Heuristic Search for a Real-World 3D Stock Cutting Problem

Katerina Klimova and Una Benlic

Abstract Stock cutting is an important optimisation problem which can be found in many industries. The aim of the problem is to minimize the cutting waste, while cutting standard-sized pieces from sheets or rolls of a given material. We consider an application of this problem arising from the packing industry, where the problem is extended from the standard one or two dimensional definition into the three dimensional problem. The purpose of this work is to help businesses determine the sizes of boxes to purchase so as to minimize the volume of empty space of their packages. Given the size of a real-world problem instances, we present an effective Adaptive Large Neighbourhood Search heuristic that is able to decrease the volume of empty space by an average of 22% compared to the previous approach used by the business.

Keywords Cutting and packing · Adaptive neighborhood search · Heuristics

1 Introduction

Stock cutting is a well-known optimisation problem arising from important practical applications. It consists in cutting standard-sized pieces of stock material (e.g., paper rolls or sheet metal) so as to minimize the amount of wasted material. According to a study conducted by a leading international packing company, 50% of the packing volume is air. Considering that Amazon alone dispatched over 5 billion orders in 2017, the potential for packing improvement is massive. In the ideal case scenario, each order would be packed into a custom-made box that fits its

K. Klimova (✉)
Satalia, Camden, London, UK
e-mail: kat@satalia.com

U. Benlic
School of Electrical and Automation Engineering, East China Jiaotong University, Nanchang, China

J. S. Neufeld et al. (eds.), *Operations Research Proceedings 2019*, Operations Research Proceedings, https://doi.org/10.1007/978-3-030-48439-2_8

dimensions. However, this is generally impossible from the practical stance as the packing process would get significantly slower and costly—a suitable box would need to be produced for each order.

The purpose of this work is to determine the three-dimensions of a given finite number of box types available for packing to help businesses reduce the amount of empty packing volume. As stock cutting is known to be NP-complete [1], we propose an Adaptive Large Neighbourhood Search heuristic based on different repair and destroy move operators. The heuristic iterates between a destruction and a repairer phase. The destruction phase is the diversification mechanism which consists in removing a subset of items (elements) from a given complete solution. This is based on fast destroy operators to escape from local optima, while the repairer phase is the intensification mechanism that makes use of greedy move operators to lead the search towards new quality solutions. Experimental results on a set of real-world instances show an average decrease of around 22% in air volume compared to the solutions used by the business.

2 Literature Review

Different formulations and applications of the cutting problem have been studied in the literature since the 60s. The first problem definition was the minimization of cost for cutting a given number of lengths of material from stock material of a given cost. A linear programming approach for this problem was proposed by Gilmore and Gomory [2]. Even though the problem was first defined as one dimensional, the definition was soon extended to consider two dimensions. For instance, the work by Gilmore and Gomory [3] presents a solution to multistage cutting stock problems with two or more dimensions. More recently, Belov et al. [4] proposed a branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. In [5], Hifi presented a combination of dynamic programming and hill climbing for the two-dimensional stock cutting problem. Both one and two dimensional stock cutting problems can be frequently found in practice, from cutting wires to cutting boxes and corrugated paper.

Despite its practical applications in the packing industry, only limited research has been done on the three-dimensional stock cutting problem [6], while more attention has been devoted to the closely related 2D and 3D packing problem that consists in packing items into minimal number of containers [7]. We present the first heuristic approach based on the Adaptive Large Neighborhood Search [8] framework for the 3D stock cutting problem.

3 Formal Definition

The problem considered in this work is encountered at almost every online shipping company, where a decision has to be made on the sizes of packing boxes that the business needs to order so as to minimize the volume of empty space of their packages. For practical reasons, the maximum number of different box types (sizes) must not be exceeded, which is generally from three to twenty box types for the majority of businesses.

Given a large number of orders consisting of different items, the problem is then to determine the box types (dimensions) that the business needs to purchase, along with their corresponding quantities, while ensuring that the permitted limit of box types is not exceeded. We further take into account the common practice of item consolidation (placement) into a single box with the aim to minimize empty volume. These consolidated items then form a single object of a cumulative volume. To determine the dimensions of this object, we rotate every item such that x and z are the longest and the shortest dimensions respectively. The longest x dimensions across each item to consolidate becomes the x dimension of the new consolidated object. We determine the y dimension of the new object in the same manner, while the z dimension is determined given the new x and y dimensions and the cumulative volume.

Let n be the maximum number of box types and let $I = \{1, \ldots, m\}$ be a set of m historical orders with corresponding dimensions $s^I_{x,i}, s^I_{y,i}, s^I_{z,i}, i \in I$, such that $s_{x,i} \geq s_{y,i} \geq s_{z,i}$. The volume v of an item is then computed as $s_{x,i} * s_{y,i} * s_{z,i}$.

