete datasets or the progressive generation of data points.

The generated data allow a high degree of flexibility in the usage of the data. For example, in the online mode, users can opt to use a generator function, such as the one provided by the code in the code repository, and receive randomly sampled data points, or they may enforce a specific delivery order, by re-grouping or re-ordering the provided dataset according to specific criteria and retrieving orders sequentially. A combined strategy is also possible, whereby the users train and benchmark their algorithms using a specific order sequence from the provided datasets and then validate the order from the generator. The toolbox does not prescribe any specific usage.

In any case, the base dataset analysis needs to be completed before any of the procedures above can be executed.

In the context above, the main parameter that must be considered is the maximum size of the communities to be considered when calling the **cosine_similarity_community_clustering**. There are a few important considerations for setting this parameter that may affect the nature of the generated datasets or points. These considerations follow from the analysis in Section 3. Allowing clusters that are too large may reduce the capacity of the toolbox to create samples with proper diversity. Allowing considerably small clusters may lead to the creation of orders that are not truly representative of real industry data. The algorithm used for creating the clusters does not allow limiting their minimum size or number. The algorithm only allows limiting the maximum size.

When using the toolbox with a new dataset, the parameter needs to be adjusted to the characteristics of the dataset. The recommended approach is to start without any limitations and evaluate the sizes of the largest clusters. If the largest clusters are in fact representative of the dataset, for example, due to the existence of an overly representative set of very similar configurations, then no further adjustment is required. However, if the larger clusters do not provide a good mapping for the dataset, then progressively reducing the cluster size is the preferred approach. Following such a procedure led the authors to the results documented.

5. Conclusions

In this paper, the authors conducted a detailed literature review on the most well-established datasets used for benchmarking the performance of 3DBPP algorithms. The literature review encompassed both online and offline datasets and evaluated their effectiveness along with limitations. A software toolbox was proposed to overcome some of the identified drawbacks of the existing datasets. The results show that the toolbox can generate good quality synthetic datasets based on real-world product information and practical constraints. Experience with using the toolbox also suggests that the proposed methodology may scale well to other industry domains with similar data; however, the functions within the toolbox may have to be parameterized according to the domain. If correctly parameterized, these functions showed a good capacity to identify small but relevant regularities in the data. This capacity is key to the creation of realistic synthetic datasets. The toolbox is, as described in this paper, restricted to the analysis of a few data dimensions: product geometric dimensions and weight. In real application scenarios, other product features may be relevant (e.g., load capacity, surface friction, and custom packing

restriction). Both the methodology and toolbox can be very easily adapted to include these additional features. However, the previous will require re-programming of the toolbox, which today operates on the dimensions specified before. The previous was also the main motivation for making the toolbox and the dataset publicly available. This work was developed under the umbrella of a large research project studying the practical applications and constraints of 3DBPP in industry. The results of the toolbox will be incorporated into the development of advanced 3DBPP solutions in the food and beverages industry. In the process of generating data for different 3DBPP solutions, the authors anticipate that the toolbox will continue to be further developed. An obvious direction in this development is to add not only the order information but also the benchmark solution when packing the order in the provided datasets.

Author Contributions: Conceptualization, L.R. and A.A.A.; methodology, L.R. and A.A.A.; software, L.R.; validation, L.R. and A.A.A.; formal analysis, L.R. and A.A.A.; investigation, L.R. and A.A.A.; resources, L.R.; data curation, L.R. and A.A.A.; writing—original draft preparation, L.R. and A.A.A.; writing—review and editing, L.R. and A.A.A.; supervision, L.R.; project administration, L.R.; funding acquisition, L.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Strategic Innovation Program Produktion2030, a joint venture between Vinnova, Formas and the Swedish Energy Agency under the scope of the project entitled "Hållbar och flexibel automatisering av säsongsproduktion genom dynamisk resurshantering (2021-01283)". The APC was waived by MDPI.

