A Scalable Approach for the K-Staged Two-Dimensional Cutting Stock Problem*

Frederico Dusberger and Günther R. Raidl

Abstract This work focuses on the K-staged two-dimensional cutting stock problem with variable sheet size. High-quality solutions are computed by an efficient beam-search algorithm that exploits the congruency of subpatterns and takes informed decisions on which of the available sheet types to use for the solutions. We extend this algorithm by embedding it in a sequential value-correction framework that runs the algorithm multiple times while adapting element type values in each iteration and thus constitutes a guided diversification process for computing a solution. Experiments demonstrate the effectiveness of the approach and that the sequential value-correction further increases the overall quality of the constructed solutions.

1 Introduction

We consider the *K-staged two-dimensional cutting stock problem with variable sheet size* (K2CSV) in which we are given a set of n_E rectangular *element types* $E = \{1,\dots,n_E\}$, each $i \in E$ specified by a height $h_i \in \mathbb{N}^+$, a width $w_i \in \mathbb{N}^+$, and a demand $d_i \in \mathbb{N}^+$. Furthermore, we have a set of n_T *stock sheet types* $T = \{1,\dots,n_T\}$, each $t \in T$ specified by a height $H_t \in \mathbb{N}^+$, a width $W_t \in \mathbb{N}^+$, an available quantity $q_t \in \mathbb{N}^+$, and a cost factor $c_t \in \mathbb{N}^+$. Both elements and sheets can be rotated by 90°. A feasible solution is a set of *cutting patterns* $\mathscr{P} = \{P_1,\dots,P_n\}$, i.e. an arrangement of the elements specified by E on the stock sheets specified by T without overlap and using guillotine cuts up to depth K. Each pattern P_j, $j = 1,\dots,n$, has an associated stock sheet type t_j and a quantity a_j specifying how often the pattern is to be applied, i.e. how many sheets of type t_j are cut following pattern P_j.

Frederico Dusberger — Günther R. Raidl
Institute of Computer Graphics and Algorithms, TU Wien, Vienna, Austria
e-mail: {dusberger|raidl}@ac.tuwien.ac.at
*We thank LodeStar Technology for their support and collaboration in this project.

The K2CSV occurs in many industrial applications and we are thus considering here in particular large-scale instances from industry. For these instances, typically the number of different element and sheet types is moderate but the demands of the element types are rather high. Nonetheless, reasonable solutions need to be found within moderate runtimes. The objective is to find a feasible set of cutting patterns $\mathscr{P}$ minimizing the number of used sheets $c(\mathscr{P})$ weighted by their cost factors, i.e.

$$\min\ c(\mathscr{P}) = \sum_{t \in T} c_t \sigma_t(\mathscr{P}) \tag{1}$$

where $\sigma_t(\mathscr{P})$ is the number of used sheets of type $t \in T$ in the set $\mathscr{P}$.

Each cutting pattern $P_j \in \mathscr{P}$ is represented by a (cutting) tree structure where the leaf nodes correspond to individual (possibly rotated) elements and the internal nodes are either *horizontal* or *vertical compounds* containing at least one subpattern. Vertical compounds always only appear at odd stages (levels), starting from stage one, and represent parts separated by horizontal cuts of the respective stage. Horizontal compounds always only appear at even stages and represent parts separated by vertical cuts. Each node thus corresponds to a rectangle of a certain size (h,w), which is in case of compound nodes the bounding box of the respectively aligned subpatterns. A pattern's root node always has a size that is not larger than the respective sheet size, i.e. $h \leq H_{t_j}$ and $w \leq W_{t_j}$. Analogously to a_j denoting the quantity of sheets cut according to pattern P_j, compound nodes store congruent subpatterns only by one subtree and maintain an additional quantity. In this tree structure, residual (waste) rectangles are never explicitly stored, but can be derived considering a compound node's embedding in its parent compound or sheet.

