Approaches for the pattern minimization problem-Supplementary material

Gabriel Gazzinelli Guimaraes, Kelly Cristina Poldi*

Instituto de Matemática, Estatística e Computação Científica (IMECC), Universidade Estadual de Campinas (UNICAMP), Rua Sérgio Buarque de Holanda, 651, 13083-859, Campinas, SP, Brasil.

1. Introduction

The supplementary material is divided into four sections, where we detail the structured usage of the proposed upper and lower bounds, discuss the application of our approaches to the bi-objective cutting stock problem with setup costs, provide additional implementation details for the approaches in our paper as well as those in Martin et al. (2022) and Cui et al. (2015), and explore adaptations of our approaches for alternative strategies to balance the number of distinct cutting patterns and overproduction.

2. A detailed explanation of the structured usage of the upper and lower bounds proposed

As follows, we present a general algorithm to find upper and lower bounds on the frequency of the cutting patterns for Formulation 1. The algorithm relies on Propositions 2 to 5, presented in Section 3.2, and on Algorithm 1: A preprocessing algorithm to determine upper bounds on the frequency of the cutting patterns, presented in Section 3.4.1.

The algorithm takes as input the number of cutting patterns used, $\mathcal{K}$, and an upper bound on the frequency of execution of the first cutting pattern, Φ^1, which can be determined using the algorithm presented in Section 3.4.1. Additionally, the algorithm also has as input the vector $\overline{\mathbf{d}} = [d_{s_1}, d_{s_2}, \ldots, d_{s_{|I|}}]$, where s_i represents the indices of the demand vector arranged in non-increasing order, such that if $i_1 \geq i_2$, then $d_{s_{i_1}} \geq d_{s_{i_2}}$, as defined in Section 3.2. As outputs, the algorithm returns the upper and lower bounds on the frequency of the cutting patterns. The following algorithm is used to determine the upper and lower bounds proposed in the paper:

*Corresponding author. ORCID 0000-0002-1649-6843

Email address: ggazzinelli9@gmail.com, poldi@unicamp.br (Kelly Cristina Poldi)

Algorithm 1 An algorithm to determine upper and lower bounds on the frequency of the cutting patterns

for k from 2 to $\mathcal{K}$ **do**
  if $d_{s1} > kd_{s2}$ **then**
    $\Phi^k = \min\{\lfloor(\underline{\beta} - max\{0, \mathcal{K} - k\})/k\rfloor, \lfloor d_{s_1}/k'\rfloor, \Phi^{k-1}\}$
  else
    if $k = 2$ **then**
      $\Phi^k = \min\{\lfloor(\underline{\beta} - max\{0, \mathcal{K} - k\})/k\rfloor, max\{d_{s_2}, \lfloor d_{s_1}/2\rfloor\}, \Phi^{k-1}\}$
    end if
    if $k = 3$ and $2d_{s_2} \leq d_{s_1}$ and $3d_{s_2} > d_{s_1}$ **then**
      $\Phi^k = \min\{\lfloor(\underline{\beta} - max\{0, \mathcal{K} - k\})/k\rfloor, d_{s_2}, \Phi^{k-1}\}$
    end if
    if $k = 3$ and $2d_{s_2} > d_{s_1}$ **then**
      $\Phi^k = \min\{\lfloor(\underline{\beta} - max\{0, \mathcal{K} - k\})/k\rfloor, max\{d_{s_3}, \lfloor d_{s_1}/2\rfloor\}, \Phi^{k-1}\}$
    end if
    if $k > 3$ **then**
      $\Phi^k = \min\{\lfloor(\underline{\beta} - max\{0, \mathcal{K} - k\})/k\rfloor, \Phi^{k-1}\}$
    end if
  end if
end for
for k from 1 to $\mathcal{K}$ **do**
  $\Theta^k = \left\lceil \frac{\beta - \overline{\Phi}^{k-1}}{\mathcal{K}-k+1} \right\rceil$
end for
return Θ^k and Φ^k, for $k = 1, \ldots, \mathcal{K}$.

The first four conditional operators in Algorithm 1 are related to Propositions 2 to 4, while the fifth conditional operator is related to the general case in which the propositions cannot be applied. Furthermore, the values of Θ^k are obtained via the closed-form expression presented in Section 3.2.

3. Remarks on the Cutting Stock Problem with Setup Cost (CSP-S)

We highlight that the contributions proposed in the previous sections are not limited to the context of the Pattern Minimization Problem (PMP). In fact, the improvements developed in this work can be reframed to suit the CSP-S with minimal changes.

An intrinsic part of the PMP definition is a predefined upper bound on the number of used objects, given by $\underline{\beta}$. Such limitation is fundamental to the development of most of the improvements proposed in Section 3. Even though the CSP-S has no pre-established limitations in the problem definition, a valid upper bound on the number of used objects, given by $\overline{\beta}$, can be determined via a simple two-stage method in the CSP-S context.

