International Journal of Production Management and Engineering

https://doi.org/10.4995/ijpme.2025.22196
Received: 2024-07-31 Accepted: 2024-11-23

Optimizing cut order planning: A comparative study of heuristics, metaheuristics, and MILP algorithms

Sharif Al-Mahmud[a1], **Jose Alejandro Cano**[b1*], **Emiro Antonio Campo**[b2], **Stephan Weyers**[a2]

[a] University of Applied Sciences and Arts, Fachhochschule Dortmund, Emil-Figge-Str. 44 - 44227 Dortmund, Germany.
[b] University of Medellin. Carrera 87 # 30-65, 050026 Medellin, Colombia.
[a1] sharif.al-mahmud@fh-dortmund.de; [b1] jacano@udemedellin.edu.co; [b2] ecampo@udemedellin.edu.co;
[a2] stephan.weyers@fh-dortmund.de

Abstract:

Cut Order Planning (COP) optimizes production costs in the apparel industry by efficiently cutting fabric for garments. This complex process involves challenging decision-making due to order specifications and production constraints. This article introduces novel approaches to the COP problem using heuristics, metaheuristic algorithms, and commercial solvers. Two different solution approaches are proposed and tested through experimentation and analysis, demonstrating their effectiveness in real-world scenarios. The first approach uses conventional metaheuristic algorithms, while the second transforms the nonlinear COP mathematical model into a Mixed Integer Linear Programming (MILP) problem and uses commercial solvers for solution. Modifications to existing heuristics, combined with tournament selection in genetic algorithms (GA), improve solution quality and efficiency. Comparative analysis shows that Particle Swarm Optimization (PSO) outperforms GA, especially for small and medium-sized problems. Cost and runtime evaluations confirm the efficiency and practical applicability of the proposed algorithms, with commercial solvers, delivering superior solutions in shorter computation times. This study suggests the use of solvers for the COP problem, especially for smaller orders, and reserves PSO and GA for larger orders where commercial solvers may not provide a solution.

Key words:

COP, cut order planning, heuristics, Metaheuristics, MILP, garment manufacturing.

1. Introduction

The apparel industry, a multi-trillion-dollar market, needs to improve production efficiency to remain competitive (Ranaweera et al., 2023; Wijethilake et al., 2023). However, enhancing business processes is crucial for productivity, efficiency, and socio-economic benefits (Yang et al., 2023). Garment production involves multiple steps, including order taking, sample preparation, material procurement, storage, inspection, cutting, sewing, finishing and outbound logistics (Abd Jelil, 2018; Ünal & Yüksel, 2020; Xu et al., 2020). These processes aim to increase productivity and reduce costs, with sewing being the most critical step (Nchalala et al., 2023). Fabric cutting, preceding sewing, accounts for 5-10% of total manufacturing costs and 50-60% of material costs (Alsamarah et al., 2022; Wong & Leung, 2008).

As shown in Figure 1, the cutting process begins with the Cut Order Planning (COP), determining garment quantities and sizes. Subsequent steps include marker making, spreading fabric on cutting tables, cutting according to the marker pattern, identifying cut pieces, and bundling for further processing. COP is crucial for garment manufacturing efficiency and cost (Prasad et al., 2022). It determines optimal fabric cutting patterns to minimize waste (Fister et al., 2010), enhancing sustainability and economic viability (Nasrin & Alam, 2023). Fabric is spread in layers (plies) on a cutting table, forming sections,

To cite this article: Al-Mahmud, S., Cano, J.A., Campo, E.A., Weyers, S. (2025). Optimizing cut order planning: A comparative study of heuristics, metaheuristics, and MILP algorithms. *International Journal of Production Management and Engineering, 13*(1), 1-26. https://doi.org/10.4995/ijpme.2025.22196

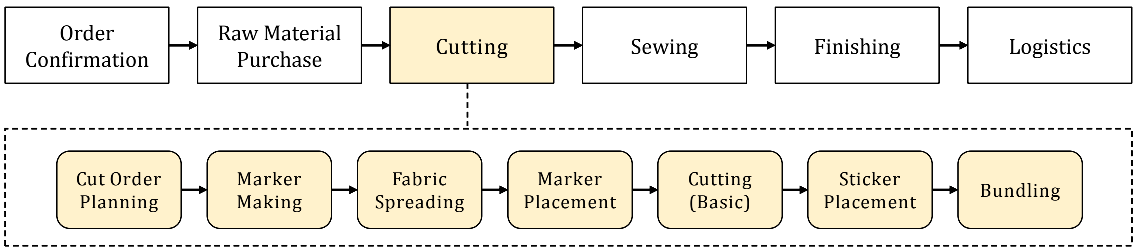


Figure 1. Garment manufacturing process.

while markers guide cutting of irregularly shaped patterns for different sizes (Martens, 2004; Tsao et al., 2022).

Although COP focuses on optimizing the number of fabric plies and sections to minimize production costs and waste, it does not explicitly address the spatial arrangement of garment patterns on the fabric. This spatial arrangement is typically handled by a related optimization problem known as marker making or nesting (Puasakul & Chaovalitwongse, 2016; Yang et al., 2024). Marker making involves placing irregularly shaped garment pieces onto the fabric in the most efficient way possible to minimize material waste. While COP determines how many layers of fabric and sections are needed for a given order, marker making ensures that the actual patterns are arranged optimally within each section, complementing the goals of COP by further reducing fabric waste (Abd Jelil, 2018).

Figure 2 shows a cutting section consisting of five fabric plies and a marker for three different garment sizes, where each color on the patterns represents a specific size. Large orders may be divided into multiple sections due to limitations in the cutting process, and each section may have varying numbers of plies and unique markers (Fister et al., 2010).

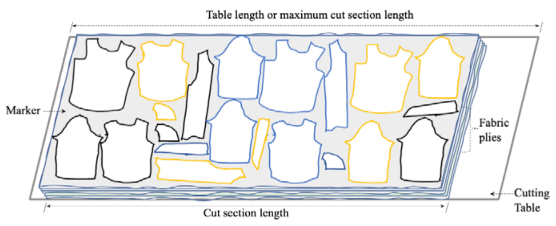


Figure 2. A cutting section of five plies with a marker containing three garments of three sizes.

The problem of COP can be solved using various methods. For instance, consider a buyer's order comprising five sizes (Table 1). This order could be addressed using: (1) a single cut section with one long fabric ply encompassing all sizes; (2) five separate sections, each dedicated to a single size; or (3) a multi-section approach, where the first section consists of 7 plies (size-1 once, size-2 three times, size-3 three times, size-4 twice, and size-5 once), the second section would have 2 plies (size-2 once, size-3 twice, size-4 once, and size-5 three times), and the third section would have 1 ply (size-4 and size-5 each appearing once). A solution for COP can be represented as a matrix with 'm' rows (sections) and 's+1' columns (sizes + ply count). Each row (α) represents a section, with $G_{\alpha\beta}$ denoting the number of occurrences of size β within that section, and P_α indicating the total fabric plies used in section α. To illustrate, consider the 4 × 6 matrix in Figure 3(a), which represents the third solution discussed earlier. The total garments produced by each section α can be calculated as $\sum G_{\alpha\beta} \times P_\alpha$. Figure 3(b) displays the total number of garments per size in each section, while Figure 3(c) confirms the overall quantity of garment output per section.

Table 1. A COP problem (size-wise demand quantity).

Size1	Size2	Size3	Size4	Size5
7	23	26	17	13

While the cutting and packing problem has been extensively researched (Yang et al., 2024), COP in garment manufacturing has received less attention (Fister et al., 2010; Nascimento et al., 2010). However, efficient COP has become increasingly important due to evolving industry demands (Shang et al., 2019; Ünal & Yüksel, 2020; Xu et al., 2020; Prasad et al., 2022; Tsao et al., 2022; Xiang et al., 2022; Ranaweera et al., 2023). COP involves complex constraints beyond size demand, including fabric ply limitations and section length restrictions (Puasakul & Chaovalitwongse, 2016). As the problem's complexity increases, manual methods become inefficient since many alternative solutions can lead to different overall costs. Optimizing COP

$G_{\alpha\beta}$					P_α
1	3	3	2	1	7
0	1	2	1	3	2
0	0	1	1	0	1

α = 1, 2, 3 ==> m=3 sections ==> 3 rows
β = 1, 2, 3, 4, 5 ==> s=5 sizes ==> 5+1columns

(a)

Size 1	Size 2	Size 3	Size 4	Size 5
7	21	21	14	7
0	2	4	2	6
0	0	1	1	0
7	**23**	**26**	**17**	**13**
86				

(b)

Section	Total Garments
1	70
2	14
3	2
Total order	**86**

(c)

Figure 3. (a) Garments per ply per section (b) Garments per size (c) Garments per section.

is NP-complete, as proved by Jacobs-Blecha et al. (1998), necessitating advanced artificial intelligence techniques for near-optimal solutions in realistic scenarios (Abd Jelil, 2018; Abualigah et al., 2021).

Jacobs-Blecha et al. (1998) proposed three greedy heuristic algorithms, though with limited production constraints. Rose and Shier (2007) faced similar limitations. Nascimento et al. (2010) offered a state-space search method with heuristic rules, allowing users to choose between optimal and near-optimal solutions. Shang et al. (2019) employed the iterated greedy algorithm (HFSC), involving initial solution construction and repeated improvement. M'Hallah and Bouziri (2016) proposed five heuristics for initial population generation, but their approach didn't account for limited fabric length. Tsao et al. (2020) modified two of these algorithms (H1 and H5) to address consistent solution production and size prioritization issues. However, these modifications still lack randomization, potentially limiting the search space for metaheuristics using such heuristics as initial solutions.

Genetic algorithms (GA) (Martens, 2004; Abeysooriya & Fernando, 2012a, 2012b; M'Hallah & Bouziri, 2016 ; Tsao et al., 2020; Xu et al., 2020; Prasad et al., 2022) and evolutionary algorithms (EA) (Filipič et al., 2006; Wong & Leung, 2008; Fister et al., 2010) are most common for COP. GA studies often use random selection (Martens, 2004) or roulette wheel selection (Abeysooriya & Fernando, 2012a, 2012b; Tsao et al., 2020). Hybrid approaches combining GA with other methods have been explored (Abeysooriya & Fernando, 2012a, 2012b; Tsao et al., 2020). Other metaheuristics include Ant Colony Optimization (2011), Simulated Annealing, and Tabu Search (M'Hallah & Bouziri, 2016; Tsao et al., 2020). Some studies used commercial LINGO software with integer programming (Silva et al., 2017; Ünal & Yüksel, 2020). These studies employed a two-step process, first building a mathematical model with integer programming and then solving it as an optimization problem using commercial software. Martens (2004) used GA based on nonlinear integer programming (NLIP) and integer programming (IP) models for real-world COP problems. Filipič et al. (2006) converted COP into a knapsack problem. Rose and Shier (2007), Wong and Leung (2008), and Silva et al. (2017) assumed equal marker area for all sizes, oversimplifying the production process. Silva et al. (2017) also predetermined the number of sections, limiting the solution's flexibility for larger orders.

This study incorporates existing heuristics (H1_T2020, H3_MB2016) and introduces three adapted ones (H1+, H3+, H5+). It proposes a GA and adapts a Particle Swarm Optimization PSO algorithm for COP. Additionally, a mixed-integer linear programming (MILP) model is developed, solvable with commercial solvers like CPLEX and OpenSolver. The study aims to compare heuristics, metaheuristics, and solvers for COP, assessing their adaptability and cost-minimization potential in the garment industry. industry. The remainder of this paper is organized as follows: Section 2 presents the mathematical model of the COP. Section 3 introduces solution approaches for the COP. Section 4 details the numerical applications. Section 5 presents the results and discussion, and Section 6 concludes the contributions of this study.

2. Mathematical Modeling

The main objective of COP is to optimize the use of fabric rolls by determining the most efficient way to cut garment components from the available fabric. This involves creating cutting plans that minimize waste while satisfying production requirements such as order quantities, garment sizes, and other constraints. Therefore, to fulfill an order, consider an arrangement for cutting garments of various sizes $\beta = 1, 2, \ldots, s$, each with a particular quantity Q_β to fulfill the order. The cutting process is divided into sections (α = 1, 2, 3, …, m). The decision variables are $G_{\alpha\beta}$, P_α, and Z_α , where $G_{\alpha\beta}$ represents garments of size β in one fabric ply in section α, P_α represents ply count in section α, and Z_α represents whether section α is used. The objective is to minimize the total cost by finding optimal values for these decision variables. The mathematical model in this study assumes that the total cost is influenced by fabric cost, setup cost, and excess production cost. The setup cost is considered constant for all sections. The indices, decision variables, parameters, and mathematical model formulation for optimizing the COP are outlined below.

Indexes

β: sizes (β =1,2,3,…s)

α: sections (α =1,2,3,…m)

Decision variables

$G_{\alpha\beta}$: Number copies of size β in section α

P_α: Number of layers in section α

Z_α: Takes the value of 1 if section α is used; otherwise, 0

Parameters

Y_β: fabric consumption rate for size β (given)

Q_β: demand for size β

CS: cost of setting up a new section (per section)

CE: cost of excess production (per product)

f: cost of using fabric (per length unit)

l_{max}: Maximum length of fabric in any section

P_{max}: Maximum number of plies in any section

P_{min}: Minimum number of plies in any section

$$Min\ \ C_T = \sum_\alpha \sum_\beta G_{\alpha\beta} Y_\beta P_\alpha f + \sum_\alpha Z_\alpha CS + + \sum_\alpha \sum_\beta CE\left(P_\alpha G_{\alpha\beta} - Q_\beta\right) \tag{1}$$

$$\sum_\beta G_{\alpha\beta} Y_\beta \le l_{max} \tag{2}$$

$$P_\alpha \ge P_{min} \tag{3}$$

$$P_\alpha \le P_{max} \tag{4}$$

$$\sum_\alpha G_{\alpha\beta} P_\alpha \ge Q_\beta \tag{5}$$

$$G_{\alpha\beta}\,,\ P_\alpha \in \mathbb{Z}^+,\ z_\alpha \in \{0,1\} \qquad \forall\, \alpha, \beta \tag{6}$$

Equation (1) represents the objective function, which aims to minimize the total cost. This cost includes the fabric cost, setup cost, and excess production cost, which are detailed in the first, second, and third components respectively. Equation (2) enforces the length constraint of a section. Equation (3) and Equation (4) set limits on the number of plies per section. Equation (5) ensures that customer demand is satisfied. Equation (6) mandates integer decision variables for the number of copies of size β in section α and for the number of layers in section α and establishes Z_α as a binary variable.

