Methods for Solving the Linear Cutting-stock Problem on Several Slitting Machines with Minimizing the Number of Waste and Knife Rearrangements

Vladislav Klimenko, Liudmila Shchegoleva
Petrozavodsk State University (PetrSU)
Petrozavodsk, Russia
{klimenko, schegoleva}@petrsu.ru

***Abstract*—This article presents the methods for solving the linear cutting problem. The linear cutting problem is, in general, an optimization problem that involves arranging the given types of material (orders) in such a way as to minimize waste and/or maximize the use of raw materials, taking into account the restrictions on the number of knives, the width of the master roll and the required orders. A special case of the problem includes an additional condition of minimizing knife rearrangements. There are also multiple slitting machines with different widths of the master roll. It makes the problem more complex and requires proper algorithms. The following approaches to its solution are considered: the full enumeration method, and a genetic search based on genetic and evolutionary algorithms. Pseudocode is presented for various methods of solving the problem. An example is provided for the genetic search method. A comparison is made between both algorithms in terms of algorithmic complexity, controllability of execution time and accuracy.**

I. The linear cutting-stock problem

Cellulose and wood pulp are sent to mixing pools, where they are blended and transformed into a uniform paper mass. This mass is then fed under pressure to the paper-making machine (PMM), where the process of producing paper sheet begins. The technological process includes several stages [1]: first, the dewatering of the paper mass occurs on the wire section, then in the press section, after which the sheet is dried and calendered to give it the desired texture and density. When the paper sheet reaches the required characteristics, it is wound onto a drum roll. Using a crane, the sheet is removed from the PMM roll and transferred to longitudinal cutting machines, where it is cut into rolls of specified formats.

Typically, several presses operate at the production facility, which can have different master roll widths. This variety allows for the optimization of the production process and adaptation to various customer requirements. Each press can be adjusted to work with specific characteristics of the paper sheet, ensuring flexibility in the production process and efficient use of resources.

The paper cutting process includes several key stages [2]. First, the paper sheet is fed into slitting machines (SM), where it is carefully aligned and secured. Then, with the help of sharp knives or cutting discs, the sheet is divided into rolls of the desired width. These machines can be set up for various formats, allowing for the production of rolls of different widths depending on the order. After cutting, the finished rolls are transported via conveyors to the packaging line. Here, they undergo final quality checks and are packaged in wrapping paper, ensuring protection of the rolls during storage and transportation.

SM play a crucial role in the finishing process of paper production, enabling the precise conversion of a master rolls into smaller, manageable formats tailored to customer specifications. These machines are designed to handle a wide range of paper types, including those with varying thicknesses and properties. Equipped with advanced technology, SM ensure that each cut is clean and accurate, minimizing waste and maximizing efficiency. The operation typically involves feeding the large paper rolls (master rolls) into the SM, where they are unwound and passed through a series of rotating blades that slice the paper into narrower strips. The resulting rolls are then automatically wound onto cores, ready for further processing or shipping. By integrating SM into the production line, manufacturers can enhance their operational flexibility, responding swiftly to market demands and providing high-quality products that meet diverse customer needs.

The interplay between ecology and money is increasingly relevant in the context of industrial processes like paper cutting [3]. As companies strive to maximize profits, the pressure to reduce waste and enhance sustainability becomes paramount. The efficient use of resources not only has environmental benefits—such as reducing deforestation and minimizing landfill contributions—but also translates into significant cost savings. By optimizing the cutting process, businesses can decrease material waste, which directly impacts their bottom line. This alignment of ecological responsibility with financial viability creates a compelling business case for adopting greener practices in manufacturing.

Moreover, as consumers become more environmentally conscious, companies that prioritize sustainable practices can gain a competitive advantage in the market. Investing in

technologies that minimize waste and enhance energy efficiency can lead to improved brand loyalty and potentially higher sales. This is particularly pertinent in industries like paper production, where the sourcing and processing of raw materials have a direct impact on the environment. By viewing ecological initiatives not just as an obligation but as a strategic investment, businesses can cultivate a positive image, attract environmentally aware consumers, and ultimately enhance their profitability while contributing to the health of the planet.

