A Bin Packing Problem with Mixing Constraints for Containerizing Items for Logistics Service Providers

Sajini Anand P S[1] and Stefan Guericke[2]

A.P. Moller Maersk R&D, Bangalore, India
A.P. Moller Maersk R&D, Copenhagen, Denmark

Abstract. Large logistics service providers often need to containerize and route thousands or millions of items per year. In practice, companies specify business rules of how to pack and transport items from their origin to destination. Handling and respecting those business rules manually is a complex and time-consuming task. We propose a variant of the variable sized bin packing problem applicable to the containerization process occurring at logistics service providers. This novel model variant extends the bin packing problem with color constraints by adding multiple item mixing constraints. We present a binary integer program along with a first-fit decreasing heuristic and compare the performance on instances from a global logistics service provider. The numerical results indicate promising results to solve this computationally hard combinatorial optimization problem.

Keywords: load planning problem; variable sized bin packing with color constraints; variable sized bin packing with multiple item mixing constraints

1 Introduction

Every year, millions of standardized containers are transported across the world [27], [25]. [26] reports that the value transported through containers is more than 50% of the worldwide seaborne value. Containerized cargo varies from automobile spare parts to scrap metals and refrigerated cargo to consumer goods. In this paper, we focus on commodities that can be packed into rectangular stackable boxes with variable weights and volumes that are transported in dry containers, as it is the case for e.g. large retail enterprises.

Organizing and orchestrating the transport is either handled by large manufactures directly or outsourced to third parties, such as freight forwarders and third/ fourth party logistics providers. We keep referring to these third parties as logistics service providers (LSP) throughout this paper. LSPs enable the end-to-end transport of cargo from an origin location (either a manufacturer or a producer) to its final point of destination (warehouse, market or a customer) ensuring a worldwide distribution of products. LSPs usually containerize the cargo into full container loads (FCL) to reduce costs by using efficient container shipping lines. This containerization process considers items of different types and origins from one (or potentially multiple) customer(s). Items have properties associated, such as their origin plant, and we refer to these item properties as cargo attributes.

Today, the containerization process is a time-consuming manual task. Customers use complex business rules that define how items can be mixed in a single container. For example, a customer would like to avoid mixing items of more than 4 different production plants. The large volume of items that needs to be containerized combined with the rules make it challenging to identify good solutions on a day-to-day basis. Fast and efficient solution methods can help the planners in their decision process. The underlying optimization problem is a variant of the one-dimensional (e.g. volume, weight restrictions) Variable Sized Bin Packing Problem (VSBPP) that minimizes the total container costs while adhering to the business rules that constrain the mixing of items.

To the best of our knowledge, a generic approach for considering multiple mixing constraints for cargo consolidation has not been presented before. Therefore, this paper contributes to the state-of-the-art by:

- Introducing a novel variant of the one-dimensional variable sized bin packing problem with multiple item mixing constraints, applicable for instance to the containerization problem of logistics service providers,
- Proposing a binary integer problem that can solve small instances to optimality,
- Proposing a fast and efficient extension to the well-known first-fit decreasing heuristic to solve the problem for practical instances.

The remainder of this paper is organized as follows: Section 1.1 provides details of the underlying containerization and load planning problem, followed by a literature review in Section 1.2. Section 2 formalizes the problem, presents a binary integer program and a heuristic solution approach. Section 3 introduces test-instances and a numerical analysis of the solution methods. We conclude the work and reflect on future extensions in Section 4.

1.1 The Load Planning and Containerization Problem

A logistics service provider operates facilities, called container freight stations (CFS), to consolidate less than container load (LCL) cargo from different customers, factories and properties into standardized containers for container shipping. CFS can either be used for consolidation or de-consolidation, e.g. to cross-dock cargo at its destination. Full container loads are then transported in an intermodal network, often using container shipping lines for the long haul. The containerization and specification of a routing on a shipping line (a specific vessel departure) is referred to as the load planning process in practice. Once containers reach their destination port, they are forwarded to the destination CFS, de-consolidated and transported to their final destination. See Figure 1 for an overview of this process.

