Solving a Weighted Set Covering Problem for Improving Algorithms for Cutting Stock Problems with Setup Costs by Solution Merging

Benedikt Klocker, Günther R. Raidl[⋆]

Institute of Computer Graphics and Algorithms, TU Wien, Vienna, Austria
{klocker|raidl}@ac.tuwien.ac.at

Abstract. Many practical applications of the cutting stock problem (CSP) have additional costs for setting up machine configurations. In this paper we describe a post-processing method which can improve solutions in general, but works especially well if additional setup costs are considered. We formalize a general cutting stock problem and a solution merging problem which can be used as a post-processing step. To solve the solution merging problem we propose an integer linear programming (ILP) model, a greedy approach, a PILOT method and a beam search. We apply the approaches to different real-world problems and compare their results. They show that in up to 50% of the instances the post-processing could improve the previous best solution.

Keywords: cutting stock problem, discrete optimization, PILOT, beam search, solution merging

1 Introduction

There are many different kinds of cutting stock problems (CSPs) occurring in practice and in theory. They have in common that they ask for a set of patterns, where each pattern is a collection of elements, to satisfy given element demands while minimizing the total costs of the patterns. The classical CSP only considers fixed costs for each individual pattern, but in many practical applications additional setup costs arise whenever the machine has to be set up to cut a different pattern. In such cases finding a solution involving a small number of different types of patterns is often crucial.

Assume we already have a method which solves a given CSP and generates and collects many different patterns during the execution. We formalize the *Cutting Stock Set Cover Problem* (CSSCP), an extension of the weighted set covering problem which exploits all these collected patterns by deriving an optimal

[⋆] We thank Lodestar Technology Ges.m.b.H. for the collaboration, the Austrian Research Promotion Agency FFG for funding this project under contract "Innovationsscheck Plus Nr. 855569" and the Vienna Graduate School on Computational Optimization, financed by the Austrian Science Fund (FWF) under grant W1260.

combination of a subset of them resembling a feasible, possibly new incumbent solution. Solving this subproblem can be seen as a kind of solution merging. It can be applied either as a post-processing or as an intermediate step to lead the pattern construction in a more promising direction.

The methods we investigate here for solving the CSSCP are more specifically used to improve the solutions found by a previously developed solver for the K-staged two-dimensional cutting stock problem with variable sheet size [1]. The solver gets used to solve real-world problems and therefore we have a strong focus on the practical applicability of the algorithm. There exists a lot of literature on the broad field of cutting stock problems [2]. The idea of using generated patterns to combine them to a good solution was already used by Cui et al. [3] who use an integer linear program (ILP) in a 2-phase-approach for the one-dimensional problem. A theoretical analysis of a general weighted set covering problem is done in [4], where no concept like setup costs were considered.

2 Problem Formulation

The goal of this section is to formalize CSSCP in a general manner so that it can be used in the context of different cutting stock problems, including different dimensions. Therefore, we first need to formalize a general setting for the cutting stock problem, which we call the *General Cutting Stock Problem* (GCSP).

Definition 1 (General Cutting Stock Problem (GCSP)).
Let $E = \{1, \dots, n\}$ be a set of elements, $(d_i)_{i=1}^n \in \mathbb{N}^n$ a demand vector and $s^{\max} \in \mathbb{N} \cup \{\infty\}$ the maximal stack size. Further, let T be a set of stock materials and $a_t^{\max} \in \mathbb{N} \cup \{\infty\}$ the maximal amount for each stock material $t \in T$.

A solution is represented by a multiset of patterns, where the structure of patterns is problem-specific. We can associate with each pattern p an element vector $(e_i^p)_{i=1}^n \in \mathbb{N}^n$ which describes how often the element i is contained in the pattern p and a stock material $t_p \in T$ out of which it gets cut. A pattern p has associated problem specific production costs c_p^P and stacking costs c_p^S. We define a solution s as a set of feasible patterns P^s and an amounts vector $(a_p^s)_{p \in P^s} \in \mathbb{N}^{|P^s|}$. The goal is to find an optimal solution s which satisfies

$$\sum_{p \in P^s} e_i^p \cdot a_p^s \geq d_i \quad \forall i = 1, \dots, n \tag{1}$$

$$\sum_{p \in P : t_p = t} a_p \leq a_t^{\max} \quad \forall t \in T \tag{2}$$

and minimizes the total costs

$$c(s) := \sum_{p \in P^s} c_p^P \cdot a_p^s + \sum_{p \in P^s} \left\lceil \frac{a_p^s}{s^{\max}} \right\rceil \cdot c_p^S. \tag{3}$$

If $s^{\max} = \infty$ we define $\left\lceil \frac{a_p^s}{s^{\max}} \right\rceil$ equals 1 if $a_p^s > 0$ and 0 if $a_p^s = 0$.

