IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 25, 2020, accepted May 12, 2020, date of publication May 14, 2020, date of current version May 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2994635

Greedy Adaptive Search: A New Approach for Large-Scale Irregular Packing Problems in the Fabric Industry

XIAOYIN HU[1,2], **JIANSHU LI**[1,2], **AND JINCHUAN CUI**[1]
[1]Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China
[2]School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

Corresponding author: Xiaoyin Hu (hxy@amss.ac.cn)

This work was supported by the Key Laboratory of Management, Decision and Information Systems, CAS.

ABSTRACT The 2-dimensional irregular packing problems are important in the fabric industry. Under several restrictions, fabric packing problems require placing a given set of parts within a fixed-width rectangular sheet, aiming at a minimum length use. In textile industry production, the fabric packing problems are usually large-scale with time limits, where the total number of parts is large, and a high-utilization solution should be computed in several minutes. However, there are few existing works on large-scale packing problems. In this paper, we propose a greedy adaptive search algorithm by constructing a new evaluation function and introducing a new restricted local search strategy. In our algorithm, with a given initial sequence of parts, we iteratively search the best-fit part in succeeding several parts and place it on sheet. Moreover, we employ a two-stage heuristic searching algorithm to search over all the possible sequences for a good initial sequence with high utilization. Numerical examples involve some large-scale industrial instances, together with some large-scale instances generated from benchmarks. Numerical tests show that our algorithm outperforms existing state-of-the-art solvers in large-scale packing problems. The results show the potential of our algorithm to large-scale packing problems in industrial production.

INDEX TERMS Evaluation function, fabric, no-fit polygon, packing, restricted local search.

I. INTRODUCTION

In the fabric industry, solving the 2-dimensional *irregular packing problems* is a critical operation in the cutting process. The main purpose of these problems is to place a given set of 2-dimensional parts within a rectangular sheet with a fixed width and find the solution with the best utilization of the sheet while no part overlaps with others. The utilization of the packing results is important both economically and environmentally. In fabric cutting problems, the total number of parts is large, while the time limit is strict. Normally, a solution should be generated within several minutes. In this paper, we focus on constructing an efficient algorithm for solving large-scale fabric packing problems. In general, large-scale fabric cutting problems have the following properties:

- The contour of the part is not necessarily convex with a large number of vertices;
- Requires the minimum distance among parts;
- Each part can only be rotated by a finite set of angles, including $\{0°, 90°, 180°, 270°\}$;
- There may be some flaws in the fabric;
- The number of parts is large.

For irregular packing problems, a few programming-based approaches are proposed. These approaches compute a solution by solving certain mathematical models for the packing problems. These proposed models include the mixed-integer programming (MIP) model with a linear objective function and mixed-integer constraints [2], [15], [22], [32], [40], [42], nonlinear programming models with a nonlinear objective function [14], [15], [29], [30], [44], and constraint-based programming models where the packing problem is described by constraints [12], [38], [39]. Interested readers are referred to the references in the survey [33]. However, since the irregular packing problem is NP-hard [23], it might not be possible to compute an optimal solution within the time limit. For example, computing the optimal solution for 5

The associate editor coordinating the review of this manuscript and approving it for publication was Yilun Shang.

parts can take more than 6 hours [33]. Moreover, since these exact mathematical models usually involve a considerable number of variables and complex constraints [31], [33], generating a high- utilization solution with commercial solvers is difficult to achieve, especially for large-scale packing problems.

As illustrated in [21], the use of meta-heuristics and hybrid algorithms as powerful optimization tools have been applied to this problem. Sato *et al.* [41] summarized that these approaches can be roughly divided into two categories: searching over the sequence [5], [6], [9], [13], [16], [26], [35], [45] and searching over the layout [21], [29], [41]. The main difference resides in the techniques for generating the final solution. In searching over the sequence approach, the final solution is represented by a sequence of parts. However, in the latter approach, the position of each item is directly represented by their positions [7].

This difference directly impacts the searching strategy. In searching over the sequence approach, the parts are placed sequentially into the sheet without overlapping. As a result, this approach applies heuristics algorithms searching for a sequence with high utilization in the following steps:

1) Determine a sequence of the placement for parts. This can be performed randomly or by sorting the parts according to some measure, e.g., area or perimeter of polygons [35];
2) Place the parts with some evaluation functions. Typically, a part is placed at the contour of the stencils already placed. Some algorithms also allow the hole-filling strategy, i.e., the part is placed in the holes formed by previously placed parts [19], [24].

The most popular placement rule is the bottom-left policy [41]. In addition, Bennell and Song [8] proposed several attributes considering mutual fitness under the bottom-left policy. With a specific evaluation function, the challenge is to determine the sequence of parts. Pinheiro *et al.* [36] adopted a random-key genetic algorithm to simultaneously determine the sequence and rotation. Moreover, Burke *et al.* [10] modified the greedy bottom-left layout construction by discretizing the horizontal search and applied a hill-climbing tabu search. However, the bottom-left strategy usually results in solutions with low utilization, and the beam search strategy proposed by Bennell and Song [8] lacks efficiency, especially for large-scale packing problems.

Searching over the layout approach searches for a solution based on an initial solution, which is usually generated by searching over the sequence approach. Starting from the initial solution, the goal is to minimize the overlap within a sheet with fixed length and height. This approach generates a feasible solution if and only if the corresponding overlap vanishes. As a result, in this approach, parts move freely, and a separation method is usually employed to minimize overlap. The no-fit polygon is used to determine the mutual overlap among the parts. This model was adopted by Bennell and Dowsland [6] to generate valid layouts using a tabu search heuristic. Gomes and Oliveira [25] hybridized the compaction and separation algorithms with a simulated annealing algorithm.

In large-scale cases, the searching over the layout approaches has a large continuous search space, which becomes an obstacle for constructing efficient solvers [41]. Therefore, it is important to design an efficient searching over the sequence algorithm to generate a high-utilization solution in time limits. In addition, an efficient searching over the sequence algorithm can provide a good initial solution for searching over the layout approaches.

For large-scale packing problems, the correlated work is limited. Most existing algorithms are tested on specific small-scale or middle-scale cases, where the parts have simple contours. The numerical performance of existing algorithms on large-scale fabric packing problems is unknown.

