Model Driven Decision-Making Methods

Elnara Galimzhanova

1 Description of the problem

A clothing company must organize cutting of parts of clothes from a given roll of (costly) material, so that they can be later sewn up into complete clothes. The roll has given (fixed) width, and potentially unlimited length. Each part is represented by a convex 2D polygon with at most k edges (with k not too large, but not too small either), and it must be cut from the roll a given number of times. Because of the markings on the tissue, the orientation of the parts w.r.t. the roll is fixed, but each part can be placed at any point. The problem is to place all parts on the shortest possible stretch of the roll.

2 Problem Representation and Geometric Features

The idea of implementing the task were taken from the article The Dotted-Board Model: a new MIP model for nesting irregular shapes. The mathematical solution is also illustrated by using the dotted-model, within the some changes.

To address nesting problems, a set of geometric constraints must be considered, involving the arrangement of pieces within a specified board. Each piece type is represented by a single piece. The reference point of each piece is denoted over one of its vertices. The board is rectangular.

2.1 Representation of the board

The board in this context is a rectangular area with a specified width W and upper bound L on the length.

- W - board width
- L - upper bound on the board length
- C - L+1 number of columns
- R - W+1 number of rows
- D - C×R total number of board dots

2.2 Indices

- $t, u \in \mathcal{T};\ \mathcal{T} = \{1, ..., T\}$ – t, u are piece types
- $c \in \mathcal{C};\ \mathcal{C} = \{0, ..., C-1\}$ – c is a board column
- $r \in \mathcal{R};\ \mathcal{R} = \{0, ..., R-1\}$ – r is a board row
- $d \in \mathcal{D};\ \mathcal{D} = 1, ..., D$ – d is a board dot. A board dot can also be represented by a pair of values $(c, r) \in \mathcal{C} \times \mathcal{R}$, in which $c = \left\lfloor \frac{d}{R} \right\rfloor$ and $r = d - \left\lfloor \frac{d}{R} \right\rfloor \times R - 1$ and $d = c \times R + r + 1$

2.3 Representation of the piece types

Each piece type t is typically characterized by a convex polygon. This polygon is defined by a set of vertices, and the coordinates of these vertices are relative to a reference point associated with the piece type. We decide on this point by finding the highest corner (the one with the biggest y-coordinate) and the leftmost corner (the one with the smallest x-coordinate) on the shape.

- (0, 0) – reference point of piece type t ($\in \mathcal{T}$)
- x_t^M maximum x limit of the rectangular envelope of piece type t
- C - L+1 number of columns
- R - W+1 number of rows
- D - C×R total number of board dots

2.4 Inner-Fit Polygon (IFP)

Simplifying the nesting problem's geometric constraint, each piece must fit entirely within the board. To ensure this, we apply the inner-fit polygon concept (IFP) introduced by Gomes and Oliveira in 2002. For a given piece type, the inner-fit polygon with respect to the board represents all valid placement points for the piece's reference point, ensuring complete coverage of the board.

IFP_t - inner-fit polygon between piece type t and the board ($t \in \mathcal{T}$)

$\mathcal{IFP}_t$ - is the maximum subset of D such that a piece type t having its reference point at $d \in \mathcal{IFP}_t$ is totally over the board.

The illustration of the IFP is displayed in Figure 1. In this figure, all potential positions of the reference point for polygon 3 are marked with dots.

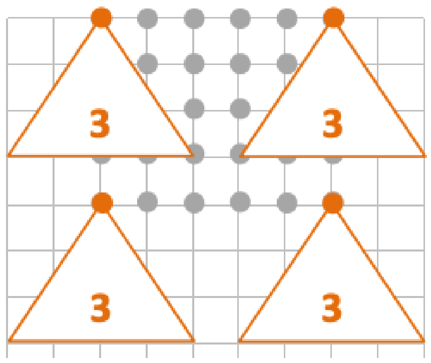

Figure 1: $\mathcal{IFP}_3$ between piece type 3 and a rectangular board.