This study adapts the mathematical formulation from Tsao et al. (2020) but modifies the length constraint due to potential infeasibility. The original model's 100-meter cumulative length limit for all sections proved impractical for large orders. Instead, we impose a length limit on individual sections based on cutting table dimensions, as reflected in Equation 2. This approach aligns with common practices in COP literature (Degraeve & Vandebroek, 1998; Wong & Leung, 2008; Nascimento et al., 2010; Yang et al., 2011; Shang et al., 2019).

3. Solution approaches

This study uses various solution approaches to tackle the COP problem. These approaches involve heuristic techniques to generate random viable solutions, which serve as the initial population for the proposed metaheuristic methods. The use of population-based metaheuristics, such as GA and PSO, can accelerate convergence by employing high-level relay hybridization of metaheuristics to seed the initial population with high-quality solutions. The approach for the final solution involves using general-purpose commercial solvers, specifically CPLEX and OpenSolver, to find the optimal solution for experimental instances that allow it in terms of size and computation time.

3.1. Constructive heuristics

The article examines five constructive heuristics: H1_T2020, H3_MB2016, H1+, H3+, and H5+. H1 and H3 provide unique solutions, while H1+, H3+, and H5+ generate random solutions each time. These were selected based on M'Hallah and Bouziri's (2016) findings on algorithm superiority. H1_T2020, adopted by Tsao et al. (2020), refines M'Hallah & Bouziri's (2016) approach by accounting for fabric length limitations in each section. Algorithm 1 (Appendix I) presents a formal statement of H1_T2020, constructing sections based on minimal demand, ply height limits, and maximum size occurrences within length constraints.

Algorithm 2 (Appendix I) presents H3_MB2016, adapted from M'Hallah & Bouziri (2016). This heuristic creates a single section containing all sizes, with ply count set as the greatest common divisor (GCD) of order quantities, respecting ply height constraints. To meet length limits, garments are removed from frequently occurring sizes. Residual demands are updated, and new sections are generated as needed. The algorithm incorporates the authors' strategy for converting solutions to feasible ones, addressing the original version's lack of length constraints.

The H1+ heuristic, presented in Algorithm 3 (Appendix I), is a slightly modified version of the H1_T2020 algorithm. The key difference in the H1+ heuristic is its randomization of size selection when placing garments within a section. Unlike the original approach, which prioritized sizes from left to right (most likely smallest to largest), H1+ assigns garment occurrences in a random size order. This randomization allows for potentially different solutions with each run of the algorithm.

Algorithm 4 (Appendix I) represents the heuristic H3+, a revised version of M'Hallah & Bouziri's (2016) H3 algorithm. The revision ensures viable solutions, following a similar approach to Tsao et al.'s (2020) modifications to H1 and H5. The quantity of garments per size in the marker was constrained by the maximum allowed without violating length constraints or by demand requirements. Like the H1+ heuristic, H3+ employs randomized size ordering to increase solution diversity.

The H5+ heuristic, described in Algorithm 5 (Appendix I), is a refined version of M'Hallah & Bouziri's (2016) random search approach, further improved by Tsao et al. (2020) and revised in this study. Randomization is introduced by assigning garment sizes in random order and iteratively constructing sections with randomized ply counts. New sections are created until a maximum count ($\overline{m}$=2s) is reached or all demands are met. Final sections use the H3+ heuristic to consolidate remaining demands, if applicable.

Figure 4 shows the solutions obtained using the constructive heuristics H1_T2020, H3_MB2016, H1+, H3+, and H5+ based on the information provided in Table 1. It is important to note that each heuristic produces a distinct COP solution, which is characterized by the total number of garments, garments per size, sections, and plies per section.

3.2. Metaheuristics

Metaheuristic algorithms are non-deterministic approximation methods designed to find near-optimal solutions within a reasonable amount of time (Ezugwu et al., 2021; Gogna & Tayal, 2013). These algorithms can be broadly categorized into two types: single-solution-based and population-based. Single-solution-based algorithms iteratively improve a single solution, while population-based algorithms maintain a diverse set of solutions throughout the search to prevent premature convergence (Katoch et al., 2021; Toaza & Esztergár-Kiss, 2023). Although some metaheuristics draw inspiration from physics, chemistry, social sciences, sports, and other fields, the majority are inspired by nature (Boussaïd et al., 2013; Ezugwu et al., 2021). This study examines the use of GA and PSO as nature-inspired population-based metaheuristics to address the COP problem.

3.2.1. Genetic Algorithm

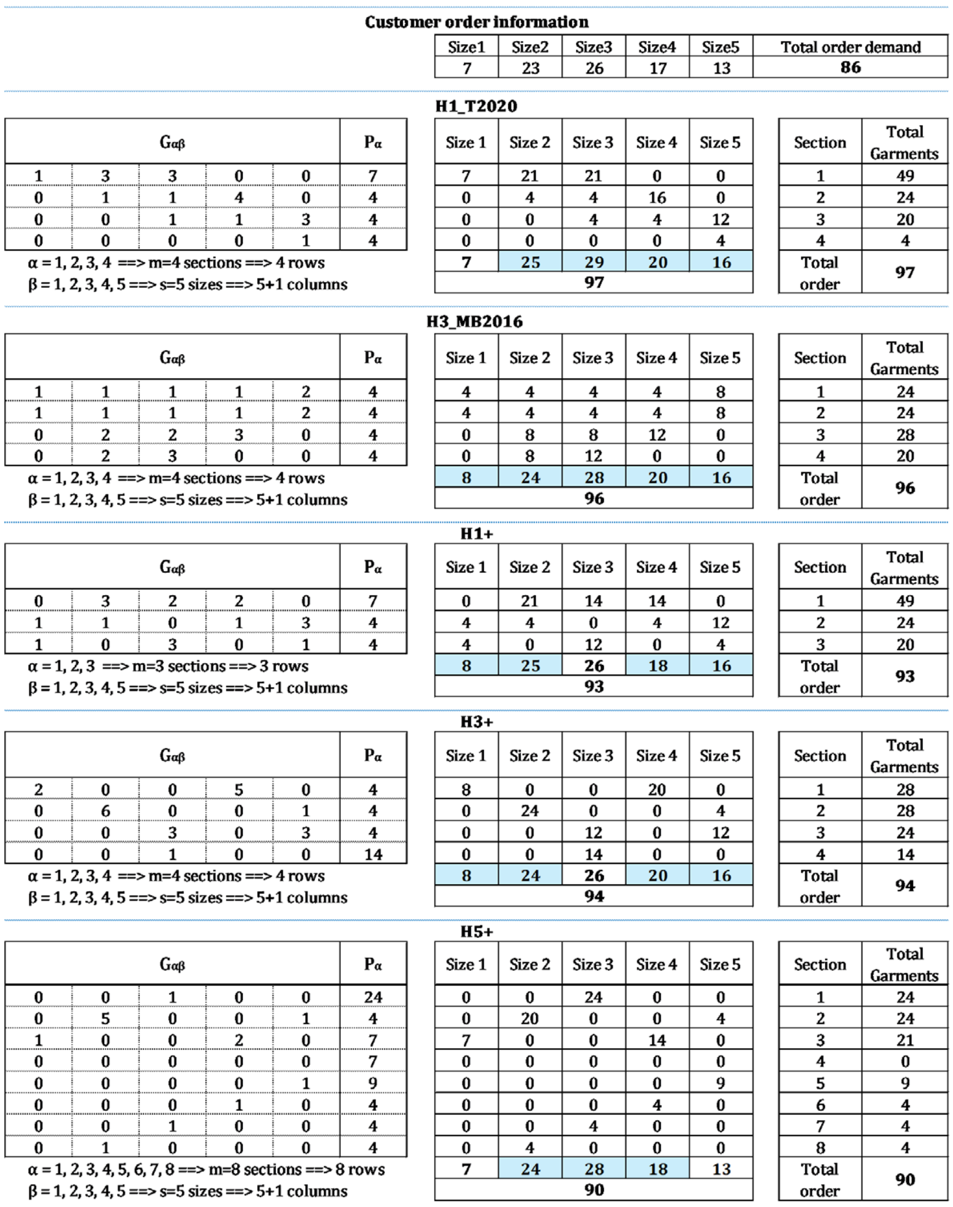
GAs are evolutionary optimization methods inspired by biological evolution and natural selection (Katoch et al., 2021; Ramos-Figueroa et al., 2021), introduced by John Holland in 1975. They generate a population of potential solutions, called chromosomes, and iteratively evaluate their fitness. Fitter solutions are more likely to reproduce, generating children for subsequent generations, evolving towards optimal solutions (Alhijawi & Awajan, 2023). In COP, each chromosome represents a potential solution evaluated by a fitness function to minimize costs. The selection, crossover, and mutation operations favor fitter chromosomes, as shown in Algorithm 6 (Appendix II). GA evolves a randomized population towards optimality through genetic operators across generations, ensuring viability and handling variable chromosome sizes.

3.2.1.1. Initial population

The initial population is generated using five heuristics presented for the COP: H1_T2020, H3_MB2016, H1+, H3+, and H5+. The first two heuristics produce single solutions, while the latter three incorporate randomization, generating a diverse set of *N* solutions to populate the initial pool.

3.2.1.2. Selection operation

The selection process involves organizing tournaments with a predetermined number of contestants (k) chosen randomly from the population. The winner of the competition is the contestant with



Customer order information

Size1	Size2	Size3	Size4	Size5	Total order demand
7	23	26	17	13	**86**

H1_T2020

Gαβ					Pα
1	3	3	0	0	7
0	1	1	4	0	4
0	0	1	1	3	4
0	0	0	0	1	4

α = 1, 2, 3, 4 ==> m=4 sections ==> 4 rows
β = 1, 2, 3, 4, 5 ==> s=5 sizes ==> 5+1 columns

Size 1	Size 2	Size 3	Size 4	Size 5
7	21	21	0	0
0	4	4	16	0
0	0	4	4	12
0	0	0	0	4
7	25	29	20	16
		97		

Section	Total Garments
1	49
2	24
3	20
4	4
Total order	97

H3_MB2016

Gαβ					Pα
1	1	1	1	2	4
1	1	1	1	2	4
0	2	2	3	0	4
0	2	3	0	0	4

α = 1, 2, 3, 4 ==> m=4 sections ==> 4 rows
β = 1, 2, 3, 4, 5 ==> s=5 sizes ==> 5+1 columns

Size 1	Size 2	Size 3	Size 4	Size 5
4	4	4	4	8
4	4	4	4	8
0	8	8	12	0
0	8	12	0	0
8	24	28	20	16
		96		

Section	Total Garments
1	24
2	24
3	28
4	20
Total order	96

H1+

Gαβ					Pα
0	3	2	2	0	7
1	1	0	1	3	4
1	0	3	0	1	4

α = 1, 2, 3 ==> m=3 sections ==> 3 rows
β = 1, 2, 3, 4, 5 ==> s=5 sizes ==> 5+1 columns

Size 1	Size 2	Size 3	Size 4	Size 5
0	21	14	14	0
4	4	0	4	12
4	0	12	0	4
8	25	26	18	16
		93		

Section	Total Garments
1	49
2	24
3	20
Total order	93

H3+

Gαβ					Pα
2	0	0	5	0	4
0	6	0	0	1	4
0	0	3	0	3	4
0	0	1	0	0	14

α = 1, 2, 3, 4 ==> m=4 sections ==> 4 rows
β = 1, 2, 3, 4, 5 ==> s=5 sizes ==> 5+1 columns

Size 1	Size 2	Size 3	Size 4	Size 5
8	0	0	20	0
0	24	0	0	4
0	0	12	0	12
0	0	14	0	0
8	24	26	20	16
		94		

Section	Total Garments
1	28
2	28
3	24
4	14
Total order	94

H5+

Gαβ					Pα
0	0	1	0	0	24
0	5	0	0	1	4
1	0	0	2	0	7
0	0	0	0	0	7
0	0	0	0	1	9
0	0	0	1	0	4
0	0	1	0	0	4
0	1	0	0	0	4

α = 1, 2, 3, 4, 5, 6, 7, 8 ==> m=8 sections ==> 8 rows
β = 1, 2, 3, 4, 5 ==> s=5 sizes ==> 5+1 columns

Size 1	Size 2	Size 3	Size 4	Size 5
0	0	24	0	0
0	20	0	0	4
7	0	0	14	0
0	0	0	0	0
0	0	0	0	9
0	0	0	4	0
0	0	4	0	0
0	4	0	0	0
7	24	28	18	13
		90		

Section	Total Garments
1	24
2	24
3	21
4	0
5	9
6	4
7	4
8	4
Total order	90

Figure 4. Solutions obtained by using heuristic algorithms.

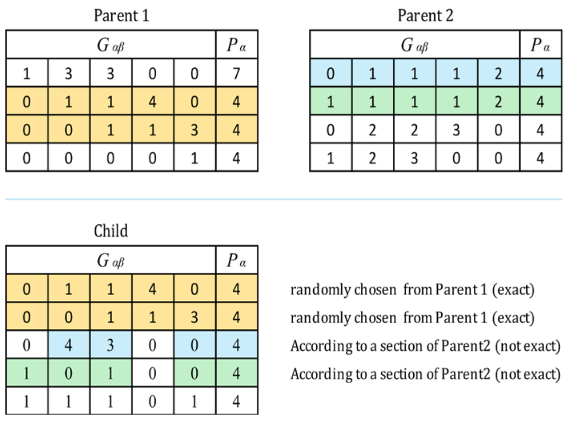
the best fitness, which in this case corresponds to the lowest value, and they move on to the mating pool for reproduction.

3.2.1.3. Crossover operation

To ensure the generation of feasible offspring solutions, this study modifies the crossover operation from M'Hallah and Bouziri (2016), addressing the original procedure's oversight of length constraints. Algorithm 7 (Appendix II) provides the pseudocode for this improved crossover operation. The selection operator is responsible for choosing two individuals from the population to participate in the crossover operation, namely *Parent1* and *Parent2*. This process results in the creation of a new chromosome, known as *Child*, which is composed of some of the m_1 sections from *Parent1* and some of the m_2 sections from *Parent2*. In this context, *G1* and *G2* represent the *G* values of *Parent1* and *Parent2*, respectively, while *P1* and *P2* indicate the *P* values of *Parent1* and *Parent2*, respectively.