In this paper, based on searching over the sequence approach, we propose a greedy adaptive search (GAS) method for large-scale fabric packing problems. We construct an evaluation function, where the weights are dynamically adjusted to achieve high fitness among parts while retaining the robustness of the algorithm. To reduce the search space and increase the one-pass utilization, inspired by the work of Bennell and Song [8], we propose a greedy searching technique where we search for the best-fit part in the constitutive α parts and then place it on the sheet. The primarily numerical experiments demonstrate that our algorithm outperforms the existing open-source solver for large-scale packing problems. In addition, on large-scale fabric packing problems constructed from the ESICUP dataset, our algorithm takes less CPU time and has comparable utilization to some existing state-of-the-art algorithms, which is usually based on searching over the layout approach.

The rest of this paper is organized as follows. We put all the preliminaries, including the preprocessing and computing no-fit polygon, in Section 2. In Section 3, we present the detailed algorithm, and when a higher utilization rate is required, we propose a two-stage heuristic algorithm. Numerical experiments are reported in Section 4. In the last section, we draw a brief conclusion.

II. PROBLEM DEFINITION

This section gives a specific problem definition of the irregular packing problem in the fabric industry. We have a list of parts $P = (P_1, P_2, \ldots, P_n)$, a list of their allowable orientations $O = (O_1, O_2, \ldots, O_n)$ and their reference points $\{r_1, \cdots, r_n\}$. The sheet is a rectangular sheet C(W, L) with fixed-width W and arbitrary length L, and our goal is to minimize L. There are some flaws $F = (F_1, F_2, \ldots, F_m)$ on the sheet, let $d(A, B)$ denote the minimum required spacing between A and B, and $\partial C(W, L)$ denotes the boundary of the container $C(W, L)$, where $d(P_i, P_j) = d_1, d(P_i, F_k) = d_2, d(P_i, \partial C(W, L)) = d_3, \forall i, j, k$.

We denote polygon $P_i \in P$ rotated by $\theta_i \in O_i$ as $P_i^{\theta_i}, P_i^{\theta_i} := \{(\hat{u}\cos\theta_i + \hat{v}\sin\theta_i, -\hat{u}\sin\theta_i + \hat{v}\cos\theta_i) | (\hat{u}, \hat{v}) \in P_i\}$, which may be written as P_i for simplicity when the orientation is $0°$. We describe translations of polygons by Minkowski sums.

Let $x_i = (x_{i1}, x_{i2})$ be the translations of polygons, and w_i be the new position of their reference points, and we have $x_i = w_i - r_i^{\theta_i}$. Thus, the polygon placed at x_i and rotated by θ_i can be represented as $P_i^{x_i,\theta_i} := P_i^{\theta_i} \oplus x_i = \{p+x_i | p \in P_i^{\theta_i}\}$. A solution to this problem is described by a set of translation vectors $(x_1, x_2, \ldots, x_n)$ and a set of orientations $(o_1, o_2, \ldots, o_n)$. Let L^* be the length of the sheet used for nesting, which is a decision variable to be minimized. Therefore, we define the problem in the fabric industry as follows:

$$\begin{aligned}
&\text{minimize } L^* \\
&\text{subject to } L^* = \max_{1\le i\le n} (u_1 | (u_1, u_2) \in P_i^{\theta_i} \oplus x_i), \\
&\qquad d(P_i^{\theta_i} \oplus x_i, P_j^{\theta_j} \oplus x_j) \ge d_1, \quad 1 \le i \le j \le n \\
&\qquad d(P_i^{\theta_i} \oplus x_i, \partial C(W, L)) \ge d_3, \quad 1 \le i \le n \\
&\qquad d(P_i^{\theta_i} \oplus x_i, F_k) \ge d_2, \quad 1 \le i \le n, 1 \le k \le m \\
&\qquad P_i^{\theta_i} \oplus x_i \subseteq C(W, L), \quad 1 \le i \le n \\
&\qquad x_i = w_i - r_i^{\theta_i}, \quad 1 \le i \le n \\
&\qquad \theta_i \in O_i, \quad 1 \le i \le n \\
&\qquad r_i \in \mathbb{R}^2, \quad 1 \le i \le n \\
&\qquad L^* \in \mathbb{R}^+,
\end{aligned} \tag{1}$$

As described in our introduction, this programming is difficult to solve due to its discreteness and nonconvexity, especially in the large-scale case. To efficiently solve the packing problem, we propose a novel algorithm based on a greedy strategy with adaptively adjusted parameters. The details of our proposed algorithm are presented in Section IV.

III. PREPROCESSING WITH GEOMETRY

A. SIMPLIFY THE PARTS

In this problem, the number of vertices of the parts can be very large, which leads to complex geometrical computation and results in numerical inefficiency. Thus, it is important for us to reduce the vertices by simplifying the boundary of the parts, for which we introduce two efficient approaches.

1) RAMER-DOUGLAS-PEUCKER ALGORITHM

Given a curve composed of line segments, Ramer-Douglas-Peucker algorithm [18], [37] is a popular method for finding a similar curve with fewer points. With a maximum simplified error δ, this algorithm first selects two ends, A and B, from the curve, and then finds the point C, which is the farthest from line segment AB. If the distance from C to line segment AB is less than δ, we simplify the curve AB as line AB and remove all the other points on curve AB; otherwise, this algorithm recursively calls itself to simplify the curve between A and C, as well as the curve between B and C, respectively. In brief, the Ramer-Douglas-Peucker algorithm uses line segments to approximate some successive vertices.

However, such approximation leads to a simplified boundary of our parts, resulting in errors in calculating their mutual distance. These errors can result in violations in the constraint on the distance among parts. To avoid such a violation,

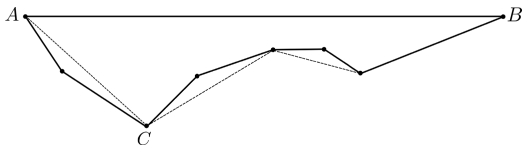


FIGURE 1. Ramer-Douglas-Peucker algorithm.