2.5 No-Fit Polygon (NFP)

The NFP is a polygon describing the feasible locations for two polygons such that they do not overlap. So the NFP will help us placing the items close to one another without overlap. An example is shown in Figure 2 for polygons 3 and 4.

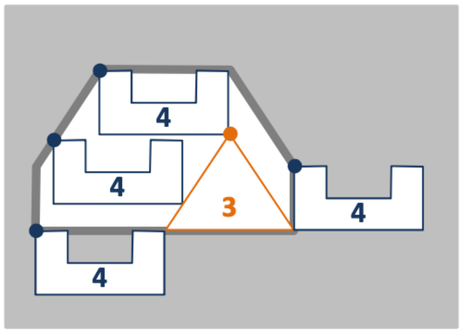

Figure 2: The NFP of polygon 3 and 4, $NFP_{3,4}$.

$NFP_{t,u}$, which stands for the nofit polygon between piece types t and u (where t and u belong to the set $\mathcal{T}$), is a region defined by a set of points. Within this region:

- If the reference point of piece type u falls inside $NFP_{t,u}$, it indicates that the two pieces overlap.
- If the reference point of piece type u is positioned on the edges of $NFP_{t,u}$, it signifies that the pieces touch each other.
- If the reference point of piece type u is situated outside $NFP_{t,u}$, it implies that the two pieces do not touch or overlap.

$NFP^d_{t,u}$ given a board represented by a set of D dots, is defined as the largest subset of D. When the reference point of piece type t is located at dot d, if piece type u has its reference point at any point within $NFP^d_{t,u}$, then pieces u and t overlap. This condition

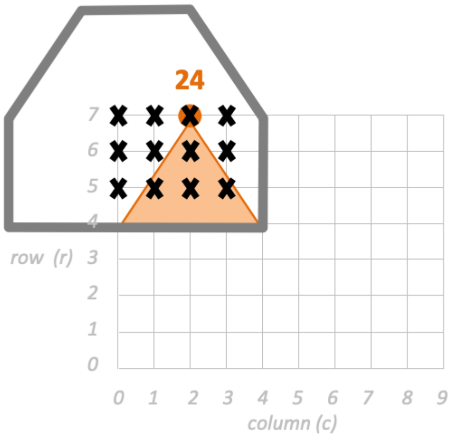

Figure 3: Set of dots in $NFP^2 4_{3,4}$.

applies for all pairs of piece types t and u within the set $\mathcal{T}$ and all dots d within the inner-fit polygon of piece type t (denoted as $\mathcal{IFP}_t$). The set of dots of $NFP^2 4_{3,4}$ is illustrated in Figure 3

The resulting NFP is also convex. It is in fact so that if both input polygons are convex then the resulting NFP is guaranteed to be convex.

2.6 Minkowski

NFP is calculated using by the concept of using Minkowski sums. Lets say we have a rectangle A and a triangle B and we want to compute the NFP with the rectangle as the fixed polygon. The Minkowski sum of two polygons A and B is defined as:

$$A \oplus B = a + b : a \in A, b \in B \tag{1}$$

In Figure 4 an example of the Minkowski sum of two polygons is depicted. Every edge of polygon A is translated by every vertex of polygon B. This results in n new (overlapping) polygons, where n is the number of vertices of B. The union of these polygons if the Minkowski sum $A \oplus B$.

The NFP is related to the Minkowski sum in the sense that:

$$NFP_{AB} := A \oplus (-B) \tag{2}$$

For an accurate calculation of the set of dots in the NFP, it is necessary to perform two steps:

- Translate the second polygon to the coordinate (0,0).
- Reverse the coordinates along the x-axis by negating the x-values.
- Reverse the coordinates along the y-axis by negating the y-values.