However, the cutting of the paper sheet must be organized in such a way as to minimize waste and ensure maximum efficiency in material usage. This leads to the necessity of a mathematical formulation of the linear cutting-stock problem (LCP), where it is essential to optimally distribute the rolls according to the width of the formats in order to achieve the best results in the cutting process, taking into account the restrictions on the widths of the master rolls and the requirements for the final product.

II. Mathematical model

Let:

- c — the total number of SM;
- $L = \{L_1, L_2, \ldots, L_c\}$ — the widths of each SM's master roll;
- m — is a total quantity of orders;
- $M = \{(l_1, q_1), (l_2, q_2), \ldots, (l_m, q_m)\} : i = \overline{1, m}, j = \overline{1, m}, l_i \neq l_j$ — the set of orders, where each order i has a format width l_i and a number of rolls q_i;
- v — is a quantity of orders in a cutting layout;
- $p = (p_1, p_2, \ldots, p_v)$ — a cutting layout, which is an ordered sequence, where for each p_j there exists an order $(l_k, q_k) \in M$ such that $p_j = l_k$, with the number of elements in different layouts possibly being different;
- a — is a quantity of cutting layouts in a cutting plan for one SM;
- $P = \{p^1, p^2, \ldots, p^a\}$ — the cutting plan for one SM, which is an ordered sequence of layouts;
- $G = \{P^1, P^2, \ldots, P^c\}$ — the global cutting plan for all SM.

It is necessary to find such a global cutting plan G for which the total width of waste will be minimal. Let the function $\tau(P)$ of waste sums is given by the expression:

$$\tau(P) = \sum_{i=1}^{a} \left(L - \sum_{j=1}^{v_i} p_j^i \right). \tag{1}$$

Then the objective function is:

$$\tau'(G) = \sum_{P \in G} \tau(P) \to \min. \tag{2}$$

The following constraints are given. The constraint imposes a condition on the layout: the sum of formats must not exceed the width of their master roll:

$$\sum_{p_j \in P^i} p_j < L_i, \quad 1 \leq i \leq c. \tag{3}$$

The constraint describes the condition for fulfilling all orders:

$$\sum_{P \in G} \sum_{i=1}^{a} \sum_{j=1}^{v_i} I(p_j^i = l_k) = q_k, \quad k = \overline{1, m}. \tag{4}$$

However, to improve the efficiency of the enterprise, a requirement can be added to the task to minimize the number of knife rearrangements for the sought cutting plan [4]. The cutting plan should be minimized, first of all, in terms of waste, and then minimized in terms of the number of knife rearrangements.

Let

$$\sigma(p^1, p^2) = v_2 - \max_{0 \leq k \leq \min\{v_1, v_2\}} \{j : j \leq k, p_j^1 = p_j^2\} \tag{5}$$

— a function calculating the knife rearrangements to transition from layout p^1 to layout p^2. v_1 and v_2 are the quantity of orders in the p^1 and p^2 respectively. It is equal to the difference between the number of formats in layout p_2 and the number of matching formats at the beginning of the ordered sequence of formats for layouts p^1 and p^2. Matching formats at the beginning of the sequence do not require knife rearrangement, so they can be ignored.

The function for calculating the number of knife rearrangements for the cutting plan P can be expressed as follows:

$$\mu(P) = v_1 + \sum_{i=1}^{a-1} \sigma(p^i, p^{i+1}). \tag{6}$$

This sum consists of the number of formats in the initial layout p^1, as the knives are not initially set. The sum also includes the number of necessary knife rearrangements for each pair of layouts p^i and p^{i+1}.

Then the function for calculating the number of knife rearrangements for the global cutting plan G can be expressed as follows:

$$\mu'(P) = \sum_{P \in G} \mu(P). \tag{7}$$

Thus, the problem of minimizing the number of knife rearrangements for the SM in the layout P for the LCP with waste minimization will be as follows:

$$\mu'(G) \to \min. \tag{8}$$

Subject to the condition that the minimum of the function: $\tau'(G)$ is achieved on the set G.

The mathematical model extends another one from the article [5]. But it's algorithms can't be applied, because there are several SM with different widths. The mathematical model requires more complex methods which can distribute orders between multiple SMs.

III. Full enumeration method

The exhaustive search method represents a comprehensive approach to solving the LCP, ensuring the discovery of a globally optimal solution. However, when faced with high structural complexities inherent in linear cutting tasks, this

method encounters significant limitations due to its computational complexity, which is so high that it sometimes renders it impractical for real-world applications [6]. The algorithmic complexity for this method will be exponential in relation to the number of orders [7].

