Combinatorial Benders Decomposition and Column Generation for Optimal Box Selection

Christoph Buchheim[0000−0001−9974−404X],
Alice Kirchheim[0000−0002−8529−425X], and
Pia Schreynemackers[0009−0006−8717−3386]

Abstract We consider a two-stage optimization problem with sparsity constraints, motivated by a common challenge in packaging logistics: minimizing the volume of transported air by optimizing the size and number of available packaging boxes, given the demand for order items. In the first stage, we select the optimal dimensions of the boxes, while in the second stage, the items demanded are assigned to the selected box types, with the goal of minimizing unused volume. To solve this two-stage problem, we propose a cutting plane approach using Benders decomposition, where the master problem essentially corresponds to the first stage. Due to the specific structure of the second-stage problem, we are able to generate the resulting Benders cuts by a purely combinatorial algorithm. Moreover, this allows us to apply column generation, as the coefficients of the constraints of the new variables can also be derived directly from the combinatorial structure. Since the number of available box types is huge in practice, the combination of both procedures provides a particularly promising approach.

1 Introduction

In package-handling operations, products are shipped in boxes of various sizes. However, selecting suboptimal box sizes can lead to excessive void space, increased pack-

Christoph Buchheim
Department of Mathematics, TU Dortmund University, Vogelpothsweg 87, 44227 Dortmund, Germany,
e-mail: christoph.buchheim@tu-dortmund.de

Alice Kirchheim
Department of Mechanical Engineering, TU Dortmund University, and Fraunhofer Institute for Material Flow and Logistics IML, Joseph-von-Fraunhofer-Str. 2-4, 44227 Dortmund, Germany,
e-mail: alice.kirchheim@tu-dortmund.de

Pia Schreynemackers
Department of Mathematics, TU Dortmund University, Vogelpothsweg 87, and Fraunhofer Institute for Material Flow and Logistics IML, Joseph-von-Fraunhofer-Str. 2-4, 44227 Dortmund, Germany,
e-mail: pia.schreynemackers@tu-dortmund.de

aging material usage and higher shipping costs. Despite these negative consequences, decisions regarding box usage are often still based on human intuition rather than systematic optimization. This leads to inefficiencies in space utilization and overall logistics performance [1, 2]. In this paper, we therefore present a two-stage algorithm for optimizing both the selection of box types and the assignment of items to the box types, aiming at a minimum total volume of all used boxes. We focus on applications where each item is packed individually, so that the arrangement of items within a box is not considered. However, the problem is significantly complicated by bounding the number of different box types, which is an important constraint in practice: while the set of feasible box types, given by the dimensions of a box, is huge, only a small subset of them should be actually used in order to simplify and speed up the packaging process.

Even if such a sparsity constraint is added, the large number of potential box types still leads to a large number of variables in our model, most of which however are zero in an optimal solution. To handle this effectively, we use column generation. Additionally, the problem's structure, which can be divided into two decisions, is exploited. First, the boxes are selected, and then items are assigned to the selected boxes. As the second step is easier than the first, this intuitive division is perfectly suited to a Benders decomposition. In this work, we apply and combine both Benders decomposition and column generation. By exploiting the specific problem structure, we are able to devise very fast combinatorial algorithms for both tasks.

To the best of our knowledge, no exact algorithms have been proposed in the literature for the specific two-stage box selection and assignment problem considered in this work. Instead, several heuristic approaches have been explored in the literature. Gurumoorthy and Hinge [1] address variable packaging box dimensions and model the problem as a clustering problem where each cluster corresponds to the group of items which will be packed into the same box type. They achieve a 4.4% reduction in shipment volume on actual shipments transported by Amazon. Das *et al.* [3] tackle a broader multi-objective version of the problem, combining warehouse assignment and carton configuration optimization. Their ant colony optimization approach consistently found optimal or non-dominated solutions on 54 different datasets, compared to a frequently used k-means clustering based evolutionary algorithm.