Each pattern $P_j \in \mathscr{P}$ can be transformed into a *normal form* of equal objective value, hence it is sufficient to consider patterns in normal form only. In normal form, subpatterns of vertical (horizontal) compounds are arranged from top to bottom (left to right), ordered by nonincreasing width (height), and aligned at their left (top) edges, i.e. in case the subpatterns have different widths (heights), remaining space appears to their right (at their bottom). Figure 1 shows a 3-staged cutting tree and the corresponding pattern.

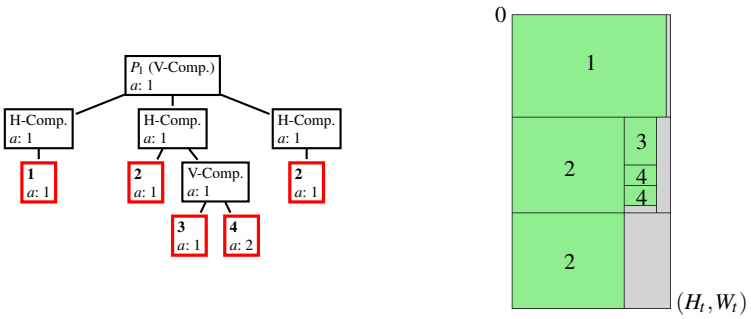

Fig. 1 A three-staged cutting tree (left) and the corresponding cutting pattern (right). The leaves represent actual elements of types 1 to 4 obtained after at most three stages of guillotine cuts.

2 Related Work

Many state of the art approaches for cutting and packing problems employ column generation, dynamic programming, or a combination thereof [10, 3]. However, in the light of large-scale instances, exact approaches cannot compute solutions within reasonable time. Instead, heuristics and metaheuristics are far more promising in obtaining competitive solutions in this context. Lodi et al. [9] give a survey on the classical construction heuristics for cutting and packing problems. In contrast to the exponential runtime of the exact approaches, their runtime complexities range from $O(n \log n)$ to $O(n^3)$, where n is the total demand over all elements. Unfortunately, this advantage is still not enough when the total demand is very high. Moreover, these heuristics are rather inflexible and a strategy that does not prematurely fix the structure of the cutting tree and the sizes of the inserted subpatterns is needed. In a recent paper, Fleszar [6] proposed three more involved construction heuristics for the cutting stock problem with only a single sheet type achieving excellent results in short time. A promising approach to boost the quality of constructed solutions is sequential value-correction, an idea that has been successfully applied, among others, to the two-dimensional strip packing problem by Belov et al. [1] and the two-dimensional bin packing problem with a single sheet type by Cui et al. [4]. The basic idea is to associate each element type with a value signifying that type's priority. Multiple solutions are constructed based on these values, which are continuously adapted according to certain quality criteria, leading to a guided diversification process. The values are increased for types that led to patterns with high relative waste, s.t. those types are used earlier in the following iterations. Considering multiple sheet types, Hong et al. [7] embedded fast construction heuristics in a backtracking framework to address the problem of a meaningful sheet type selection. Solutions are constructed sheet by sheet, where backtracking is applied to revise the choice of a sheet type that led to a poor solution. In recent work, we followed a similar strategy by successfully employing a beam-search algorithm to the K2CSV [5], which serves as a basis for the sequential value-correction framework we propose here and is summarized in the following section.

3 A Congruency-Aware Construction Heuristic for the K2CSV

The main component of the algorithm is a construction-heuristic for computing the pattern for a single sheet, more precisely, the heuristic solves the K-staged two-dimensional knapsack problem. By operating on element types rather than the single elements separately, the heuristic is highly scalable w.r.t. element type demands avoiding the excessive runtimes of element-based construction heuristics. A solution is constructed by considering for each element type $i \in E$ the insertion of a completely filled grid of $a_i^{\text{vert}} \times a_i^{\text{hor}}$ instances of i. The best position and grid size in the sheet pattern are determined according to a heuristic fitness criterion. Note that in contrast to the original algorithm, we use here the more complex *sufficiency cri-*

terion as proposed in [2]. Due to the compact solution representation in the cutting tree, where congruent subpatterns are stored only once with an associated quantity, the heuristic can then simultaneously apply this insertion to as many congruent subpatterns as possible. Another drawback of conventional construction heuristics we avoid is their inflexibility w.r.t. already placed elements. The heuristic considers the current pattern flexible in the sense that compounds are not fixed after their initialization, but can be resized, if necessary to accommodate additional subpatterns.