One variant of exhaustive search using recursion is proposed. This algorithm finds a global cutting plan with minimal waste and the number of knife rearrangements. At each iteration, it considers the current global cutting plan and returns the best available option. Initially, the considered global cutting plan is empty. The algorithm can be divided into two main parts: the first part checks that the current solution satisfies Constraint (4), and if so, it can be returned as the best for the current iteration; the second part is executed if the current solution does not satisfy Constraint (4), and thus, additional formats need to be added to the cutting plan. At each iteration of the algorithm, a new cutting plan is created and stored in the variable *newP*. The new format is added to the last cut (the variable *LastCut*) if it does not violate Constraint (3); otherwise, a new cut is added (the variable *newLastCut*). A new recursive iteration of the enumeration is called for the new cutting plan, and the resulting global cutting plan (the variable *NewBestG*) is compared with the current best global cutting plan (the variable *BestG*). The current best global cutting plan will be replaced with the plan from the variable *NewBestG* in three cases:

1) if the variable *BestG* contains an empty global cutting plan;
2) if the global cutting plan in the variable *NewBestP* is better than the global cutting plan in the variable *BestG* in terms of waste;
3) or if the global cutting plan in the variable *NewBestG* is equal in waste but better in the number of knife rearrangements.

After considering all possible options for adding orders, the best global cutting plan is returned from the variable *BestG*. The pseudo code of the recursive enumeration algorithm is presented in Algorithm 1.

The input of the algorithm is:

- A set of orders $M = \{(l_1, q_1), (l_2, q_2), \ldots, (l_m, q_m)\}$;
- the widths of SM $L = \{L_1, L_2, \ldots, L_c\}$;
- the current global cutting plan G (initially, this is an empty global cutting plan).

The output of the algorithm is:

- The global cutting plan in the variable *BestG*, minimizing waste and the number of knife rearrangements.

Algorithm 1 Enumerate Global Plan Cuts

```
1: EnumerateGlobalPlanCuts(M, L, G={{∅}, ..., {∅}}):
2:   BestG = ∅
3:   # Counter for order formats
4:   MCount = {l_1 : 0, l_2 : 0, ..., l_m : 0}
5:   for P in G do
6:     for p in P do
7:       for l in p do
8:         MCount[l]++
9:   # Check if the current format fulfills the order
10:  orderCompleted = 1
11:  for i = 1 to m do
12:    if M[i].q ≠ MCount[M[i].l] then
13:      orderCompleted = 0
14:  if orderCompleted = 1 then
15:    # If G contains the necessary number of orders,
16:    # it cannot be expanded with new formats
17:    BestG = G
18:    return BestG
19:  P = ∅
20:  for i = 1 to m do
21:    if M[i].q < MCount[M[i].l] then
22:      # Skip fulfilled orders
23:      continue
24:    for j = 1 to c do
25:      P = G_j
26:      PQuantity = |P|
27:      # Last cut in the current cutting plan
28:      LastCut = P[PQuantity]
29:      # Total width of the last cut
30:      LastCutLen = Σ_{j=1}^{LastCut} LastCut[j]
31:      # If adding a new format to the last cut
32:      # does not violate the maximum width constraint
33:      if LastCutLen + M[i].l ≤ L_j then
34:        newLastCut = LastCut + {M[i].l}
35:        newP = P
36:        # Add the new format to the last cut
37:        newP[PQuantity] = newLastCut
38:      else
39:        # Add a cut to the new cutting layout
40:        newP = P + {M[i].l}
41:      G' = G
42:      G'_j = newP
43:      NewBestG = EnumeratePlanCuts(M, L, G')
44:      # Check if the new cutting plan is the best option
45:      if NewBestG ≠ ∅ then
46:        if BestG = ∅ then
47:          BestG = NewBestG
48:        if τ'(BestG) > τ'(NewBestG) then
49:          BestG = NewBestG
50:        ifτ'(BestG) = τ'(NewBestG)andμ'(BestG) >
   μ'(NewBestG) then
51:          BestG = NewBestG
52:    return BestG
```

IV. Genetic search method

Genetic and evolutionary algorithms are search and optimization methods that can significantly improve the processes of solving cutting stock problems with the minimization of knife rearrangements. One of the fundamental advantages of these algorithms is the ability to flexibly control execution time, allowing them to be adapted to the specific requirements of a given task. At the same time, each iteration does not degrade the quality of the current solution, making the process less vulnerable to local minima [8].