Figure 5 presents an example of the crossover operator based on the COP problem outlined in Table 1. The operator clones sections i_1 through i_2

from *Parent1* to *Child*, where i_1 and i_2 are randomly selected from the discrete uniform distribution $[1, m_1]$. In this instance, m_1=4, so i_1 and i_2 are equal to 2 and 3. This implies that *Child* inherits sections 2 and 3 from *Parent1*, and residual demands are computed for this stage of the child's solution with 2 sections: $r_\beta = [7, 19, 18, -3, 1], \ \forall \beta \in s$. After positive residual demands persist, *Child* is then finished based on the sections from *Parent2*.



Parent 1

$G_{\alpha\beta}$					P_α
1	3	3	0	0	7
0	1	1	4	0	4
0	0	1	1	3	4
0	0	0	0	1	4

Parent 2

$G_{\alpha\beta}$					P_α
0	1	1	1	2	4
1	1	1	1	2	4
0	2	2	3	0	4
1	2	3	0	0	4

Child

$G_{\alpha\beta}$					P_α	
0	1	1	4	0	4	randomly chosen from Parent 1 (exact)
0	0	1	1	3	4	randomly chosen from Parent 1 (exact)
0	4	3	0	0	4	According to a section of Parent2 (not exact)
1	0	1	0	0	4	According to a section of Parent2 (not exact)
1	1	1	0	1	4	

Figure 5. Crossover operator for COP.

Let $\alpha' = 1$, which represents the index number of sections from *Parent2*. Since $\alpha' \leq m_2$, and $\max_{\beta \in s}\left(r_\beta\right) = 19 > P_{min}$, a temporary variable $gcd = GCD\left(r_\beta : r_\beta > 0,\ G2_{\alpha'\beta} > 0,\ \beta \in s\right)$ is determined. In this instance, *gcd*= Greatest common divisor of (19,18,1)=1, considering that the residual demands of sizes 1, 2, 4, and 5 are strictly positive and $G2_{1,\beta}$ values of only size 2, 3, and 5 are >0. Therefore, section 3 of *Child* has $P_3 = max\{P_{min}, min P2_{\alpha'}, gcd\}\} = max\{4, min\{4,1\}\}$=4. The $G_{\alpha\beta}$ of this section are the determined by $\left\lfloor \frac{r_\beta}{P_3} \right\rfloor$, if the residual demand of size β is strictly positive and $G2_{1,\beta}$ values of that size β is >0, until the maximum length is reached. This results in $G_{3,1} = 0$, since $G2_{1,1} \ngtr 0$. Then, $G_{3,2} = \left\lfloor \frac{19}{4} \right\rfloor = 4$, which fulfills the condition that $G2_{1,2}$>0; $G_{3,3}$=3, because $G2_{1,3}$>0, but a maximum 3 garments can be placed without violating length constraint; $G_{3,4}$=0 , because there is no residual demand for this size (r_3=–3; and $G_{3,5}$>0, because although $G2_{1,5}$>0 no garments can be placed without violating the length constraint. With the addition of the third section in the *Child* solution, the remaining demands are now $r_\beta = [7, 3, 6, -3, 1], \ \forall \beta \in s$. As there are still positive demands, a new section is created based on α'=2, which corresponds to the second section of *Parent2*.

Now, let α'=2≤m_2 and $\max_{\beta \in s}\left(r_\beta\right)$=6>$P_{min}$, the temporary variable $gcd = GCD(r_\beta : r_\beta > 0, G2_{\alpha'\beta} > 0, \beta \in s)$ is determined. In this case, *gcd*=GCD(7,3,6,1)=1, given that the residual demands of sizes 1, 2, 3, and 5 are strictly positive, and $G2_{2,\beta}$ values of all these sizes are >0. Therefore, section 4 of *Child* has $P_4 = max\{P_{min}, min\{P2_{\alpha'}, gcd\}\}$ $= max\{4, min\{4,1\}\}$=4. Similar to the previous section, $G_{4,1} = \left\lfloor \frac{7}{4} \right\rfloor = 1$, $G_{4,2} = \left\lfloor \frac{3}{4} \right\rfloor = 0$, $G_{4,3} = \left\lfloor \frac{6}{4} \right\rfloor = 1$, $G_{4,5} = \left\lfloor \frac{2}{4} \right\rfloor = 0$, after ensuring that $G2_{2,\beta}$ values of all these sizes are >0. Then, $G_{4,4} = 0$ since residual demand of size 4 is not strictly positive. Hence, with the fourth section in *Child*, the remaining demands are: $r_\beta = [3, 3, 2, -3, 1], \ \forall \beta \in s$. As positive demands persist, a new section is formed. This time $\alpha'=3 \leq m_2$, however $\max_{\beta \in s}\left(r_\beta\right) = 3 \ngeq P_{min}$. Therefore, P_5 and $G_{5,\beta}$ are calculated without reflecting any sections from either parent. $P_5 = max\{P_{min}, GCD(r_\beta : r_\beta > 0, \beta \in s)\}$ $= max\{4, GCD(3,3,2,1)\} = max\{4,1\}$=4.

The $G_{\alpha\beta}$ of this section are the determined by $\left\lceil \frac{r_\beta}{P_5} \right\rceil$, only if the residual demand of size β is strictly positive, until the maximum length is reached. Hence, $G_{5,1} = \left\lceil \frac{3}{4} \right\rceil = 1$, $G_{5,2} = \left\lceil \frac{3}{4} \right\rceil = 1$, $G_{5,3} = \left\lceil \frac{2}{4} \right\rceil = 1$, $G_{5,5} = \left\lceil \frac{1}{4} \right\rceil = 1$, and $G_{5,4}$=0, since there is no residual demand of size 4. After appending this fifth section to *Child* chromosome, all the residual demands become zero or negative: $r_\beta = [-1, -1, -2, -3, -3], \ \forall \beta \in s$, which ends the crossover operation. Therefore, the resulting *Child* solution is now the output of the crossover operation.

3.2.1.4. *Mutation operation*

The Mutation operator, presented in Algorithm 8 (Appendix II), is also adopted and modified by the mutation procedure of M'Hallah and Bouziri (2016) to generate feasible solutions after alteration. The mutation is performed on the *Child* chromosome after the crossover operation.

Figure 6 illustrates an example of the mutation process. Suppose that the Mutation operator randomly selects size 5 in section 2 ($G_{2,5}$) to be altered. The Mutate function is then invoked, which slightly modifies the solution and ensures that it satisfies all constraints. Initially, the value of $G_{2,5}$=0 is set, and the residual demand is calculated as $r_\beta = [-1, -1, -2, -3, 9], \ \forall \beta \in s$. A collection of sections, excluding section 2, is then identified where the number of plies (P) is less than or equal to $\max_{\beta \in s}(r_\beta)$. $\mathbb{I}=\{\alpha'=1\ldots.m, : \alpha' \neq \alpha, \text{and } P_{\alpha'} \leq \max_{\beta \in s}(r_\beta)\}=\{1,3,4,5\}$. Section 4 is chosen because it has the shortest length out of all that belongs to set $\mathbb{I}$. Then, $G_{4,5}=0+\left\lceil \frac{9}{4} \right\rceil=3$ is set where, 0 is the previous value of $G_{4,5}$, 9 is the residual demand for size 5, and 4 is the number of plies in section 4. All residual needs have been reduced to zero or negative values, indicating that all constraints have been satisfied and the mutation is complete. If there is still positive demand, section 4 will be removed from set $\mathbb{I}$, and the same actions will be repeated with the updated set $\mathbb{I}$. If the set $\mathbb{I}$ is empty and there are still residual demands >0, the mutation will generate a new section to meet the demand.

Child

$G_{\alpha\beta}$					P_α
0	1	1	4	0	4
0	0	1	1	**3**	4
0	4	3	0	0	4
1	0	1	0	0	4
1	1	1	0	1	4

Mutated child

$G_{\alpha\beta}$					P_α
0	1	1	4	0	4
0	0	1	1	**0**	4
0	4	3	0	0	4
1	0	1	0	**3**	4
1	1	1	0	1	4

Figure 6. Mutation operator for COP.

3.2.2. Particle Swarm Optimization

PSO, which stands for Particle Swarm Optimization, is a metaheuristic algorithm proposed by Kennedy and Eberhart (1995). It involves candidate solutions, referred to as 'particles,' navigating the search space in a manner like a bird swarm seeking food (Shami et al., 2022). Although PSO was initially designed for D-dimensional continuous optimization problems, it has since been adapted for various applications (Poli et al., 2007; Shami et al., 2022). Kennedy and Eberhart (1997) introduced the binary version of PSO, which paved the way for its application in discrete search spaces. As a result, PSO has been used in numerous studies to solve discrete optimization problems (Chang et al., 2024; Gómez-Montoya et al., 2020; Jarboui et al., 2008). Moreover, some researchers have improved the optimization capabilities of basic or discrete PSO algorithms by incorporating perturbation processes (Chen et al., 2021).

In the context of COP, each solution is represented by a particle's position vector $X_p=\left\{x_{p1}, x_{p2}, \cdots, x_{pm}\right\}$, where $x_{p\alpha}$ represents cutting section α of particle p. Each particle p has a velocity vector $V_p=\left\{v_{p1}, v_{p2}, \cdots, v_{pm}\right\}$, a personal best position $Pbest_p=\{pbest_{p1}, pbest_{p2}, \ldots, pbest_{pm1}\}$ and the global best position $Gbest_p=\{gbest_1, gbest_2, \ldots, pbest_{m2}\}$, where m, m_2, and m_2 are the respective number of dimensions or sections. Algorithm 9 (Appendix III) demonstrates how PSO iteratively improves solutions by adjusting particles' velocities and positions to explore the search space over T iterations or until a stopping criterion is met.

The COP problem involves integer variables and a varying number of dimensions. This research uses an adapted combinatorial PSO from Jarboui et al. (2008). While the original PSO considers constant dimensions in a discrete search space, the adaptation accommodates solutions with varying section numbers to suit the COP optimization problem.

3.2.2.1. Swarm Initialization

The PSO begins by initializing a swarm of particles, represented by position vectors, using the same five heuristics (H1_T2020, H3_MB2016, H1+, H3+, H5+) that were used for the GA. This ensures that the initial solutions are feasible. The velocities of the particles are randomly generated to explore the search space.

3.2.2.2. Position and Velocity Vector Update

To facilitate the transition between continuous and discrete search spaces, we introduced two dummy variable vectors, $Y_p=\left\{y_{p1}, y_{p2}, \ldots, y_{pn}\right\}$ and $\Lambda_p=\left\{\lambda_{p1}, \lambda_{p2}, \ldots, \lambda_{pn}\right\}$, alongside the basic PSO variables. However, unlike Jarboui et al. (2008), n represents the minimum number of sections among the particle p's current position X, its personal best solution $Pbest$, and the global best solution $Gbest$. In other words, n is the number of cutting sections that can be compared between X, $Pbest$, and $Gbest$. Equation (7) determines the elements of Y_p^i based on comparisons between the particle's current position, its personal best, and the global best solutions from the previous iteration. The velocity elements for the i^{th} iteration are updated using Equation (8), which incorporates factors such as the inertia weight (w), cognitive and social acceleration coefficients (c_1 and c_2), and random numbers (r_1 and r_2). The number of elements in Y_p^i may not necessarily match the number of elements in V_p^{i-1}. Therefore, if a corresponding value is available in Y_p^i, the velocity element is updated; otherwise, it remains unchanged.

$$y_{p\alpha}^{i} = \begin{cases} 1, & if\ x_{p\alpha}^{i-1} = gbest_{\alpha}^{i-1} \\ -1, & if\ x_{p\alpha}^{i-1} = pbest_{p\alpha}^{i-1} \\ -1\ or\ 1, & randomly\ if \left(x_{p\alpha}^{i-1} = gbest_{\alpha}^{i-1} = pbest_{p\alpha}^{i-1}\right) \\ 0, & if\ x_{p\alpha}^{i-1} \neq gbest_{\alpha}^{i-1}\ AND\ x_{p\alpha}^{i-1} \neq pbest_{p\alpha}^{-1} \end{cases} \quad (7)$$

$$v_{p\alpha}^{i} = \begin{cases} w \times v_{p\alpha}^{i-1} + r_1 \times c_1\left(-1 - y_{p\alpha}^{i}\right) + r_2 \times c_2 \times \left(1 - y_{p\alpha}^{i}\right), & if\ y_{p\alpha}^{i}\ available \\ v_{p\alpha}^{i-1}, & otherwise \end{cases} \quad (8)$$

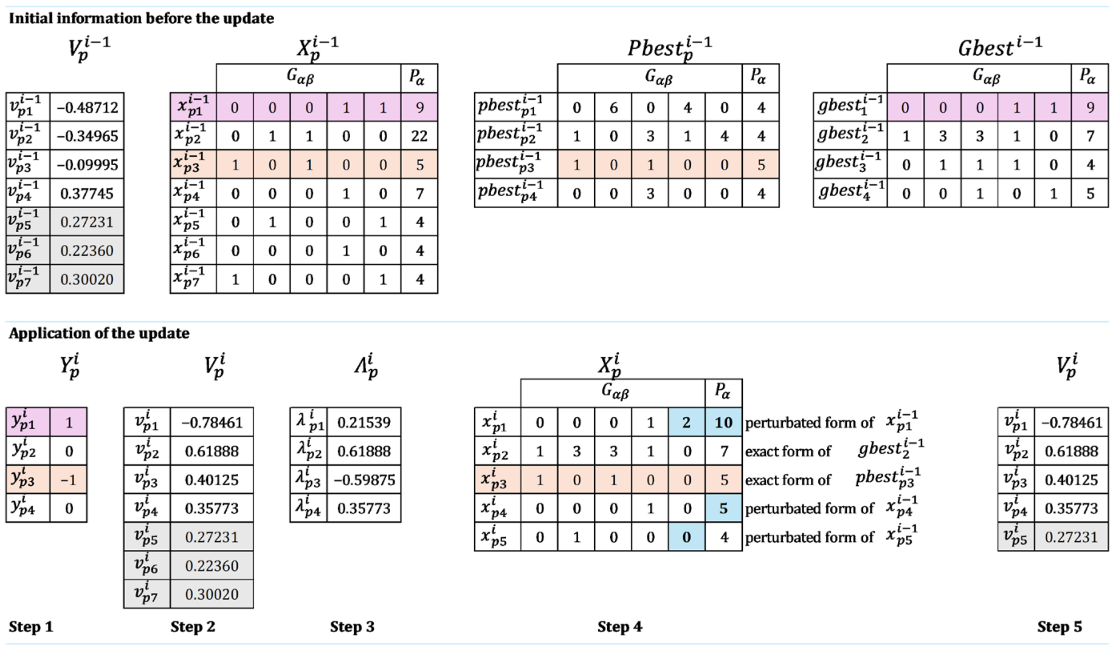
The updated velocity is then utilized in Equation (9) to calculate the second dummy variable, Λ_p^i, which, in turn, is used in Equation (10) to update the particle's position vector $X_p^i = \left\{x_{p1}^i, x_{p2}^i, \ldots, x_{pm'}^i\right\}$, where m' denotes the new section count in the updated solution. The explore factor φ controls the position vector update process and may impact the algorithm's convergence or divergence. In contrast to Jarboui et al. (2008), a perturbation procedure is integrated at this point for the sections transitioning from the old position vector. This inclusion aims to introduce fluctuations within the sections, promoting diversification in the solution space.

$$\lambda_{p\alpha}^{i} = y_{p\alpha}^{i} + v_{p\alpha}^{i} \quad (9)$$

$$x_{p\alpha}^{i} = \begin{cases} gbest_{\alpha}^{i-1}, & if\ \lambda_{p\alpha}^{i} > \varphi \\ pbest_{p\alpha}^{i-1}, & if\ \lambda_{p\alpha}^{i} < -\varphi \\ pertubarted\left(x_{p\alpha}^{i-1}\right), & otherwise \end{cases} \quad (10)$$

Finally, after updating particle p's position vector X_p^i to have m' cutting sections, its velocity vector V_p^i is resized to match this new dimension. If $m' \leq m$, it keeps the first m' values and deletes the remaining. If $m' > m$, it randomly generates $(m' - m)$ number of values and appends them. Algorithm 10 (Appendix III) provides a detailed explanation of how to update a position vector.