As mentioned above, our exact approach is based on the combination of Benders decomposition and column generation. This combined method has been effective in solving large-scale two-stage problems, particularly when structural decomposability can be exploited. Several authors [4, 5, 6, 7, 8, 9] have shown that integrating Benders decomposition with column generation leads to a scalable framework applicable across various domains such as scheduling, assignment, and routing. However, these contributions do not address the particular problem discussed in this work.

Moreover, an important novel feature in our approach is the explicit modeling of a sparsity constraint. While we do not focus on complexity theoretic questions here, it is worth mentioning that such constraints often lead to NP-hard problems, e.g., when considered in continuous knapsack [10] or in regression problems [11], and that sparse optimization is an interesting area of optimization in its own right. Sparsity is the key to both our Benders decomposition approach, where the master problem is responsible for determining the support of the sparse solution, and to column generation, as it explicitly limits the number of non-zero variables in an integer solution.

2 Problem Formulation

In order to formalize the problem considered, let $I = \{1, ..., n\}$ be the index set for the given items to be packed and $J = \{1, ..., m\}$ the index set for the possible box types. In the basic two-stage formulation of the optimal box selection problem, we minimize an objective function $c^\top x + Q(x)$ over $x \in \mathbb{Z}_+^m$, where the second stage is defined by

$$\begin{array}{rll} Q(x) = \min & \sum_{i,j} g_{ij} y_{ij} & \\ \text{s.t.} & \sum_j y_{ij} \geq d_i & \forall i \in I \\ & \sum_i y_{ij} \leq x_j & \forall j \in J \\ & y_{ij} = 0 & \forall (i,j) \in I \times J \colon f_{ij} = 0 \\ & y_{ij} \in \mathbb{Z}_+ & \forall (i,j) \in I \times J \ . \end{array}$$

Here, the first stage variable x_j represents the number of selected boxes of type j and the second stage variable y_{ij} represents the number of items of type i assigned to box type j. The objective is to select the cost-minimal box types that satisfy the given demand $d \in \mathbb{Z}_+^n$ of each item type, where the costs could depend on the box ($c_j \in \mathbb{R}$) and on the assignment ($g_{ij} \in \mathbb{R}$). The value $f_{ij} \in \{0, 1\}$ determines whether item i fits into box j or not, for all $i \in I$ and $j \in J$. In the second stage, we obtain a problem with a totally unimodular constraint matrix, so that we can relax the integrality constraints on y without changing the objective value $Q(x)$, provided that x is integer.

We note that the basic problem formulation stated above is actually trivial to solve: For every item, we can select the cheapest box available it fits into. However, we next introduce a sparsity constraint, which makes the problem significantly harder. In fact, in practical applications, limiting the selection to a small number of box types is essential for speeding up the packaging process. The corresponding sparsity constraint can be modeled in the first stage by binary variables s and big M constraints as follows:

$$\begin{array}{rl} \min & \sum_j c_j x_j + Q(x) \\ \text{s.t.} & x_j \leq M s_j \ \forall j \in J \\ & \sum_j s_j \leq K \\ & x \in \mathbb{Z}_+^m, \ s \in \{0,1\}^m \ . \end{array}$$

The goal is thus to choose at most $K \in \mathbb{Z}_+$ box types in an optimal way.

The resulting two-stage formulation can be simplified by exploiting that, in every optimal solution, we have $x_j = \sum_i y_{ij}$ for all $j \in J$. Moreover, we assume for sake of simplicity that the cost coefficients g_{ij} and c_j are both given by the volume of the box j, for all $j \in J$. Overall, this leads to a new two-stage formulation with a master problem

$$\min \quad Q'(s) \quad \text{s.t.} \quad \sum_j s_j \leq K, \ s \in \{0,1\}^m \tag{M}$$

and a subproblem

$$\begin{array}{rll} Q'(s) = \min & \sum_{i,j} c_j y_{ij} & \\ \text{s.t.} & y_{ij} \leq f_{ij} d_i s_j & \forall (i,j) \in I \times J \\ & \sum_j y_{ij} \geq d_i & \forall i \in I \\ & y_{ij} \geq 0 & \forall (i,j) \in I \times J \ . \end{array} \tag{S}$$