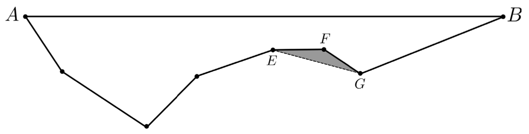


FIGURE 2. Clean the concave based on area.

we need to expand the parts, which causes the loss of the overall utilization of the generated results. Hence, we need to perform a more detailed simplification of the parts with fewer vertices.

2) CLEAN THE CONCAVE BASED ON AREA

The main purpose of this algorithm is to simplify those vertices that form a concave boundary of the parts, which is called a locally tiny concave structure in the following paragraph. Here, the locally tiny concave structure denotes three consecutive vertices E, F, G where the area of triangular EFG is less than δ_s, and if we draw a line between E and G, F is in the interior of the newly formed contour. Fig. 2 provides an illustrative example for the so-called locally tiny concave structure.

With a fixed tolerance δ_s, in each iteration, our algorithm searches for a locally tiny concave structure among all vertices of the given contour. If we find such vertices E, F, G, we remove the vertex F, edges EF and FG, then add a new edge EG to the contour. The algorithm stops when there is no locally tiny concave structure in the simplified contour.

B. PREPROCESSING WITH THE MINKOWSKI SUM

1) GENERATING NO-FIT POLYGON (NFP)

The no-fit polygon (NFP) and inner-fit polygon (IFP) were first proposed by Art Jr [3] and applied in detecting the overlap between two parts. The NFP/IFP is a polygon that defines the legal placement relationship of one part to another part or sheet. More precisely, each part has a reference point, which can be any point inside or outside the part. Given two parts P_i and P_j, the NFP of parts P_i and P_j is the set of points where if the reference point of part P_j is placed, then the two parts overlap. In the rest of the paper, we denote the no-fit polygon between two parts, P_i and P_j by NFP$_{ij}$. The boundary of NFP$_{ij}$ and its exterior are the feasible regions where part P_j can be placed so as not to overlap part P_i. Similar to NFP, with a selected reference point for any part P_i, its IFP denotes the set of points where if the reference point is placed, part P_i is placed inside the sheet. In the rest of our paper, we denote the inner-fit polygon of the part i by IFP$_i$.

To compute the NFP and IFP for all parts, several approaches have been proposed in the literature, such as the sliding method [11], the Minkowski sum [7], a combination

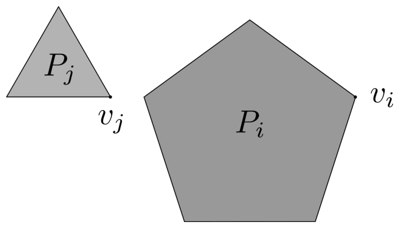


FIGURE 3. Polygon P_i and P_j.

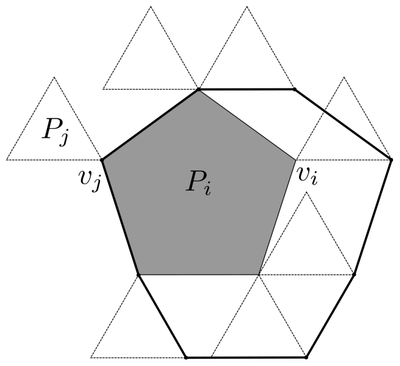


FIGURE 4. Construction of NFP_{ij}.

of region splitting and the Minkowski sum approach [1]. Assume parts P_i and P_j have e_i and e_j edges, respectively, then the computational complexity of computing NFP_{ij} is up to $\mathcal{O}(e_i^2 e_j^2)$ [4], [20].

In this paper, we apply the Minkowski sum to generate pairwise NFP among parts. The mathematical formulation of the Minkowski sum for generating NFP of parts P_i and P_j with reference point r_i, r_j can be defined as follows:

$$\text{NFP}(P_i, P_j) = \{r_i + v_i - v_j | v_i \in P_i, v_j \in P_j\}. \tag{2}$$

The no-fit polygon has the following properties:

- P_i and P_j overlap if and only if $r_j \in \text{NFP}(P_i, P_j)$;
- P_i and P_j touches if and only if $r_j \in \partial\text{NFP}(P_i, P_j)$;
- If $P_k = -P_i$, $P_l = -P_j$, then $\text{NFP}(P_k, P_l) = -\text{NFP}(P_i, P_j) + r_i - r_k$.

Similarly, with sheet $C(W, L)$, the IFP of part P_i can be expressed as

$$\text{IFP}(P_i) = \bigcup_v \{v - r_i + P_i \subseteq C(W, L)\}. \tag{3}$$

2) OFFSET TO KEEP SPACING

As described in the introduction, a minimum distance among any parts, denoted as d_1, is required in irregular fabric shape packing problems. To guarantee that the minimum distance among the parts is no less than d_1, we choose to dilate the contour of each part. Moreover, when these dilated parts do not intersect, the minimum distance of all placed parts is no less than d_1. For part i, to compute its enlarged parts $P_i^{x_i,\theta_i}$, we enlarge its original contour $P_i^{x_i,\theta_i}$ by computing its Minkowski sum with a circle:

$$\tilde{P}_i^{x_i,\theta_i} = \{v + w | v \in P_i^{x_i,\theta_i}, w \in B_{\frac{d_1}{2}}\}. \tag{4}$$

where $B_{\frac{d_1}{2}}$ denotes the circle with radius $\frac{d_1}{2}$. Based on the enlarged parts, the constraint $d(P_i^{\theta_i} \oplus x_i, P_j^{\theta_j} \oplus x_j) \geq d_1$ in (1) is equivalent to $\tilde{P}_i^{x_i,\theta_i}$ not intersecting $\tilde{P}_j^{x_j,\theta_j}$. In addition, in the packing problem, a minimum distance between parts and the boundary of the sheet is also required. In this case, we shrink the boundary of the sheet by $d_2 - \frac{d_1}{2}$ and denote the shrunk sheet as $\tilde{C}(W, L)$, i.e.,

$$\tilde{C}(W, L) = \bigcap_{w \in B_{d_3 - \frac{d_1}{2}}} C(W, L) - w.$$

With such notations, the second constraint in (1) can be reformulated as $\tilde{P}_i^{x_i,\theta_i} \subset \tilde{C}(W, L)$ for $1 \leq i \leq n$. Additionally, the minimum distance between the parts and flaws should be larger than d_3. Similarly, we can enlarge the boundary contour of the flaws by $d_3 - \frac{d_1}{2}$ and denote them as $\tilde{F}_i$. More precisely,

$$\tilde{F} = \{v + w | v \in F, w \in B_{d_2 - \frac{d_1}{2}}\}.$$

Thus, the third constraint in (1) can be reshaped as the non-overlapping between $\tilde{F}_i$ and $\tilde{P}_j^{x_j,\theta_j}$.