The effectiveness of applying genetic and evolutionary algorithms is closely related to the power of computational resources, which often implies the use of high-performance computing systems, such as supercomputers and clusters [9].

A. The algorithm

The following random search algorithm with elements of a genetic algorithm is proposed [10]. It was called genetic search (GS). It iteratively forms new populations $S = \{S_1, S_2, \ldots, S_C\}$, where S_t is the t-th global cutting plan G, and $S_{t,i}$ is the i-th cutting plan P in the t-th global cutting plan G. The initial population S_0 consists of random global cutting plans that meet the Constraints (3) and (4). The size of the initial population determines the number of iterations, thus affecting the accuracy and speed of the algorithm.

Each iteration begins with mutation (function GlobalMutate) for the global cutting plans of the current population S_t. The Mutate function creates a new cutting plan P', and if it is better than the old cutting plan, that is, $\tau'(G) > \tau'(G')$, the global cutting plan is replaced, $G = G'$. After that, crossover (function Crossingover) is performed for a pair of adjacent cutting plans $S_{t,2i-1}$ and $S_{t,2i}$. The Crossingover function returns the better of the two cutting plans, thereby forming a global cutting plan $S_{t+1,i}$ in the new population. This reduces the size of the population S_{t+1} by half compared to S_t. As a result of the algorithm's operation, the population is reduced to a single global cutting plan G', which will be no worse than the cutting plans from the initial population $G \in S_0$. The larger the size of the initial population, the more iterations there are, and consequently, the higher the probability of obtaining a global cutting plan G' with minimal waste and knife rearrangements. Let,

- t be the iteration number of the search and population.
- S be the set of global cutting plans G that meet the Constraints (3)-(4); in other words, the population of cutting plans. And S_t is the population at the search iteration t.
- C be the size of the current population.
- Algorithm 2: Mutate(P) is the mutation function that randomly shuffles the cutting layouts $p \in P$ and the order of formats within them. The function shuffle randomly shuffles the order of the cutting formats p. The use of the Mutate function increases the likelihood of obtaining an improved cutting plan in terms of minimizing the number of knife rearrangements. Input: cutting plans P. Output: cutting plan with altered order P'.

Algorithm 2 Mutate Algorithm

```
1: Mutate(P):
2:    P' = P
3:    for i = 1 to a do
4:       # Shuffle the cutting layout P'_i
5:       shuffle(P'[i])
6:    # Shuffle the cutting layouts in P'
7:    shuffle(P')
8:    return P'
```

- Algorithm 3: GlobalMutate(P) is the mutation function for global cutting plan that applies Mutate function that increases the likelihood of obtaining an improved global cutting plan in terms of minimizing the number of knife rearrangements. Input: global cutting plan G. Output: global cutting plan with altered order G'.

Algorithm 3 Mutate Algorithm for a global cutting plan

```
1: GlobalMutate(G):
2:    G' = G
3:    for P ∈ G' do
4:       # Mutate the cutting plan P
5:       P' = Mutate(P)
6:       if τ(P') < τ(P) then
7:          P = P'
8:       else if τ(P') = τ(P) and μ(P') < μ(P) then
9:          P = P'
10:   return G'
```

- Algorithm 4: Crossingover(G^1, G^2) is a crossover function for two global cutting plans G^1 and G^2. This function selects the best of the global cutting plans G^1 and G^2 based on the objective function (8). Using the Crossingover function allows the population to get rid of the less optimal plans. Input: global cutting plans G^1, G^2. Output: new global cutting plan G'.

Algorithm 4 Crossingover

```
1: Crossingover(G^1, G^2) :
2:    if τ'(G^1) < τ'(G^2) then
3:       return G^1
4:    else
5:       if τ'(G^1) = τ'(G^2) and μ'(G^1) < μ'(G^2) then
6:          return G^1
7:       else
8:          return G^2
```

The genetic search algorithm is presented in Algorithm 5. Input: S_0 – the initial population of global cutting plans. Output: final cutting plans P'.