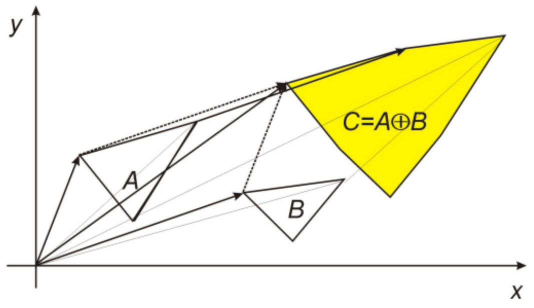

Figure 4: The Minkowski sum of polygons A and B.

2.7 Decision variable

A binary decision variable δ_t^d is defined for each pair (dot, piece type) such that $d \in \mathcal{IFP}_t$.

$$\delta_t^d = \begin{cases} 1, & \text{if the reference point of a piece of type } t \text{ is positioned on dot } d \\ 0, & \text{otherwise} \end{cases}$$

2.8 Objective function and constraints

A feasible solution is achieved by arranging all the pieces within a rectangular board. Therefor we need to minimize the the amount of board length used.

Minimize the total length of fabric:

$$min\ z \tag{3}$$

subject to:

Minimizes the length of the board:

$$x_t^M \times \delta_t^d \le z \qquad \text{for } \forall d \in \mathcal{IFP}_t, \forall t \in \mathcal{T} \tag{4}$$

Ensure that each piece is placed on the fabric only once:

$$\sum_{d \in \mathcal{IFP}_t} \delta_t^d = 1 \qquad \text{for } \forall t \in \mathcal{T} \tag{5}$$

The constraint ensures that the pieces do not overlap:

$$\delta_u^e + \delta_t^d \le 1 \qquad \text{for } \forall e \in \mathcal{NFP}_{t,u}^d, \forall t, u \in \mathcal{T}, \forall d \in \mathcal{IFP}_t \tag{6}$$

Constraint restricts the definition of decision variables to positions where the pieces are entirely contained within the board:

$$\delta_t^d \in \{0,1\} \qquad \text{for } \forall d \in \mathcal{IFP}_t, \forall t \in \mathcal{T} \tag{7}$$

The domain of variable z:

$$z \geq 0$$

2.9 Boundaries

2.9.1 Lower boundary:

The lower bound is determined by the length of the longest piece that must also meet the width requirement of the board.

2.9.2 Upper boundary:

For the upper bound, we repeatedly run a process until we obtain a feasible solution. The variable representing this feasible solution will serve as our upper limit.

3 Experiment

3.0.1 Initialization:

The programming language employed in this experiment is Python. We utilized several libraries, including pulp for developing the MILP model, shapely for working with polygons, NumPy for data manipulation, and matplotlib for polygon visualization.

The data utilized in the experiment was partly obtained from the RCO dataset, which exclusively consists of convex polygons. This dataset includes information such as the dimensions of the board and the polygon shapes that were input.

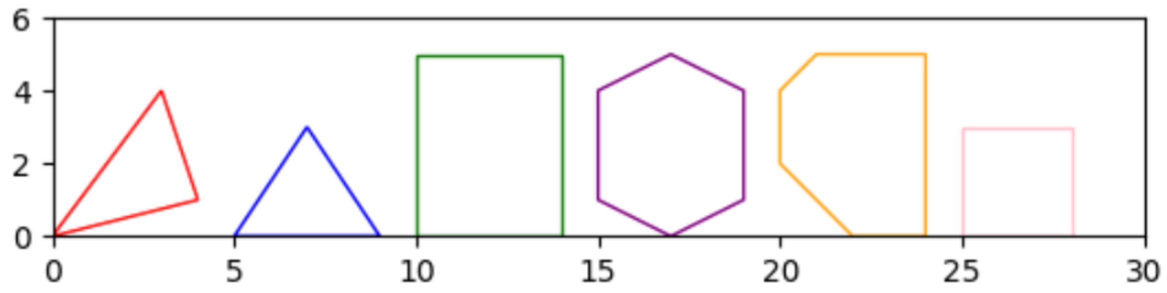

Figure 5: Piece types of instances

solver	time(secs)
GLPK	74.1
COIN	128.00
CBC	109.08
MOSEK	77.59
GUROBI	2.45

Table 1: The result of running solvers on four different polygons.

