Irregular Polygon Strip Packing for the Production Process Transition of the Fashion Industry

ANDRÉ ANDRINGA, University of Twente, The Netherlands

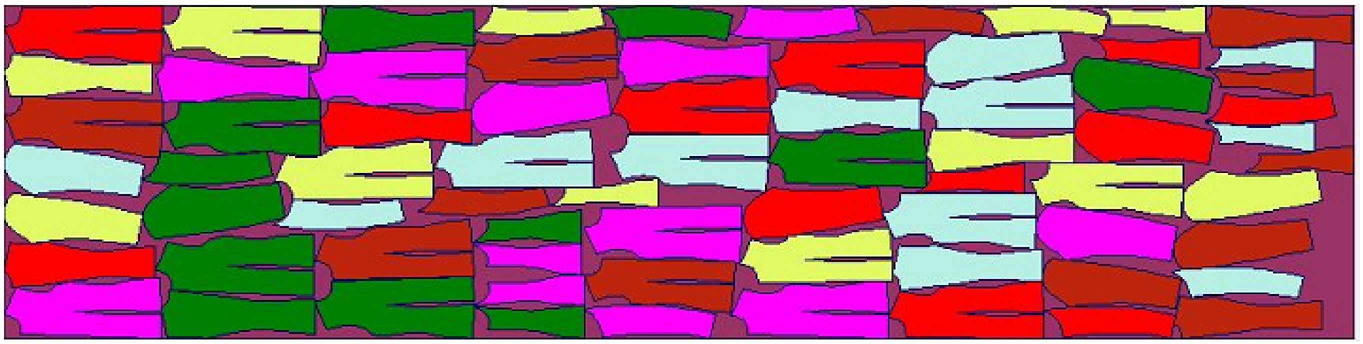

Fig. 1. Example of a generated marker file.

The fast fashion industry is widely recognised as one of the most polluting industries globally. In order to address this issue, a shift from mass production to made-to-order or made-to-measure models has emerged as a potential solution. These alternative models address overproduction by manufacturing clothing items upon order and tailoring garments to each client's specific measurements, reducing returns. However, implementing these models in the fashion industry brings forth new challenges in the production process, particularly in generating marker files for pattern placement on fabric. Existing algorithms used for pattern placement optimisation can take days to converge, which poses a significant problem for unique garment production. In this paper, I propose two improvements to the sliding no-fit polygon generation algorithm, significantly enhancing its speed. Additionally, I introduce a novel ordering heuristic that improves efficiency and serves as a partial seed for the genetic algorithm and Q-learning. Furthermore, I propose a hybrid placement heuristic that prioritises border placement and then switches to the bottom-left strategy, resulting in improved packing efficiency. In conclusion, my advancements offer promising results for addressing the challenges of made-to-order and made-to-measure in the fashion industry and can contribute to reducing waste and improving sustainability.

Additional Key Words and Phrases: 2D irregular cutting and packing problem, no-fit polygon, heuristics, genetic algorithm, reinforcement learning

1 INTRODUCTION

Fast fashion has become a global phenomenon, with the fashion industry producing 150 billion garments each year [16]. Most of these clothes are manufactured using the ready-to-wear model, where garments are produced in standard sizes and then sold. Accurately predicting consumer demand for each item remains a challenge, leading to a significant sustainability issue: overproduction. Annually, clothing worth over $600 million goes unsold and is ultimately incinerated [25]. Another problem is the high rate of clothing returns, which range from 10% to 60% across different categories, resulting in a substantial number of unsold items [5], These issues contribute to the fashion industry's significant environmental impact, particularly due to the production cost of fabric [27]. Cotton, the most used fabric for garments, requires enormous amounts of water, pesticides, insecticides and energy to produce [24], making the fashion industry one of the most polluting sectors worldwide [8].

To address these challenges, a potential solution is transitioning from mass production to made-to-order (MTO) or made-to-measure (MTM) models. In both approaches, garments are produced after they are ordered, and with MTM garments are tailored to each individual's specific measurements. These models offer the advantage of reducing waste and have been shown to be profitable [7].

However, implementing such a shift in the production process presents challenges. Traditionally, in the fashion industry, the process involves designing a garment and then using an algorithm to determine the most efficient marker file. A marker file contains the pattern packing layout, specifying how the garment pieces should be arranged on the fabric to minimize waste. Based on this marker file hundreds of thousands of identical garments are produced. In contrast, the MTO and MTM models introduce more variation in the produced garments. Each variation requires the generation of a new marker file. This poses a problem as existing algorithms often take days to converge upon a solution. With the introduction of unique garments in newer models, running the algorithm for every few ordered garments becomes time-consuming and fails to meet the demand for clothing.

Therefore, these new production models require a shift in the evaluation criteria for packing algorithms. Time efficiency becomes crucial as the algorithm needs to generate marker files for every few ordered garments. For example, Zara sold approximately 2,981,965 units in the fiscal year 2018/2019, averaging to about one garment

TScIT 39, July 7, 2023, Enschede, The Netherlands

sold every 10 seconds [28]. To effectively address fluctuating demands, marker files are required to be generated within a maximum time frame of 1-2 minutes. It is essential to develop packing algorithms that can quickly generate marker files. Second to time is the importance of minimising fabric waste. MTM and MTO limit fabric waste by reducing overproduction. Therefore, slightly less efficient packing algorithms are acceptable due to time constraints. The main contributions of this work include:

- Two enhancements to the sliding no-fit polygon generation algorithm, greatly improving its speed.
- A novel ordering heuristic that improves efficiency and serves as a seed for the genetic algorithm (GA) and Q-learning.
- A hybrid placement heuristic which starts with prioritising placement near the borders, and ends with the bottom-left strategy, which improves packing efficiency.

In conclusion, the advancements presented in this paper show promising results in extensive testing with various datasets. These advancements have the potential to address the challenges faced by the fashion industry in MTO and MTM scenarios, contributing towards waste reduction and improved sustainability.

2 RELATED CONCEPTS

Solutions to the packing problem can be subdivided into three major parts. Firstly, a method is needed for overlap detection. Secondly, the order in which the pieces will be packed is decided. Lastly, the placement position is determined for the pieces in the given order. In the following part I will explain the major concepts that this thesis builds upon.