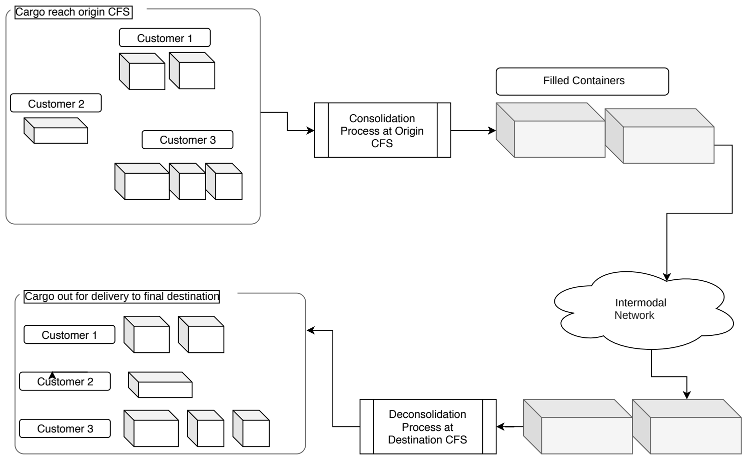

Cargo reach origin CFS
Customer 1
Customer 2
Customer 3
Consolidation Process at Origin CFS
Filled Containers
Intermodal Network
Deconsolidation Process at Destination CFS
Cargo out for delivery to final destination
Customer 1
Customer 2
Customer 3

Fig. 1: Consolidation and deconsolidation of cargo at container freight stations.

A LSP receives cargo bookings with a list of attributes that contain information about the cargo, such as date of arrival, factory, stock keeping unit (SKU), vendor or cargo type. We have observed on average 35 attributes that influence the load planning process. These can be distinguished between those 1) influencing the mix of cargo within a container (called cargo attributes) and those 2) determining the vessel departure (routing attributes). An example of a mixing constraint is 'cargo from two different vendors or containing more than four different SKUs cannot go in the same container'. The objective of business rules is to simplify the handling at the destination CFS, support the packing at the warehouses and ensure safety regulations. In practice, those rules usually differ by origin – destination pair and customer.

The objective of packing cargo into containers is to minimize the total container costs. Different types of standardized dry containers are used, for instance twenty-foot equivalent units (TEU) and forty-foot equivalent units (FFE), each with different volume and weight capacities. Each cargo needs to be assigned to a container while respecting capacity and cargo mixing constraints. In the remainder of this paper we will refer to items and bins instead of cargo and containers to be consistent with previous work in the bin-packing literature.

1.2 Literature Review

The classic Bin Packing Problem (BPP) of assigning items to a minimum number of identical bins without exceeding their capacity [15] is a well explored NP-hard optimization problem [20]. We refer to the work of [5], [6], [4], [9] and [10] for extensive surveys throughout the last decades. In this paper we focus on one-dimensional bin packing problems and refer to [23] and [3] for recent advances in multi-dimensional bin packing algorithms. BPPs are solved to optimality using mathematical programming [9], metaheuristics [21] and well-known approximation algorithms such as First-Fit Decreasing (FFD) [16].

A relevant extension to the classic BPP are variable sized bins (VS-BPP) and variable costs [13]. This variant has been solved using various algorithms, such as Branch-and-Price or an extended FFD, and we refer to [1], [14] and [17] for an overview.

Another extension related to the containerization problem presented above is the Bin Packing with Conflicts (BPPC) that assign items to a minimum number of bins while avoiding joint assignments of items that are in conflict [11]. A graph structure is used where nodes represent the items and edges represents the conflicts between a pair of items. [11] do not consider variable bin sizes, multiple capacity constraints and, most importantly, mixing constraints beyond pairwise conflicts. We refer to [24] and [2] for further solution methods on the BPPC.

A relatively recent work that is closely related to our problem is the variable sized bin packing problem with color constraints (VBPC) [8]. Items have an associated color and not more than two colors can be packed together in the same bin. [8] also introduce the generalized VBPC (GVBPC) problem where at most p colors can be packed into the same bin. They propose a first-fit variant to solve the GVBPC and derive bounds of their approach. [19] solve the bin packing problem with color constraints using a Branch-and-Price method. [18] apply a variable neighborhood search matheuristic to solve a variant of the VBPC where each item can have a set of colors. [7] study another variant where each color can only be associated with one item.