The 3D cutting problem consists in (1) determining the x, y, z dimensions of each box b, represented by the decision variables $s^B_{d,b} \geq 0, d \in \{x, y, z\}, b = 1 \ldots n$; and (2) in determining the assignment of each item $i \in I$ to boxes, where $u_{b,i} \in 0, 1, b = 1 \ldots n, i \in I$ is a binary variable that indicates if item i is assigned to box b. The complete mathematical model is given below.

$$\min \sum_{i \in I, b \in B} u_{b,i}(v^B_b - v^I_i), s.t. \tag{1}$$

$$\sum_{b \in 1..n} u_{b,i} = 1, \forall i \in I \tag{2}$$

$$s^B_{x,b} \geq s^B_{y,b} \geq s^B_{z,b}, \forall b = 1 \ldots n \tag{3}$$

$$s^B_{d,b} \geq u_{b,i} s^I_{d,i}, \forall i \in I, b = 1 \ldots n, d \in \{x, y, z\} \tag{4}$$

$$u_{b,i} \in \{0, 1\}, \forall i \; I, b = 1 \ldots n \tag{5}$$

$$s^B_{d,b} \in \mathbb{N}, \forall b = 1 \ldots n, d \in \{x, y, z\} \tag{6}$$

Equation (1) defines the objective which is to minimize the difference between the box volume v^B and the item volume v^I if item is assigned to the given box. Equation (2) ensures that each order is assigned to exactly one box type, while Eq. (3) ensures that dimensions x and z are the largest and the shortest box dimensions respectively. Equation (4) ensures that each item fits the box assigned to it.

Although the above formulation could be linearized, the problem still remains hard to solve for the existing exact solvers. The reason for this is the definition of problem where input can be millions of items which need to be assigned to one of tens of boxes while applied constraints leave the search space too large.

4 Proposed Approach

4.1 General Framework

A solution to the 3D stock cutting problem can be represented as an array S of integers, where each element of the array corresponds to an item $i \in I$, while $S(i)$ is the box $1 \leq b \leq n$ assigned to item i. Starting from a random assignment of items to boxes, the proposed algorithm iterates between a destroy and a repair procedure, where the destroyer consists in deallocating a selection of items from the solution for the purpose of diversification, while the repairer reconstructs the partial solution by reallocating all the items removed in the destroyer phase. A distinguishing feature of the proposed Adaptive Large Neighborhood Search (ALNS) approach is the use of multiple move operators during both the destroyer and the repairer phase.

Let $\mathcal{M} = \{(m_1^d, m_1^r), \ldots, (m_k^d, m_k^r)\}$ be the set of combinations (pairs), where m^d and m^r are the move operators used in the next destroyer and repairer phase respectively. Each iteration of ALNS first consists in adaptively selecting a pair $(m^d, m^r) \in \mathcal{M}$ as described in Sect. 4.3. The algorithm then proceeds by applying α moves with operator m^d to the current solution to diversify the search, followed by α moves with operator m^r to reconstruct the solution, where α is a parameter that controls the diversification strength ($\alpha = 100$ in our experiments). Finally, the algorithm updates the best recorded solution if the solution following the repair phase constitutes an improvement. The main algorithmic framework of the proposed ALNS is given in Algorithm 1.

4.2 Move Operators

Move operators are the key element of a Large Neighborhood Search algorithm. We distinguish between two types of move operators—destroyers and repairers. Given a complete solution, each move of a destroyer deallocates an item from

Algorithm 1 ALNS framework

$S \Leftarrow buildInitialSolution$
$\mathcal{M} \leftarrow \{(m_1^r, m_1^d), \ldots, (m_k^r, m_k^d)\}$ /*set of move operator pairs*/
$S_{best} \Leftarrow S$
while Stopping condition is not met **do**
  $(m^r, m^d) \leftarrow selectMoveOperatorPair(M)$
  $S \Leftarrow destroy(S, m^d, \alpha)$
  $S \Leftarrow repair(S, m^r, \alpha)$
  if $cost(S_{best}) > cost(S)$ **then**
    $S_{best} \Leftarrow S$
  end if
end while

its allocated box type leading to a partial solution. Given a partial solution, each move of a repairer reassigns an item to a box. Since escaping from local minima is especially difficult for very large data sets with small number of box types available, the number of destroyers for our ALNS exceeds the number of repairers.

The proposed approach makes use of five types of destroy operators: (1) random operator consists in deallocating from a solution a randomly selected item; (2) best operator consists in removing from the solution an item with the largest volume of empty space; (3) smaller container operator removes the smallest item from a randomly selected box type; (4) larger container removes the largest item from a randomly selected box type; and (5) clustered operator deallocates from the solution an item from a selected cluster, where a cluster is formed of α items of similar dimensions.