2.1 Overlap Detection

There are three options when it comes to overlap detection between polygons. Firstly, trigonometry functions can be used for overlap detection. Secondly, polygons can be mapped as pixels in a grid, which can be used for overlap detection. And lastly, the no-fit polygon (NFP) method can be used. The concept of NFPs is introduced in 1966 by Art [3], under the term ‘shape envelop.’ The term ‘no-fit polygon’ was later coined by Adamowicz and Albano [1]. Multiple approaches exist for constructing NFPs, including the sliding approach implemented by Mahadevan in 1984 [22].

The main advantage of using the NFP method over other approaches is its reusability. In many packing algorithms, multiple iterations are required to find the optimal solution. With trigonometry or the pixel method, overlap detection needs to be performed repeatedly, leading to increased computational overhead. However, with the NFP method, the majority of calculations are performed once at the beginning. Subsequently, only the point-in-polygon test needs to be conducted, which is computationally less intensive [9]. Therefore the NFP is the most time-efficient way of doing overlap detection [17]. Nevertheless, adoption of the NFP method in real world industries has been slow. This is mainly because of the complex implementation and the numerous edge cases NFP creation methods need to handle [4].

2.1.1 No-Fit Polygon Generation Methods. There are multiple different methods of generating the NFP. The best method depends on the properties of the polygon as well as the use case.

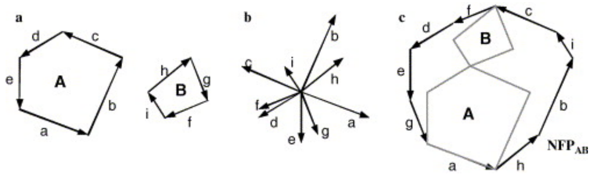

Fig. 2. NFP generation of convex polygons.[4]

For convex polygons, the generation of an NFP is straightforward. Given polygons A and B, where the edges of polygon A are oriented in the counterclockwise direction, and those of polygon B in the clockwise direction (Fig. 2a), the process involves translating all edges to a single starting point (Fig. 2b) and constructing the NFP by appending all edges in counterclockwise order (Fig. 2c).

For non-convex polygons the process of creating the NFP is more complex, there are two main approaches for doing this.

The first approach entails converting the non-convex polygon into multiple convex polygons, generating NFPs for each of these polygons, and then combining them to form one complete NFP. Nonetheless, a disadvantage of this approach is that the final step is often time-consuming, and for larger polygons, it can take up to twenty minutes [2]. There are improvements that could be made which can speed up this process.

In this paper, the focus will be on exploring the sliding approach, which is a widely used method for solving the irregular polygon packing problem. Unlike the previously discussed approach, the sliding approach does not have the disadvantages that results in increased creation times of NFPs. The sliding method involves the controlled movement of one polygon, typically referred to as the orbital polygon B, along a stationary polygon A. By following this procedure, the no-fit polygon of A and B (NFP$_{AB}$) can be derived through a three-step process. Initially, polygons A and B should be positioned in a manner where they make contact without overlapping, as illustrated in Figure 3a. Subsequently, polygon B is incrementally slid around polygon A, ensuring continuous contact while avoiding any overlap between the two, as exemplified in Figure 3b. Finally, the trajectory of a reference point on polygon B is traced throughout the sliding motion, resulting in the formation of the NFP, as depicted in Figure 3c. It is important to note that the choice of reference point on polygon B is arbitrary but serves as a basis for subsequent overlap checks between polygons A and B. Upon obtaining the NFP$_{AB}$, three distinct collision scenarios may arise. Firstly, if the reference point precisely aligns with the edges of the NFP$_{AB}$, it implies that polygons A and B make contact without overlapping as is showcased in Figure 4a. Secondly, if the reference point on polygon B lies within the boundaries of the NFP$_{AB}$, it indicates an overlap between polygons A and B as illustrated in Figure 4b. Lastly, if the reference point falls outside the NFP$_{AB}$, it signifies the absence of any contact between polygons A and B, see Figure 4c.

2.2 Packing Order

The second part of the packing process is determining the order in which the pieces are placed. This packing order significantly impacts the packing efficiency. To optimise the packing order and

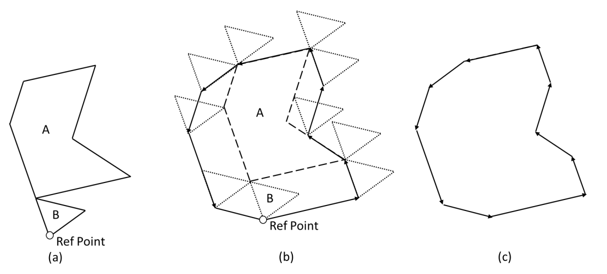

Fig. 3. Sliding algorithm for creating an NFP.

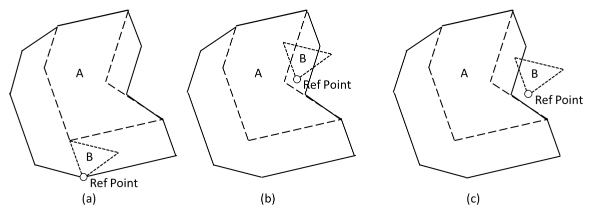

Fig. 4. The different cases of the NFP intersection check (a) touching (b) intersection (c) no touch

minimise fabric waste, two algorithms will be utilised in this study. Firstly, the GA, which is a widely used approach in addressing the irregular polygon packing problem. Secondly, a reinforcement learning technique called Q-learning will be employed.

2.2.1 Genetic Algorithm. The GA was introduced by Holland in 1975 [13]. It is rooted in the concept of survival of the fittest and draws inspiration from biological evolution. Genes serve as the fundamental building blocks of the GA, representing solutions to the given problem. Additionally, the GA comprises a fitness function and offspring creation process based on these genes. This latter step typically involves parent selection, crossover to combine the genes of two parents into new combinations, and mutation to introduce random exploration [15].