In order to make meaningful decisions on the choice of sheet types, this construction heuristic is used in a beam-search algorithm generating the solution sheet by sheet. Each node in the search tree corresponds to a (partial) solution, starting with the empty solution at the root. A branch from a node reflects the decision for one of the sheet types from T, the computation of a new pattern on a sheet of that type using the construction heuristic and adding this pattern to the solution with a quantity as high as possible considering the residual element type demands. At each level, all the nodes on that level are evaluated and all but the BW best ones are pruned, where BW is the chosen beam-width. The quality of a partial solution is determined by the average relative waste over all used sheets weighted by the cost factors of the respective types. The lower the relative waste, the better the solution. This procedure continues until all requested elements are used.

4 Sequential Value-Correction

The effectiveness of the beam-search algorithm is demonstrated in [5], where experimental results showed that the approach computes high-quality solutions in relatively short time. Nonetheless, being heuristic in nature, the algorithm is not guaranteed to always find good solutions. In order to compensate for this drawback, we embed it in a sequential value-correction framework.

We associate each element type $i \in E$ with a value v_i that is initially equal to its area, i.e. $v_i = h_i w_i$. The beam-search algorithm is then called for a certain number of iterations and each time a solution has been computed, these values are updated. Let $\text{elems}_i(P_j)$ be the number of elements of type i in pattern P_j and let further $\text{wr}(P_j)$ denote the waste ratio of pattern P_j, i.e. the ratio of unused area to total area in P_j. For a given solution, the value v_i of each element type $i \in E$ is adapted for each of the sheet patterns $P_j \in \mathscr{P}$ according to the following formula:

$$v_i \leftarrow (1-g) \cdot v_i + \frac{g \cdot (h_i w_i)^p}{1 - \text{wr}(P_j)}, \tag{2}$$

where p is a parameter slightly larger than 1 (e.g. 1.02) and g is defined as

$$g = \frac{\text{elems}_i(P_j) \cdot a_j}{d_i + d_i^r} \tag{3}$$

where d_i^r is the residual demand of element type i after computing pattern P_j.

The intuition behind this formula is the following:

- The deterministic weighting factor g ensures that the value of each element type i is updated to an extent that is proportional to the number of elements of type i used in P_j. In particular, we have $g = 0$ if $\text{elems}_i(P_j) \cdot a_j = 0$, i.e. v_i remains unchanged if no element of type i occurs in the pattern.
- As $p > 1$, the values are overproportional to the element types' areas. The intention is to prefer element types that have a relatively large area and are therefore harder to pack.
- Similarly, combinations of elements that yield a high waste ratio will lead to higher values for the respective element types and to their preference in the following iterations as they are hard to combine.

Since the decisions in both the beam-search algorithm as well as the fitness-criterion of the underlying construction heuristic are based on the element types' areas, they can easily be adapted to utilize the values v_i, for $i \in E$, instead.

5 Computational Results

Our algorithms were implemented in C++, compiled with GCC version 4.8.4, and executed on a single core of a 3.40 GHz Intel Core i7-3770. We tested the sequential value-correction approach for 30 iterations and BW=20 (SVC) on the benchmark set by Hopper and Turton [8]. It comprises three instance categories of increasing complexity, each consisting of five randomly generated instances with $|T| = 6$, $2 \leq q_t \leq 4$ and c_t proportional to t's area, for all $t \in T$, and $d_i = 1$, for all $i \in E$. We compared SVC with the HHA algorithm by Hong et al. [7] and with applying the pure beam-search for BW=500 (BS500) and, using the original fitness criterion from [5], for BW=5000 (BS5000). As HHA does not use a stage limit, we set $K = 10$ for our algorithms. Table 1 reports for each category the average percentage of the used area on the sheets $\overline{a(\mathscr{P})}$, to be comparable with the results from [7], and the average runtime $\bar{t}$. Although $d_i = 1$, for all $i \in E$, i.e. we cannot exploit congruency,