3.0.2 Analysis of results:

The tested solvers include GLPK, GUROBI, MOSEK, CBC, and COIN.

In Table 1, you can examine multiple runs using GLPK, CBC, MOSEK, COIN and GURIBI solvers. The table clearly demonstrates that GUROBI attained the quickest solution.

In Figure 6, the solutions of CBC, COIN, GLPK, MOSEK and GUROBI for four different polygons are presented. Although the optimal solution for all solvers is 8, their respective results vary, with differences in the width. Notably, CBC and COIN share identical settings.

Figure 7 shows the time in seconds (in the "numbers") it takes for different solvers to solve problems with a varying number of polygon sides. The solvers included in the analysis are CBC, COIN, MOSEK, GUROBI, and GLPK, and the number of polygon sides ranges from 2 to 6.

CBC, exhibits a substantial increase in solving time as the number of polygon sides increases. Notably, the solving time spikes from 16.08 seconds for three sides to 101.59 seconds for four sides and further escalates to 737.42 seconds for five sides. However, the solving time for six sides remains nearly constant at 735.04 seconds.

COIN, follows a similar trend as CBC, but its solving times are consistently higher. As the number of polygon sides increases, COIN's solving time increases dramatically, exceeding 1000 seconds for five sides.

MOSEK, shows a different pattern. While it exhibits a relatively high solving time for two sides (51.27 seconds), its solving time decreases as the number of sides increases, reaching 85.27 seconds for five sides.

GUROBI, consistently outperforms the other solvers. Its solving times are significantly lower across all polygon side counts. GUROBI takes less than 5 seconds to solve problems with up to five sides, making it a favorable choice for applications where efficiency is primary.

GLPK, shows an unusual trend in the data. Its solving time increases dramatically with the number of sides, peaking at 2161.7 seconds for six sides. However, it's important to note that this particular data point was canceled, implying that GLPK may have struggled to solve the problem within a reasonable time frame. GLPK proves to be valuable when the number of convex elements is fewer than five. In cases where