2.2.2 Q-learning. The name Q-learning was introduced by Watkins in 1989 [32]. Q-learning is a machine learning algorithm, which has three main building blocks. An agent, a set of states and a set of actions for every state. The agent interacts with the environment by choosing an action, which makes the state change. Then a reward is calculated and given as feedback to the agent. In the case of Q-learning a so-called Q-table is updated with the newly received reward. The agent then chooses the next action based on the Q-table. This decision is based on where the biggest rewards are, which are exploitative actions. Nonetheless, a certain degree of randomness is incorporated to ensure both exploration and exploitation.

2.3 Placement Heuristics

The last step of the packing algorithm is deciding where the polygons get placed. This placement is done repeatedly when deciding the order based on a meta heuristic or reinforcement learning algorithm. Therefore, achieving fast placement is important. The speed of the placement algorithm directly correlates with the number of iterations the algorithms can undergo, leading to improved results.

2.3.1 Bottom-Left. Bottom-left (BL) is the most well-known placement heuristic. It is used widely in the rectangular packing problem, but also in the irregular polygon packing problem [9]. The BL heuristic operates by sliding a piece downwards as far as possible, followed by sliding it to the left as far as possible. This process is repeated, alternating between the downwards and leftwards directions. When using NFPs, there is no need to execute this algorithm since the NFPs themselves offer a convenient way to identify the bottom left point. This can be achieved by examining all the points that form the NFP and choosing the bottom left point.

3 STATE OF THE ART

The state of the art in packing problems revolves around efficient solutions for mass production of clothes. Algorithmic approaches have proven highly effective, in recent years they have surpassed human performance, which previously had a 5% advantage [6, 23]. However, these algorithms often require days to optimise a solution, posing a challenge in terms of time efficiency. This section explores key aspects of the current research, including overlap detection, packing order optimisation, and placement heuristics, to address these challenges and improve the packing process.

3.1 Overlap Detection

The NFP method has proven to be the most efficient form of overlap detection for the problem at hand. In particular, the sliding NFP creation approach holds promise due to its superior efficiency compared to decomposing and reconstructing irregular polygons. The initial demonstration of the sliding approach was presented by Mahadevan [22]. Subsequently, a more robust algorithm was described by Burke et al. [4], and the most recent improvement was presented by Luo and Rao in 2022 [20].

Considering the importance of generating solutions within a limited timeframe, the NFP approach appears promising due to its highly efficient collision detection capability [4]. However, it is worth noting that there are recent algorithms developed to address the packing problem for mass production, which utilise the pixel approach [29, 30]. This choice is likely influenced by the complexity involved in implementing the NFP creation algorithm, an issue that the pixel method does not encounter [31].

3.2 Packing Order

Most of the research conducted in the area of packing problems revolves around optimising the packing order, as it has a significant impact on packing efficiency [12]. To achieve this optimisation, a wide range of heuristics and meta-heuristics are experimented with. Among these approaches, the GA has been widely utilised in solving packing problems. For instance, Luo et al. recently employed a hybrid approach combining the GA with variable neighborhood search, resulting in promising results [21]. Similarly, Junior et al. also employed the GA to determine the optimal packing order [14].

In the field of irregular polygon packing, reinforcement learning algorithms, like Q-learning, are still in the early stages of application [10]. However, it is worth noting that reinforcement learning has demonstrated remarkable progress in other fields. For example, in the realm of gaming, reinforcement learning algorithms have managed to defeat professional teams in complex multiplayer games [26]. Additionally, in various domains, reinforcement learning techniques have achieved significant advancements. This is demonstrated by studies on deep reinforcement learning [18].

In conclusion, optimizing the packing order is crucial for improving efficiency in packing problems. The GA has shown promise in solving these problems, while reinforcement learning algorithms like Q-learning are still in the early stages of application. However, considering their success in other domains, exploring and comparing RL and GA techniques offer potential for advancements in packing efficiency and sustainability.

3.3 Placement Heuristics

While the BL heuristic remains the standard and most commonly used placement heuristic, alternatives exist that offer improvements [12]. Building upon the BL heuristic, Liu and Teng introduced an improved version known as the Improved Bottom-Left (IBL) heuristic [19]. Another enhancement, the Bottom-Left Fill (BLF) heuristic, was developed by Gomes and Oliveira to address the issue of filling holes in the packing process [11]. These algorithms are typically designed for use in conjunction with the pixel method for overlap detection or the standard trigonometry approach. However, when working with polygons, these algorithms become more straightforward, and even the basic BL implementations already consider hole-filling in their decision-making process.

Not all studies rely solely on simple heuristics for polygon placement. A recent study by Tsao et al. utilised particle swarm optimisation to determine piece placement, yielding highly competitive results in terms of fabric and time efficiency for mass production [29]. However, in the context of MTO and MTM scenarios, where time constraints are critical, these algorithms are not feasible. For instance, the process of creating a marker file for six blouses with the algorithm developed by Tsao et al. consumed an excessive 30,891 seconds of CPU time, equivalent to approximately 8.6 hours. Considering the urgent need for swift solutions in MTO and MTM applications, this algorithm proves unsuitable for such emerging requirements.

3.4 Conclusion

Advancements in overlap detection, packing order optimisation, and placement heuristics have been made in packing problems. The NFP method, especially sliding NFP creation, shows promising results [4], while the pixel approach offers a solution which is simpler to implement. Packing order optimisation benefits from the GA and has potential for reinforcement learning techniques. Placement heuristics, like BL, have improved versions such as IBL and BLF. Time constraints in custom manufacturing scenarios pose challenges for efficient solutions. Further research is needed to address these emerging application requirements.

4 PROPOSED METHODS

The aim of this study is to address the packing problem under time constraints and optimise each step of the packing process. Specifically, improvements will be explored in overlap detection, followed by investigating the potential of GA or Q-learning for optimising the packing order. Lastly, a search will be conducted for a more efficient placement heuristic to optimise the final step of the process.

4.1 Overlap Detection

As stated before, NFP is the most time efficient overlap detection method. However, the initial creation of the needed NFPs still takes a significant amount of time. Hence, efforts have been made to enhance the generation process by minimising the number of intersection checks required.

4.1.1 New Touchpoint detection method. To explain the changes made to the algorithm proposed by Luo and Rao [20], it is necessary to understand the structure of their original algorithm. This original algorithm includes additional steps to handle edge cases. Since these edge cases are not relevant to clothing patterns, they have been disregarded in this study.