IV. ONE-PASS PACKING ALGORITHM

Let Ω contain the indexes of all placed parts; then, the legal movement for part i is

$$W(\tilde{P}_i^{x_i,\theta_i}) := IFP(\tilde{P}_i^{x_i,\theta_i}) \setminus \bigcup_{j \in \Omega} NFP(\tilde{P}_j^{x_j,\theta_j}, \tilde{P}_i^{x_i,\theta_i}), \tag{5}$$

where

$$IFP(\tilde{P}_i^{x_i,\theta_i}) := \bigcap_{v \in \tilde{C}(W,L) \setminus \tilde{F}} \left\{v + r_i^{\theta_i} - \tilde{P}_i^{x_i,\theta_i}\right\}, \tag{6}$$

denotes the inner-fit polygon of $\tilde{P}_i^{x_i,\theta_i}$ for sheet $\tilde{C}(W, L)$ with flaws $\tilde{F}$.

The heuristic algorithm TOPOS-"***T***écnicas de ***O***ptimizacão para o ***Pos***icionamento de Figuras Irregulares" proposed by Oliveira *et al.* [35] has two essential concepts that differ from the conventional bottom-left approach: how to select the next part and how to place the selected part. To select the next part, as well as its position and orientation, Oliveira *et al.* [35] defined two heuristics strategy: local search and initial sort. Their TOPOS algorithm proposed evaluation criteria to evaluate the score for all unplaced parts with any possible placement and rotations. Then, the TOPOS algorithm performs the local search for all unplaced parts, where it selects a part with the smallest score and chooses the position and rotation corresponding to the smallest score. The framework of the TOPOS algorithm can be stated as follows:

1) Compute the score for all unplaced parts with all possible positions and rotations;
2) Choose the part as well as the corresponding position and rotation with the smallest score, and place it on the sheet;
3) Return to step 1 until all parts are placed.

In each iteration of the TOPOS algorithm, the part to be placed is selected from all the unplaced parts, which leads to

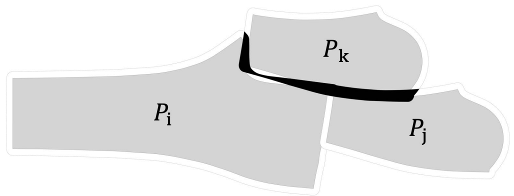


FIGURE 5. The overlap among parts.

expensive geometric computations in computing the score of all unplaced parts, especially in large-scale cases. In addition, although TOPOS selects the best-fit part from all unplaced parts, as discussed in Section IV-B, the pursuit of a local optimum may lead to a low-utilization solution.

Based on the searching-over-sequence approaches and TOPOS algorithm, we propose an algorithm for large-scale fabric packing problems, where the solution is generated by selecting a new part and adding to the partial solution of already placed parts. In our algorithm, we propose several improved evaluation criteria to evaluate the score for all unplaced parts. In addition, to balance the local optimum and global optimum, we restrict the depth of the greedy searching and propose an improved local search strategy that adopts the previous information to adjust the current score. Moreover, inspired by the initial sorting strategy in [8], we propose a new choice for the initial sorting in our algorithm.

A. EVALUATION FUNCTION

In this subsection, we present a detailed evaluation criterion to evaluate the score of part $\tilde{P}_i^{x_i,\theta_i}$, which is denoted as $s(\tilde{P}_i^{x_i,\theta_i})$ in the rest of our paper. As described in [8], [35], we select the following attributes as the basis of our criteria for determining the next piece and its position: length of placed parts L_0 and area of overlap between certain enclosures S_0. When we select part P_i and place it to u_i with rotation angle θ_i, the length of all placed parts can be represented as

$$L_1(\tilde{P}_i^{x_i,\theta_i}) = \max_{(u,v)\in\bigcup_{j\in\Omega\cup i}(\tilde{P}_j^{x_j,\theta_j})} \{u\}. \tag{7}$$

In addition, we denote the dilation enclosure of $\tilde{P}_i^{x_i,\theta_i}$ as $Q_i^{x_i\theta_i}$, i.e.,

$$Q_i^{x_i,\theta_i} = \{u+v | u \in \tilde{P}_i^{x_i,\theta_i}, v \in \mathcal{B}_w\}. \tag{8}$$

Therefore, the overlap regions for any selected part P_k with translation x_k and rotation θ_k can be defined as:

$$S_0(\tilde{P}_i^{x_i,\theta_i}) = \text{Area}\left(\{Q_k^{x_k,\theta_k}\} \bigcap \left(\bigcup_{i\in\Omega} Q_i^{x_i,\theta_i}\right)\right). \tag{9}$$

Fig. 5 illustrates the overlap among parts P_i, P_j and P_k. Parts P_i, P_j are already placed into the sheet, and part P_k is selected. In Fig. 5, the grey regions denote the origin contour for all three parts, while the regions enclosed in the dashed line denote their dilation. In addition, the black region denotes their overlapping regions S_0.