Algorithm 5 GeneticSearch

```
1: GeneticSearch(S_0) :
2:   t = 0
3:   while S_t > 1 do
4:     n = S_t
5:     for i = 1 to n do
6:       S_{t,i} = GlobalMutate(S_{t,i})
7:     # Write the crossover result to the next population
8:     for i = 1 to ⌊n/2⌋ do
9:       S_{t+1,i} = Crossingover(S_{t,2i-1}, S_{t,2i})
10:    # If the population size is odd,
11:    # transfer the remaining element
12:    if n mod 2 = 1 then
13:      S_{t+1,i} = S_{t,⌈n/2⌉}
14:    t = t + 1
15:  return S_{t,1}
```

B. Algorithmic complexity

In total, the GS will perform $O(\log(|S_0|))$ iterations, as the size of the populations is halved each time, rounding up. However, each iteration contains the processing of cutting plans in the functions GlobalMutate, Crossingover, and τ. A cutting plan consists of $\sum_{i=1}^{|M|} q_i$ formats, as it must satisfy Constraint (4). Thus, the overall algorithmic complexity of genetic search is equal to

$$O\left(\sum_{i=1}^{|M|} q_i \log(|S_0|)\right) \tag{9}$$

C. Example

Let:

- $c = 2$ — the total number of SM;
- $L = \{L_1, L_2\}$ — the widths of each SM, where $L_1 = 60$ and $L_2 = 40$;
- $M = \{(10, 2), (20, 3), (30, 1), (40, 2), (50, 1)\}$ — the set of orders.

In the table I the initial population S_0 is presented.

TABLE I
INITIAL POPULATION S_0

Plan	Global cutting Plans	τ'	μ'
$S_{0,1}$	$P^1 = \{\{10, 20, 30\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	20	6
$S_{0,2}$	$P^1 = \{\{10, 20\}, \{30\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	80	6
$S_{0,3}$	$P^1 = \{\{10, 30\}, \{20\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	80	6
$S_{0,4}$	$P^1 = \{\{10, 20, 30\}\}, P^2 = \{\{40\}, \{10\}, \{30\}\}$	40	6
$S_{0,5}$	$P^1 = \{\{10, 20, 30\}, \{10, 30\}\}, P^2 = \{\{40\}\}$	20	5

Each iteration of genetic search is described below for the S_0.

1) $t = 0$: The results of applying genetic search on S_0 are presented in the Table II. There are no better mutated plans. So no one is replaced with a new mutated global cutting plan G'. The results of applying GlobalCrossingover(G) on S_0 are $S_{0,1}$, $S_{0,4}$ and $S_{0,5}$. $S_{0,1}$ was chosen from $S_{0,1}$ and $S_{0,2}$ because the value of τ' for $S_{0,1}$ is less. The same for $S_{0,3}$ and $S_{0,4}$. $S_{0,5}$ has not a pair. The final population S_1 is presented.

TABLE II
ITERATION $t = 0$

Plan	New global cutting plans *G'* in GlobalMutate	τ'	μ'
$S_{0,1}$	$P^1 = \{\{30, 10, 20\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	20	6
$S_{0,2}$	$P^1 = \{\{30\}, \{10, 20\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	80	6
$S_{0,3}$	$P^1 = \{\{20\}, \{10, 30\}, \{40\}\}, P^2 = \{\{30, 10\}\}$	80	6
$S_{0,4}$	$P^1 = \{\{10, 20, 30\}\}, P^2 = \{\{40\}, \{30\}, \{10\}\}$	40	6
$S_{0,5}$	$P^1 = \{\{30, 10, 20\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	20	6
Choice	**Selected global cutting plans *G'***	τ'	μ'
Old	$P^1 = \{\{10, 20, 30\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	20	6
Old	$P^1 = \{\{10, 20\}, \{30\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	80	6
Old	$P^1 = \{\{10, 30\}, \{20\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	80	6
Old	$P^1 = \{\{10, 20, 30\}\}, P^2 = \{\{40\}, \{10\}, \{30\}\}$	40	6
Old	$P^1 = \{\{10, 20, 30\}, \{10, 30\}\}, P^2 = \{\{40\}\}$	20	5
Choice	**Crossingovered plans**	τ'	μ'
$S_{0,1}$	$P^1 = \{\{10, 20, 30\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	20	6
$S_{0,4}$	$P^1 = \{\{10, 20, 30\}\}, P^2 = \{\{40\}, \{10\}, \{30\}\}$	40	6
$S_{0,5}$	$P^1 = \{\{10, 20, 30\}, \{10, 30\}\}, P^2 = \{\{40\}\}$	20	6
Plan	**Global cutting plans S_1**	τ'	μ'
$S_{1,1}$	$P^1 = \{\{10, 20, 30\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	20	6
$S_{1,2}$	$P^1 = \{\{10, 20, 30\}\}, P^2 = \{\{40\}, \{10\}, \{30\}\}$	40	6
$S_{1,3}$	$P^1 = \{\{10, 20, 30\}, \{10, 30\}\}, P^2 = \{\{40\}\}$	20	5