We further consider the problem variant GCSP' in which demands must exactly be satisfied, that means we replace condition (1) *by*

$$\sum_{p \in P^s} e_i^p \cdot a_p^s = d_i \quad \forall i = 1, \ldots, n. \tag{4}$$

Definition 2 (Cutting Stock Set Cover Problem (CSSCP)).
Let E, $(d_i)_{i=1}^n$, $s^{\max}$, T and $a_t^{\max}$ for $t \in T$ be given as in Definition 1. Furthermore, let P be a given set of feasible patterns (e.g. collected from different heuristic solutions to a GCSP). The CSSCP asks for a solution to the underlying GCSP consisting of patterns in P, i.e. $P^s \subseteq P$ which satisfies the conditions (1) *and* (2) *and minimizes the costs $c(s)$ as defined in* (3).

If we replace condition (1) *by* (4) *we call the problem CSSCP'.*

In our case the set P for the CSSCP is constructed during a very large neighborhood search by collecting all patterns occurring during the search.

3 Solution Approaches

In this section we present four different approaches to solve the Cutting Stock Set Cover Problem, an integer linear programming formulation, which can solve the problem exactly, a greedy approach, which can find good solutions very fast, a PILOT-approach and a beam search.

3.1 ILP Formulation

We start by modeling the CSSCP as integer linear program. Theoretically it can solve the problem exactly, but in practice the approach does not scale well to large instances. Therefore, if we use a time limit it may produce solutions with large optimality gaps. We use integer variables a_p for the amount of each pattern p and helper variables s_p for the number of stacks of the pattern p.

$$\min_{(a_p)_{p \in P}, (s_p)_{p \in P}} \sum_{p \in P} a_p \cdot c_p^{\mathrm{P}} + s_p \cdot c_p^{\mathrm{S}}$$

$$\text{s.t.} \sum_{p \in P} a_p \cdot e_i^p \geq d_i \qquad \forall i \in \{1, \ldots, n\} \tag{5}$$

$$\sum_{p \in P : t_p = t} a_p \leq a_t^{\max} \qquad \forall t \in T \tag{6}$$

$$s_p \cdot s^{\max} \geq a_p \qquad \forall p \in P \tag{7}$$

$$a_p \in \mathbb{N}, s_p \in \mathbb{N} \qquad \forall p \in P$$

If we want to solve CSSCP' we replace constraint (5) by

$$\sum_{p \in P} a_p \cdot e_i^p = d_i. \tag{8}$$

The constraints (5) or (8) ensure that the demands get satisfied and the inequalities (6) guarantee that the maximal amounts for each stock material get respected. Furthermore, the constraints (7) couple the s_p variables with the a_p variables by ensuring that there are enough stacks, so that the maximal stack size $s^{\max}$ gets not exceeded.

3.2 Greedy Heuristic

The idea of this greedy construction heuristic is to rate each pattern depending on the current unsatisfied demands and pick the best pattern as the next one in a greedy manner. It is a fast approach, usually resulting in reasonable solutions. To also consider stacking costs we allow to add a pattern with a given amount at once. Thus, we do not only pick a pattern but also an amount for this pattern in a greedy way. As a rating criteria we use the volume of the elements on the pattern whose demand is not yet satisfied divided by the cost of the pattern and the pattern stack.