The first criterion we select is the length of all placed parts, proposed by Bennell and Song [35] and Bennell and Song *et al.* [8] to reduce the total length of the used sheet, which is equivalent to improving the utilization of the final result. However, directly using L_0 in the evaluation will cause an imbalance among different parts. For example, if an absolute measure of the length is used, the algorithm will choose to pack the long pieces at its final stage and undermine the global utilization of the final solution. Hence, we must define attributes as a relative measure of the piece length or area. To balance the magnitude of these measures for all pieces, we modify the measure of the part $\tilde{P}_i^{x_i,\theta_i}$ with length l_i in the first criteria as

$$L_1(\tilde{P}_i^{x_i,\theta_i}) = \frac{L_0(\tilde{P}_i^{x_i,\theta_i})}{l_i}. \tag{10}$$

In addition, the area of overlap between enclosures S_0 is an essential factor in the evaluation function to force the parts placed close to each other and thus improve the total utilization. However, if we directly maximize the total overlap area between parts to provide an incentive for the highest area of overlap, the algorithm prefers to choose large parts and place them first. To balance the choice among parts, we modify the measure for overlap by introducing the perimeter of the selected part as a penalty. More precisely, for part $\tilde{P}_i^{x_i,\theta_i}$ with perimeter q_i, the measure is

$$S_1(\tilde{P}_i^{x_i,\theta_i}) = \frac{S_0(\tilde{P}_i^{x_i,\theta_i})}{q_i}. \tag{11}$$

Based on the two attributes above, part P_i is placed on the sheet with position x_i and rotation θ_i at the k-th round scored

$$s(\tilde{P}_i^{x_i,\theta_i}) = L_1(\tilde{P}_i^{x_i,\theta_i}) - \gamma_2 S_1(\tilde{P}_i^{x_i,\theta_i}). \tag{12}$$

It is worth mentioning that the importance of these two attributes is different at each stage of the algorithm. At the beginning, only a few parts are placed on the sheet. Hence, placing the selected parts to the position that fits well is more important than minimizing the total length. As a result, in our algorithm, we first estimate a utilization rate γ_1 for our solution and its corresponding length $\tilde{L} = \frac{\sum_{i=1}^{n} Area(P_i)}{\gamma_1 W}$, then compute the score by

$$s(\tilde{P}_i^{x_i,\theta_i}) = \begin{cases} 0.01 \cdot L_1(\tilde{P}_i^{x_i,\theta_i}) - \gamma_2 S_1(\tilde{P}_i^{x_i,\theta_i}), & L_0(\tilde{P}_i^{x_i,\theta_i}) \le \tilde{L}; \\ L_1(\tilde{P}_i^{x_i,\theta_i}) - \gamma_2 S_1(\tilde{P}_i^{x_i,\theta_i}), & L_0(\tilde{P}_i^{x_i,\theta_i}) > \tilde{L}. \end{cases} \tag{13}$$

B. RESTRICTED LOCAL SEARCH WITH DEPTH

In the TOPOS algorithm, the local search heuristic does not pack the parts following a predetermined order. Instead, the next part is selected from all available unpacked parts greedily according to the given evaluation criteria [8], [35]. However, for large-scale packing problems with hundreds of parts, recurrently evaluating all unplaced parts is costly. Therefore, in our algorithm, we only search for the best

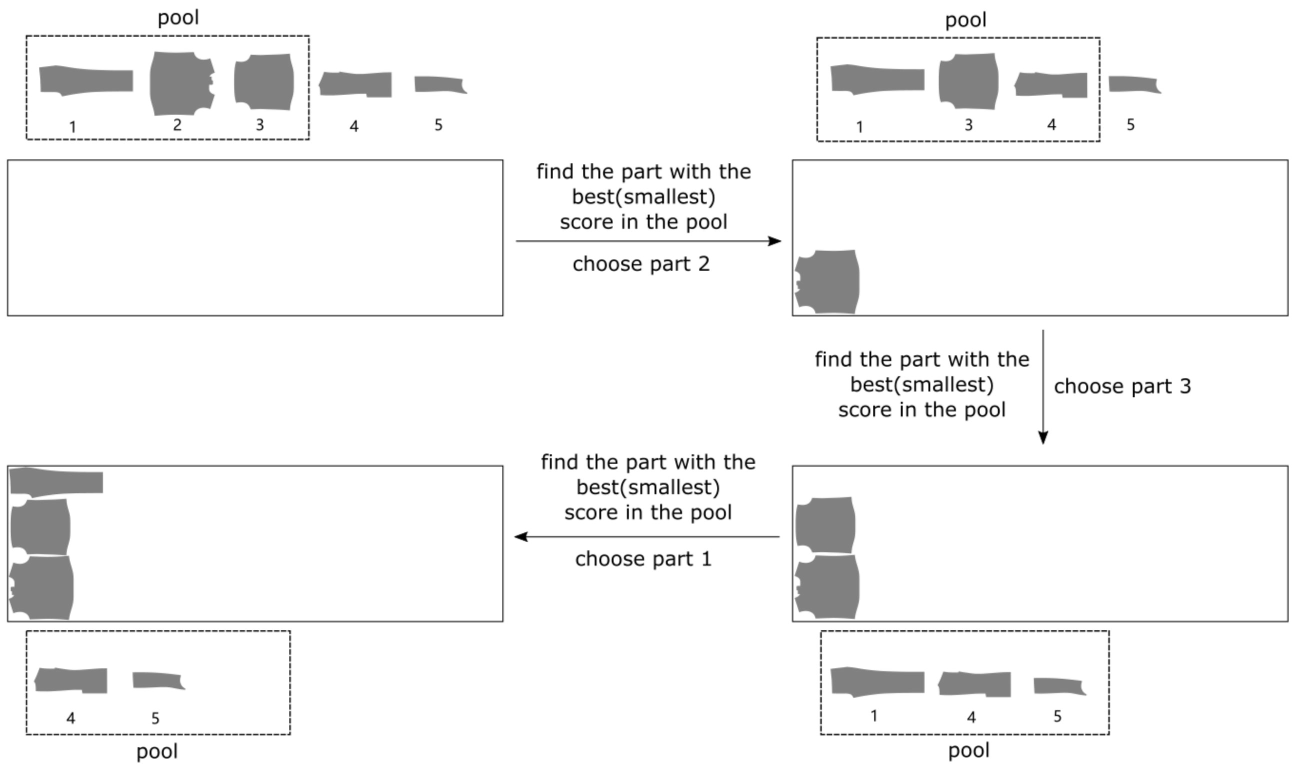

pool
1
2
3
4
5
find the part with the best(smallest) score in the pool
choose part 2
pool
1
3
4
5
find the part with the best(smallest) score in the pool
choose part 3
find the part with the best(smallest) score in the pool
choose part 1
4
5
pool
1
4
5
pool

FIGURE 6. **Example of restricted local search with depth, where depth=3.**

part in the consecutive α parts, where α is the searching depth. We call this searching procedure searching in the pool for simplicity in the rest of the paper. This behavior of the algorithm is displayed in the example in Fig. 6. As illustrated in our numerical experiments, limiting the searching depth is efficient in computing the placement. In most cases, this strategy can generate better results when compared with the classical TOPOS algorithm. In this paper, we denote $\text{Score}(P_i, k)$ as the smallest score for part P_i at the k-th round. More precisely, let $W_{i,k}$ denote the possible movement for part P_i at the k-th round, and $\text{Score}(P_i, k)$ has the following expression:

$$\text{Score}(P_i, k) = \min_{\theta_i \in O_i, x_i \in W_{i,k}} s(\tilde{P}_i^{x_i,\theta_i}). \tag{14}$$

When P_i is not in the pool at the k-th round, we set $\text{Score}(P_i, k) = +\infty$. Since all possible movements are infinite, in our algorithms, we choose $W_{i,k}$ as all the vertices of $W(\tilde{P}_i^{x_i,\theta_i})$.