The main components of the algorithm consist of three steps: finding touchpoints, determining the translation vector from these touchpoints, and computing the translation distance along the given vector. Among these steps, finding touchpoints is the most time-consuming step with a complexity of $O(n*m)$, where n and m represent the number of vertices in polygons A and B, respectively.. This means that every vertex of polygon A needs to be checked against every vertex of polygon B.

The key to this optimisation lies in the step determining the translation length. The results of the intersection checks done during this step can be saved and used to speed up the search of touchpoints.

To enhance the speed of finding touchpoints, the translation length determination step stores the vertices that collide when moving along the vector. With this information, the subsequent step of finding touchpoints only needs to determine the type of touchpoint that occurs, such as a vertex touching another vertex or a vertex touching an edge. This approach is significantly less time-consuming, and implementing this improvement has resulted in considerably faster generation times for the NFPs.

4.1.2 Improved Point Exclusion. As suggested by Luo and Rao, the calculation of the sliding vector's length involves creating boundaries parallel to the movement vector and excluding points based on these boundaries [20]. In addition to these parallel boundaries, I propose perpendicular boundaries. These perpendicular boundaries are generated by considering the starting points of the polygons as well as the end points after applying the movement vector. Figure 5 illustrates these perpendicular boundaries alongside the method's parallel boundaries.

The introduction of perpendicular boundaries offers the potential to expedite the collision check process by eliminating numerous unnecessary points in certain cases. It is essential to recognise that this approach also introduces additional overhead, which could potentially slow down the collision check.

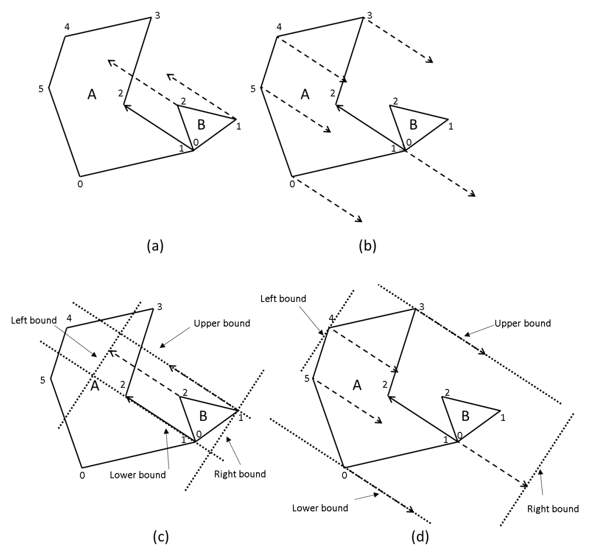

Fig. 5. Extended Point exclusion, (a, b) polygons with translation vectors, (c, d) boundaries based on those points and vectors.

4.2 Packing Order

Packing order optimisation generally has the biggest effect on the fabric usage efficiency. This step is the most time-consuming in current solutions. The problem with optimising this step is that there is a very clear trade off between solution quality and solution time. I have tried to get a solution of the highest possible quality in the limited time. To do this I have looked at both existing solutions in the space of irregular polygon packing and at Q-learning which has been used more and more in optimisation problems in other fields.

4.2.1 Improved ordering heuristics. Starting of, I will examine simple sorting methods that can yield satisfactory, but not optimal solutions. One of the most intuitive approaches is sorting the pieces from largest to smallest and packing them in that order. This ordering method aligns with our natural inclination to start with the largest pieces and fit the smaller ones into the spaces created by them. However, it is important to note that depending on the shape of the polygons, a situation may arise where the pieces with a smaller area, are significantly taller. If these pieces cannot fit into the gaps created by larger pieces, it significantly increases fabric consumption and significantly reduces packing efficiency.

To address this issue, one potential solution is to consider sorting the pieces by height. This approach can be beneficial when dealing with elongated polygons that have minimal area, as it improves the packing efficiency in such cases. However, it is important to note that sorting by height may lead to suboptimal results in other scenarios where the area of the polygons plays a more significant role.

A novel formula has been invented to determine the ordering of pieces based on a combination of area and height. The formula can be found in equation 1. This formula takes into account the quadratic nature of the area variable and reduces the influence of height on the outcome by dividing it by two. Testing has shown that this hybrid method outperforms sorting by area or height alone, providing better solutions for the packing problem.

$$Score = area + \frac{height^2}{2} \tag{1}$$

4.2.2 Genetic Algorithm. The GA can help with further optimisation of the packing order and is widely applied to the given problem. One disadvantage of the GA is that it can take a long time for it to converge to an optimal solution. However, the heuristics explained in the previous section, used for ordering the pieces, outperform random ordering. Hence, there is no need to start the GA from scratch. The created heuristics can be used to generate an initial population for the GA, which can then further improve upon these heuristics. This process is called seedling. Because there is not one heuristic which performs best in all situations, the seedling for the GA is done using all three described heuristics. The GA is initialised with one third of the population sorted by area, another third sorted by height and the last third sorted by the custom combination heuristic. This process of seeding the GA improves the time it costs to find a reasonable solution dramatically.

Besides the aforementioned seeding, there are several design decisions that need to be made to adapt this problem for the GA. Initially, it is necessary to determine how the genes will represent the solution to the problem. In this paper, the approach taken is to assign a unique number between 0 and n (where n is the number of polygons) to each gene. The genome then consists of n genes, forming an ordered list of pieces.

Once the solution is represented by genes, the next decision involves implementing the crossover operation. Unlike binary strings, this task is not straightforward since each piece can only appear once in the genome. To address this, the crossover is performed by randomly selecting a number of genes from the first parent, followed by the genes that are not yet present in the genome, in the order they appear in the second parent. For the other child, the same process is followed, but the genes are taken first from the second parent and then filled up with the genes of the first parent. An example illustrating this process with a total of ten polygons and a randomly selected number of six genes from the parent can be found in Figure 6. Finally, the reward function needs to be implemented, in this case the reward is equal to the fabric efficiency of the packing order.

The parameters were chosen as follows: the crossover rate is set to 0.9 and the mutation rate is set to 0.2, based on a study by Junior et al. [14]. Additionally, the population size is set to be equal to the number of polygons in the given problem.