Summarizing, our work builds upon and extends work in the field of the generalized (one-dimensional) variable sized bin packing problems with color constraints (GVBPC). To the best of our knowledge, multiple attribute constraints have not been studied in literature before. We'll refer to the containerization problem as a generalized (one-dimensional) variable sized bin packing problems with multiple item mixing constraints (GVBPMC).

2 Solution Approach

In this section we present the formal definition of the problem, propose a binary integer program formulation and a first-fit decreasing based heuristic.

2.1 Problem Formalization

Let J be the set of bins, where each bin $j \in J$ has cost Q_j and a maximum capacity of resource type R, $V_{r,j}$. In this work, we focus on resource cubic meters, but others, such as weight, could be considered as well. Let I be the set of items with a utilization $v_{i,r}$ of resource $r \in R$.

We now present the extensions for multiple item mixing constraints, i.e. limiting the number of different values of items attributes in a single bin. Let A be a set of relevant item attributes and C_a the set of unique values for attribute a.

Partitions $P_c \subseteq I$ define all items sharing the same attribute value c. Note that $\{P_c\}$ represents a partition of I into subsets such that each of the subset contains items belonging to the uniquely chosen attribute $a \in A$ such that $\cup_{c \in C_a} P_c = I$ and $\cap_{c \in C_a} P_c = \emptyset$. We denote $|P_c|$ as the number of items sharing the attribute value. Finally, parameter d_a defines how many different attributes values can occur in a single bin concurrently, similar to [8].

2.2 Mathematical Program

We propose a binary integer programming formulation for the generalized (one-dimensional) variable sized bin packing problems with multiple item mixing constraints (GVBPMC). It extends the classic variable sized bin packing problem and its variant with color constraints.

The objective of the optimization problem is to assign all items to bins while minimizing total bin costs so that no conflicts between items occur and no capacity constraints are violated. Binary variables $x_{i,j}$ are used to assign item $i \in I$ to bin $j \in J$ and y_j indicate whether bin j is used in the solution. Binary variables $w_{g,j}$ indicate whether any item of mixing constraint group g's attribute value is present in bin j.

$$\min \sum_{j \in J} Q_j y_j \tag{1}$$

$$s.t. \sum_{j \in J} x_{i,j} = 1 \qquad \forall i \in I \tag{2}$$

$$\sum_{i \in I} v_{i,r} x_{i,j} \leq V_{r,j} y_j \qquad \forall j \in J, r \in R \tag{3}$$

$$\sum_{i \in P_c} x_{i,j} \leq |P_c| w_{c,j} \qquad \forall a \in A, c \in C_a, j \in J \tag{4}$$

$$\sum_{c \in C_a} w_{c,j} \leq d_a \qquad \forall a \in A, j \in J \tag{5}$$

$$x_{i,j} \in \{0,1\}, \forall i \in I, j \in J \tag{6}$$

$$w_{c,j} \in \{0,1\}, \forall a \in A, c \in C_a, j \in J, \tag{7}$$

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

The objective function (1) minimizes the total cost of used bins after assignment of all items. Constraints (2) ensure that each item is assigned to exactly one bin. Constraints (3) ensure that the capacity of bin j in respect to resource type r is respected by summing the total resource utilization of items assigned to the bin. Constraints (4) and (5) limit the mix of items in the same bin. Specifically, Constraints (4) set the indicator variable $w_{c,j}$ to 1 if any item with attribute value c is assigned to bin j. Constraints (5) set an upper bound on the number of different attribute values that are allowed to be together in a single bin. Finally, (6) – (8) define the variable domains.

2.3 First-Fit Decreasing based Heuristic

Integer models for the bin packing problem are known to suffer from symmetry due to identical bins and thus long runtimes for practical instances [12]. An algorithm that has been extensively used in previous work is the first-fit decreasing heuristic, [22], [6]. We extend this heuristic by multiple counters to respect the constraints placed on the attributes of items. We present the pseudo-code in Algorithm 1 and refer to it as FFD with Multiple Constraints (FFDMC).