Formally, we need a volume value $v_i \in \mathbb{R}_+$ for each element i, which represents the difficulty to put an element i on some pattern. For the one dimensional cutting stock problem this may be the length of the element and for the two-dimensional cutting stock problem, as in our specific case, this can be the area of an element. For a given partial solution s, a pattern p and an amount a we define the following rating criteria

$$r^s(p,a) := \frac{\sum_{i=1}^n \min(a \cdot e_i^p, r_i^s) \cdot v_i}{c_p^{\mathrm{P}} \cdot a + c_p^{\mathrm{S}} \left\lceil \frac{a}{s^{\max}} \right\rceil}$$

where the remaining demand r_i^s is defined by

$$r_i^s := \max\left(0, d_i - \sum_{p \in P} e_i^p \cdot a_p^s\right).$$

The complete greedy approach is given by Algorithm 1.

Algorithm 1: Set Cover Greedy Heuristic

```
$(r_i^s)_{i=1}^n \leftarrow (d_i)_{i=1}^n$, $(a_p^s)_{p \in P} \leftarrow 0$
**while** $\exists i \in \{1,\dots,n\} : r_i^s > 0$ **do**
    $(a, p^{\text{best}}) \leftarrow \arg\max_{(a,p) \in \mathbb{N} \times P : 0 < a \le t_p^{\max}} r^s(p,a)$
    $a_{p^{\text{best}}}^s \leftarrow a_{p^{\text{best}}}^s + a$
    $u_i \leftarrow u_i - e_i^{p^{\text{best}}} \cdot a \quad \forall i = 1,\dots,n$
```

To determine a pattern p and an amount a with a maximal value it is enough to check for each pattern p the values a from the following set

$$A := \{s^{\max}\} \cup \left\{ \left\lfloor \frac{r_i^s}{e_i^p} \right\rfloor, \left\lceil \frac{r_i^s}{e_i^p} \right\rceil : i = 1,\dots,n \right\} \cap \{1,\dots,s^{\max}\}$$

and search the pair with the maximal value $r^s(p, a)$. To do this we iterate for each pattern p through the set A in a descending order and can stop the iteration through A when the value $r^s(p, a)$ decreases.

If we want to solve the problem CSSCP' we have to restrict the algorithm to only use patterns p and amounts a which do not lead to an overproduction. Note, however, that this restriction leads often to bad solutions or to no feasible solution at all since there are few or no possible patterns left at some point. Therefore, we apply a repairing mechanism instead of restricting the patterns, which is presented in the following section.

3.3 Solution Repairing for CSSCP'

To still be able to produce good results with the greedy heuristic when exact demands need to be satisfied, we modify the problem CSSCP' in the following way. We allow that a solution may contain patterns which are a substructure of a pattern in P. This means they get constructed by removing some elements from a pattern p in P. We call this new problem CSSCP".

If we want to solve this new problem we can allow patterns and amounts that lead to overproduction and then try to remove the overproduced elements. This may involve checking some problem specific constraints to verify that the new pattern is still feasible. If we obtain a new feasible pattern we can use it and continue with the greedy algorithm.

3.4 PILOT Approach

The main idea of PILOT is to evaluate each potential extension of a current partial solution by individually completing the extended solution in a greedy way and using the obtained solution value for the considered extension [5]. The extension with the best rating is then chosen and the whole process iterates until a final complete solution is obtained.

In our case we more specifically realize the PILOT approach as follows. For each $p \in P$ compute the best amount a according to the greedy criterion rating $r^s(p, a)$, then filter the best ℓ patterns p according to the same rating. For each of these patterns we copy the current solution, add the pattern with the corresponding amount, apply the greedy heuristic to complete the copied solution and compute the objective value of the completed solution. Then we select a pattern p that leads to the best complete solution. For solving the problems CSSCP' and CSSCP" we can proceed in the same way as in the greedy heuristic.

3.5 Beam Search

The idea of beam search is to perform a branching tree search in a breath-first manner, but since this would need too much time in general, it limits the number of considered solutions on each level by some constant k[6]. To get from one level to the next one all extensions of the current solutions are considered and again the best k solutions get stored for the next level. Since our extensions strongly depend on the amount value, we have no clear levels in the search tree. If we consider each addition of a pattern regardless of the amount as one level, we would end up with current solutions of completely different sizes. This is bad since solutions closer to the finished solution tend to be harder to extend and therefore the rating usually decreases. This would mean that smaller partial solutions, still further away from the finished solution, are preferred and eliminate possibly better larger partial solutions.