2) $t = 1$: The results of applying genetic search on S_1 are presented in the Table III. Mutation function can improve a plan as it happened with $S_{1,3}$. The new value of $S_{1,3}$ is $P^1 = \{\{10, 30, 20\}, \{10, 30\}\}, P^2 = \{\{40\}\}$, because μ' is 4. The old values was 5, τ' is the same. Results of applying GlobalCrossingover(G) on S_1 are $S_{1,1}$ and $S_{1,3}$. The final population S_2 is presented.

TABLE III
MUTATION ON S_1

Plan	New global cutting plans *G'* in GlobalMutate	τ'	μ'
$S_{1,1}$	$P^1 = \{\{20, 10, 30\}, \{40\}\}, P^2 = \{\{30, 10\}\}$	20	6
$S_{1,2}$	$P^1 = \{\{30, 10, 20\}\}, P^2 = \{\{40\}, \{30\}, \{10\}\}$	40	6
$S_{1,3}$	$P^1 = \{\{10, 30, 20\}, \{10, 30\}\}, P^2 = \{\{40\}\}$	20	4
Choice	**Selected global cutting plans *G'***	τ'	μ'
Old	$P^1 = \{\{10, 20, 30\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	20	6
Old	$P^1 = \{\{10, 20, 30\}\}, P^2 = \{\{40\}, \{10\}, \{30\}\}$	40	6
New	$P^1 = \{\{10, 30, 20\}, \{10, 30\}\}, P^2 = \{\{40\}\}$	20	4
Choice	**Crossingovered plans**	τ'	μ'
$S_{1,1}$	$P^1 = \{\{10, 20, 30\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	20	6
$S_{1,3}$	$P^1 = \{\{10, 30, 20\}, \{10, 30\}\}, P^2 = \{\{40\}\}$	20	4
Plan	**Global plans**	τ'	μ'
$S_{2,1}$	$P^1 = \{\{10, 20, 30\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	20	6
$S_{2,2}$	$P^1 = \{\{10, 30, 20\}, \{10, 30\}\}, P^2 = \{\{40\}\}$	20	4

3) $t = 2$: Results of applying GlobalMutate(G) on S_2 are presented in the Table IV. There are no better mutated plans. So no one is replaced with a new mutated global cutting plan G'. The results of applying GlobalCrossingover(G) on S_2 is $S_{2,2}$. It was chosen from $S_{2,1}$ and $S_{2,2}$ because the value of μ' for $S_{2,2}$ is less, when τ' value is the same. The final population S_3 is presented.

TABLE IV
MUTATION ON S_2

Plan	**Mutated global cutting plans *G'***	τ'	μ'
$S_{2,1}$	$P^1 = \{\{20, 10, 30\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	20	6
$S_{2,2}$	$P^1 = \{\{30, 10, 20\}, \{10, 30\}\}, P^2 = \{\{40\}\}$	20	6
Choice	**Selected global cutting plans *G'***	τ'	μ'
Old	$P^1 = \{\{10, 20, 30\}, \{40\}\}, P^2 = \{\{10, 30\}\}$	20	6
Old	$P^1 = \{\{10, 30, 20\}, \{10, 30\}\}, P^2 = \{\{40\}\}$	20	4
Choice	**Crossingovered plans**	τ'	μ'
$S_{2,2}$	$P^1 = \{\{10, 30, 20\}, \{10, 30\}\}, P^2 = \{\{40\}\}$	20	4
Plan	**Global plans**	τ'	μ'
$S_{3,1}$	$P^1 = \{\{10, 30, 20\}, \{10, 30\}\}, P^2 = \{\{40\}\}$	20	4