Algorithm 1: FFDMC heuristic for the load planning problem.

```
1  $R = \{\}$;
2  $J' \leftarrow sort\_descending(J, V_{j,cbm})$;
3  $I' \leftarrow sort\_descending(I, v_{i,cbm})$;
4  while $|I'| > 0$ do
5      $j \leftarrow pop(J')$
6      $V^R \leftarrow V_j$
7      $I^* \leftarrow \{i \in I' : v_{i,r} \leq V_{j,r} \forall r \in R\}$
8      if $|I^*| > 0$ then
9          while $|I^*| > 0$ do
10             $i* \leftarrow pop(I^*)$
11             if $!constr\_violated(j, i, R, V^R)$ then
12                 $R \leftarrow R \cup (j, i)$
13                 $V^R \leftarrow V^R - v_i$
14                 $remove(I', i)$
15             end
16         end
17     else
           Result: $Infeasible$
18     end
19 end
   Result: $reduce\_bin\_sizes(R)$
20
```

The assignment process works as follows: First, the set of bin-item tuples holding the solution is initialized in line 1. In lines 2 and 3 the bins J and the item set I are sorted and indexed in descending order of the volume parameter v (utilization) and V (capacity) such that the lowest indexed items have the highest values. Note that we are focusing on on cubic meter capacity only.

The while loop in line 4 continues while there are still items to be assigned to a bin. Starting with the largest bin, line 6 initializes the residual capacity to its total capacity and selected items I^* that fit into the bin. If I^* is empty, this means that $|I'| > 0$ but no item is fitting into the largest bin. Thus, the algorithm returns an $Infeasible$ state (line 17). Next, loop in line 9 tries to assign items while constraints are not validated. The function $constr\_violated(j, i, R, V^R)$, returning $\{true, false\}$ for bin j, item i, solution R and residual capacity vector V^R, is defined in Equation 9. It returns $true$, if item i fits into the residual capacity V_r^R for all of resource types r and if the mixing constraints are not violated. Therefore, I'' defines the set of items currently assigned to bin j. The second line in equation 9 checks that the upper bounds d_a for all attributes hold by summing the number of unique values C_a for attribute a. Attribute value $c \in C_a$ appears in bin j, if the intersection between items already associated to bins I'' and items with that property P_C is not empty.

$$\begin{aligned} constr\_violated(j,i,R,V^R) = & \forall r \in R : V^R_r - v_{i,r} \geq 0 \wedge \\ & \forall a \in A : d_a \geq \sum_{c \in C_a} \begin{cases} 0, if\ I'' \cap P_c = \emptyset \\ 1, else \end{cases} \\ & \text{with } I'' = \{i' : (j', i') \in R : j' = j\} \cup \{i\} \end{aligned} \tag{9}$$

If no constraints are violated, line 12 adds the bin-item assignment to the solution, updates the residual capacity vector and removes i from the items to be processed. Note that we assume the *pop* operator to remove items from the lists. The process repeats until all items of I' are assigned to bins. The post-processing function $reduce\_bin\_sizes(R)$ in line 19 tries to improve the solution further by assessing if a next smaller bin can accommodate an already assigned item volume in a bin (provided smaller bins are still available). Otherwise, the FFDMC algorithm returns the current load plan.

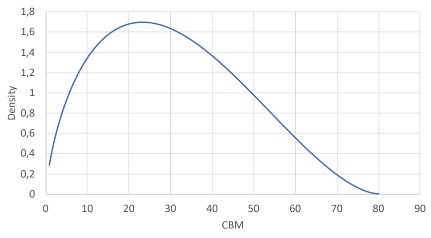
3 Numerical Results

In this section we present the test instances, numerical results for the single item mixing constraints are presented and solution approaches compared. Afterwards, the much more complex multi-constraint case is evaluated. The results are calculated on an Azure Virtual Machine (*standard_a1_v2*) with one CPU, 2 GB RAM and Gurobi 7.5.2 with default settings. For all results, a maximum runtime of 30 minutes has been set. Our FFDMC heuristic is implemented in Python 3.6.