4.2.3 Q-learning. Reinforcement learning has proven effective in optimising complex problems, as demonstrated in the application of Q-learning in optimising complex tasks like playing the Dota 2

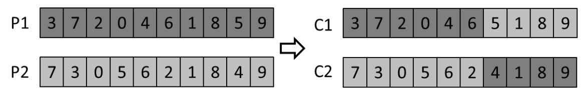

Fig. 6. Visualisation of crossover between parents P1 and P2, creating children C1 and C2

video game [26]. When applying Q-learning to the irregular polygon packing problem, it is necessary to represent the problem in states. However, using the order of the pieces as states would result in an excessively large number of states, making it infeasible for the agent to explore them within the given time constraints. To address this, states are represented as tuples of two sets. The first set contains the packed pieces, while the second set contains the pieces that are yet to be packed. For example, the initial state would be represented as ({}, {0, ..., *n*}), where *n* represents the total number of pieces, and each piece is denoted by a number. A subsequent state, where piece 1 is chosen as the first to be placed, could be represented as (1, 0, 2, ..., *n*). The set of possible actions for each state corresponds to the second set in the tuple, allowing any unpicked piece to be selected for packing. The challenge arises in defining the reward function. Since evaluating the quality of non-final states is infeasible, the feedback received is limited to the final reward. This reward is calculated based on the fabric usage efficiency of the packing order. The score is added to the results list for visited states, and the Q-table is updated with the new average. The pseudo-code for this process is presented in Algorithm 1.

Algorithm 1 Q-learning algorithm

```
1:  Run the evaluation for the seeding heuristics
2:  while $time \leq duration$ do
3:      $episode \leftarrow episode + 1$
4:      $eps \leftarrow epsylon\_decay(Episode)$
5:      $state \leftarrow get\_start\_state()$                ▷ ({}, {0, ..., n})
6:      $order \leftarrow emptylist$
7:      while $!state.is\_terminal()$ do
8:          $action \leftarrow select\_action(state[1], eps)$    ▷ Next piece
9:          $order \leftarrow take\_action(order, eps)$
10:     end while
11:     $reward \leftarrow get\_reward()$                     ▷ Fabric efficiency
12:     if $reward \geq best\_reward$ then
13:         $best\_reward \leftarrow reward$
14:         $best\_order \leftarrow order$
15:     end if
16:     for $i = 0...n$ do                   ▷ Update results- and Q-table
17:         $sub\_order \leftarrow set(order[: i])$
18:         Append $reward$ to $R[sub\_order]$
19:         $Q[sub\_order] \leftarrow average(R[sub\_order])$
20:     end for
21: end while
22: Return $best\_order, best\_reward$
```

4.3 Placement Heuristics

The BL placement heuristic is very well established in the field of packing problems, however it is not optimal. Therefore I have tried to improve on this placement heuristic. In the following section, I will detail the steps I have taken to build upon the bottom-left placement heuristic.

4.3.1 Bottom and Left. The initial issue I observed with the BL heuristic was its tendency to place pieces in the middle of the packing area, when this meant only a slight downward shift, compared

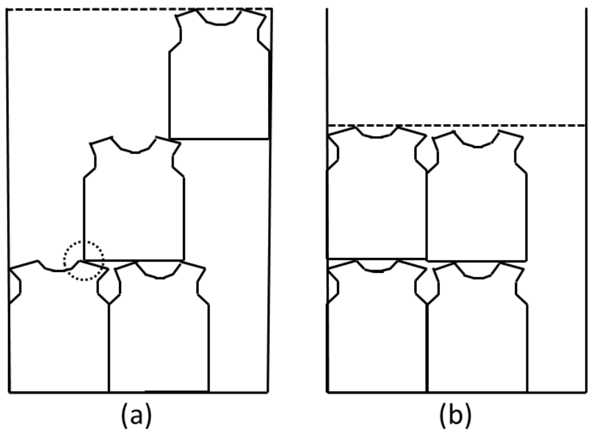

Fig. 7. Visualisation of how bottom-left can lead to inefficiencies, (a) BL heuristic, (b) bottom and left heuristic

to placing pieces against the left border. This situation is illustrated in Figure 7. The height gain achieved by such placements was minimal for the affected piece and could cause subsequent pieces to be positioned higher, resulting in inefficiencies within the BL algorithm's outcomes. To address this problem, a new heuristic has been developed that incorporates both the height (y-value) and width (x-value) of the pieces. Specifically, the x-value is multiplied by a small coefficient, ensuring that priority is still given to placing pieces as low as possible. However, when faced with the choice between placing a piece slightly lower in the middle or positioning it against the left border but slightly higher, the latter option is preferred. The formula utilised for ranking potential packing positions depicted in equation 2.

$$placement\_score = y + 0.05x \tag{2}$$

By employing this formula, the ranking of packing positions is designed to mitigate the influence of small differences in the y-value that would otherwise prompt the placement of pieces in the middle. As a result, more efficient packings are achieved. The visualisation of this heuristic can be observed in Figure 8.b, where a piece placed along the line receives an equal score. This line demonstrates that pieces positioned against the left border but slightly higher are assigned the same score as pieces placed in the middle but slightly lower. It serves as a visual representation of the formula's priority of placing pieces as much as possible to the left.

4.3.2 Border Heuristic. The second problem I found was that the big pieces were all placed to the left. This can result in inefficiencies because of the shapes of the polygons. An example of where this leads to inefficiencies can be found in figure 9. The polygons are irregular and therefore cause gaps between them. These gaps are generally bigger compared to the gaps between the piece and the boundary. A solution to this given problem would be a heuristic which prioritises placing pieces near the border. This is the reasoning behind the novel heuristic, shown in equation 3. Where max_width is the fabric width minus the width of the piece. Figure 8.c shows a line on which a polygon has the same score. Showing that towards

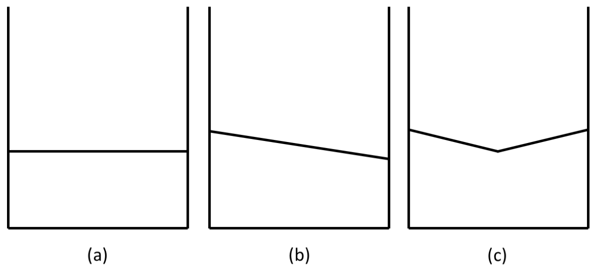

Fig. 8. Visualisation of heuristics which shows a line which receives the same score for the heuristic: (a) Bottom-Left, (b) Bottom and Left (c) Border.