Data Availability Statement: The data and the software toolbox reported in this paper are available at https://github.com/luferi/3DBPP, accessed on 22 June 2023.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Wang, F.; Hauser, K. Robot packing with known items and nondeterministic arrival order. *IEEE Trans. Autom. Sci. Eng.* **2020**, *18*, 1901–1915. [CrossRef]
2. Bortfeldt, A.; Wäscher, G. Constraints in container loading—A state-of-the-art review. *Eur. J. Oper. Res.* **2013**, *229*, 1–20. [CrossRef]
3. Allen, S.D.; Burke, E.K.; Kendall, G. A hybrid placement strategy for the three-dimensional strip packing problem. *Eur. J. Oper. Res.* **2011**, *209*, 219–227. [CrossRef]
4. Bortfeldt, A.; Mack, D. A heuristic for the three-dimensional strip packing problem. *Eur. J. Oper. Res.* **2007**, *183*, 1267–1279. [CrossRef]
5. Yarimcam, A.; Asta, S.; Özcan, E.; Parkes, A.J. Heuristic generation via parameter tuning for online bin packing. In Proceedings of the IEEE Symposium on Evolving and Autonomous Learning Systems (EALS), Orlando, FL, USA, 9–12 December 2014; pp. 102–108. [CrossRef]
6. Ali, S.; Ramos, A.G.; Carravilla, M.A.; Oliveira, J.F. On-line three-dimensional packing problems: A review of off-line and on-line solution approaches. *Comput. Ind. Eng.* **2022**, *168*, 108–122. [CrossRef]
7. Hong, Y.D.; Kim, Y.J.; Lee, K.B. Smart pack: Online autonomous object-packing system using RGB-D sensor data. *Sensors* **2020**, *20*, 4448. [CrossRef] [PubMed]
8. Bischoff, E.E.; Ratcliff, M.S.W. Issues in the development of approaches to container loading. *Omega* **1995**, *23*, 377–390. [CrossRef]
9. Davies, A.P.; Bischoff, E.E. Weight distribution considerations in container loading. *Eur. J. Oper. Res.* **1999**, *114*, 509–527. [CrossRef]
10. Eley, M. A bottleneck assignment approach to the multiple container loading problem. *Oper. Res. Spectr.* **2003**, *25*, 45–60. [CrossRef]
11. Che, C.H.; Huang, W.; Lim, A.; Zhu, W. The multiple container loading cost minimization problem. *Eur. J. Oper. Res.* **2011**, *214*, 501–511. [CrossRef]
12. Ren, J.; Tian, Y.; Sawaragi, T. A priority-considering approach for the multiple container loading problem. *Int. J. Metaheuristics* **2011**, *1*, 298–316. [CrossRef]
13. Ivancic, N.; Mathur, K.; Mohanty, B.B. An integer-programming based heuristic approach to the three-dimensional packing problem. *J. Manuf. Oper. Manag.* **1989**, *2*, 268–289.
14. Martello, S.; Pisinger, D.; Vigo, D. The three-dimensional bin packing problem. *Oper. Res.* **2000**, *48*, 256–267. [CrossRef]
15. Mohanty, B.B.; Mathur, K.; Ivancic, N.J. Value considerations in three-dimensional packing—A heuristic procedure using the fractional knapsack problem. *Eur. J. Oper. Res.* **1994**, *74*, 143–151. [CrossRef]