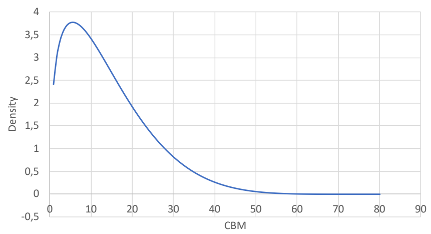
3.1 Test Instances

Preliminary experiments indicate that the binary program is very sensitive to the input data, in particular to the item size distribution. Based on real sample sets provided by a global logistics service provider, we generate instances covering a variety of item sizes (in terms of cubic meters), different number of items and containers, and mixing constraints. Thereby, we can test the applicability of our solution methods to different input data and ensure that the algorithms are applicable to customers with different item-size patterns. To generate realistic item-sizes, we fitted a beta distribution to the cubic meter attribute of four different real-world instances. Figure 2 visualizes density functions using the estimated parameters for the four instances in a scale of $[0, 80]$ cbm (which is the maximum for a 45 foot container).

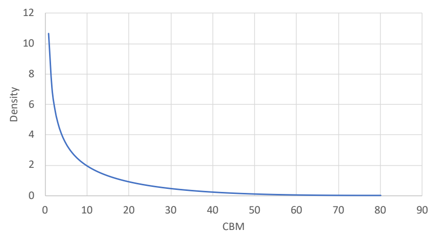
In total, we have generated 340 instances for two different analyses. The first set contains 275 and the second 65 instances. *Set 1* has a single item mixing



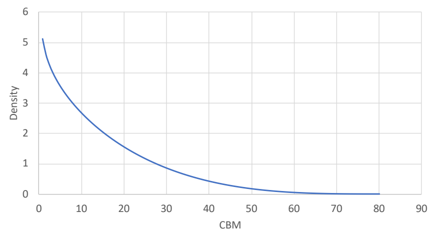
(a) Sample 1, $\alpha = 1.70, \beta = 2.71$.



(b) Sample 2, $\alpha = 1.41, \beta = 6.60$.



(c) Sample 3, $\alpha = 0.45, \beta = 3.55$.
s



(d) Sample 4, $\alpha = 0.89, \beta = 4.03$.

Fig. 2: Density functions of the beta-distribution for sample 1, 2, 3 and 4.

constraint but different number of bins, parameters of the beta-distribution and number of items. For each of the four estimated beta-distributions, 50, 100, 150 and 200 items, for a uniqueness[1] of 5%, 10% and 25%, we created 25 samples. Additionally, we modified the number of bins to be 10, 20, 30, 40, 50, 75, 100 and 150 for a fixed number of items, uniqueness and distribution to evaluate the influence of that parameter on the runtime. *Set 2* evaluates 1..4 item mixing constraints and upper bounds while using a fixed amount of 100 items, 120 bins and $\alpha = 1.41$ and $\beta = 6.6$, which is the most common pattern provided by the LSP.

3.2 Comparison of Mixed Integer Program and FFDMC Heuristic for a Single Item Mixing Constraint

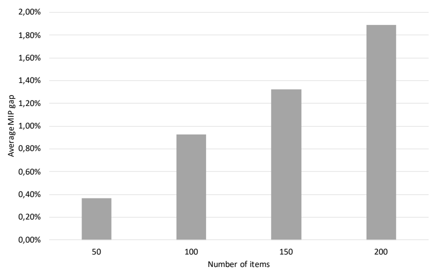
In this section, MIP gaps[2] and runtimes between Gurobi and our suggested FFDMC heuristic are compared for a single item mixing constraint (similar to the GVBPC but with different costs).

As indicated in Figure 3, all instances could be solved to an average gap of 1.1% (the maximum observed gap was less than 5%) within 30 minutes using Gurobi. Figure 3a shows that the average gap highly depends on the number of

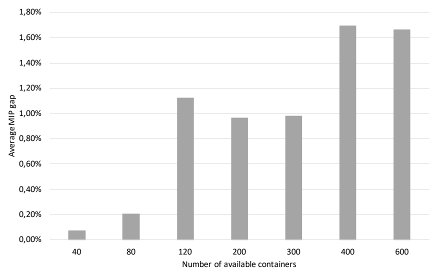
[1] Uniqueness indicates how many items are sharing the same property value in percent, averaged over all rules of the instances.

[2] We define MIP gap as $\frac{|best\_bound - incumbent|}{max(|best\_bound|, |incumbent|)}$.