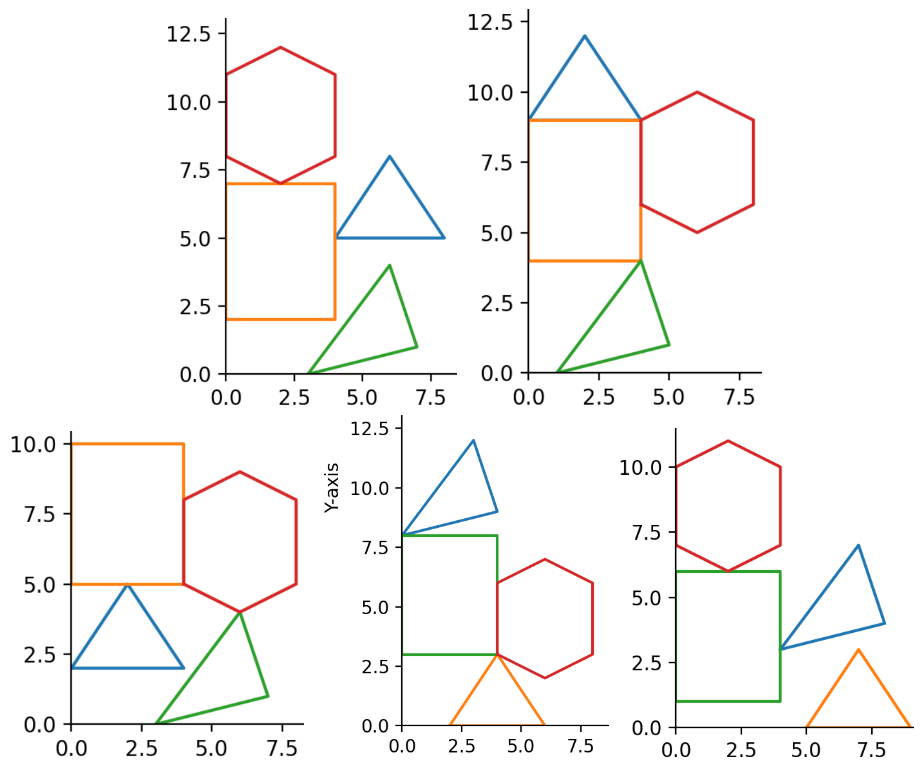

Figure 6: The results obtained from the CBC, COIN, GLPK, MOSEK and GUROBI solvers when using four different polygons.

there are a higher number of convex elements, we encountered issues with GLPK.

The execution of solvers on 6 polygons with default parameters is illustrated in Table 2, and these results will be optimized in the next section.

4 Optimization

The primary issue lies in the fact that when we aim to minimize the fabric's length, the model maximizes the fabric's width. Therefore, an alternative approach is to minimize the fabric's width.

solver	time(sec)
CBC	735.04
COIN	1330.65
MOSEK	191.07
GUROBI	11.71
GLPK	2161.7 (Canceled)

Table 2: Running computations on a set of six polygons.

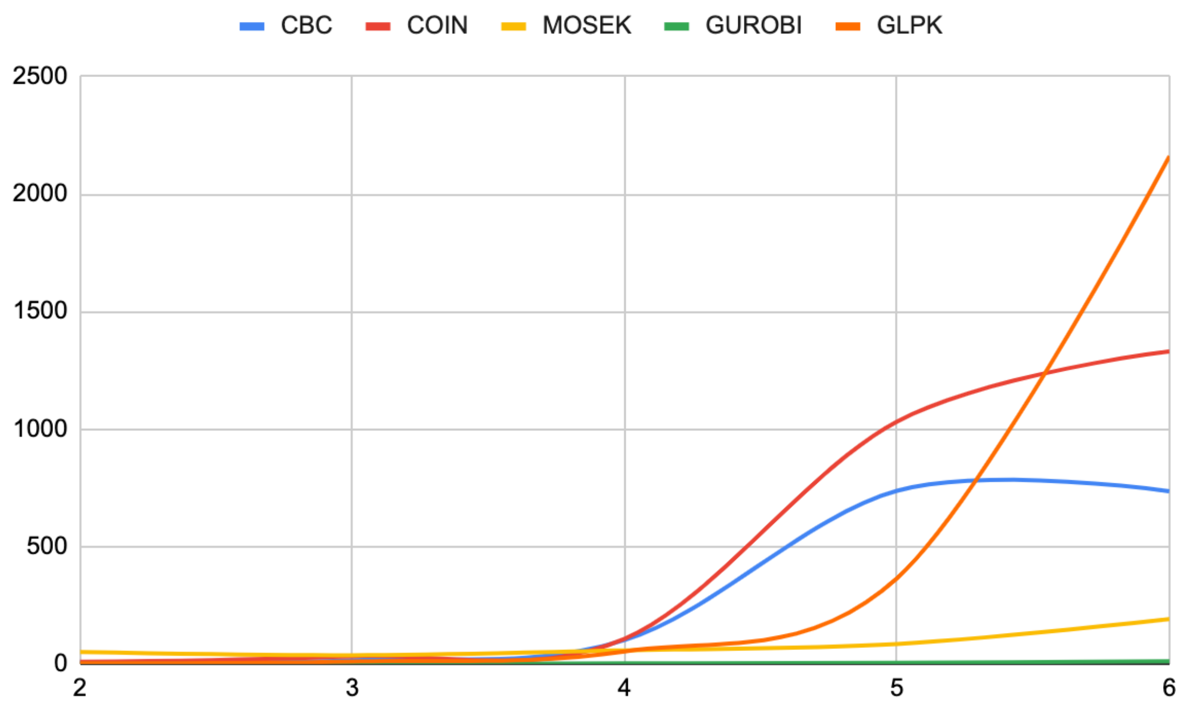

Figure 7: Running solvers on sets of polygons comprising 2 to 6 polygons using their default parameters(time in sec/number of polygons).