Although selecting the best-fit part in our algorithm results in a partial solution with high utilization in the early stage, there may still exist some irregular parts that do not fit well with others. In the TOPOS algorithm, these irregular parts are never selected until all others are placed. In this case, the TOPOS algorithm generates a poor placement in its final stage, which results in poor utilization.

To solve this problem, we introduce two factors to adjust the final score and select the part with the smallest score.

Here are two rules to balance the preference for all unplaced parts:

1) The longer part P_i is in the pool, the more likely it will be selected in the next evaluation;
2) When the score of part P_i is considerably smaller than its largest score, it is likely to be selected in this evaluation.

For those irregular parts, these two rules guarantee that they will not be backlogged until the final stage of our algorithm. If a part is kept in the pool for a long time, it must be difficult for this part to fit well with the placed parts, so we add a penalty to its score and force a choice in the next few iterations. In this paper, $t(P_i, k)$ represents the number of rounds that P_i remains in the pool at the k-th iteration.

In addition, when an irregular part makes considerable progress in its score, we consider that this part has found a relatively suited position at this moment. Thus, this part is preferred in the current selection. This factor is defined as:

$$c_4 := \text{Score}(P_k, k) - \max_{j \le k, \text{Score}(P_i, j) \neq +\infty} \text{Score}(P_i, j). \tag{15}$$

Then, we obtain the final evaluation score in (16).

$$\text{Finalscore}(P_i, k) = \text{Score}(P_i, k) - \gamma_3 t(P_i, k) + \gamma_4 c_4. \tag{16}$$

Then, based on FinalScore, when given parameters $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ and depth α, we compute the final solution by

Algorithm 1. We also provide a flowchart in Fig. 7 to describe the algorithm vividly. It is worth mentioning that the best-fit part corresponds to the smallest score.

Algorithm 1 Greedy Adaptive Search (GAS)

```
Input: Predefined sequence $j_1, ..., j_n$, parameters $\gamma_1, \gamma_2, \gamma_3, \gamma_4$, depth $\alpha$
Output: Final utilization rate and packing results
1:  Initialization: $k = 0$, $\Omega = \emptyset$, Pool $= \{j_{k+1}, ..., j_{k+\alpha}\}$;
2:  while $k \le n$ do
3:      Computes Score$(P_i, k)$ for $i \in$ Pool;
4:      Choose $i_k^* := \arg\min_{i \in \text{Pool}} \text{Score}(P_i, k)$ and its corresponding best position $x_{i_k}^*$ and rotation $\theta_{i_k}^*$:
            $$x_{i_k}^*, \theta_{i_k}^* = \arg\min_{\theta_{i_k} \in O_{i_k}, w \in W(\tilde{P}_{i_k}^{x_{i_k}, \theta_{i_k}})} s(\tilde{P}_{i_k}^{x_{i_k}, \theta_{i_k}});$$
5:      Place part $P_{i_k}$ with position $x_{i_k}^*$ and rotation $\theta_{i_k}^*$, $\Omega = \Omega \cup \{i_k^*\}$;
6:      if $k + \alpha + 1 \le n$ then
7:          Pool $=$ (Pool $\cup \{j_{k+\alpha+1}\}) \setminus \{i_k^*\}$;
8:      end if
9:      $k++$;
10: end while
```

C. INITIAL SEQUENCE

Our algorithm starts from a predefined sequence and searches for a solution with high utilization. To enhance the performance of the local search, we select the predefined sequence in descending order according to the following rules:

- area of the polygon.
- perimeter of the polygon.
- area of the smallest rectangle containing the polygon.
- area of polygon $\cdot$ (1 $-$ irregularity), where irregularity is calculated from the area of the polygon divided by the area of the smallest rectangle containing the polygon.

D. TWO-STAGE HEURISTIC

In our algorithm, if we wish to compute a solution with higher utilization, we can search over the parameters $\gamma_1^k, \gamma_2^k, \gamma_3^k, \gamma_4^k, \alpha^k$ as well as the sequences. However, since the number of all possible sequences equals $n!$, directly searching over the sequence is costly and lacks efficiency. As described in Algorithm 1, in each iteration, our algorithm chooses the best-fit part in the consecutive α parts, which can be regarded as a modification to the predefined sequence. Consequently, adjusting parameters results in a different evaluation function, leading to a different inserting sequence for the parts. Since our algorithm only involves 5 parameters, searching for a better parameter in such a low-dimensional space is more efficient than searching over all possible predefined sequences. As a result, we propose a two-stage searching strategy, where we first search for good parameters and then search over the predefined sequences for better utilization with fixed parameters.