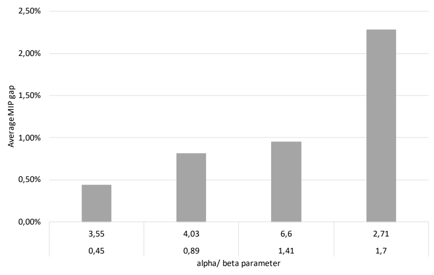
items in the instances. Practical instances typically contain between 100 - 150 items, showing the applicability of a binary integer formulation for a single item constraint. The number of bins (see Figure 3b) have a clear impact on the runtime and the gap as they increase the complexity of the model. Figure 3c shows that the instances container larger items (in terms of CBM) are clearly more difficult to solve which is challenging for the reliability of solution approaches for practical applications.



(a) Gaps by number of items.



(b) Gaps by number of available bins.



(c) Gaps by beta-distribution parameters.

Fig. 3: Gurobi MIP gaps in percent for different instance properties for test *Set 1*.

In Table 1 the gap of Gurobi is compared to the FFDMC heuristic for different number of items. We observe that the gap for Gurobi is consistently under 5% for all instances, also for larger instances. FFDMC performs well, with average gaps between 2.17 – 7.11%, depending on the number of items. However, for some instances the worst-case gap is relatively large (up to 26.57%). Table 1 also indicates that between 70% and 22% of the instances could be solved to optimality with Gurobi. However, the number of instances without proven optimality increase with more items.

Table 2 compares the runtime of Gurobi and the FFDMC heuristic. As indicated, for a significant number of instances Gurobi could not prove optimality

		Gap Gurobi [%]				Gap FFDMC [%]			
Items	% Instances	Min	Avg	Med	Max	Min	Avg	Med	Max
50	Opt. (70%)	0	0	0	0	0	6.9	7.65	26.57
	TL (30%)	0.43	1.23	0.81	3.93	0.55	2.17	1.15	7.65
100	Opt. (41%)	0	0	0	0	0.00	6.28	4.66	18.43
	TL (59%)	0.09	1.52	0.55	4.25	0.83	4.54	3.94	19.55
150	Opt. (22%)	0	0	0	0	0.64	4.46	3.47	14.29
	TL (78%)	0.05	1.54	1.41	3.76	0.39	6.06	5.40	20.03
200	Opt. (22%)	0	0	0	0	4.37	7.11	6.37	10.72
	TL (78%)	0.22	1.96	2.04	4.47	1.12	6.13	5.84	17.04

Table 1: MIP gaps for Gurobi and FFDMC for different number of items for test *Set 2*. *Opt.* shows the number of instances solved to optimality with Gurobi, whereas *TL* (timelimit)) refers to an occurred timeout of 30 minutes.

within 30 minutes. The runtime in which cases Gurobi could prove optimality, varies between on average 19.42 seconds for 50 items and 564.32 seconds for 200 items. FFDMC in contrast never took more than 1/10th of a second. While measuring these low runtimes, we expect a large impact of side-processes running on the virtual machine and should thus been interpreted carefully.

		Runtime for Gurobi [s]				Runtime for FFDMC [s]			
Items	% Instances	Min	Avg	Median	Max	Min	Avg	Median	Max
50	Opt. 70%	0.91	19.42	7.29	159.25	0.0010	0.003	0.0014	0.0100
	TL 30%	1800	1800	1800	1800	0.0011	0.0032	0.0013	0.0071
100	Opt. 41%	6.95	286.83	103.41	1528.83	0.0031	0.0096	0.0095	0.0204
	TL 59%	1800	1800	1800	1800	0.0030	0.0098	0.0096	0.0190
150	Opt. 22%	32.45	240.0	104.70	1481.89	0.0120	0.0175	0.0184	0.0293
	TL 78%	1800	1800	1800	1800	0.0115	0.0187	0.0192	0.0277
200	Opt. 22%	21.03	564.32	277.41	1707.26	0.0216	0.0266	0.0270	0.0303
	TL 78%	1800	1800	1800	1800	0.0205	0.0315	0.0296	0.0685

Table 2: Runtimes in seconds for Gurobi and FFDMC for different number of items for test *Set 2*. *Opt.* shows the number of instances solved to optimality with Gurobi, whereas *TL* refers to an occurred timeout of 30 minutes.

Concluding the single item mixing constraint, our FFDMC heuristic provides solutions much faster than Gurobi but results in significantly larger gaps.