So, we decided to minimize the total width and total length of fabric, instead of only length.

Minimize the total width and length of fabric:

$$min\ z_{length} + z_{width} \tag{8}$$

Minimizes the total length of the board:

$$x_t^M \times \delta_t^d \leq z_{length} \qquad \text{for } \forall d \in \mathcal{IFP}_t, \forall t \in \mathcal{T} \tag{9}$$

Minimizes the total width of the board:

$$y_t^M \times \delta_t^d \leq z_{width} \qquad \text{for } \forall d \in \mathcal{IFP}_t, \forall t \in \mathcal{T} \tag{10}$$

The domain of variable z:

$$z_{width}, z_{length} \geq 0$$

Unfortunately, with other solvers, the computation of 8 polygons takes more than 3000 seconds. Therefore, we have decided to use only the GUROBI solver. The modified model executed using the GUROBI solver, which was the fastest solver for up to 8 polygons. The results are shown in Figure 8.In the figure, we can observe the difference

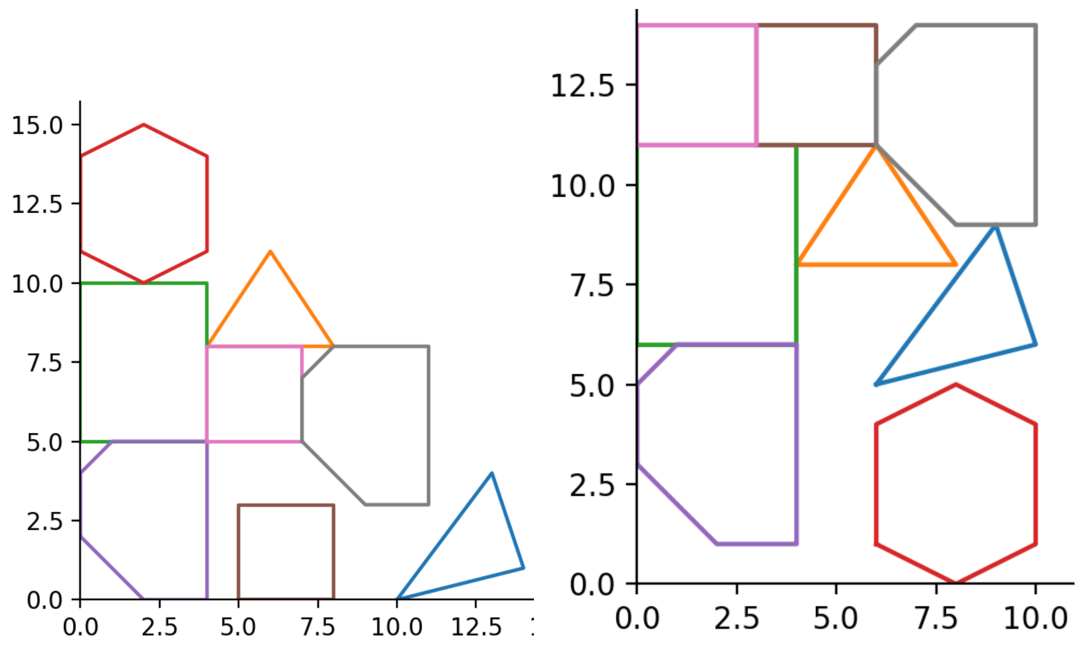

Figure 8: Result of GUROBI solver on 8 polygons without and with changes.

that occurs after changing the model. We are not only minimizing the length value but also the width.