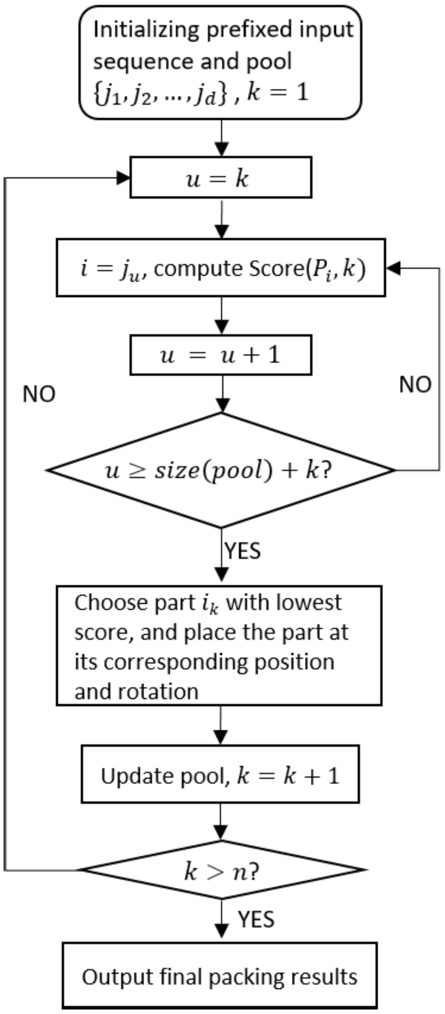

Initializing prefixed input sequence and pool $\{j_1, j_2, ..., j_d\}$, $k = 1$
$u = k$
$i = j_u$, compute Score(P_i, k)
$u = u + 1$
NO
NO
$u \ge size(pool) + k?$
YES
Choose part i_k with lowest score, and place the part at its corresponding position and rotation
Update pool, $k = k + 1$
$k > n?$
YES
Output final packing results

FIGURE 7. Flowchart of the GAS algorithm.

In our algorithm, we first search by heuristics for parameters $\{\gamma_1^*, \gamma_2^*, \gamma_3^*, \gamma_4^*, \alpha^*\}$. In each iteration of our algorithm, the score of parameters $\{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \alpha\}$ is evaluated as the best utilization of the final solution with all four predefined sequences. Then, with the fixed parameters $\{\gamma_1^*, \gamma_2^*, \gamma_3^*, \gamma_4^*, \alpha^*\}$, we heuristically search for a better predefined sequence. The detailed algorithm is presented in Algorithm 2. The genetic algorithm framework, as well as the scheme for selection, crossover and mutation in Algorithm 2, can be found in [43].

V. COMPUTATIONAL RESULTS

Two groups of instances were tested using the GAS: the data provided by the Alibaba Cloud and some of the benchmark instances. The data provided by the Alibaba Cloud are real data in industrial production; shirts, trousers and other parts are combined for packing, and the numbers of both parts and vertices are relatively large. The classic benchmark instances include various materials, and we chose fabric

TABLE 1. Data from Alibaba Cloud and Benchmark ESICUP instances. TNI:Total number of instances. NIT: number of item types. ANV: average number of vertices. CANV: cleaned average number of vertices. AO: admissible orientations.

Case	TNI	NIT	ANV	CANV	AO	Flaw	d_1(mm)	d_2(mm)	d_3(mm)
Data1	314	121	81.57	39.77	$0°, 180°$	N	5	-	0
Data2	283	221	77.43	30.58	$0°, 180°$	N	5	-	0
Data3	252	113	72.28	40.94	$0°, 180°$	Y	5	5	5
Data4	293	240	77.42	30.27	$0°, 180°$	Y	5	5	5
Shirts*4	396	8	6.63	6.63	$0°, 180°$	N	0	-	0
Trousers*4	256	17	5.06	5.06	$0°, 180°$	N	0	-	0

TABLE 2. Maximum and average utilization of real cases from Alibaba Cloud obtained by GAS and DeepNest. Avg:average. Uti:Utilization.

Case	GAS+GA (10 runs)			GAS		GAS+GA ($\alpha = 1$,10 runs)			DeepNest(10 runs)		
	Best %	Avg %	Time(s)	Uti%	Time(s)	Best %	Avg %	Time(s)	Best %	Avg %	Time(s)
Data1	85.36	85.17	1200	84.91	60	84.22	84.10	1200	81.10	80.71	1200
Data2	86.06	85.92	1200	85.62	60	84.87	84.62	1200	82.53	82.32	1200
Data3	86.31	86.19	1200	86.10	60	85.21	85.09	1200	83.79	83.56	1200
Data4	85.31	85.21	1200	85.13	60	84.31	84.17	1200	81.55	81.27	1200

Algorithm 2 Greedy Adaptive Search Modified by Genetic Algorithm (GAS + GA)

```
Input: Initial sequences $seq_1, ..., seq_4$, size of the population $N$, initial population for parameters $\{\gamma_1^{u,0}, \gamma_2^{u,0}, \gamma_3^{u,0}, \gamma_4^{u,0}, \alpha^{u,0}\}$, where $u = 1, ..., N$, population size for sequences $M$
Output: Final utilization rate and packing results
1: Initialization: $k = 0$;
2: while Continue do
3:     Compute the fitness for each individual in the population by
           $\max_{i=1,2,3,4}$(utilization of GAS($seq_i, \gamma_1^{u,k}, \gamma_2^{u,k}, \gamma_3^{u,k}, \gamma_4^{u,k}, \alpha^{u,k}$)),
       for $u = 1, ..., N$;
4:     Select individuals by tournament selection;
5:     Crossover by arithmetic crossover;
6:     Mutate by Gaussian mutation;
7:     $k = k + 1$;
8: end while
9: Choose the parameters $\gamma_1^*, \gamma_2^*, \gamma_3^*, \gamma_4^*, \alpha^*$ that corresponds to the highest fitness;
10: Mutate and crossover $seq_1, ..., seq_n$ to generate the initial population for sequence $\{seq_1, ..., seq_M\}$;
11: while Continue do
12:     Compute the fitness for each sequence in the population by GAS($seq_i, \gamma_1^*, \gamma_2^*, \gamma_3^*, \gamma_4^*, \alpha^*$), for $i = 1, ..., M$;
13:     Select sequences by tournament selection;
14:     Perform crossover by partially matched crossover;
15:     Mutate by inversion mutation;
16: end while
```

material including shirts and trousers to test the GAS. For each instance, independent of the group, the algorithm was executed ten times. The algorithm was tested on an Intel(R) Core(R) Silver 4110 CPU @ 2.1 GHz, with 32 cores and 394 GB of memory.

A. TEST INSTANCES

In Table 1, we summarize some basic characteristics of the test instances; all the cases admit orientation $0°, 180°$.

Data 1-4 are large-scale industrial datasets collected by the Alibaba Cloud and can be retrieved from their website.[1] These datasets have over 250 parts with the mean vertices over 70, showing that the problems are large-scale with the complex contour of the parts. As illustrated in Table 1, the proposed two preprocessing algorithms in Section 2.1 significantly reduce the mean vertices for the parts. Moreover, a certain cutting gap needs to be reserved among the parts, and some of the sheets have flaws in it, which need to be avoided during packing.