3.3 Comparison of Mixed Integer Program and FFDMC Heuristic for Multiple Item Mixing Constraints

In this section, we compare results of Gurobi with the FFDMC heuristic for instances with up to four item mixing constraints. We limit the number to four in the experimental study to still receive reasonable results for Gurobi.

Mixing Constraints	Gap Gurobi [%]	Gap FFDMC [%]
1	1.69%	5.74%
2	18.94%	16.46%
3	34.10%	15.62%
4	48.35%	12.40%

Table 3: Average MIP gaps for Gurobi and FFDMC for instances with 1..4 multiple item mixing constraints of test *Set 2*.

Average gaps for Gurobi and FFDMC for different number of mixing constraints are presented in Table 3. As observed in the previous section, FFDMC's gap is worse than Gurobi's for a single mixing constraint. With increasing number of constraints, we observe a non-monotonic performance of FFDMC and believe that is connected to the randomness of the test instances. However, with an increasing number of constraints, FFDMC clearly outperforms Gurobi in terms of gap. Already for two constraints, FFDMC's gap is lower than Gurobi's. When considering four mixing constraints, FFDMC's gap is 75% lower that Gurobi's.

Mixing Constraints	Runtime Gurobi [s]	Runtime FFDMC [s]
1	1739.19	0.01
2	1800.02	0.02
3	1800.04	0.03
4	1800.08	0.03

Table 4: Average runtime in seconds for Gurobi and FFDMC for instances with 1..4 multiple item mixing constraints of test *Set 2*.

The runtimes of Gurobi and FFDMC are presented in Table 4. Gurobi cannot prove optimality for almost all instances with more than one mixing constraint as it hits the maximum runtime. In contrast, FFDMC is able to find solution in less of 1/10th of a second. The slight increase in runtime with more constraints can be explained by a more complex violation check function (see Equation 9) that involves checking additional conditions.

Summarizing the findings for multiple item mixing constraints, we can state that FFDMC clearly outperforms our binary program solved with a state-of-the-art commercial solver, both in terms of runtime and MIP gaps. The very low runtimes suggest that it is very efficient for cases with many item mixing constraints. Thus, it can serve as an efficient construction heuristic or provide solution for warm starting the MIP solver. However, the results indicate that FFDMC still has relatively large remaining gaps and can be improved further.

4 Conclusion and future work

In this paper, we proposed a generalization of the classic bin packing problem with multiple item mixing constraints to solve the containerization problem arising at logistics service providers (such as third- or fourth-party logistics providers).

The problem is an extension to the BPP with color constraints. The problem addresses the packing issue where constraints can be placed on item attributes and thus limiting the mixing of items in each bin. Practical applications are, among others, creating load plans for multiple customers, reducing item type variations in containers and improving the item management.

We present a binary integer program to solve instances to optimality and a first-fit decreasing based heuristic to determine good solutions fast. To evaluate both methods, we generate a few hundred test-instances based on samples of a global LSP. Numerical results indicate that Gurobi performs reasonably well on instances with one mixing constraint, leading to an average gap of 1 - 2%. Instances with two item mixing constraints increase the average gap of 18.94%, and instance with four to more than 48%. Gurobi almost always reaches the timelimit of 30 minutes. Our FFDMC heuristic creates solutions for one mixing constraint with a gap of less than 6%, with a runtime of less than 1/10 of a second for all instances. For instances with four constraints, FFDMC provides a remaining MIP gap of less than 13% and clearly outperforms Gurobi for large, practical instances in both runtime and solution quality. We conclude that FFDMC is a reasonable starting point to solve the introduced problem for practical sized instances and can support future research as a benchmark algorithm.

Future work should be done in the area of solution algorithms. For instance, Branch-and-Price has been successfully applied to the bin packing problem with color constraints and is a promising approach. Besides, meta- or matheuristics such as [18] have also been used successfully. Finally, theoretical analyses of the approximation quality could provide new insights for solution algorithms.