A similar phenomenon occurs if patterns have different production costs, because then patterns with smaller costs could get preferred although maybe the volume/costs ratio is smaller than for a more expensive pattern. To prevent this we use levels based on the costs of a pattern.

Let $c^{\text{unit}} := \min_{p \in P}(c_p^{\text{P}})$ be the cost unit for one level. The level of a partial solution s is defined by $l(s) := \left\lceil \frac{c(s)}{c^{\text{unit}}} \right\rceil$. For a partial solution we define the rating

$$r(s) := \frac{\sum_{i=1}^{n} \max\left(d_i, \sum_{p \in P} e_i^p \cdot a_p^s\right) \cdot v_i}{c(s)}.$$

The complete beam search is sketched in Algorithm 2. For solving the problems CSSCP' and CSSCP" we can proceed in the same way as in the greedy heuristic.

Algorithm 2: Set Cover Beam Search Approach

```
Add empty solution to storage of level 0, $l \leftarrow 0$, $F \leftarrow \emptyset$
while $|F| < k$ and partial solution for a level larger or equal $l$ exists do
    for $s$ in storage of level $l$ do
        compute best amount $a$ for pattern $p$ according to $r^s(p, a)$; create
          copy $s'$ of $s$ and add $p$ with amount $a$ to $s'$
        if $r(s')$ is one of the best $k$ ratings in level $l(s')$ then
            add $s'$ to storage of level $l(s')$
    increase $l$
return best solution in $F$
```

4 Computational Results

In this section we present results for our algorithms tested with real-world instances for the K-staged two-dimensional cutting stock problem with variable sheet size. For generating the patterns we use the VLNS described in [1].

The algorithms are implemented in C++ and compiled with g++ 4.8.4. For solving the integer linear program we use Gurobi 7.0 [7]. All tests were performed on a single core of an Intel Xeon E5540 processor with 2.53 GHz and 10 GB RAM.

The input instances consist of real-world instances for the K-staged two-dimensional cutting stock problem with variable sheet size together with a collection of feasible patterns found by the VLNS within five minutes runtime for each instance. If more than 5000 patterns were found in this time only the best

Table 1: Results for 10 instances for the CSSCP.

		VLNS			ILP		Greedy		PILOT		Beam Search	
nr	*demands*	*obj.*	*num patterns*	*total patterns*	*obj.*	*time*	*obj.*	*time*	*obj.*	*time*	*obj.*	*time*
1	118	10.98	12	5000	11.50	3600	13.99	1.58	**11.50**	21.89	13.00	19.34
2	648	15.00	21	5000	16.00	3600	18.50	2.15	18.00	24.32	**16.00**	24.07
3	100	19.51	31	120	18.84	74.06	19.35	0.01	**19.01**	0.17	**19.01**	2.23
4	522	29.00	65	5000	30.25	3600	31.85	0.22	29.20	261.05	**29.10**	133.27
5	153	30.00	43	5000	30.00	3600	32.00	2.21	31.00	43.21	**30.50**	38.70
6	79	44.88	61	219	44.89	0.14	44.93	0.05	**44.89**	2.84	44.93	1.90
7	350	48.53	85	5000	48.04	3600	48.89	2.23	48.37	35.19	**48.12**	89.14
8	2000	117.00	190	5000	114.00	3600	130.48	0.29	**117.00**	18.71	119.00	129.91
9	4830	157.00	252	5000	164.50	3600	184.50	2.91	178.00	2145.45	**176.50**	438.10
10	1600	193.50	320	1554	193.00	0.24	211.00	0.07	200.50	6.35	**194.00**	63.92

5000 patterns were taken. The instances are all real-world instances we got from different users, who are already using the system. It is a broad set of 192 instances of different sizes and different configurations. The parameter ℓ for the PILOT approach and the parameter k for the beam search are both set to 30. We selected 10 representative instances to compare the proposed algorithms.