Furthermore, we conducted a comparison of gap results when using the *warmStart* parameter in the 'on' mode (Figure 9). The WarmStart parameter is initialized with delta values from the solver's first computation. Subsequently, one delta value is added at a time, starting from 1. This means that we are incrementally incorporating the delta values. As illustrated in the results, compiling without warm start and providing 4 deltas yield almost identical results. The longest computation time is observed when using 2 delta values. Notably, with only one delta, the gap starts at 20, while the others start from 23.1.

5 Conclusion

The nesting problem emerged as a notably complex problem, requiring an extended execution time. To address this, the problem was tackled using the Dotted Board approach. Various solvers from the pulp library were employed, including CBC, COIN, GLPK, GUROBI, and MOSEK. Regrettably, GLPK proved impractical due to the dataset's size. The fastest solution with a significant difference was achieved by the GUROBI solver.

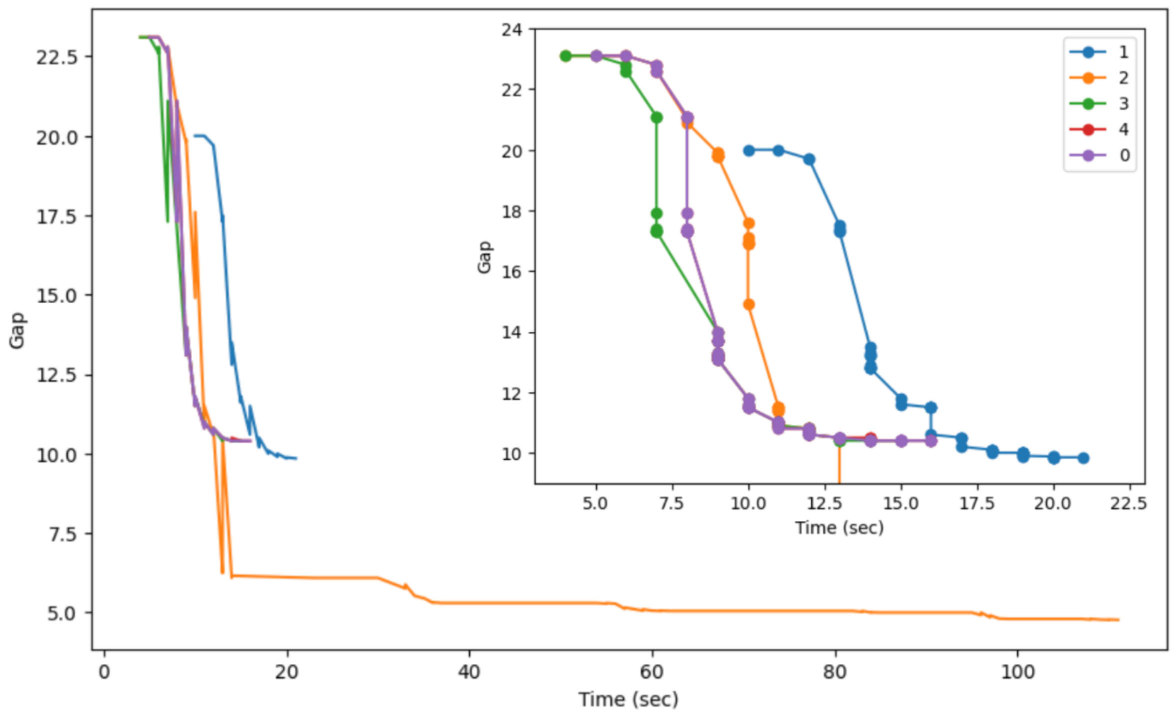

Figure 9: Comparison of gap using the WarmStart parameter: The numbers indicate how many delta values are provided to the model at the beginning.