References

1. Belov, G., Scheithauer, G.: A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. European Journal of Operational Research **141**(2), 274–294 (2002)
2. Capua, R., Frota, Y., Ochi, L.S., Vidal, T.: A study on exponential-size neighborhoods for the bin packing problem with conflicts. Journal of Heuristics **24**(4), 667–695 (2018)
3. Christensen, H.I., Khan, A., Pokutta, S., Tetali, P.: Approximation and online algorithms for multidimensional bin packing: A survey. Computer Science Review **24**, 63–79 (2017)
4. Coffman, E.G., Csirik, J., Galambos, G., Martello, S., Vigo, D.: Bin packing approximation algorithms: survey and classification. In: Handbook of combinatorial optimization, pp. 455–531. Springer New York (2013)

5. Coffman, E.G., Garey, M.R., Johnson, D.S.: Approximation algorithms for bin-packing: an updated survey. In: Algorithm design for computer system design, pp. 49–106. Springer (1984)
6. Coffman, E.G., Garey, M., Johnson, D.: Approximation algorithms for bin packing: A survey. Approximation algorithms for NP-hard problems pp. 46–93 (1996)
7. Crévits, I., Hanafı, S., Mahjoub, A.R., Taktak, R., Wilbaut, C.: A special case of variable-sized bin packing problem with color constraints. In: 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT). pp. 1150–1154. IEEE (2019)
8. Dawande, M., Kalagnanam, J., Sethuraman, J.: Variable sized bin packing with color constraints. Electronic Notes in Discrete Mathematics **7**, 154–157 (2001)
9. Delorme, M., Iori, M., Martello, S.: Bin packing and cutting stock problems: Mathematical models and exact algorithms. European Journal of Operational Research **255**(1), 1–20 (2016)
10. Delorme, M., Iori, M., Martello, S.: Bpplib: a library for bin packing and cutting stock problems. Optimization Letters **12**(2), 235–250 (2018)
11. Epstein, L., Levin, A.: On bin packing with conflicts. SIAM Journal on Optimization **19**(3), 1270–1298 (2008)
12. Fasano, G.: A mip approach for some practical packing problems: Balancing constraints and tetris-like items. Quarterly Journal of the Belgian, French and Italian Operations Research Societies **2**(2), 161–174 (2004)
13. Friesen, D.K., Langston, M.A.: Variable sized bin packing. SIAM journal on computing **15**(1), 222–230 (1986)
14. Haouari, M., Serairi, M.: Heuristics for the variable sized bin-packing problem. Computers & Operations Research **36**(10), 2877–2884 (2009)
15. Johnson, D.S.: Near-optimal bin packing algorithms. Ph.D. thesis, Massachusetts Institute of Technology (1973)
16. Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Graham, R.L.: Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM Journal on computing **3**(4), 299–325 (1974)
17. Kang, J., Park, S.: Algorithms for the variable sized bin packing problem. European Journal of Operational Research **147**(2), 365–372 (2003)
18. Kochetov, Y., Kondakov, A.: Vns matheuristic for a bin packing problem with a color constraint. Electronic Notes in Discrete Mathematics **58**, 39–46 (2017)
19. Kondakov, A., Kochetov, Y.: A core heuristic and the branch-and-price method for a bin packing problem with a color constraint. In: International Conference on Optimization Problems and Their Applications. pp. 309–320. Springer (2018)
20. Korte, B., Vygen, J.: Bin-Packing, pp. 407–422. Springer Berlin Heidelberg (2000)
21. Levine, J., Ducatelle, F.: Ant colony optimization and local search for bin packing and cutting stock problems. Journal of the Operational Research society **55**(7), 705–716 (2004)
22. Martello, S., Toth, P.: Knapsack problems: algorithms and computer implementations. Wiley-Interscience series in discrete mathematics and optimization (1990)
23. Pisinger, D., Sigurd, M.: The two-dimensional bin packing problem with variable bin sizes and costs. Discrete Optimization **2**(2), 154–167 (2005)
24. Sadykov, R., Vanderbeck, F.: Bin packing with conflicts: a generic branch-and-price algorithm. INFORMS Journal on Computing **25**(2), 244–255 (2013)
25. Statista Research Department: Estimated containerized cargo flows on major container trade routes in 2018, by trade route (in million teus) (Accessed May 2020), `https://www.statista.com/statistics/253988/estimated-containerized-cargo-flows-on-major-container-trade-routes/`

26. UN Conference on Trade and Development: Review of maritime transport 2013. Tech. rep., United Nations (2013)
27. World Shipping Council: World shipping council trade statistics (Accessed May 2020), `http://www.worldshipping.org/about-the-industry/global-trade/trade-statistics`