The first stage consists of solving the associated Bin Packing Problem (BPP) and obtaining a set of cutting patterns corresponding to an optimal solution to the problem. Then, in the second stage, the frequency of the cutting patterns previously determined is post-processed to be equal to the highest value of demand of the items generated by the execution of the cutting patterns. This method was proposed in Martin et al. (2022) as a means to determine an initial value for $\overline{\beta}$.

Naturally, the initial bound may be too loose to produce significant improvements during the early stages of the process. However, it is worth emphasizing that this bound can be iteratively refined using the

solution for the CSP-S obtained in the previous iteration, as demonstrated in Martin et al. (2022). Within a bi-objective approach, we expect the bound to progressively improve as the algorithm advances, ultimately enhancing its overall performance.

Therefore, inequality (9) can be applied in the context of the CSP-S by switching $\underline{\beta}$ with $\overline{\beta}$. As a consequence, the propositions and valid inequalities, as well as the preprocessing procedure, can be applied to the CSP-S by changing $T_{\underline{\beta}}$ to $T_{\overline{\beta}}$ whenever necessary.

4. Additional details regarding the implementation specifics in each paper

The approaches in Martin et al. (2022) were implemented in C++ and utilized GUROBI v.9.1.1 as the ILP solver. The experiments were conducted on a PC equipped with an Intel Xeon E5-2680 processor (2.7 GHz), limited to 4 threads, 32 GB of RAM, and running Ubuntu 16.04 LTS.

The sequential heuristic from Cui et al. (2015) was implemented in C++, utilizing the CPLEX 12.5 solver, and executed on a computer with an Intel Core i7-3632QM CPU (2.20 GHz) and 8 GB of RAM.

As mentioned in our paper, our approaches were implemented in the Julia programming language (v 1.9.4) using GUROBI (version 11.0.3) as the ILP solver. The experiments were conducted on a computer with an Intel i7-8700 processor (3.20 GHz) and 16 GB of RAM.

To solve the instances generated by the CUTGEN1 generator, the authors Martin et al. (2022) adopted a weighted sum approach. They assigned a weight 10 times greater to minimizing the number of objects than to minimizing the number of distinct cutting patterns, following the same strategy as Cui et al. (2015). We also employed this strategy when implementing the model proposed in Cui et al. (2015).

It is worth noting that this strategy may produce solutions that do not minimize total trim loss, although it did not occur in our experiments. To prevent such outcomes, one could assign a higher weight to the objective of minimizing trim loss—for example, a weight equal to the total demand of all item types. While we tested higher weights, they negatively impacted the computational performance of the model proposed in Cui et al. (2015). Therefore, we opted to use the exact same weights as the original paper to better reflect the model's performance.

In Martin et al. (2022), for the Fiber instances, the authors adopted a bi-objective approach, aiming to identify all efficient solutions. This contrasts with our work, which focuses solely on solutions that achieve minimum trim loss.

5. Alternative approaches to balancing the number of distinct cutting patterns and overproduction

Both approaches proposed in this paper can be adapted to either include the overproduction of item types as an additional objective or penalize overproduction within the objective function. To achieve this, we can adopt the same strategy as the post-processing approach presented in Section 7.2, which introduces a set of variables, s_i, to account for the overproduction of items i. In this framework, inequalities (4) and (24) are modified as follows:

$$\sum_{k \in K} \sum_{f \in F_k \setminus \{0\}} j x_{kfi} = d_i + s_i, \quad \forall i \in I,$$

$$\sum_{s \in R} \alpha_i^s r_s + \sum_{u \in Q} \alpha_i^u q_u = d_i + s_i, \quad \forall i \in I.$$

To explicitly address item overproduction as an additional objective, one can augment Formulation 1 and Formulation 2 with the objective of minimizing $\sum_{i \in I} s_i$. The resulting multi-objective problem can then be solved using conventional multi-objective optimization methods.

Alternatively, to address overproduction using a penalizing strategy, the term $\sum_{i \in I} s_i$, multiplied by a small weight, can be added to the original objective functions of Formulation 1 and Formulation 2. This modification prioritizes minimizing the number of cutting patterns while simultaneously reducing item overproduction as a secondary objective.

References

Cui Y, Zhong C, Yao Y. Pattern-set generation algorithm for the one-dimensional cutting stock problem with setup cost. European Journal of Operational Research 2015;243:540–6. doi:10.1016/j.ejor.2014.12.015.

Martin M, Yanasse HH, Salles-Neto LL. Pattern-based ilp models for the one-dimensional cutting stock problem with setup cost. Journal of Combinatorial optimization 2022;44:557—582. doi:10.1007/s10878-022-00848-z.