Figure 7 shows how to update a particle's position and velocity vector based on the COP problem described in Table 1. Let X_n^{i-1} be a randomly initialized particle solution, V_p^{i-1} be its associated velocity vector, $Pbest_p^{i-1}$ be the personal best solution found by this particle so far, and $Gbest^{i-1}$ be the globally best solution identified among the entire swarm. Equation (7) is used to determine Y_p^i, which initially contains four values. This is because only the first four sections of X_p^{i-1} can be compared with $Pbest_p^{i-1}$ and $Gbest^{i-1}$. Sections 1 and 3 in X_p^{i-1} match $Gbest^{i-1}$ and $Pbest_p^{i-1}$, respectively, resulting in y_{p1}^i being 1 and y_{p3}^i being –1 (highlighted in the same colors). The remaining sections in X_p^{i-1} do not align with *Pbest* and *Gbest*, resulting in the other Y_p^i values being zero.



Initial information before the update

V_p^{i-1}

v_{p1}^{i-1}	−0.48712
v_{p2}^{i-1}	−0.34965
v_{p3}^{i-1}	−0.09995
v_{p4}^{i-1}	0.37745
v_{p5}^{i-1}	0.27231
v_{p6}^{i-1}	0.22360
v_{p7}^{i-1}	0.30020

X_p^{i-1}

	$G_{\alpha\beta}$					P_α
x_{p1}^{i-1}	0	0	0	1	1	9
x_{p2}^{i-1}	0	1	1	0	0	22
x_{p3}^{i-1}	1	0	1	0	0	5
x_{p4}^{i-1}	0	0	0	1	0	7
x_{p5}^{i-1}	0	1	0	0	1	4
x_{p6}^{i-1}	0	0	0	1	0	4
x_{p7}^{i-1}	1	0	0	0	1	4

$Pbest_p^{i-1}$

	$G_{\alpha\beta}$					P_α
$pbest_{p1}^{i-1}$	0	6	0	4	0	4
$pbest_{p2}^{i-1}$	1	0	3	1	4	4
$pbest_{p3}^{i-1}$	1	0	1	0	0	5
$pbest_{p4}^{i-1}$	0	0	3	0	0	4

$Gbest^{i-1}$

	$G_{\alpha\beta}$					P_α
$gbest_1^{i-1}$	0	0	0	1	1	9
$gbest_2^{i-1}$	1	3	3	1	0	7
$gbest_3^{i-1}$	0	1	1	1	0	4
$gbest_4^{i-1}$	0	0	1	0	1	5

Application of the update

Y_p^i

y_{p1}^i	1
y_{p2}^i	0
y_{p3}^i	−1
y_{p4}^i	0

Step 1

V_p^i

v_{p1}^i	−0.78461
v_{p2}^i	0.61888
v_{p3}^i	0.40125
v_{p4}^i	0.35773
v_{p5}^i	0.27231
v_{p6}^i	0.22360
v_{p7}^i	0.30020

Step 2

Λ_p^i

λ_{p1}^i	0.21539
λ_{p2}^i	0.61888
λ_{p3}^i	−0.59875
λ_{p4}^i	0.35773

Step 3

X_p^i

	$G_{\alpha\beta}$					P_α	
x_{p1}^i	0	0	0	1	2	10	perturbated form of x_{p1}^{i-1}
x_{p2}^i	1	3	3	1	0	7	exact form of $gbest_2^{i-1}$
x_{p3}^i	1	0	1	0	0	5	exact form of $pbest_{p3}^{i-1}$
x_{p4}^i	0	0	0	1	0	5	perturbated form of x_{p4}^{i-1}
x_{p5}^i	0	1	0	0	0	4	perturbated form of x_{p5}^{i-1}

Step 4

V_p^i

v_{p1}^i	−0.78461
v_{p2}^i	0.61888
v_{p3}^i	0.40125
v_{p4}^i	0.35773
v_{p5}^i	0.27231

Step 5

Figure 7. Position and velocity vector update process in PSO.

In Step 2, the velocity of the particle is updated for iteration i using Equation (8). The values of c_1 and c_2 are considered as 2, w as 0.4, and random values of r_1 (0.147440) and r_2 (0.585727) between 0 and 1. The first value of V_p^i is calculated as –0.78461 using Equation (8). The next three V_p^i values are updated similarly. As Y_p^i has only four values, the velocity remains unchanged after the fourth section. In Step 3, Equation (9) combines the Y_p^i and V_p^i components to produce Λ_p^i, which is another dummy variable with four values. In Step 4, Equation (10) is used to adjust X_p^{i-1} and build X_p^i. The new solution, X_p^i is initialized with 0 sections. The first value of Λ_p^i is compared to a threshold of $\varphi=0.5$. Since $-\varphi \leq \lambda_{p1}^i = 0.21539 \leq \varphi$, the first section of X_p^{i-1} is temporarily assigned as $temp_{x_{p1}^i}$, subject to perturbation.

The perturbation process initially focuses on the P value. A generated random number is compared to the perturbation rate (η). If the random number is less than η, two integers, p1 and p2, are determined as $p1,p2 = \text{Sorted}\left[P_\alpha, min(P_{max}, max\{P_{min}, \underset{\beta \in s}{max}(r_\beta)\}\right]$. In this case, with P_α=P_1=9; P_{max}=35; P_{min}=4; and $\underset{\beta \in s}{max}\left(r_\beta\right) = 26$, we obtain $p1,p2$=9&26 respectively. Subsequently, a random integer between [9,26]=10 is set as P_1 of X_p^i. The next phase involves perturbing the G values. Assume that the G of Size5 ($G_{1,5}$) is randomly selected for alteration. It begins by setting $G_{1,5}$=0. At this point, particle X_p^i consists of only one temporary section, $temp_{x_{p1}^i}$. The calculated residual demands with this modified section are: $r_\beta = \left[7,23,26,7,13\right], \ \forall \beta \in s$.

Next, a temporary variable temp is calculated as $temp = min\left(\left\lceil \frac{r_\beta}{P_\alpha} \right\rceil, \left\lfloor \frac{l_{max} - l_\alpha}{Y_\beta} \right\rfloor\right)$. With r_5=13, P_α=10, l_{max}=20, l_α=1.79, $G_{1,5}$=0, Y_5=1.51, we obtain $temp$=min(2,12)=2. Subsequently, $G_{1,5}$ is updated from 0 to a random integer in [2,12]=2. After perturbation, the section is permanently added to X_p^i. Perturbed values are visually highlighted. With this definitive section, residual demands are: $r_\beta = \left[7,23,26,7,-7\right], \ \forall \beta \in s$. While demand remains positive, the process continues. The second section is taken from the global best ($Gbest$) and appended to X_p^i since $\lambda_{p2}^i = 0.61888 > \varphi$. For the next Λ_p^i value $\lambda_{p3}^i = -0.59875 < -\varphi$, the third section is taken from $Pbest_p$ and merged into X^i. It is important to note that none of these sections undergoes perturbation.

The process iteratively adds sections from X_p^{i-1}, $Pbest_p$ or $Gbest$ into X_p^i following Equation (10), as long as demands remain positive. The fourth section is added from X_p^{i-1} using the same perturbation procedure as the first section, as long as $-\varphi \leq \lambda_{p4}^i = 0.35773 \leq \varphi$. Note that due to the probabilistic nature of the approach, $G_{\alpha\beta}$, P_α, or neither may go through perturbation. After adding four sections, the residual demands now stand as follows: $r_\beta = \left[-5, 2, 0, -5, -7\right], \ \forall \beta \in s$. The process continues to add new sections with remaining demand. However, perturbed sections are taken from X_p^{i-1} when there are no more sections from Λ_p^i to compare against the threshold. This process continues until there are no more sections left in X_p^{i-1} or the remaining demands are non-positive. If there is still demand after adding all X_p^{i-1} sections, the H1+ algorithm packs the remaining demand to ensure feasibility.

In this example, X_p^i is completed after the fifth section from X_p^{i-1}, because no non-zero positive residual demand remains. The final version of the particle's position vector, X_p^i, is illustrated in Figure 7 (Step 4). In Step 5, the velocity vector V_p^i is resized to match the dimension of the position vector X_p^i. The first 5 values of V_p^i are preserved, discarding the rest, as X_p^i now has five sections instead of the previous seven. If the new solution had more sections than the previous one, additional velocity values would be randomly generated for the extra sections.

3.3. Exact solution using commercial solvers

The COP model presented in Equations 1-6 is a Mixed-Integer Nonlinear Programming (MINLP) model. The nonlinearity arises from the product of decision variables $G_{\alpha\beta}$ and P_α in the objective function (1) and constraints (5). To solve the COP problem using commercial solvers, it is necessary to convert the MINLP model into a (MILP) model. This can be achieved by introducing additional binary and integer variables and constraints. This process of linearization enables the replacement of nonlinear terms in the objective function and constraints with equivalent linear expressions, thereby rendering the problem solvable by standard MILP solvers. The MILP model modifies certain indexes and variables while keeping the same parameters as the MINLP model. Below are the indices, decision variables, and mathematical formulation for optimizing the COP.

Indexes

β: sizes (β=1,2,3,…s)

α: sections (α=1,2,3,…m)

k: digits (k=1,2,3,…K)

Decision variables

$G_{\alpha\beta}$: Number copies of size β in section α

$v_{\alpha k}$: Technical auxiliary variable to convert an integer variable into a limited set of binary variables

$w_{\alpha\beta k}$: Technical auxiliary variable to convert a product into an integer variable to linearize the problem

Z_α: Takes the value of 1 if section α is used; otherwise, 0

$$Min\ C_T = \sum_\alpha \sum_\beta Y_\beta f \sum_k 2^k w_{\alpha\beta k} + \sum_\alpha Z_\alpha CS + + \sum_\beta CE\Big(\sum_\alpha \sum_k 2^k w_{\alpha\beta k} - Q_\beta\Big) \quad (11)$$

$$\sum_\beta G_{\alpha\beta} Y_\beta \le l_{max} \quad (12)$$

$$\sum_k v_{\alpha k} 2^k \le P_{max} Z_\alpha \quad (13)$$

$$\sum_k v_{\alpha k} 2^k \ge P_{min} Z_\alpha \quad (14)$$

$$\sum_\alpha \sum_k 2^k w_{\alpha\beta k} \ge Q_\beta \quad (15)$$

$$w_{\alpha\beta k} \le v_{\alpha k} Q_\beta \quad (16)$$

$$w_{\alpha\beta k} \ge G_{\alpha\beta} - Q_\beta(1 - v_{\alpha k}) \quad (17)$$

$$w_{\alpha\beta k} \le G_{\alpha\beta} \quad (18)$$

The conversion from the MINLP model to the MILP model starts with encoding the ply count as binary. The MILP model introduces new binary decision variables, $v_{\alpha k}$, to indicate whether section α includes the value 2^k. This enables the representation of P_α (a decimal number) as a linear combination of these binary variables: $P_\alpha = \sum_k v_{\alpha k} 2^k$. The index k ranges from 1 to a maximum value K, which determines the maximum number of binary digits allowed for representing P_α. K is given for any concrete problem as P_α is limited by P_{max}. To define new integer variables, $w_{\alpha\beta k}$, we take the product of $v_{\alpha k}$ and $G_{\alpha\beta}$. Using $w_{\alpha\beta k}$, the nonlinear part of Equation (11) can be linearly rewritten as $\sum_\alpha \sum_k 2^k w_{\alpha\beta k} = \sum_\alpha G_{\alpha\beta} P_\alpha$. Constraint (12) in the MILP model is equivalent to Constraint (2) in the MINLP model. Constraints (13) and (14) replace the Constraints (3) and (4) from the MINLP model, respectively, enforcing the minimum and maximum values for P_α using the binary representation. Constraint (15) replaces the nonlinear Constraint (5) in the MINLP model. This ensures that the total production of each size meets the Q_β demand. The variables $w_{\alpha\beta k}$ are connected to $v_{\alpha k}$ and $G_{\alpha\beta}$ through the constraints (16), (17), and (18). If $v_{\alpha k}$=0, constraint (16) results to $w_{\alpha\beta k}$=0. When $v_{\alpha k}$=1, the constraint (17) reduces to $w_{\alpha\beta k} \ge G_{\alpha\beta}$. Together with constraint (18) this means $w_{\alpha\beta k} = G_{\alpha\beta}$. Therefore, in all possible cases ($v_{\alpha k}$=0 and $v_{\alpha k}$=1) indeed $w_{\alpha\beta k} = v_{\alpha k} G_{\alpha\beta}$ holds true.