3 Benders decomposition and Benders cuts

Benders decomposition [12] is a well-known technique for solving two-stage problems. In this method, the second stage must be a linear program that can be dualized, to form constraints that are linear in the first stage variables. The function Q' in (M) is replaced by a new variable θ, which is bounded below by the optimal value of the dual problem. The general procedure can be summarized as follows:

- Generate a candidate solution $\bar{s}$ for the first stage. So-called feasibility cuts ensure that $\bar{s}$ is always feasible for the second stage.
- If $\bar{s}$ is feasible but not optimal for the whole system (i.e., if it does not correspond to the current value $\bar{\theta}$), linear constraints that cut off the pair $(\bar{s}, \bar{\theta})$ are added to the main problem, using the subproblem's structure. These are so-called optimality cuts.
- Resolving the adjusted first-stage problem leads to another candidate to be checked. Iterating leads to more optimality cuts.
- If a candidate is optimal, the produced potential optimality cut is not violated, and the algorithm terminates with an optimal solution.

In our case, the subproblem (S) decomposes into one separate problem for each $i \in I$,

$$\begin{array}{rll} \min & \sum_j c_j y_{ij} & \\ \text{s.t.} & y_{ij} \le f_{ij} d_i \bar{s}_j & \forall j \in J \\ & \sum_j y_{ij} \ge d_i & \\ & y_{ij} \ge 0 & \forall j \in J\,, \end{array} \tag{S_i}$$

where $\bar{s} \in [0,1]^m$ is a given solution of the LP-relaxation of the master problem. Therefore, we introduce a new variable θ_i for every item $i \in I$, representing the optimal value of (S_i). The objective of the main problem then becomes $\sum_i \theta_i$. The dual problem to (S_i), which is needed to derive the Benders cuts, reads

$$\begin{array}{rll} \max & \sum_j f_{ij} d_i \bar{s}_j \lambda_{ij} + d_i \mu_i & \\ \text{s.t.} & \mu_i - \lambda_{ij} \le c_j & \forall j \in J \\ & \lambda_{ij} \ge 0 & \forall j \in J \\ & \mu_i \ge 0\,. & \end{array} \tag{D_i}$$

The main feature of our algorithm is that Benders cuts can be derived by a combinatorial algorithm. We first claim that feasibility cuts have a very simple structure in our case, they just ensure that every item fits into at least one selected box type:

Theorem 1 *All feasibility cuts for (M) are given as $\sum_j f_{ij} s_j \ge 1$ for $i \in I$.*

Therefore, all feasibility cuts can be added to (M) from the beginning and no separation algorithm is needed. To derive optimality cuts, we use the following algorithm to compute an optimal solution of (D_i):

Algorithm 1 Combinatorial solution of (D_i)

1: Sort the index set J by ascending costs $c_1 \le \ldots \le c_m$.
2: Determine the smallest index $k_i \in J$ such that $\sum_{j=1}^{k_i} f_{ij} \bar{s}_j \ge 1$.
3: Return the dual optimal solutions $\mu_i^* = c_{k_i}$ and $\lambda_{ij}^* = (c_{k_i} - c_j)^+ := \max\{c_{k_i} - c_j, 0\}$.

Theorem 2 *Assume that $\bar{s} \in [0,1]^m$ satisfies all feasibility cuts. Then Algorithm 1 terminates in $O(m \log m)$ time with an optimal solution of (D_i).*

This result can be shown by duality. Note that k_i is well-defined in Step 2 due to the feasibility cuts stated in Theorem 1. The resulting optimality cut candidate for (S_i) now reads $\theta_i \geq -\sum_j (f_{ij} d_i \lambda^*_{ij}) s_j + d_i \mu^*_i$. Whenever it is violated, it can be added to the master problem (M) as a valid cutting plane. The sorting in Step 1 can be done in a preprocessing phase, as it does not depend on $\bar{s}$. This reduces the total running time for computing optimality cuts for all n subproblems to $O(nm)$ per iteration.