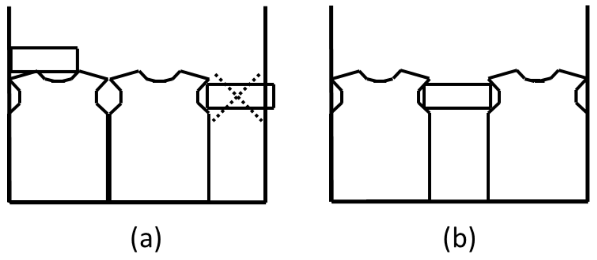

Fig. 9. Example of when the border placement heuristic would improve packing efficiency.

the borders, the polygon can be placed higher while keeping the score constant. This last placement heuristic has gotten the best results in testing.

$$placement\_score = y + (min(0.05 * x, 0.05 * (max\_width - x))) \quad (3)$$

4.3.3 Hybrid Placement Heuristic. Once implemented, it became evident that this heuristic also possesses certain drawbacks. The most notable flaw I observed was the suboptimal placement of the last few pieces. Examining the packing arrangement, it became apparent that the heuristic strongly favors the borders, resulting in vacant space in the middle where pieces could be shifted to minimise the overall packing area. To address this issue, I have devised a hybrid approach. At the start, the algorithm employs the border heuristic until a certain number of pieces are placed, and subsequently transitions to using the BL heuristic. To determine the optimal point for switching from the border heuristic to the BL heuristic, various percentages ranging from 30% to 80% were tested. The results indicate that the most effective percentage is 60%. By combining these two heuristics, more efficient packings have been achieved.

5 RESULTS

In order to test the improvements made they are tested against six benchmark problems. The data-sets for these benchmarks are found on the website of ESICUP [1]. These data-sets can be subdivided into two groups. Firstly, there are realistic test-problems from the fashion industry, the data-sets in this group are: Albano, Dagli, Mao and Marques. Secondly, there are artificial test problems, which do not resemble clothing patterns, but they are useful for testing the robustness and speed of the NFP creation algorithm. The number of polygons for every dataset can be found in Table 1. The number of NFPs that are created is the number of polygons squared, as for every polygon a NFP needs to be created with every other polygon.

[1]https://www.euro-online.org/websites/esicup/data-sets/

Table 1. Comparison of NFP creation times between the proposed sliding algorithm (PSA) and Luo[15]

Dataset	N	Luo and Rao[20]	PSA	Diff. (%)	PSA_PPE	Diff. (%)
Albano	8	1.87	0.62	66.64	0.62	0.83
Dagli	10	2.03	0.82	59.43	0.83	-0.54
Fu	12	1.12	0.75	32.67	0.77	-1.76
Jakobs1	25	17.10	7.37	56.92	7.21	2.19
Mao	9	4.65	0.98	78.99	0.96	1.69
Marques	8	1.94	0.67	65.29	0.67	0.72

5.1 NFP

The first test using these benchmarks is to evaluate the two improvements in generating the no-fit polygons. The algorithms are implemented in Python. Table 1 presents the times taken to generate all possible NFPs, considering each polygon with every other polygon. To ensure reliable results, these measurements have been repeated 100 times.

The table clearly demonstrates that the proposed sliding algorithm (PSA) outperforms the algorithm described by Luo and Rao [20]. On average, the algorithm with the new touchpoint detection method is 59.99% faster compared to Luo and Rao's algorithm. Additionally, the perpendicular point exclusion (PPE) can accelerate the algorithm in cases where perpendicular borders exclude many points. However, in other scenarios, it can slightly slow down the algorithm due to the extra computational checks required . On average, the PPE step reduces runtime by 0.52%.

5.2 Packing Order

The same data was used to test the different ordering heuristics. Every problem was solved using the three different ordering heuristics. The resulting fabric usage efficiencies can be found in table 2. Firstly, the pieces were sorted by area, starting with the piece with the biggest area. Secondly, the pieces were sorted by their height, again starting with the highest pieces and working towards the shorter pieces. Lastly, the custom ordering heuristics is used which combines the height and the area. On average, this hybrid heuristic achieves a performance improvement of 4.03% and 4.54% compared to sorting by area and sorting by height, respectively.

5.2.1 Genetic Algorithm. To test the improvement of the GA the same data-sets were used. The algorithm ran for 60 seconds with the population size set to the number of pieces in the packing problem. The cross rate was set at 0.9 and the mutation rate at 0.2. These tests have been repeated 10 times, since the GA relies on randomness for exploration, the results can differ per test. The average fabric utilisation and the best utilisation are shown in table 3. The mean average improvement compared to the maximum efficiency from the

Table 2. Fabric usage percentage of different ordering heuristics.

Dataset	area	height	hybrid
Albano	75.25	77.38	80.08
Dagli	72.76	71.42	72.67
Fu	75	65.77	79.75
Jakobs1	70	70	75.38
Mao	65.77	71.02	65.77
Marques	76.01	76.15	83.63

Table 3. Results of Seeding and the GA in terms of fabric efficiencies

Dataset	Seed best	GA avg	GA best
Albano	80.08	81.47	83.03
Dagli	72.76	76.95	78.94
Fu	79.75	83.82	86.36
Jakobs1	75.38	80.16	81.67
Mao	71.02	76.14	77.06
Marques	83.63	83.63	83.63

Table 4. Results of Seeding and Q-learning in terms of fabric efficiencies

Dataset	Seed best	Q-learning avg	Q-learning best
Albano	80.08	81.20	82.99
Dagli	72.76	76.17	77.09
Fu	79.75	83.23	83.82
Jakobs1	75.38	75.38	75.38
Mao	71.02	75.38	77.56
Marques	83.63	86.83	89.84

seeds is 3.26%, these results are promising. Comparing the results for the test-set Jakobs1, the average fabric efficiency found after running the GA for 60 seconds is only .04% worse compared to the hybrid methodology presented by Junior et al. in 2013 which ran for 13,498 seconds [14].