The results for the CSSCP are shown in Table 1. The *demands*-column shows the total sum of all demands d_i for each instance. The *num patterns*-column shows the number of patterns of the best solution found during the VLNS and the *total patterns*-column shows the total amount of collected patterns $|P|$ during the VLNS. The columns *obj.* contain the objective values and the columns *time* contain the used time in seconds for each algorithm and each instance. As the ILP approach uses much more time its values can be seen as reference values, although for some large instances the heuristics perform better, since the ILP approach has a large remaining optimality gap. The best value of the three heuristic approaches is printed bold for each instance.

The PILOT approach and the beam search outperform the greedy approach but they need much more time compared to the greedy heuristic. The beam search needs in general more time than the PILOT approach although there are some exceptions like in case of instance 9.

Since in practice all users of our algorithm use it to solve the CSSCP" we also want to investigate the results for this problem. Because ILP approach can only solve the CSSCP' and cannot make use of the relaxed conditions of the CSSCP" we omit it from these tests. Table 2 shows the results for the CSSCP" for the same instances as in Table 1. The best values for each instance for the three heuristics VLNS excluded are printed bold.

Also for the CSSCP" the PILOT approach and the beam search outperform the greedy heuristic, although they need much more time. We also see that it is quite hard to improve upon the VLNS, but especially the PILOT approach and the beam search are able to do so in quite a few instances.

If we compare the three heuristics with the solutions from the VLNS over the whole set of 192 instances the greedy heuristic can improve the found solution for 6% of the instances, the PILOT approach can improve 21% of the found solutions and the beam search can improve 32% of the solutions. If we only

Table 2: Results for 10 instances for the CSSCP".

		VLNS			Greedy		PILOT		Beam Search	
nr	*demands*	*obj.*	*patterns*	*coll. patterns*	*obj.*	*time*	*obj.*	*time*	*obj.*	*time*
1	118	10.98	12	5000	14.90	1.64	**12.94**	25.19	**12.94**	50.98
2	648	15.00	21	5000	19.43	2.28	**15.99**	20.01	16.97	38.53
3	100	19.51	31	120	19.54	0.01	**19.01**	0.18	**19.01**	2.89
4	522	29.00	65	5000	33.10	1.82	32.10	275.36	**31.00**	181.39
5	153	30.00	43	5000	32.48	0.79	31.98	44.19	**30.98**	47.61
6	79	44.88	61	219	**44.88**	0.02	**44.88**	2.80	**44.88**	2.61
7	350	48.53	85	5000	48.53	0.28	48.70	32.94	**47.96**	99.06
8	2000	117.00	190	5000	130.47	0.34	**118.00**	18.64	119.99	500.66
9	4830	157.00	252	5000	191.94	3.07	182.96	2271.01	**182.95**	501.86
10	1600	193.50	320	1554	210.98	0.07	200.50	6.27	**194.50**	120.38

consider large solutions with at least 100 sheets the greedy can improve 15%, the PILOT approach 54% and the beam search 50% of the solutions.

5 Conclusion and Future Work

In this paper we formulated a variant of the weighted set cover problem, the CSSCP, which solves a cutting stock problem if a set of feasible patterns is already given. It can be used as a second phase after constructing patterns to improve or find a solution for the cutting stock problem as a post processing or as an intermediate step during the pattern construction. We proposed four solution approaches, an exact ILP model and three heuristics, to solve the problem. Furthermore, we compared them with each other by testing them with real-world instances. The tests have shown that the approaches can be used to improve the found solutions in many cases. Future work may be to combine the presented approaches with a construction heuristic that completes a solution when there are no good patterns anymore in the pattern set.

References

1. 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, Springer (2015) 384–392
2. Cheng, C., Feiring, B., Cheng, T.: The cutting stock problem — a survey. International Journal of Production Economics **36**(3) (1994) 291–305
3. Cui, Y., Zhong, C., Yao, Y.: Pattern-set generation algorithm for the one-dimensional cutting stock problem with setup cost. EJOR **243**(2) (2015) 540–546
4. Yang, J., Leung, J.Y.T.: A generalization of the weighted set covering problem. Naval Research Logistics **52**(2) (2005) 142–149
5. Duin, C., Voß, S.: The Pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs. Networks **34**(3) (1999) 181–191
6. Lowerre, B.T.: The HARPY speech recognition system. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA (1976)
7. Gurobi Optimization, I.: Gurobi optimizer reference manual, version 7.0.1 (2016)