4) $t = 3$: The population S_3 consists of only one global cutting plan. It means that the global cutting plan $S_{3,1}$ is a final solution:

$$S_{3,1} = \{\{\{10, 30, 20\}, \{10, 30\}\}, \{\{40\}\}\} \\ \tau'(S_{3,1}) = 20, \mu'(S_{3,1}) = 4 \quad (10)$$

It is better than the solutions from the initial population S_0. But it is not the best, there are better solutions. For example, the solution G is provided:

$$G = \{\{\{20, 40\}\}, \{\{10, 30\}, \{10, 30\}\}\} \\ \tau'(G) = 0, \mu'(G) = 4 \quad (11)$$

V. COMPARISON OF ALGORITHMS

The full enumeration algorithm has exponential algorithmic complexity relative to the number of orders, while GS has $O\left(\sum_{i=1}^{|M|} q_i \log(|S_0|)\right)$. It is evident that GS is faster than complete enumeration; however, this speed is achieved at the expense of considering fewer possible options.

The full enumeration method forces the consideration of all possible options. In contrast, for GS, the number of iterations can be set based on the size of the initial population, which allows for control over execution time.

Moreover, the full enumeration method guarantees consideration of all options and thus finds the best cutting plan. GS can only find a global cutting plan G' that is no worse than the initial population S_0, but increasing the size of the initial population increases the likelihood of finding a better global cutting plan.

VI. CONCLUSION

The article examines a particular case of the one-dimensional cutting stock problem with the minimization of waste and knife rearrangements. The mathematical model includes several SM with different widths of the master roll. For this case, two algorithms are proposed to find the optimal solution: the full enumeration method and genetic search (GS). The full enumeration method guarantees finding the optimal solution. However, it is significantly slower than GS. The GS is faster and allows for the regulation of execution time, but it does not guarantee that the answer will be optimal. Thus, GS is better suited for solving the problem.

In further research, it is planned to expand the mathematical model and methods using the following criteria and constraints: order fulfillment timelines, queue for PMM, and consideration of multiple PMMs.

REFERENCES

[1] U. A. Romoldovich and V. A. Kuzneczov, "Matematicheskie modeli i metody‘ ucheta srokov produkcii v zadache raskroya tamburov bumagodelatel‘ny‘x mashin," *Ucheny‘e zapiski Petrozavodskogo gosudarstvennogo universiteta*, no. 4 (141), pp. 112–115, 2014.

[2] M. H. Correia, J. F. Oliveira, and J. S. Ferreira, "Reel and sheet cutting at a paper mill," *Computers & Operations Research*, vol. 31, no. 8, pp. 1223–1243, 2004.

[3] R. Khan, C. I. Pruncu, A. S. Khan, K. Naeem, M. Abas, Q. S. Khalid, and A. Aziz, "A mathematical model for reduction of trim loss in cutting reels at a make-to-order paper mill," *Applied Sciences*, vol. 10, no. 15, p. 5274, 2020.

[4] M. Martin, H. H. Yanasse, and L. L. Salles-Neto, "Pattern-based ilp models for the one-dimensional cutting stock problem with setup cost," *Journal of Combinatorial Optimization*, vol. 44, no. 1, pp. 557–582, 2022.

[5] V. Klimenko and L. Shchegoleva, "Methods for solving the linear cutting-stock problem with minimizing the number of waste and knife rearrangements," *Inženernyj vestnik Dona (RUS)*, no. 3, 2025.

[6] O. I. El-Dessouki and W. H. Huen, "Distributed enumeration on between computers," *IEEE Transactions on Computers*, vol. 29, no. 09, pp. 818–825, 1980.

[7] A. B.-F. Branch, "Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm," 1997.

[8] C. Salto, E. Alba, and J. M. Molina, "Analysis of distributed genetic algorithms for solving cutting problems," *International Transactions in Operational Research*, vol. 13, no. 5, pp. 403–423, 2006.

[9] A. M. Easwaran and S. Drossopoulou, "A parallel genetic algorithm approach to the knife change minimisation problem," in *the Proceedings of the sixth Parallel Computing Workshop (PCW'96), Japan*, 1996.

[10] R. V. Voronov, A. I. Shabaev, and V. V. Klimenko, "Genetic algorithm for the linear cutting problem with allowances for production volumes," *Software Engineering*, no. 1, pp. 35–43, 2024.