4 Column generation

In Benders decomposition, we first ignore a part of the problem formulation to make it easier to handle, and then add necessary constraints step by step. Column generation [13] follows a similar principle, particularly when handling a large number of variables. It exploits the fact that in such problems, in a basic solution, almost all of the variables are at value zero. For a variable, having a value of zero or not belonging to the model at all makes no difference, which motivates the idea of adding new variables to the model, i.e., adding new columns to the constraint matrix, in the course of the algorithm.

The critical task in a column generation approach is to identify those variables with negative reduced costs, even if they do not belong to the master problem (M) yet, in order to decide which variables could improve the objective value of the corresponding LP relaxation. In our application, the master problem (M) has three types of constraints: optimality cuts, feasibility cuts, and the sparsity constraint $\sum_j s_j \leq K$. Unlike in most Benders decomposition approaches, we can exploit here that our Benders cuts have a specific combinatorial structure, as shown in the previous section.

More precisely, let $\bar{\nu} \in \mathbb{R}^r$ denote a dual optimal solution to the LP relaxation of (M), obtained with the current set of variables and constraints. To compute the reduced costs for a potential new variable s_j, we need to determine its coefficient in all constraints of (M). It can be shown that these coefficients can be computed as follows:

$$a_{pj} = \begin{cases} f_{ij} d_i (c_{k_i} - c_j)^+, & \text{if } p \text{ corresponds to an optimality cut of the } i\text{th subproblem,} \\ f_{ij}, & \text{if } p \text{ corresponds to a feasibility cut for item } i, \\ -1, & \text{if } p \text{ corresponds to the sparsity constraint.} \end{cases}$$

In the first case, k_i denotes the index that has been computed in Step 2 of Algorithm 1 when determining the optimality cut corresponding to p; this value can be saved along with the cut after generation. Altogether, the resulting reduced costs $\sum_{p=1}^{r} a_{pj} \bar{\nu}_p$ can be computed in $O(r)$ time for a given candidate variable s_j. Note that all objective coefficients for s-variables in (M) are zero.

In our approach, we assume that a potentially huge, but fixed list of available box types is given. We thus do not have to deal with a complex pricing problem but only need to check all candidates from the list one after another, which is possible due to the fast computation of reduced costs.

5 Numerical Experiments

The focus of our preliminary experimental evaluation lies on the comparison of different approaches for the column generation subroutine. They differ in the choice of variables to be added when there are multiple candidates with negative reduced costs. More precisely, we compare the following approaches:

- **mostneg1**: only the candidate with the most negative reduced costs is added;
- **mostneg5**: only the five candidates with the most negative reduced costs are added;
- **random5**: five random candidates with negative reduced costs are added.

For a clear comparison that is independent of further implementation details, we only present results for the LP-relaxation of the master problem (M), where the sparsity constraint as well as all feasibility cuts are added from scratch and optimality cuts are separated dynamically until all of them are satisfied.

In particular, all strategies lead to the same dual bound. We are thus mostly interested in a comparison of the resulting running times, which are given in CPU seconds in the following table (**time**). Moreover, we state the percentage of variables that are explicitly used in the LP relaxation, out of all potential variables (**%var**), and the percentage of variables with non-zero value in the final solution of the LP relaxation, with respect to the present variables (**%nz**). Finally, we report the number of generated optimality cuts (**#cuts**), the number of successful column generation iterations (**#cg**), and the number of times the LP relaxation is solved (**#lp**).

For our experiments, we produced all instances as follows: given a number of items (**n**), a step size in cm (**step**), and a solution cardinality (**K**), we define the set of potential box types by enumerating all dimensions that are positive multiples of **step**, up to 200 cm. The items are generated by choosing all dimensions uniformly at random from the range $\{1\,\text{cm}, \ldots, 200\,\text{cm}\}$; we allow rotating items in order to fit them into a box. The demands d_i are chosen uniformly at random from $\{1, \ldots, 5\}$.

Some typical results are shown in Table .1, where each row represents averages over 10 different random instances. Here, we initialize all approaches with only one box type, which measures 200 cm in all dimensions. Our preliminary results suggest that the fastest of the considered column generation strategies is **mostneg5**, followed by **random5** for the hardest instances. Concerning the number of LP iterations, again **mostneg5** yields the best results, but **mostneg1** now outperforms **random5**. In all approaches, the percentage of generated variables (**%var**) is rather small, in particular for the instances with more potential box types. This makes column generation a promising approach, although many further improvements are necessary to make it competitive, e.g., a more reasonable initial set of variables.