			HHA		BS500		BS5000		SVC	
Instance Category	$\lvert E\rvert$	$\lvert T\rvert$	$\overline{a(\mathscr{P})}$	$\bar{t}$(s)	$\overline{a(\mathscr{P})}$	$\bar{t}$(s)	$\overline{a(\mathscr{P})}$	$\bar{t}$(s)	$\overline{a(\mathscr{P})}$	$\bar{t}$(s)
M1	100	6	**98.4**	60	97.7	21.6	**98.4**	34.2	**98.4**	18.6
M2	100	6	95.6	60	**96.3**	12.2	96.1	32.0	**96.3**	13.0
M3	150	6	97.4	60	96.8	45.8	96.5	103.1	**97.6**	35.9

Table 1 Comparison of area utilization for the three instance categories M1 to M3. The best value for $\overline{a(\mathscr{P})}$ in each row is printed in bold.

our algorithms yield competitive results in comparison to HHA. While the pure beam search variants achieve the same or slightly worse results, SVC yields the best results for all categories demonstrating the effectiveness of our value-correction

strategy. Moreover, the runtimes of SVC are comparable to those of BS500 and better than both those of BS5000 and HHA, which always runs for 60 seconds.

6 Conclusions and Future Work

In this work, we extended a successful constructive algorithm for the K2CSV by a sequential value-correction framework in order to improve the overall quality of the constructed solutions. The basic algorithm, which was presented in [5], is a beam-search constructing a solution sheet by sheet using a congruency-aware construction heuristic, which makes the approach highly scalable and allows it to solve large real-world instances within reasonable time. By computing multiple solutions while adapting values associated to the element types after each iteration, this approach can be seen as a guided diversification process. Experiments on benchmark instances document the effectiveness of this strategy.

In future work, we intend to develop a subsequent improvement heuristic for which the excellent results provided by this algorithm are used as initial solutions.

References

1. Belov, G., Scheithauer, G., Mukhacheva, E.A.: One-Dimensional Heuristics Adapted for Two-Dimensional Rectangular Strip Packing. The Journal of the Operational Research Society **59**(6), 823–832 (2008)
2. Charalambous, C., Fleszar, K.: A constructive bin-oriented heuristic for the two-dimensional bin packing problem with guillotine cuts. Computers & Operations Research **38**(10), 1443–1451 (2011)
3. Cintra, G., Miyazawa, F., Wakabayashi, Y., Xavier, E.: Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. European Journal of Operational Research **191**(1), 61–85 (2008)
4. Cui, Y., Yang, L., Zhao, Z., Tang, T., Yin, M.: Sequential grouping heuristic for the two-dimensional cutting stock problem with pattern reduction. International Journal of Production Economics (2013)
5. Dusberger, F., Raidl, G.R.: A Scalable Approach for the K-Staged Two-Dimensional Cutting Stock Problem with Variable Sheet Size. In: Computer Aided Systems Theory – EUROCAST 2015, LNCS. Springer (2015). *to appear*
6. Fleszar, K.: Three Insertion Heuristics and a Justification Improvement Heuristic for Two-Dimensional Bin Packing with Guillotine Cuts. Computers & Operations Research **40**(1), 463–474 (2013)
7. Hong, S., Zhang, D., Lau, H.C., Zeng, X., Si, Y.: A hybrid heuristic algorithm for the 2D variable-sized bin packing problem. European Journal of Operational Research **238**(1), 95–103 (2014)
8. Hopper, E., Turton, B.C.H.: An Empirical Study of meta-Heuristics Applied to 2D Rectangular Bin Packing - Part I. Studia Informatica Universalis **2**(1), 77–92 (2002)
9. Lodi, A., Martello, S., Vigo, D.: Recent advances on two-dimensional bin packing problems. Discrete Applied Mathematics **123**(13), 379–396 (2002)
10. Pisinger, D., Sigurd, M.: The two-dimensional bin packing problem with variable bin sizes and costs. Discrete Optimization **2**(2), 154–167 (2005)