16. Parreño, F.; Alvarez-Valdés, R.; Oliveira, J.F.; Tamarit, J.M. Neighborhood structures for the container loading problem: A VNS implementation. *J. Heuristics* **2010**, *16*, 1–22. [CrossRef]
17. Zhang, Z.; Guo, S.; Zhu, W.; Oon, W.C.; Lim, A. Space defragmentation heuristic for 2D and 3D bin packing problems. In Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, Catalonia, Spain, 16–22 July 2011; pp. 699–704.
18. Bortfeldt, A.; Gehring, H. Two metaheuristics for strip packing problems. In *Proceedings of the 5th International Conference of the Decision Sciences Institute, Athens 1999*; Despotis, D.K., Zopounidis, C., Eds.; New Technologies Publications: Athens, Greece, 1999; Volume 2, pp. 1153–1156.
19. Chen, M.; Huo, J.; Duan, Y. A hybrid biogeography-based optimization algorithm for three-dimensional bin size designing and packing problem. *Comput. Ind. Eng.* **2023**, *180*, 109–239. [CrossRef]
20. Nguyen, T.H.; Nguyen, X.T. Space Splitting and Merging Technique for Online 3-D Bin Packing. *Mathematics* **2023**, *11*, 1912. [CrossRef]
21. Zhao, H.; She, Q.; Zhu, C.; Yang, Y.; Xu, K. Online 3D bin packing with constrained deep reinforcement learning. *AAAI Conf. Artif. Intell.* **2021**, *35*, 741–749. [CrossRef]
22. Zhao, H.; Zhu, C.; Xu, X.; Huang, H.; Xu, K. Learning practically feasible policies for online 3D bin packing. *Sci. China Inf. Sci.* **2022**, *65*, 112105. [CrossRef]
23. Jia, J.; Shang, H.; Chen, X. Robot Online 3D Bin Packing Strategy Based on Deep Reinforcement Learning and 3D Vision. In Proceedings of the IEEE International Conference on Networking, Sensing and Control (ICNSC), Shanghai, China, 15–18 December 2022; pp. 1–6. [CrossRef]
24. Puche, A.V.; Lee, S. Online 3D Bin Packing Reinforcement Learning Solution with Buffer. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 8902–8909. [CrossRef]
25. Ha, C.T.; Nguyen, T.T.; Bui, L.T.; Wang, R. An online packing heuristic for the three-dimensional container loading problem in dynamic environments and the Physical Internet. In *Proceedings of the Applications of Evolutionary Computation: 20th European Conference, (EvoApplications), Amsterdam, The Netherlands, 19–21 April 2017*; Part II; Springer International Publishing: Cham, Switzerland, 2017; pp. 140–155.
26. Wang, R.; Nguyen, T.T.; Kavakeb, S.; Yang, Z.; Li, C. Benchmarking dynamic three-dimensional bin packing problems using discrete-event simulation. In *Proceedings of the Applications of Evolutionary Computation: 19th European Conference, (EvoApplications), Porto, Portugal, 30 March–1 April 2016*; Part II; Springer International Publishing: Cham, Switzerland, 2016; pp. 266–279.
27. Duan, L.; Hu, H.; Qian, Y.; Gong, Y.; Zhang, X.; Xu, Y.; Wei, J. A multi-task selected learning approach for solving 3D flexible bin packing problem. *arXiv* **2018**, arXiv:1804.06896.
28. Li, T.H.S.; Liu, C.Y.; Kuo, P.H.; Fang, N.C.; Li, C.H.; Cheng, C.W.; Hsieh, C.Y.; Wu, L.F.; Liang, J.J.; Chen, C.Y. A three-dimensional adaptive PSO-based packing algorithm for an IoT-based automated e-fulfillment packaging system. *IEEE Access* **2017**, *5*, 9188. [CrossRef]
29. Asta, S.; Özcan, E.; Parkes, A.J. CHAMP: Creating heuristics via many parameters for online bin packing. *Expert Syst. Appl.* **2016**, *63*, 208–221. [CrossRef]
30. Özcan, E.; Parkes, A.J. Policy matrix evolution for generation of heuristics. In Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, Dublin, Ireland, 12 July 2011; pp. 2011–2018. [CrossRef]
31. Drake, J.H.; Swan, J.; Neumann, G.; Özcan, E. Sparse, continuous policy representations for uniform online bin packing via regression of interpolants. In *Proceedings of the Evolutionary Computation in Combinatorial Optimization: 17th European Conference, EvoCOP 2017, Amsterdam, The Netherlands, 19–21 April 2017*; Springer International Publishing: Cham, Switzerland, 2017; Volume 17, pp. 189–200. [CrossRef]
32. Yang, S.; Song, S.; Chu, S.; Song, R.; Cheng, J.; Li, Y.; Zhang, W. Heuristics Integrated Deep Reinforcement Learning for Online 3D Bin Packing. *IEEE Trans. Autom. Sci. Eng.* 2023, *early access*. [CrossRef]
33. Zhao, H.; Yu, Y.; Xu, K. Learning Efficient Online 3d Bin Packing on Packing Configuration Trees. In Proceedings of the International Conference on Learning Representations. 2022. Available online: https://openreview.net/forum?id=bfuGjlCwAq (accessed on 24 April 2023).
34. Bódis, A.; Balogh, J. Bin packing problem with scenarios. *Cent. Eur. J. Oper. Res.* **2019**, *27*, 377–395. [CrossRef]
35. Lin, T.D.; Hsu, C.C.; Hsu, L.F. Optimization by ant colony hybrid local search for online class constrained bin packing problem. *Appl. Mech. Mater.* **2013**, *311*, 123–128. [CrossRef]
36. Nguyen, T.H.; Tran, V.T.; Doan, P.Q.; Mac, T.T. A novel heuristic algorithm for online 3D bin packing. In Proceedings of the 21st International Conference on Control, Automation and Systems (ICCAS), Jeju, Republic of Korea, 12–15 October 2021; pp. 1993–1997. [CrossRef]
37. Ojha, A.; Agarwal, M.; Singhal, A.; Sarkar, C.; Ghosh, S.; Sinha, R. A generalized algorithm and framework for online 3-dimensional bin packing in an automated sorting center. In Proceedings of the Seventh Indian Control Conference (ICC), Mumbai, India, 20–22 December 2021; pp. 135–140. [CrossRef]