5.2.2 Q-learning. The Q-learning algorithm has been tested using a similar approach to the GA. Q-learning also ran for 60 seconds per problem. The number of episodes is determined based on the time constraint. Additionally, the initial epsilon value is set to 0.9, and the epsilon decay rate is set to 0.01. The tests with Q-learning are conducted 10 times, following the same reasoning as the GA. Q-learning manages to improve fabric usage efficiency in most cases. The mean of the average improvements is 2.60%. This improvement is smaller than the improvements made by the GA. Nonetheless, it is worth to note that for the Marques data-set the improvements ares significantly better compared to the GA refer to table 4. This suggests that Q-learning outperforms GA in specific cases.

5.3 Placement Heuristics

For testing the different placement heuristics proposed in this paper the same problems are used. The pieces are arranged in order of their area, which is a commonly used heuristic. Using the GA algorithm for ordering could introduce variations due to the degree of randomness in GA training, and the specific placement heuristic used during training. Consequently, the overall efficiencies may be lower compared to using GA-based ordering. However, these results are still valuable for comparing different placement heuristics. The four placement heuristics that will be used are bottom-left, bottom and left, the border heuristic and the hybrid heuristic, which uses the border heuristic for the first part of the packing and the bottom-left heuristic for the final pieces. The results of these tests can be found in table5. These results show that there is no one size fits all placement heuristic. For different polygon packing problems different placement heuristics will perform the best. Considering the testing data-sets, the hybrid border and bottom-left heuristic provided on average an 2.44% improvement in fabric efficiency.

Table 5. Fabric efficiencies of different placement heuristics in combination with the area heuristic for ordering the pieces.

Dataset	bottom-left	bottom and left	border	hybrid
Albano	77.11	77.11	75.25	75.61
Dagli	73.17	71.71	72.45	73.59
Fu	67.86	67.86	75.00	75.00
Jakobs1	70.00	70.00	75.38	77.36
Mao	65.77	65.77	65.77	65.77
Marques	76.16	76.15	76.15	77.40

6 CONCLUSION

In conclusion, in this paper I have proposed two improvements for the sliding NFP generation algorithm, one of which more than doubles the generation speed. The other can have a positive effect on the speed but can also be slower. Furthermore a new ordering heuristic is proposed which in general achieves higher efficiencies compared to ordering by height or area. This new heuristic is then used as a seed for the GA and the Q-learning in combination with sorting by height and width. This seeding helped the GA and reinforcement learning algorithm to focus on using the time they have as efficiently as possible. Testing has shown that given these implementations of the GA and Q-learning, the GA outperforms Q-learning. Lastly, a new hybrid placement heuristic is proposed which starts of prioritising placement of pieces close to the borders after placing 60% of the pieces it switches to the well known bottom-left strategy. This placement strategy has improved the packing efficiency with a few percent, but at the scale at which clothing gets produced these few percent can have a large impact.

7 DISCUSSION

The NFP generation optimisation steps proposed in this study require further testing on edge cases, highlighting the need for additional research in scenarios where edge cases are prevalent. Furthermore, while the GA outperforms Q-learning in this study, Q-learning has shown superior performance on the dataset of Marques. Therefore, more research is required to fully explore the potential of Q-learning in optimising irregular polygon packing, leveraging advancements in deep Q-learning and other optimization techniques. Exploring transfer-learning solutions is also an interesting avenue, despite longer training times, as they offer near-instant determination of the packing order.

REFERENCES

[1] Michael Adamowicz and Antonio Albano. 1976. Nesting two-dimensional shapes in rectangular modules. *Computer-Aided Design* 8, 1 (Jan. 1976), 27–33. https://doi.org/10.1016/0010-4485(76)90006-3

[2] Pankaj K. Agarwal, Eyal Flato, and Dan Halperin. 2002. Polygon decomposition for efficient construction of Minkowski sums. *Computational Geometry* 21, 1 (Jan. 2002), 39–61. https://doi.org/10.1016/S0925-7721(01)00041-4

[3] Richard Carl Art Jr. 1966. *AN APPROACH TO THE TWO DIMENSIONAL, IRREGULAR. CUTTING STOCK PROBLEM.* Ph. D. Dissertation. MIT. https://dspace.mit.edu/bitstream/handle/1721.1/97782/25841973-MIT.pdf?sequence=1

[4] E. K. Burke, R. S. R. Hellier, G. Kendall, and G. Whitwell. 2007. Complete and robust no-fit polygon generation for the irregular stock cutting problem. *European Journal of Operational Research* 179, 1 (May 2007), 27–49. https://doi.org/10.1016/j.ejor.2006.03.011

[5] Sharon Cullinane and Kevin Cullinane. 2021. The Logistics of Online Clothing Returns in Sweden and How to Reduce its Environmental Impact. *Journal of Service Science and Management* 14 (Jan. 2021), 72–95. https://doi.org/10.4236/jssm.2021.141006

[6] Karen Daniels and Victor J. Milenkovic. 1996. Column-based strip packing using ordered and compliant containment. In *Applied Computational Geometry Towards Geometric Engineering (Lecture Notes in Computer Science)*, Ming C. Lin and Dinesh Manocha (Eds.). Springer, Berlin, Heidelberg, 91–107. https://doi.org/10.1007/BFb0014488

[7] D.G.K. Dissanayake. 2020. Does Mass Customization Enable Sustainability in the Fashion Industry? In *Fashion Industry - An Itinerary Between Feelings and Technology*, Riccardo Beltramo, Annalisa Romani, and Paolo Cantore (Eds.). IntechOpen. https://doi.org/10.5772/intechopen.88281

[8] D.G.K. Dissanayake and D. Weerasinghe. 2022. Towards Circular Economy in Fashion: Review of Strategies, Barriers and Enablers. *Circular Economy and Sustainability* 2, 1 (March 2022), 25–45. https://doi.org/10.1007/s43615-021-00090-5

[9] Kathryn A. Dowsland, Subodh Vaid, and William B. Dowsland. 2002. An algorithm for polygon placement using a bottom-left strategy. *European Journal of Operational Research* 141, 2 (Sept. 2002), 371–381. https://doi.org/10.1016/S0377-2217(02)00131-5