The relative MIP gap tolerance in CPLEX, which balances solution quality and computational complexity (IBM, 2022), is a crucial parameter when solving the proposed MILP model. A lower tolerance yields a more accurate solution, but at the cost of increased computation time, while a higher tolerance prioritizes speed but may result in a less optimal solution. In this study, a default tolerance of 0.1% was initially used, with adjustments made to 0.5-1.6% for cases of extended runtime or no solution.

4. Computational experiments

4.1. Parameter tuning

The parameters for the two proposed metaheuristics were determined through tuning, starting with initial values from previous studies (Cano et al., 2023; Fister et al., 2008, 2010; Jarboui et al., 2008; M'Hallah & Bouziri, 2016; Shen et al., 2014; Tsao et al., 2020; Wong & Leung, 2008). A grid search evaluated different combinations using three COP cases from the industry, each tested five times. For the PSO algorithm, we tested η = [0.1, 0.2, 0.3], φ = [0.3, 0.5, 0.7, 0.9], c_1 c_2 = [1, 1.5, 2], and w = [0.4, 0.6, 0.9] resulting in 324 combinations of these values on three cases, each executed five times, for a total of 4860 runs. For the GA algorithm, we tested ρ = [0.8, 0.9, 0.95, 1], η = [0.003, 0.01, 0.05, 0.1], and K = [2, 3, 4, 5, 6]. This led to 80 possible combinations of these values for the same three cases. Each combination was run five times, resulting in a total of 1200 runs. The average fitness values and CPU runtime were recorded, and combinations were ranked by average fitness. Table 2 shows the selected PSO values: w=0.4; c_1=c_2=2;

η=0.2; and φ=0.5, with a stopping criterion of 90 iterations without improvement. The optimal swarm size was N=250 and iteration size T=200. For GA, the selected values were K=2, ρ=0.95; and η=0.003, with a stopping criterion of 99 iterations without improvement. The optimal population size was N=200 and generation size T=200.

Table 2. Parameter values for the GA and PSO.

Parameters	Symbols	PSO	GA
Swarm size or Population size	N	250	200
Iteration size or Generation size	T	200	200
Crossover rate	ρ	-	0.95
Perturbation rate or Mutation rate	η	0.2	0.003
Tournament size	K	-	2
Cognitive acceleration coefficient	c_1	2	-
Social acceleration coefficient	c_2	2	-
Inertia weight	w	0.4	-
Explore factor	φ	0.5	-
Stopping criteria	s_c	90	99

4.2. Experimental instances

To evaluate the proposed solution approaches (metaheuristics and commercial solvers), nine datasets (S4, S5, S7, M4, M5, M7, B4, B5, B7) from Tsao et al. (2020) were tested and compared. Table 3 lists the size requirements and fabric consumption for these datasets. Parameters include a setup cost CS=\$500 per section, excess production cost CE=\$20 per garment, fabric cost f=\$10 per meter, maximum fabric length per section l_{max}=20 meters, and 4 to 40 fabric layers per section. This study differs from Tsao et al. (2020) by limiting each section to 20 meters instead of a cumulative 100 meters, as sections in Tsao et al. (2020) did not exceed 20 meters. This adjustment was necessary because the original restriction was impractical for large order sets in M5, B5, and B7 (e.g., the M5 solution in Tsao et al. (2020) had a total section length of 101.549 meters).

The metaheuristic algorithms, written in Python 3.8.5, were run on an Apple M1 chip machine with an 8-core processor and 8 GB of RAM. Anaconda Navigator and Jupyter Notebook were used to write and execute the algorithms, finding solutions to a total of 9 COP sets. Each algorithm was run 10 times for each of the nine datasets. The MILP model was implemented using two general-purpose commercial solvers: CPLEX and OpenSolver (an add-on to MS Excel). The comparison results show the total cost and runtime by solution method, as well as the total cost savings achieved with the proposed solution methods.

5. Results and discussion

The proposed solution approaches, including metaheuristics and commercial solvers, were applied to the same datasets from Tsao et al. (2020) for comparison. Table 4 shows the averaged results for each solution method, with metaheuristics averaging ten executions per dataset due to their stochastic nature. These results were compared with Tsao et al. (2020), indicating that the TS algorithm with high parameter settings (TS high) provided the most cost-effective solutions in most cases, while CPLEX delivered the best performance among the proposed methods.

When comparing the GAs in Table 4, the proposed GA shows superior performance, consistently generating lower-cost solutions than the GA (Low)

Table 3. Size wise demand quantity and fabric consumption rate.

Dataset	Size 1	Size 2	Size 3	Size 4	Size 5	Size 6	Size 7
S4	80	95	58	28			
S5	25	95	145	130	25		
S7	8	105	92	56	34	10	5
M4	654	443	938	245			
M5	100	443	938	245	234		
M7	143	443	1004	345	756	257	156
B4	8751	11 425	14 526	12 498			
B5	7051	11 425	14 526	12 498	2256	569	
B7	6780	14 526	16 473	18 767	112 498	8569	6532
Fabric Consumption (Y_β) in meters	1.420	1.441	1.462	1.500	1.523	1.634	1.700

and generally outperforming the GA (High), except for datasets S7 and M4. On average, the proposed GA reduces costs by 4.2% and 4.0% compared to the GA (Low) and GA (High), respectively. This improvement is attributed to modifications in the initial solution heuristics (H1+, H3+, H5+) and the implementation of the tournament selection process. Table 4 shows that the cost for the M4 dataset by GA (high) and SA-GA (high) is lower than the cost by CPLEX with the default gap tolerance, suggesting 35 559 as the optimal solution within 0.1% tolerance. However, the cost of 35 237 reported by Tsao et al. (2020) is outside this range. Their study didn't publish the M4 solution, preventing verification. The discrepancy could be due to changes in the mathematical model or possible non-adherence to some constraints in their M4 solution, resulting in an artificially lower cost.

Table 5 shows cost savings from the proposed solution methods. GA and PSO achieve the highest savings compared to GA (High) in dataset S5 (18.8% and 19.9%, respectively), while CPLEX and Open Solver achieve the highest savings in dataset

Table 4. Total cost by solution method.

		Dataset								
Source	Solution method	S4	S5	S7	M4	M5	M7	B4	B5	B7
Tsao et al. (2020)	SA (Low)	5308	8175	7187	37 115	32 825	52 878	735 239	763 145	2 975 734
	SA (High)	5273	8175	6935	36 559	32 567	52 417	735 239	756 755	2 973 446
	GA (Low)	5307	8175	7752	37 059	33 025	53 015	735 739	764 145	2 975 234
	GA (High)	5773	9210	6647	35 237	33 290	53 126	735 738	757 755	2 974 342
	TS (Low)	5273	8175	6716	37 059	32 464	52 773	735 739	763 645	2 976 233
	TS (High)	4773	7175	6647	36 559	32 256	51 737	734 739	756 755	2 973 342
	SA-GA (Low)	5376	8626	7464	37 059	33 756	54 553	736 739	766 214	2 977 234
	SA-GA (High)	5273	8175	6719	35 237	32 256	53 198	736 739	757 755	2 975 234
	TS-GA (Low)	5273	7850	6855	37 059	32 393	53 491	735 239	764 145	2 975 234
	TS-GA (High)	5483	8175	6647	36 836	32 894	53 374	735 239	757 255	2 973 342
	Minimum cost	4773	7175	6647	35 237	32 256	51 737	734 739	756 755	2 973 342
This study	GA	5013	7479	6733	36 231	31 654	51 445	734 439	756 532	2 973 330
	PSO	4884	7375	6694	36 231	31 578	51 329	734 559	756 564	2 973 352
	CPLEX	4773	7175	5785	35 559	31 256*	50 133	697 739	718 467*	2 830 683*
	OpenSolver	4773	7175	5785	36 362	31 708	50 740	700 506	722 240	Infinite
	Minimum cost	4773	7175	5785	35 559	31 256	50 133	697 739	718 467	2 830 683

* Solutions generated with non-default relative MIP gap tolerance.

Table 5. Cost Savings.

		Dataset									
Measure	Solution method	S4	S5	S7	M4	M5	M7	B4	B5	B7	Overall
Max Saving	GA	13.2%	18.8%	13.1%	2.4%	6.2%	5.7%	0.3%	1.3%	0.1%	18.8%
	PSO	15.4%	19.9%	13.6%	2.4%	6.5%	5.9%	0.3%	1.3%	0.1%	19.9%
	CPLEX	17.3%	22.1%	25.4%	4.2%	7.4%	8.1%	5.3%	6.2%	4.9%	25.4%
	Open Solver	17.3%	22.1%	25.4%	2.0%	6.1%	7.0%	4.9%	5.7%	0.0%	25.4%
Min Saving	GA	-5.0%	-4.2%	-1.3%	-2.8%	1.9%	0.6%	0.0%	0.0%	0.0%	-5.0%
	PSO	-2.3%	-2.8%	-0.7%	-2.8%	2.1%	0.8%	0.0%	0.0%	0.0%	-2.8%
	CPLEX	0.0%	0.0%	13.0%	-0.9%	3.1%	3.1%	5.0%	5.1%	4.8%	-0.9%
	Open Solver	0.0%	0.0%	13.0%	-3.2%	1.7%	1.9%	4.7%	4.6%	0.0%	-3.2%
Avg. Saving	GA	5.4%	8.4%	3.0%	0.9%	3.4%	3.0%	0.2%	0.6%	0.1%	2.8%
	PSO	7.9%	9.6%	3.5%	0.9%	3.6%	3.2%	0.1%	0.5%	0.1%	3.3%
	CPLEX	10.0%	12.1%	16.6%	2.7%	4.6%	5.5%	5.2%	5.6%	4.8%	7.5%
	Open Solver	10.0%	12.1%	16.6%	0.6%	3.2%	4.4%	4.8%	5.1%	-	7.1%
Min (This study) vs Min (Tsao et al. (2020))		0.0%	0.0%	13.0%	-0.9%	3.1%	3.1%	5.0%	5.1%	4.8%	0.0%

S7 (25.4% for both). The worst performance for GA is in S4 compared to TS (High) (-5.0%), and for PSO in M4 compared to GA (High) and SA-GA (High) (-2.8%). CPLEX underperforms only in M4 compared to GA (High) (-0.9%), and Open Solver underperforms only in M4 compared to GA (High) and SA-GA (High). Comparing the best performance of benchmark methods with proposed methods, CPLEX and Open Solver offer up to 13% savings in dataset S7 compared to GA (High), TS (High), and TS-GA (High).

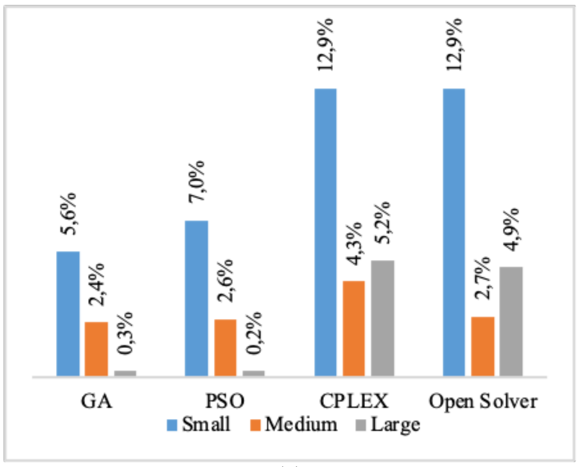
When analyzing the cost savings by demand quantity, Figure 8(a) shows that the average savings of the GA, PSO, CPLEX, and Open Solver with respect to the benchmark solutions are higher in scenarios with smaller demand quantities per size (small), while for the metaheuristics, the lowest average savings are generated in scenarios with large demand quantities. When analyzing the cost savings by number of sizes, Figure 8(b) shows that the average savings of CPLEX and Open Solver with respect to the benchmark solutions are higher in scenarios with larger number of sizes, while for GA and PSO, the most notable average savings are generated in scenarios with sizes equal to 5.

When comparing solution approaches, it is important to consider both solution quality and computation time, as there is a trade-off between the two when solving an NP-complete problem. While TS (High) was the best solution in terms of cost, it was one of the slowest algorithms in Tsao et al. (2020). Conversely, SA (Low) was the fastest algorithm presented in Tsao et al. (2020), but its solutions were less optimal. These two algorithms serve as benchmarks for comparing solution quality and execution time. Figure 9 plots CPU runtime (log scale x-axis) and total cost (y-axis) for COP datasets, with points near the lower left corner indicating better performance in terms of both quality and computation time. Similarly, for the datasets considered in the experiments, the average computation time for each solution approach is shown in Table 6.

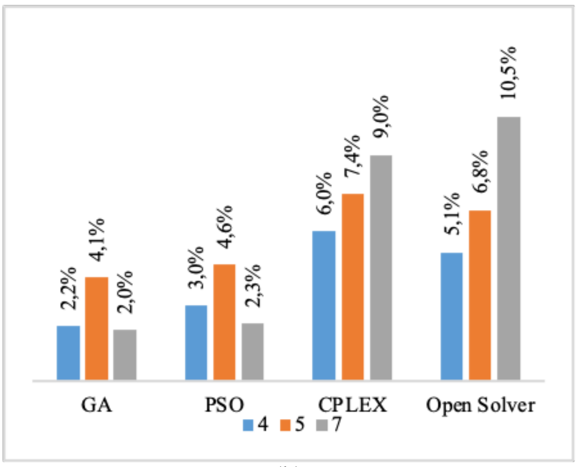
The proposed approaches generally outperform TS (High) in solution quality, with CPLEX and OpenSolver often achieving efficient solutions with computation time comparable to SA (Low).

For larger instances, the computational time solving the MILP model increases significantly due to the growth in variables and constraints. To mitigate this, we adjusted the relative MIP gap tolerance, applying higher tolerances (0.5% to 1.6%) for larger datasets to reduce runtime. For larger order quantities, computation time increases, but CPLEX excelled in cases like M5, B5, and B7 due to a higher gap tolerance. The proposed metaheuristics, PSO and GA, outperformed TS (High) for medium and large orders, with PSO performing better in terms of cost, with an average of 0.6%, but its execution time increased exponentially for large order sets.