Table .1 Comparison of column generation strategies

instances			mostneg1						mostneg5						random5					
n	step	K	time	%var	%nz	#cuts	#cg	#lp	time	%var	%nz	#cuts	#cg	#lp	time	%var	%nz	#cuts	#cg	#lp
20	20	5	12.6	7.7	47.1	79.9	15.8	30.3	5.0	11.9	31.3	80.7	5.4	15.6	9.1	20.9	16.5	118.6	44.9	80.7
20	20	10	18.4	11.4	52.4	80.8	24.0	44.5	6.5	17.1	34.1	84.6	7.6	21.1	7.8	26.9	21.3	112.7	58.3	103.2
20	10	5	111.6	1.3	48.0	81.3	18.5	34.5	59.1	2.4	23.7	87.6	7.7	19.5	120.1	4.9	10.1	162.4	74.8	126.1
20	10	10	186.9	2.0	45.8	87.1	29.2	52.9	82.7	3.4	24.2	93.7	10.6	27.3	104.8	6.8	12.2	138.3	103.1	166.0

References

1. Gurumoorthy, K.S., Hinge, A.: Go Green: A Decision-Tree Framework to Select Optimal Box-Sizes for Product Shipments. In *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, vol. 13717, pp. 598–613, 2023.
2. Que, Q., Yang, F., Zhang, D.: Solving 3D packing problem using Transformer network and reinforcement learning. *Expert Systems With Applications*, vol. 214, pp. 119153, 2023.
3. J. N. Das, M. K. Tiwari, A. K. Sinha, and V. Khanzode, Integrated warehouse assignment and carton configuration optimization using deep clustering-based evolutionary algorithms. *Expert Systems with Applications*, vol. 212, pp. 118680, 2023.
4. V. Zeighami and F. Soumis: Combining Benders' Decomposition and Column Generation for Integrated Crew Pairing and Personalized Crew Assignment Problems. *Transportation Science*, vol. 53, no. 5, pp. 1479–1499, 2019.
5. Y. Rist and M. Forbes: A column generation and Combinatorial Benders Decomposition algorithm for the Selective Dial-A-Ride-Problem. *Computers & Operations Research*, vol. 140, pp. 105649, 2022.
6. M. I. Restrepo, B. Gendron, and L.-M. Rousseau: Combining Benders decomposition and column generation for multi-activity tour scheduling. *Computers & Operations Research*, vol. 93, pp. 151–165, 2018.
7. C. Wang, R. Liu, and Z. Wu: Combining Benders Decomposition and Column Generation for Physician Scheduling in Fever Clinics During Covid-19 Pandemic. *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 4, pp. 5969–5982, 2024.
8. Z. Lan, S. He, and Y. Xu: Combining Benders decomposition and column generation for scheduled service network design problem in rail freight transportation. *Transportmetrica A: Transport Science*, vol. 17, no. 4, pp. 1382–1404, 2021.
9. İ. Muter, Ş. İ. Birbil, and K. Bülbül: Benders decomposition and column-and-row generation for solving large-scale linear programs with column-dependent-rows. *European Journal of Operational Research*, vol. 264, no. 1, pp. 29–45, 2018.
10. I.R. de Farias Jr., G.L. Nemhauser, A polyhedral study of the cardinality constrained knapsack problem, *Mathematical Programming*, vol. 96, pp. 439–467, 2003.
11. D. Bertsimas, B. Van Parys, Sparse high-dimensional regression, *The Annals of Statistics*, vol. 48, no. 1, pp. 300–323, 2020.
12. J.F. Benders, Partitioning procedures for solving mixed-variables programming problems, *Numerische Mathematik*, vol. 4, no. 1, pp. 238-252, 1962.
13. G. Desaulniers, J. Desrosiers, M. M. Solomon, *Column Generation*, Springer New York, 2005.