[10] Jie Fang, Yunqing Rao, Xusheng Zhao, and Bing Du. 2023. A Hybrid Reinforcement Learning Algorithm for 2D Irregular Packing Problems. *Mathematics* 11, 2 (Jan. 2023), 327. https://doi.org/10.3390/math11020327 Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[11] A. Miguel Gomes and José F. Oliveira. 2002. A 2-exchange heuristic for nesting problems. *European Journal of Operational Research* 141, 2 (Sept. 2002), 359–370. https://doi.org/10.1016/S0377-2217(02)00130-3

[12] Baosu Guo, Yu Zhang, Jingwen Hu, Jinrui Li, Fenghe Wu, Qingjin Peng, and Quan Zhang. 2022. Two-dimensional irregular packing problems: A review. *Frontiers in Mechanical Engineering* 8 (2022). https://www.frontiersin.org/articles/10.3389/fmech.2022.966691

[13] John H. Holland. 1984. Genetic Algorithms and Adaptation. In *Adaptive Control of Ill-Defined Systems*, Oliver G. Selfridge, Edwina L. Rissland, and Michael A. Arbib (Eds.). Springer US, Boston, MA, 317–333. https://doi.org/10.1007/978-1-4684-8941-5_21

[14] Bonfim A. Junior, Plácido R. Pinheiro, and Rommel D. Saraiva. 2013. A Hybrid Methodology for Tackling the Irregular Strip Packing Problem*. *IFAC Proceedings Volumes* 46, 7 (May 2013), 396–401. https://doi.org/10.3182/20130522-3-BR-4036.00041

[15] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. 2021. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications* 80, 5 (Feb. 2021), 8091–8126. https://doi.org/10.1007/s11042-020-10139-6

[16] Taime Koe. 2020. Can Technology Eliminate Fashion's $500B Overproduction Problem? – Fashion Mannuscript. https://www.mannpublications.com/fashionmannuscript/2020/11/06/can-technology-eliminate-fashions-500b-overproduction-problem/

[17] A.A.S. Leao, F.M.B. Toledo, J.F. Oliveira, M.A. Carravilla, and R. Alvarez-Valdés. 2020. Irregular packing problems: A review of mathematical models. *European Journal of Operational Research* 282, 3 (2020), 803–822. https://doi.org/10.1016/j.ejor.2019.04.045

[18] Yuxi Li. 2018. Deep Reinforcement Learning: An Overview. http://arxiv.org/abs/1701.07274 arXiv:1701.07274 [cs] version: 6.

[19] Dequan Liu and Hongfei Teng. 1999. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles1This work is supported by the National Natural Science Foundation of China (Grant No. 69573004).1. *European Journal of Operational Research* 112, 2 (Jan. 1999), 413–420. https://doi.org/10.1016/S0377-2217(97)00437-2

[20] Qiang Luo and Yunqing Rao. 2022. Improved Sliding Algorithm for Generating No-Fit Polygon in the 2D Irregular Packing Problem. *Mathematics* 10, 16 (Jan. 2022), 2941. https://doi.org/10.3390/math10162941 Number: 16 Publisher: Multidisciplinary Digital Publishing Institute.

[21] Qiang Luo, Yunqing Rao, Xiaoqiang Guo, and Bing Du. 2022. A biased genetic algorithm hybridized with VNS for the two-dimensional knapsack packing problem with defects. *Applied Soft Computing* 118 (March 2022), 108479. https://doi.org/10.1016/j.asoc.2022.108479

[22] Anantharam Mahadevan. 1984. *OPTIMIZATION IN COMPUTER-AIDED PATTERN PACKING (MARKING, ENVELOPES).* Ph. D. Dissertation. North Carolina State University. https://www.proquest.com/openview/0d510654f5a3e9a881247029b6a97999/1?pq-origsite=gscholar&cbl=18750&diss=y

[23] Victor J. Milenkovic. 1998. Rotational polygon overlap minimization and compaction. *Computational Geometry* 10, 4 (July 1998), 305–318. https://doi.org/10.1016/S0925-7721(98)00012-1

[24] Sudeshna Mukherjee. 2015. Environmental and Social Impact of Fashion: Towards an Eco-friendly, Ethical Fashion. *International Journal of Interdisciplinary and Multidisciplinary Studies* 2, 3 (2015), 22–35.

[25] E. Napier and F. Sanguineti. 2018. Fashion merchandisers' slash and burn dilemma: A consequence of over production and excessive waste? *Rutgers Business Review* 3, 2 (2018), 159–174.

[26] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. http://arxiv.org/abs/1912.06680 arXiv:1912.06680 [cs, stat].

[27] Sohel Rana, Subramani Pichandi, Shabaridharan Moorthy, Amitava Bhattacharyya, Shama Parveen, and Raul Fangueiro. 2015. Carbon Footprint of Textile and Clothing Products. 141–166. https://doi.org/10.1201/b18428-10

[28] P Smith. 2021. European fast fashion companies ranked by units sold 2019. https://www.statista.com/statistics/1094257/european-fast-fashion-brands-ranked-by-units-sold/

[29] Yu-Chung Tsao, Magda Delicia, and Thuy-Linh Vu. 2022. Marker planning problem in the apparel industry: Hybrid PSO-based heuristics. *Applied Soft Computing* 123 (July 2022), 108928. https://doi.org/10.1016/j.asoc.2022.108928

[30] Yu-Chung Tsao, Chia-Hsin Hung, and Thuy-Linh Vu. 2021. Hybrid Heuristics for Marker Planning in the Apparel Industry. *Arabian Journal for Science and Engineering* 46, 10 (Oct. 2021), 10077–10096. https://doi.org/10.1007/s13369-020-05210-1

[31] Shunji Umetani and Shohei Murakami. 2022. Coordinate descent heuristics for the irregular strip packing problem of rasterized shapes. *European Journal of Operational Research* 303, 3 (Dec. 2022), 1009–1026. https://doi.org/10.1016/j.ejor.2022.03.034

[32] Christopher Watkins. 1989. Learning From Delayed Rewards. (Jan. 1989).