Three move operators are used during the repairer phase: (1) random operator that assign a deallocated item to a randomly selected box type; (2) best operator that assigns a deallocated item to the best fitting box type so as to minimize the volume of empty space; and (2) dimension-fixed repairer that assigns a deallocated item to a box type only if the assignment does not lead to a change in the box dimensions.

4.3 Adaptive Procedure for Operator Selection

Given five destroy and three repair operators, the number of operator combinations in $\mathcal{M}$ (see Algorithm 1) is fifteen. Before the first iteration of the destroy/repair phase, each pair $p_k \in \mathcal{M}$ has an equal probability $p_k = 1/|\mathcal{M}|$ of selection. This probability is then adaptively updated based on the performance of the selected operator pair at the end of the ALNS iteration. Let $times(k), k \in \mathcal{M}$ be the number of times that operator pair p was used by ALNS, and let $score(k)$ be the number of times that the solution obtained after an application of k is better than the solution from the previous ALNS iteration in terms of the objective value. The updated probability p_k of using k in the next ALNS iteration is determined as $p_k = v_k/q$, where $v_k = p_k * (1 - \epsilon) + \epsilon * (score(k)/times(k))$, and $q = \sum_{k \in \mathcal{M}} p_k$. ϵ is a parameter that takes a value in the range [0, 1].

Table 1 Results of 10 independent runs

Data set	Scenario	Orders	Templates	Best total (m^3)	Avg total (m^3)	Avg void/order (L)
1	1	277,000	4	1482	24,415	6.53
1	2	277,000	9	757	918	2.73
2	1	2,100,000	4	145,514	148,556	69.29
2	2	2,100,000	9	143,952	146,711	68.55

A move operator pair $n \in \mathcal{M}$ to be used in the next iteration of ALNS is then determined using the well-known roulette selection strategy based on its selection probability p_n. To avoid premature convergence towards a single move operator pair, a pair $p_k \in \mathcal{M}$ is selected at random with a probability γ, where γ is a parameter.

5 Computational Results

This section presents computational results on two real-world data instances and two scenarios. First scenario's maximum number of available box types is limited to four types and second scenario's limit is nine types. Unfortunately, we are unable to disclose any details on the used data instances or the actual solutions used by the business.

We perform 10 independent runs for each instance and scenario, where each run is limited to 20 min that was deemed acceptable for the client. For each case, Table 1 shows the best and the average total volume of empty space across all the runs, as well as the average void in liters per order. We include average void per order as it was one of main KPIs, the value in table represents this value averaged over the 10 runs.

In case of the first data instance, the dimension of the largest box for scenario 1 is $600 \times 590 \times 420$ mm with 44,501 orders ($\sim$16%) larger than $330 \times 280 \times 265$, and $600 \times 592 \times 590$ mm for scenario 2 with 771,040 ($\sim$37%) of items being larger than $444 \times 374 \times 195$ mm. It is important to note that the data sets are strongly heterogeneous in dimensions.

6 Conclusion

This paper presents the first application of Adaptive Large Neighborhood Search (ALNS) framework to a real-work 3D stock cutting problem that arises from online shipping industry. The key elements of ALNS is a set of destroy and repair move operators that are selected in a probabilistic and adaptive manner. The proposed approach has been adopted by our client (a leading packing company) and is able to

report a reduction in the total volume of empty space of their packages by around 22% on average compared to their previous solution.

References

1. Blazewicz, M., Drozdowski, M., Boleslaw, S., Walkowiak, R.: Two dimensional cutting problem: basic complexity results and algorithms for irregular shapes. Found. Control Eng. **14**(4), (1989)
2. Gilmore, P.C., Gomory, R.E.: A linear programming approach to the cutting-stock problem. Oper. Res. **9**(6), 849–859 (1961)
3. Gilmore, P.C., Gomory, R.E.: Multistage cutting stock problems of two and more dimensions. Oper. Res. **13**(1), 94–120 (1965)
4. Belov, G., Guntram S.: A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. Eur. J. Oper. Res. **171**(1), 85–106 (2006)
5. Hifi, M.: Dynamic programming and hill-climbing techniques for constrained two-dimensional cutting stock problems. J. Comb. Optim. **8**(1), 65–84 (2004)
6. De Queiroz, T.A., et al.: Algorithms for 3D guillotine cutting problems: unbounded knapsack, cutting stock and strip packing. Comput. Oper. Res. **39**(2), 200–212 (2012)
7. Martello, S., Pisinger, D., Vigo, D.: The three-dimensional bin packing problem. Oper. Res. **48**(2), 256–267 (2000)
8. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. **40**(4), 455–472 (2006)