Regarding the time complexity of GA, it is O(T×N^2), where T is the number of generations and N is the population size. This complexity grows quadratically as the population size increases. The tournament selection process is proportional to N, while the crossover operation, which combines



(a)



(b)

Figure 8. Cost savings by (a) demand quantity and (b) number of sizes.

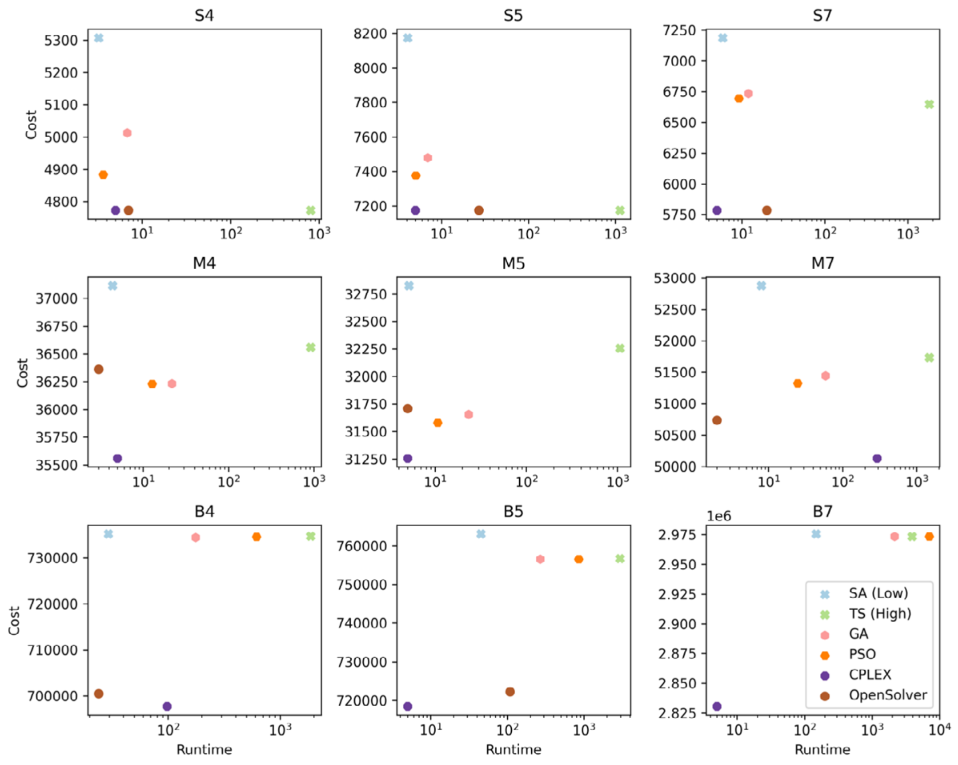


Figure 9. Scatter plot comparing runtime and solution quality (cost) of different solution approaches.

Table 6. Computation time by solution method.

Dataset	GA	PSO	OpenSolver	CPLEX
S4	6.8	3.6	7	5
S7	12.0	9.3	20	5
S5	6.9	5.0	27	5
M5	23.3	10.7	5	5
M4	21.7	12.7	3	5
M7	58.8	24.7	2	292
B4	175.3	615.8	24	98
B5	269.0	856.4	109	5
B7	2146.0	6998.7	900	5

pairs of individuals, results in approximately N(N−1)/2 pairwise combinations. As a result, the crossover complexity is $O(N^2)$. Selection and crossover dominate GA's computational cost, requiring $O(N^2)$ operations per generation, leading to a total complexity of $O(T \times N^2)$ over T generations.

In the case of PSO, the complexity is O(T×N), where N is the number of particles and T is the number of iterations. This indicates that the computation time grows linearly with both the number of particles and iterations. At each iteration, PSO updates the velocity and position of all N particles, resulting in O(N) operations per iteration. Across T iterations, the overall time complexity becomes O(T×N). Unlike GA, PSO does not require pairwise operations, which keeps its complexity linear.

The MILP model used in this study poses significant computational challenges, particularly for large instances, as MILP problems are NP-hard, leading to exponential growth in solution time as problem size increases. This complexity arises from the combination of binary decision variables, continuous variables, and linear constraints. The worst-case time complexity, using branch-and-bound or branch-and-cut techniques (e.g., CPLEX), is exponential in the number of binary variables, as the solver must explore many possible combinations.

Consequently, these results indicate that the proposed algorithmic approaches are useful for solving the COP and provide high quality solutions within reasonable computation time. In the first instance, the transformation of MINLP models into MILP models is preferred. In cases where it is not possible to find a solution within reasonable computation time using solvers such as CPLEX or OpenSolver, it is recommended to proceed with the execution of metaheuristics such as the proposed PSO and GA. These metaheuristics also produce satisfactory solutions for the COP compared to existing solution methods.

Therefore, the solution methods proposed in this study are well-suited for industrial application in garment manufacturing due to their ability to solve the COP problem. For small and medium-sized production orders, the MILP model provides highly efficient, near-optimal solutions using commercial solvers like CPLEX and OpenSolver. These solvers can be easily integrated into existing production planning systems, offering precise control over fabric usage and operational costs. In practical settings, especially where accuracy is paramount, the MILP approach could serve as an effective tool for short production runs and customized orders, helping companies optimize material usage while maintaining flexibility.

For larger production orders, where the complexity of the problem increases significantly, the metaheuristic approaches (GA and PSO) demonstrate their value by offering near-optimal solutions within reasonable computing time. These algorithms are highly adaptable, making them suitable for real-world scenarios where rapid decision-making and scalability are crucial. Their ease of implementation and lower computational cost compared to exact methods make them particularly attractive for large-scale operations or when real-time optimization is required.

Future industrial implementation could involve integrating these algorithms into production planning software, where they can dynamically optimize cut order plans based on changing demand and production constraints. Additionally, these methods can be applied in industries beyond apparel, such as automotive or furniture manufacturing, where efficient material cutting is equally critical.

6. Conclusions

In this study, we proposed two different solution approaches to solve the cut order planning problem in the garment manufacturing industry. The first approach used conventional metaheuristic algorithms, while the second approach involved transforming the nonlinear mathematical model into a linear one and using commercial solvers to solve it. We modified certain heuristics from previous literature and used them to generate the initial population for the metaheuristics. This modification, along with the use of tournament selection in the mating process, contributed to the improvement of the GA compared to previous versions in the literature. The proposed PSO showed better performance than the GA in most cases, especially for small and medium-sized problems.

The cost and runtime analysis of the proposed algorithms compared to benchmarks validates their efficiency and indicates their suitability for practical implementation in industry. In particular, the commercial solvers, CPLEX and OpenSolver, outperformed the metaheuristic approaches, providing better solutions in faster computational times than the metaheuristics. Commercial solvers provided better solutions than the best solution provider algorithm while maintaining a computational time comparable to the fastest solution provider algorithm presented in the literature.

Although heuristics and metaheuristics are conventional approaches for solving NP-hard problems such as the COP problem, the results of this study show that CPLEX and OpenSolver are often superior options. Furthermore, the use of metaheuristics in industry may require a development and implementation phase, whereas off-the-shelf software does not. Therefore, this study recommends the use of general-purpose software, such as CPLEX and OpenSolver, to solve COP problems, especially for small and medium-sized orders. For large orders, where commercial solvers cannot provide a solution, the use of the proposed PSO and GA is recommended.

The proposed solution methods not only demonstrate their effectiveness in solving the COP problem but also contribute to broader industrial goals, particularly in terms of sustainability. By optimizing COP, our methods significantly reduce fabric waste, which is crucial in an industry where material costs represent a significant portion of total expenses. The ability to minimize waste while meeting production demands directly supports sustainable practices, helping manufacturers reduce their environmental footprint while improving operational efficiency.

Future research could focus on the development of hybrid metaheuristics that combine the strengths of GA and PSO, leveraging the exploration capabilities of GA and the fast convergence of PSO. This hybrid approach could deliver even more efficient solutions, particularly for large-scale, complex instances of the COP problem. Additionally, future research could explore these hybrid enhancements applied not only in garment manufacturing but also in other industries with similar cutting and planning challenges. Finally, further research could explore decomposition methods such as column generation and Benders decomposition to address the scalability issues of the MILP model, breaking the problem into smaller, more manageable subproblems.

7. Data Availability

The data will be available upon request.

Authors contribution

All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Sharif Al-Mahmud, Jose Alejandro Cano, Emiro Antonio Campo, and Stephan Weyers. The first draft of the manuscript was written by Sharif Al-Mahmud and Jose Alejandro Cano and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

References

Abd Jelil, R. (2018). Review of Artificial Intelligence Applications in Garment Manufacturing. In S. Thomassey & X. Zeng (Eds.), *Artificial Intelligence for Fashion Industry in the Big Data Era. Springer Series in Fashion Business.* (pp. 97–123). Springer Singapore. https://doi.org/doi.org/10.1007/978-981-13-0080-6_6

Abeysooriya, R. ., & Fernando, T. G. . (2012a). Canonical Genetic Algorithm To Optimize Cut Order Plan Solutions in Apparel. *Journal of Emerging Trends in Computing and Information Sciences*, *3*(2), 150–154.

Abeysooriya, R. ., & Fernando, T. G. . (2012b). Hybrid Approach to Optimize Cut Order Plan Solutions in Apparel Manufacturing. *International Journal of Information and Communication Technology Research*, *2*(4), 348–353.

Abualigah, L., Gandomi, A. H., Elaziz, M. A., Hamad, H. Al, Omari, M., Alshinwan, M., & Khasawneh, A. M. (2021). Advances in meta-heuristic optimization algorithms in big data text clustering. *Electronics (Switzerland)*, *10*(2), 101. https://doi.org/10.3390/electronics10020101

Alhijawi, B., & Awajan, A. (2023). Genetic algorithms: theory, genetic operators, solutions, and applications. *Evolutionary Intelligence*. https://doi.org/10.1007/s12065-023-00822-6

Alsamarah, W., Younes, B., & Yousef, M. (2022). Reducing waste in garment factories by intelligent planning of optimal cutting orders. *The Journal of The Textile Institute*, *113*(9), 1917–1925. https://doi.org/10.1080/00405000.2021.1956711

Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, *237*, 82–117. https://doi.org/10.1016/j.ins.2013.02.041

Cano, J. A., Cortés, P., Muñuzuri, J., & Correa-Espinal, A. (2023). Solving the picker routing problem in multi-block high-level storage systems using metaheuristics. *Flexible Services and Manufacturing Journal*, *35*(1), 376–415. https://doi.org/10.1007/s10696-022-09445-y

Chang, D., Shi, H., Liu, C., & Meng, F. (2024). Scheduling optimization of flexible flow shop with buffer capacity limitation based on an improved discrete particle swarm optimization algorithm. *Engineering Optimization*, 1–27. https://doi.org/10.1080/0305215X.2024.2328191

Chen, B., Zhang, R., Chen, L., & Long, S. (2021). Adaptive Particle Swarm Optimization with Gaussian Perturbation and Mutation. *Scientific Programming*, *2021*, 6676449. https://doi.org/10.1155/2021/6676449

Degraeve, Z., & Vandebroek, M. (1998). A Mixed Integer Programming Model for Solving a Layout Problem in the Fashion Industry. *Management Science*, *44*(3), 301–310. https://doi.org/10.1287/mnsc.44.3.301

Ezugwu, A. E., Shukla, A. K., Nath, R., & Akinyelu, A. A. (2021). Metaheuristics: a comprehensive overview and classification along with bibliometric analysis. In *Artificial Intelligence Review* (Vol. 54, Issue 6). Springer Netherlands. https://doi.org/10.1007/s10462-020-09952-0

Filipič, B., Fister, I., & Mernik, M. (2006). Evolutionary search for optimal combinations of markers in clothing manufacturing. *GECCO ’06: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 1661–1666. https://doi.org/10.1145/1143997.1144270

Fister, I., Mernik, M., & Filipič, B. (2008). Optimization of markers in clothing industry. *Engineering Applications of Artificial Intelligence*, *21*(4), 669–678. https://doi.org/10.1016/j.engappai.2007.06.002

Fister, I., Mernik, M., & Filipic, B. (2010). A hybrid self-adaptive evolutionary algorithm for marker optimization in the clothing industry. *Applied Soft Computing*, *10*, 409–422. https://doi.org/10.1016/j.asoc.2009.08.001

Gogna, A., & Tayal, A. (2013). Metaheuristics: Review and application. In *Journal of Experimental and Theoretical Artificial Intelligence* (Vol. 25, Issue 4, pp. 503–526). Taylor & Francis. https://doi.org/10.1080/0952813X.2013.782347

Gómez-Montoya, R. A., Cano, J. A., Cortés, P., & Salazar, F. (2020). A discrete particle swarm optimization to solve the put-away routing problem in distribution centres. *Computation*, *8*(4), 1–17. https://doi.org/10.3390/computation8040099

IBM. (2022). *IBM ILOG CPLEX Optimization Studio*. Relative MIP Gap Tolerance. https://www.ibm.com/docs/en/icos/22.1.1?topic=parameters-relative-mip-gap-tolerance

Jacobs-Blecha, C., Ammons, J. C., Schutte, A., & Smith, T. (1998). Cut order planning for apparel manufacturing. *IIE Transactions*, *30*(1), 79–90. https://doi.org/10.1080/07408179808966439

Jarboui, B., Damak, N., Siarry, P., & Rebai, A. (2008). A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, *195*(1), 299–308. https://doi.org/10.1016/j.amc.2007.04.096

Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: past, present, and future. In *Multimedia Tools and Applications* (Vol. 80, Issue 5). Multimedia Tools and Applications. https://doi.org/10.1007/s11042-020-10139-6

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. *IEEE International Conference on Computational Cybernetics and Simulation*, 4104–4108. https://doi.org/10.1109/ICSMC.1997.637339

M'Hallah, R., & Bouziri, A. (2016). Heuristics for the combined cut order planning two-dimensional layout problem in the apparel industry. *International Transactions in Operational Research*, *23*(1), 321–353. https://doi.org/10.1111/itor.12104

Martens, J. (2004). Two genetic algorithms to solve a layout problem in the fashion industry. *European Journal of Operational Research*, *154*(1), 304–322. https://doi.org/10.1016/S0377-2217(02)00706-3

Nascimento, D. B., Neiva De Figueiredo, J., Mayerle, S. F., Nascimento, P. R., & Casali, R. M. (2010). A state-space solution search method for apparel industry spreading and cutting. *International Journal of Production Economics*, *128*(1), 379–392. https://doi.org/10.1016/j.ijpe.2010.07.035

Nasrin, U., & Alam, S. M. R. (2023). Implementing circular economy principles in the apparel production process: Reusing pre-consumer waste for sustainability of environment and economy. *Cleaner Waste Systems*, *6*(April), 100108. https://doi.org/10.1016/j.clwas.2023.100108

Nchalala, A., Alexander, T., & Taifa, I. W. R. (2023). Establishing standard allowed minutes and sewing efficiency for the garment industry in Tanzania. *Research Journal of Textile and Apparel*, *27*(2), 246–263. https://doi.org/10.1108/RJTA-09-2021-0112

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization: An overview. *Swarm Intelligence*, *1*, 33–57. https://doi.org/10.1007/s11721-007-0002-0

Prasad, S., Panghal, M., & Ali, T. M. (2022). Developing a cost-effective and heuristic tool to solve cut order planning problems in the apparel industry. *International Journal of Mathematics in Operational Research*, *21*(1), 26–45.

Puasakul, K., & Chaovalitwongse, P. (2016). The review of mark planning problem. *Engineering Journal*, *20*(3), 91–112. https://doi.org/10.4186/ej.2016.20.3.91

Ramos-Figueroa, O., Quiroz-Castellanos, M., Mezura-Montes, E., & Kharel, R. (2021). Variation Operators for Grouping Genetic Algorithms: A Review. *Swarm and Evolutionary Computation*, *60*($1 2020). https://doi.org/10.1016/j.swevo.2020.100796

Ranaweera, R. N. M. P., Rathnayaka, R. M. K. T., & Chathuranga, L. L. G. (2023). Optimal Cut Order Planning Solutions using Heuristic and Meta-Heuristic Algorithms: A Systematic Literature Review. *KDU Journal of Multidisciplinary Studies*, *5*(1), 86–97. https://doi.org/10.4038/kjms.v5i1.66

Rose, D. M., & Shier, D. R. (2007). Cut scheduling in the apparel industry. *Computers & Operations Research*, *34*(11), 3209–3228. https://doi.org/10.1016/j.cor.2005.12.001

Shami, T. M., El-saleh, A. A., & Member, S. (2022). Particle Swarm Optimization: A Comprehensive Survey. *IEEE Access*, 10031–10061. https://doi.org/10.1109/ACCESS.2022.3142859

Shang, X., Shen, D., Wang, F.-Y., & Nyberg, T. R. (2019). A heuristic algorithm for the fabric spreading and cutting problem in apparel factories. *IEEE/CAA Journal of Automatica Sinica*, *6*(4), 961–968. https://doi.org/10.1109/JAS.2019.1911573

Shen, M., Zhan, Z., Chen, W., Gong, Y., & Member, S. (2014). Bi-Velocity Discrete Particle Swarm Optimization and Its Application to Multicast Routing Problem in Communication Networks. *IEEE Transactions on Industrial Electronics*, *61*(12), 7141–7151. https://doi.org/10.1109/TIE.2014.2314075

Silva, P. H. H. P. N. De, Lanel, G. H. J., & Perera, M. T. M. (2017). Integer Quadratic Programming (IQP) Model for Cut Order Plan. *IOSR Journal of Mathematics*, *13*(02), 76–80. https://doi.org/10.9790/5728-1302027680

Toaza, B., & Esztergár-Kiss, D. (2023). A review of metaheuristic algorithms for solving TSP-based scheduling optimization problems. *Applied Soft Computing*, *148*(January). https://doi.org/10.1016/j.asoc.2023.110908

Tsao, Y.-C., Vu, T.-L., & Liao, L.-W. (2020). Hybrid heuristics for the cut ordering planning problem in apparel industry. *Computers & Industrial Engineering*, *144*(1), 106478. https://doi.org/10.1016/j.cie.2020.106478

Tsao, Y.-C., Delicia, M., & Vu, T. L. (2022). Marker planning problem in the apparel industry: Hybrid PSO-based heuristics. *Applied Soft Computing*, *123*, 108928. https://doi.org/10.1016/j.asoc.2022.108928

Ünal, C., & Yüksel, A. D. (2020). Cut Order Planning Optimisation in the Apparel Industry. *Fibres and Textiles in Eastern Europe*, *28*(1), 8–13. https://doi.org/10.5604/01.3001.0013.5851

Wijethilake, C., Upadhaya, B., & Lama, T. (2023). The role of organisational culture in organisational change towards sustainability: evidence from the garment manufacturing industry. *Production Planning & Control*, *34*(3), 275–294. https://doi.org/10.1080/09537287.2021.1913524

Wong, W. K. Ã., & Leung, S. Y. S. (2008). Genetic optimization of fabric utilization in apparel manufacturing. *International Journal of Production Economics*, *114*(1), 376–387. https://doi.org/10.1016/j.ijpe.2008.02.012

Xiang, W., Hui, D., Li, Y., & Wen-An, Z. (2022). Hybrid optimization algorithm for cut order planning of multicolor garment. *Control and Decision*, *37*(6), 1531–1540. https://doi.org/10.13195/j.kzyjc.2020.1749

Xu, Y., Thomassey, S., & Zeng, X. (2020). Optimization of garment sizing and cutting order planning in the context of mass customization. *The International Journal of Advanced Manufacturing Technology*, *106*(1), 3485–3503.

Yang, C. L., Huang, R. H., & Huang, H. L. (2011). Elucidating a layout problem in the fashion industry by using an ant optimisation approach. *Production Planning and Control*, *22*(3), 248–256. https://doi.org/10.1080/09537287.2010.498600

Yang, Yali, Zhang, Y., Zuo, H., & Yan, N. (2023). The effective practical application of modern intelligent manufacturing technology in textile and garment industry. *International Journal on Interactive Design and Manufacturing (IJIDeM)*. https://doi.org/10.1007/s12008-023-01559-3

Yang, Yizhe, Liu, B., Li, X., Jia, Q., Duan, W., & Wang, G. (2024). Fidelity-adaptive evolutionary optimization algorithm for 2D irregular cutting and packing problem. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-024-02329-y

Appendices

Appendix I. Pseudocodes for constructive heuristics

Algorithm 1. Pseudocode of H1_T2020

```
Set residual demand $r_\beta = Q_\beta$ for $\beta \in s$
Set number of sections, $\alpha = 0$

Repeat While $\max_{\beta \in s}(r_\beta) > 0$
    Increase the value of $\alpha$: $\alpha = \alpha + 1$
    Set $P_\alpha = max\left\{P_{min}, min\left\{\left(r_\beta:\ r_\beta > 0,\ \beta \in s\right),\ P_{max}\right\}\right\}$
    Set length of section $\alpha$, $l_\alpha = 0$
    For $\beta \in s$:
        If $r_\beta \geq P_{min}$:
            If $l_\alpha \leq l_{max}$ :
                Set $temp\_G_{\alpha\beta} = \left\lfloor \frac{r_\beta}{P_\alpha} \right\rfloor$
                Determine $temp\_l_\alpha = \sum temp\_G_{\alpha\beta} Y_\beta$
                If $temp\_l_\alpha \leq l_{max}$: set $G_{\alpha\beta} = temp\_G_{\alpha\beta}$; and $l_\alpha = temp\_l_\alpha$
                **Else**: set $G_{\alpha\beta} = \left\lfloor \frac{l_{max} - l_\alpha}{Y_\beta} \right\rfloor$ and $l_\alpha = \sum_\beta G_{\alpha\beta} Y_\beta$
            **Else**: set $G_{\alpha\beta} = 0$
        **Else If** $P_{min} > r_\beta > 0$:
            **If** $l_\alpha \le l_{max}$:
                Set $temp\_G_{\alpha\beta} = \left\lceil \frac{r_\beta}{P_\alpha} \right\rceil$
                Determine $temp\_l_\alpha = \sum_\beta temp\_G_{\alpha\beta} Y_\beta$
                **If** $temp\_l_\alpha \le l_{max}$: set $G_{\alpha\beta} = temp\_G_{\alpha\beta}$; and $l_\alpha = temp\_l_\alpha$
                **Else**: set $G_{\alpha\beta} = \left\lfloor \frac{l_{max} - l_\alpha}{Y_\beta} \right\rfloor$ and $l_\alpha = \sum_\beta G_{\alpha\beta} Y_\beta$
            **Else**: set $G_{\alpha\beta} = 0$
        **Else**: set $G_{\alpha\beta} = 0$
    Update residual demand, $r_\beta = r_\beta - P_\alpha \times G_{\alpha\beta}$; $for \beta \in s$
```

Algorithm 2. Pseudocode of H3_MB2016

```
Set residual demand $r_\beta = Q_\beta$ for $\beta \in s$
Set number of sections,$\alpha$ = 0
Repeat **While** $\max_{\beta \in s}(r_\beta) > 0$:
    Increase the value of $\alpha$: $\alpha = \alpha + 1$
    Set $P_\alpha = min\left\{P\_max, max\left\{GCD\left(r_\beta: r_\beta > 0, \beta \in s\right), P_{min}\right\}\right\}$
    Set length of section $\alpha$, $l_\alpha = 0$
    **For** $\beta \in s$:
        Set $temp\_G_{\alpha\beta} = \left\lceil \frac{r_\beta}{P_\alpha} \right\rceil$
    Determine $temp\_l_\alpha = \sum_\beta temp\_G_{\alpha\beta} Y_\beta$
    Repeat **While** $temp\_l_\alpha > l_{max}$:
        Determine $\beta' = \beta$ where $temp\_G$ has the maximum occurrence of garments.
        $temp\_G_{\alpha\beta'} = temp\_G_{\alpha\beta'} - 1$
        Update length $temp\_l_\alpha = \sum_\beta temp\_G_{\alpha\beta} Y_\beta$
    Set $G_{\alpha\beta} = temp\_G_{\alpha\beta}$
    Update Residual demand, $r_\beta = r_\beta - P_\alpha \times G_{\alpha\beta}$; $for \beta \in s$
```

Algorithm 3. Pseudocode of H1+

```
Set Residual demand $r_\beta = Q_\beta$ for $\beta \in s$
Set number of sections, $\alpha$ = 0
Repeat **While** $\max_{\beta \in s}(r_\beta) > 0$
    Increase the value of $\alpha$: $\alpha = \alpha + 1$
```

```
        Set $P_\alpha = max\left\{P_{min}, min\left\{\left(r_\beta\colon r_\beta > 0,\ \beta \in s\right),\ P_{max}\right\}\right\}$
        Set length of section $\alpha$, $l_\alpha = 0$
        Set $K = [1, 2, 3,......, s]$; a list of all sizes.
        Randomly Shuffle the order of values in $K$
        For $\beta \in K$:
                If $r_\beta \geq P_{min}$:
                        If $l_\alpha \leq l_{max}$:
                                Set $temp\_G_{\alpha\beta} = \left\lfloor \frac{r_\beta}{P_\alpha} \right\rfloor$
                                Determine $temp\_l_\alpha = \sum temp\_G_{\alpha\beta} Y_\beta$
                                If $temp\_l_\alpha \leq l_{max}$: Set $G_{\alpha\beta} = temp\_G_{\alpha\beta}$; and $l_\alpha = temp\_l_\alpha$
                                ELSE: Set $G_{\alpha\beta} = \left\lfloor \frac{l_{max} - l_\alpha}{Y_\beta} \right\rfloor$ and $l_\alpha = \sum G_{\alpha\beta} Y_\beta$
                        Else: Set $G_{\alpha\beta} = 0$
                Else If $P_{min} > r_\beta > 0$:
                        If $l_\alpha \leq l_{max}$:
                                Set $temp\_G_{\alpha\beta} = \left\lceil \frac{r_\beta}{P_\alpha} \right\rceil$
                                Determine $temp\_l_\alpha = \sum_\beta temp\_G_{\alpha\beta} Y_\beta$
                                If $temp\_l_\alpha \leq l_{max}$: Set $G_{\alpha\beta} = temp\_G_{\alpha\beta}$; and $l_\alpha = temp\_l_\alpha$
                                Else: Set $G_{\alpha\beta} = \left\lfloor \frac{l_{max} - l_\alpha}{Y_\beta} \right\rfloor$ and $l_\alpha = \sum_\beta G_{\alpha\beta} Y_\beta$
                        Else: Set $G_{\alpha\beta} = 0$
                Else: Set $G_{\alpha\beta} = 0$
        Update Residual demand, $r_\beta = r_\beta - P_\alpha \times G_{\alpha\beta}$; $for\, \beta \in s$
```

Algorithm 4. Pseudocode of H3+

```
Set Residual demand $r_\beta = Q_\beta$ for $\beta \in s$
Set number of sections, $\alpha$ = 0
Repeat While $\max_{\beta \in s}(r_\beta) > 0$:
        Increase the value of $\alpha$: $\alpha = \alpha + 1$
        Set $P_\alpha = min\left\{P\_max,\ max\left\{GCD\left(r_\beta\colon r_\beta > 0, \beta \in s\right),\ P_{min}\right\}\right\}$
        Set length of section $\alpha$, $l_\alpha = 0$
        Set $K = [1, 2, 3,......, s]$; a list of all sizes
        Randomly Shuffle the order of values in $K$
        For $\beta \in K$:
                If $r_\beta > 0$:
                        Set $G_{\alpha\beta} = max\left\{0,\ min\left(\left\lceil \frac{r_\beta}{P_\alpha} \right\rceil, \left\lfloor \frac{l_{max} - l_\alpha}{Y_\beta} \right\rfloor\right)\right\}$
                        Set $l_\alpha = \sum G_{\alpha\beta} Y_\beta$
                Else: Set $G_{\alpha\beta} = 0$
        Update Residual demand, $r_\beta = r_\beta - P_\alpha \times G_{\alpha\beta}$; $for\, \beta \in s$
```

Algorithm 5. Pseudocode of H5+

```
Set Residual demand $r_\beta = Q_\beta$ for $\beta \in s$
Set $\alpha$ = 1, and $\overline{m}=2$
Repeat While $\max_{\beta \in s}\left(r_\beta\right) > 0$:
        For $\beta \in s$:
                IF $0 < r_\beta < P_{min}$: then Set $r_\beta = P_{min}$
        Set $P_\alpha =$ a random integer between $\left[max\{P_{min}, 1\},\ min\left\{P_{max},\ max\left\{P_{min},\ \max_{\beta\in s}(r_\beta)\right\}\right\}\right]$
        Set length of section $\alpha$, $l_\alpha = 0$
        Set $K = [1, 2, 3, \ldots\ldots, s]$; a list of all sizes.
        Randomly Shuffle the order of values in $K$
        For $\beta \in K$:
                If $r_\beta \le 0$; OR $l_\alpha \ge l_{max}$:
                        Set $G_{\alpha\beta} = 0$
                        SKIP the next parts and continue the FOR Loop for next value of $\beta$.
                Set $temp\_G_{\alpha\beta} =$ a random integer between $\left[0,\ \left\lfloor \frac{r_\beta}{P_\alpha} \right\rfloor\right]$
                Determine $temp\_l_\alpha = \sum_\beta temp\_G_{\alpha\beta} Y_\beta$
                If $temp\_l_\alpha \le l_{max}$:
                        Set $G_{\alpha\beta} = temp\_G_{\alpha\beta}$
                        Set $l_\alpha = temp\_l_\alpha$
                Else:
                        Set $G_{\alpha\beta} =$ a random integer between $\left[0,\ \left\lfloor \frac{l_{max} - l_\alpha}{Y_\beta} \right\rfloor\right]$
                        Set $l_\alpha = \sum_\beta G_{\alpha\beta} Y_\beta$
        Update Residual demand, $r_\beta = r_\beta - P_\alpha \times G_{\alpha\beta}$;where $\beta \in s$
        Increase the value of $\alpha$: $\alpha = \alpha + 1$
        If $\alpha \ge \overline{m}$ AND $\max_{\beta\in s}(r_\beta) > 0$:
              Use H3+ algorithm to pack all remaining demands
```

Appendix II. Pseudocodes for genetic algorithms

Algorithm 6. Pseudocode of GA

```
Input: population size ($N$); number of generations ($T$);: tournament size ($k$); crossover rate ($\rho$); mutation rate ($\eta$).
Output: A near global optimum solution
Generate 2 solutions using H1_T2020, and H3_MB2016 and append them to $Population$
Generate solutions randomly Using H1+, H3+ & H5+ until there are $N$ amount of feasible solution in $Population$
Evaluate Fitness value of each solution in $Population$
For $t \in T$:
        For $n \in N$:
                Select $Parent1$, and $Parent2$ using Tournament Selection
                Get $Child$ by calling Crossover $(Parent1, Parent2, \rho)$ Function.
                Get $Mutated\_Child$ by calling Mutation $(Child, \eta)$ Function.
                Evaluate Fitness value of $Mutated\_Child$ .
                Append $Mutated\_Child$ in variable $Children$
        $new\_Population =$ Merge $Population$ and $Children$
        Sort the $new\_Population$ ($2N$ population size) in a non-decreasing order of Fitness.
        Set $Population$= first $N$ individuals of the $new\_Population$.
        If meets the stopping criteria:
                Stop and Go To step 6
Return First solution in $Population$
```

Algorithm 7. Pseudocode of Crossover Operator

```
Input: Two selected individuals for mating ($Parent1, Parent2$); crossover rate ($\rho$)
Output: A solution as a result of their crossover ($Child$)
FUNCTION Crossover ($Parent1, Parent2, \rho$):
If Random Uniform [0,1] > $\rho$:
        Stop and Return $Parent1$
Set $G1_{\alpha\beta}$, $P1_{\alpha} = G_{\alpha\beta}$, $P_{\alpha}$ of $Parent1$;  and  $G2_{\alpha\beta}$, $P2_{\alpha} = G_{\alpha\beta}$, $P_{\alpha}$ of $Parent2$
Set $m_1$, $m_2 =$ number of sections in $Parent1$,and in $Parent2$, respectively.
Set $i_1$, $i_2 = 2$ Randomly chosen values from the discrete uniform $[1, m_1]$, where $i_1 < i_2$.
Set the first $\alpha_c = i_2 - i_1 + 1$ sections of $Child$ by copying sections $i_1$ through $i_2$ from $Parent1$
Update residual demand of $Child$, $r_\beta$, $\beta \in s$
Set $\alpha' = 1$, where $\alpha'$ is the index number of sections from $Parent2$.
Repeat While $\max_{\beta \in s}\left(r_\beta\right) > 0$:
        Create a new section of $Child$, $\alpha_c = i_2 - i_1 + 1 + \alpha'$
        If $\alpha' \le m_2$ AND $\max_{\beta \in s}\left(r_\beta\right) > P_{min}$:
                Determine variable $gcd = GCD\left(r_\beta : r_\beta > 0,\ G2_{\alpha'\beta} > 0,\ \beta \in s\right)$
                Set $P_{\alpha_c} = max\left\{P_{min},\ min\left\{P2_{\alpha'},\ gcd\right\}\right\}$, where $P2_{\alpha'}$ is the number of plies in section $\alpha'$ of $Parent2$
                Set length of section $\alpha_c$, $l_{\alpha_c} = 0$
                For $\beta \in s$:
                        If $r_\beta > 0$, AND $G2_{\alpha'\beta} > 0$ :
                                Set $G_{\alpha_c\beta} = min\left\{\left\lfloor\frac{r_\beta}{P_{\alpha_c}}\right\rfloor, \left\lfloor\frac{l_{max} - l_{\alpha_c}}{Y_\beta}\right\rfloor\right\}$
                                Update $l_{\alpha_c} =$ lengths of the section $\alpha_c$
                        Else Set $G_{\alpha_c\beta} = 0$
                Set $\alpha' = \alpha' + 1$
        Else:
                Set $P_{\alpha_c} = max\left\{P_{min},\ GCD\left(r_\beta : r_\beta > 0,\ \beta \in s\right)\right\}$
                Set length of section $\alpha_c$, $l_{\alpha_c} = 0$
                For $\beta \in s$:
                        If $r_\beta > 0$, THEN
                                Set $G_{\alpha_c\beta} = min\left\{\left\lceil\frac{r_\beta}{P_{\alpha_c}}\right\rceil, \left\lfloor\frac{l_{max} - l_{\alpha_c}}{Y_\beta}\right\rfloor\right\}$
                                Update $l_{\alpha_c} =$ lengths of the section $\alpha_c$
                        Else Set $G_{\alpha_c\beta} = 0$
                Stop While Loop
        Update residual demand for Child $r_\beta$, $\beta \in s$
Return $Child$
```

Algorithm 8. Pseudocode of Mutation Operator

```
Input: Chromosome solution for mutation operation; $\eta$: probability of mutating of a gene $(Child)$.
Output: A solution as a result of mutation operation $(Mutated\_Child)$
FUNCTION Mutation $(Child,\ \eta)$ :
For $\alpha \in m$: where $m$ is number of sections in $Child$
        For $\beta \in s$:
                If random uniform [0,1] $< \eta$:
                        $Mutated\_Child =$ Call FUNCTION Mutate $(\alpha, \beta, G_{\alpha\beta}, P_\alpha)$
Return $Mutated\_Child$
FUNCTION Mutate $(\alpha, \beta, G_{\alpha\beta}, P_\alpha)$:
Set $G_{\alpha\beta} = 0$
Update residual demand $r_\beta$, $\beta \in s$
Set $\mathbb{I} = \{\alpha' = 1....m,\ where\ \alpha' \neq \alpha,\ and P_{\alpha'} \leq r_\beta\}$
Repeat While $\max_{\beta \in s}(r_\beta) > 0$:
        If $\mathbb{I}$ is not NULL:
                Determine $l_{\alpha'} =$ list of lengths of the sections for $\alpha' \in \mathbb{I}$
                Determine $\alpha'' = \alpha'$ and $P_{\alpha''} = P_{\alpha'}$ where $l_{\alpha''} = min\{l_{\alpha'}\}$
                Set $G_{\alpha''\beta} =$ previous value $+min\left\{\left\lceil\frac{r_\beta}{P_{\alpha''}}\right\rceil, \left\lfloor\frac{l_{max} - min\{l_{\alpha'}\}}{Y_\beta}\right\rfloor\right\}$
                Set $\mathbb{I} = \mathbb{I} \setminus \{\alpha''\}$
        Else:
                Create new section $\alpha'' = m + 1$
                Set $P_{\alpha''} = min\left\{P_{max}, max\{P_{min}, r_\beta\}\right\}$
                If $P_{\alpha''} \geq r_\beta$:
                        Set $G_{\alpha''\beta} = 1$
                Else:
                        Set $G_{\alpha''\beta} = min\left\{\left\lfloor\frac{r_\beta}{P_{\alpha''}}\right\rfloor, \left\lfloor\frac{l_{max}}{Y_\beta}\right\rfloor\right\}$
        Update residual demand $r_\beta$, $\beta \in s$
Return $Mutated\_Child$
```

Appendix III. Pseudocodes for particle swarm optimization

Algorithm 9. Pseudocode of PSO Algorithm

```
Input: Swarm size $(N)$; number of iterations $(T)$; explore factor $(\varphi)$; perturbation rate $(\eta)$
Output: A near global optimum solution $(Gbest)$
Generate 2 solutions using H1_T2020, and H3_MB2016 Algorithm and $(N-2)$ solutions randomly
Using H1+, H3+ & H5+ algorithms in $Swarm$, where each particle is $X_P$
Generate Initial velocity $V_P$ in range $[-\varphi, \varphi]$ for each particle $X_P$ in $Swarm$
For $p \in N$:
        Set $Pbest_P = X_P$
Set $Gbest$ = Best position vector of a particle in the swarm with lowest fitness value (local
search method)
For $i \in T$:
        For $p \in N$:
                Update $X_P^i$ & $V_P^i$ by calling $Particle_{Update}(X_P^{i-1}, V_P^{i-1}, Gbest, Pbest_P, \varphi, \eta)$ function.
                If Fitness $(X_P^i)$ < Fitness $(Pbest_P)$:
                        Set $Pbest_P = X_P^i$
                If Fitness ($X_P^i$) < Fitness ($Gbest$):
                     Set $Gbest = X_P^i$
        If meets the stopping criteria:
                Stop the For loop and go to step 6
Return $Gbest$
```

Algorithm 10. Pseudocode of Updating Position & Velocity Vector ($Particle_{Update}$)

```
Input: Particle's position vector ($X_P^{i-1}$); particle's position vector ($V_P^{i-1}$); global best position vector ($Gbest$); particle's personal best position vector ($Pbest$); explore factor ($\varphi$); perturbation rate ($\eta$)
Output: Updated particle ($X_P^i$)
FUNCTION $Particle_{Update}\left(X_P^{i-1}, V_P^{i-1}, Gbest, Pbest_P, \varphi, \eta\right)$ :
Calculate $Y_P^i = \left\{y_{p1}^i, y_{p2}^i, \dots, y_{p\alpha}^i \dots, y_{pn}^i\right\}$ by Equation (7)
Update velocity $V_P^i = \left\{v_{p1}^i, v_{p2}^i, \dots, v_{p\alpha}^i \dots, v_{pm}^i\right\}$ by Equation (8)
Calculate $\Lambda_P^i = \left\{\lambda_{p1}^i, \lambda_{p2}^i, \dots, \lambda_{p\alpha}^i \dots, \lambda_{pn}^i\right\}$ by Equation (9)
Initialize $new\_X$ with 0 section.
$R = \left(r_\beta : \beta \in s\right)$ : Update residual demand for solution $new\_X$
Set $sec =$ number of sections in $\Lambda_P^i$
For $\alpha \in sec$:
        If $\lambda_{p\alpha}^i > \varphi$:
                $new\_x_\alpha = gbest_\alpha$: Copy $\alpha^{th}$ section of $Gbest$ and create a new section $\alpha$ for $new\_X$
        Else If $\lambda_{p\alpha}^i < -\varphi$:
                $new\_x_\alpha = pbest_\alpha$: Copy $\alpha^{th}$ section of $Pbest_P$ and create a new section $\alpha$ for $new\_X$
        Else:
                $temp\_x_\alpha = x_\alpha^{i-1}$ : Copy $\alpha^{th}$ section of $X_P^{i-1}$ and create a new section $\alpha$ for $new\_X$
                $new\_x_\alpha = Perturbation\left(temp\_x_\alpha, R, \eta\right)$
        $R = \left(r_\beta : \beta \in s\right)$ : Update residual demand for solution $new\_X$
        If $max\left(R = r_\beta : \beta \in s\right) \le 0$:
                STOP and Go to Update dimension of velocity vector to match the Position vector's dimension
Set $q =$ Total number of sections in $X_P^{i-1}$
For $\alpha \in range\left[sec + 1, q\right]$:
        $temp\_x_\alpha = x_\alpha^{i-1}$ : Copy $\alpha^{th}$ section of $X_P^{i-1}$ and create a new section $\alpha$ for $new\_X$
        $new\_x_\alpha = Perturbation\left(temp\_x_\alpha, R, \eta\right)$
        $R = \left(r_\beta : \beta \in s\right)$: Update residual demand for solution $new\_X$
        If $max\left(R = r_\beta : \beta \in s\right) \le 0$:
                Stop and Go to Update dimension of velocity vector to match the Position vector's dimension
If Positive residual demand: Use H1+ algorithm to pack all remaining demands.
Update dimension of velocity vector to match the Position vector's dimension
Return $new\_X$ as $X_P^i$, and $V_P^i$
```

```
FUNCTION $Perturbation(new\_x_\alpha, R, \eta)$:
Set $G_\alpha\ P_\alpha = G_\alpha\ P_\alpha$ of $new\_x_\alpha$
If Random Uniform $[0,1] <\ p\_rate$:
        $p1,\ p2 = Sorted\left[P_\alpha,\ min\langle P\_max,\ max\left\{P\_min,\ \max_{\beta \in s}\left(r_\beta\right)\right\}\rangle\right]$
        Set $P_\alpha$= Random Integer between $[p1,p2]$
For $\beta \in s$:
        If Random Uniform $[0,1] <\ p\_rate$:
                Set $G_{\alpha\beta} = 0$
                If $r_\beta > 0$:
                        Determine length of the section $L_\alpha$
                        Determine $temp = max\left\{0,\ min\left(\left\lceil \frac{r_\beta}{P_\alpha} \right\rceil,\ \left\lfloor \frac{l_{max} - l_\alpha}{Y_\beta} \right\rfloor\right)\right\}$
                        Set $G_{\alpha\beta} =$ Random integer $[0,temp]$
Return $new\_x_\alpha = (G_\alpha\ P_\alpha)$
```