The other two datasets were generated from benchmarks in ESICUP[2] datasets. Since our algorithm is dedicated to solving the large-scale fabric packing problem, we select the fabric packing datasets 'shirts' and 'trousers' from the ESICUP datasets. In addition, these two datasets are small-scaled, so we generate large-scale instances by copying the shapes of these two instances four times.

B. TEST RESULTS

In our numerical examples, GAS + GA denotes our greedy adaptive search modified by the generic algorithm. We tested the performance of both GAS and GAS+GA. The GAS algorithm is a fast deterministic algorithm, so we only need to run one time. Based on GAS, GAS+GA is a modified nondeterministic algorithm. Therefore, we present both the best utilization and average utilization in Table 2-3.

In our numerical experiments, the mutation rate in GAS + GA equals 0.3, the crossover rate equals 1 and the tournament size is 2. In addition, since the search space of the sequences is much larger than the search space of all parameters, we choose the population sizes $N = 16$ and $M = 64$ in Algorithm 2. Additionally, the max generation for the first

[1]https://tianchi.aliyun.com/competition/entrance/231749/information.
[2]https://www.euro-online.org/websites/esicup/data-sets/.

TABLE 3. Maximum and average utilization of benchmark instances obtain by GAS and other best solutions in the literature. Avg:average. Uti:Utilization.

Case	GAS+GA(10 runs)			GAS		Hu et al. [28]		Best in the literature	
	Best %	Avg %	Time(s)	Uti%	Time(s)	Best %	Time(s)	Best %	Time(s)
Shirts*4	89.30	89.02	1200	88.73	10	84.44	4.64	88.96 [21]	1200
Trousers*4	91.10	91.01	1200	90.71	10	-	-	91.06 [43]	1200

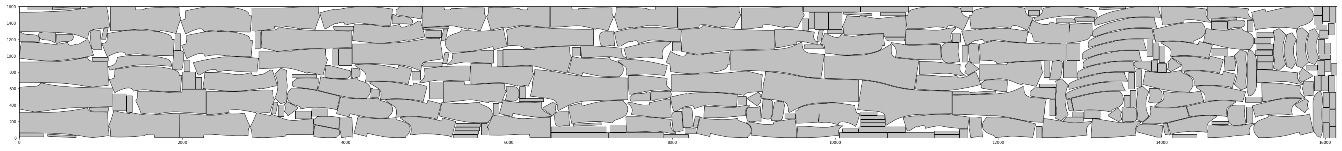


FIGURE 8. **Data1 (85.36%).**

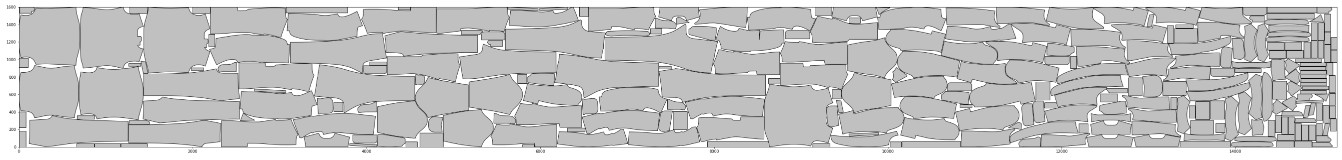


FIGURE 9. **Data2 (86.06%).**

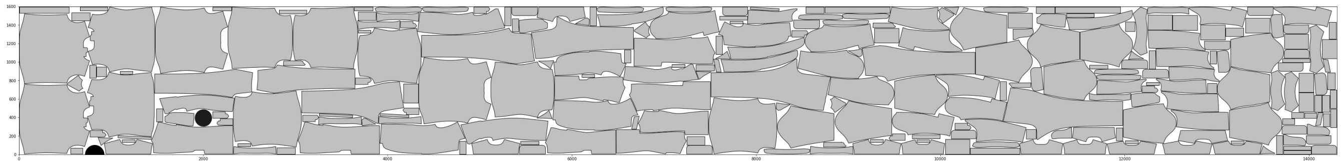


FIGURE 10. **Data3 (86.31%). Black parts denote the flaws.**

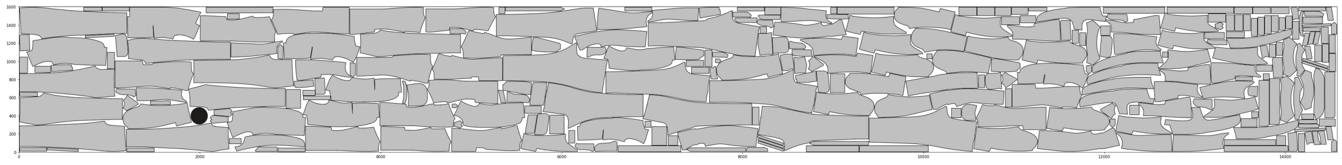


FIGURE 11. **Data4 (85.31%). Black parts denote the flaws.**

stage is 10, while the second stage of Algorithm 2 stops when its runtime exceeds the time limit.

Data 1-4 were made public recently, so no paper has comparable test results on these instances. Since the algorithm code in previous papers is not open-source, we compared the results by running the open-source software DeepNest [17] and showed the results in Table 2. The final results of GAS + GA on Data 1-4 are presented in Fig. 8-11. From Table 2, we can conclude that GAS + GA is more stable and efficient than DeepNest. On data 1-4, the results of our algorithm were 3% - 4% higher than the optimal value of the DeepNest's. Furthermore, we ran GAS+GA with two different evaluation functions to illustrate the superiority of restricted local search with depth, as presented in Section IV-B. One uses (16) as its evaluation function, and the other uses (16) with fixed $\alpha = 1$. It is worth mentioning that when we fix $\alpha = 1$, both γ_3 and γ_4 are inactive and set to 0 since they are designed for adjusting the placing order in the pool. As can be seen from Table 2, the restricted local search with depth strategy indeed improves the utilization and delivers better packing results.

Benchmark instances have been massively tested in the literature, but few articles tested the cases that have been multiplied. Hu *et al.* [27] proposed a fast algorithm and tested that algorithm on 'shirts*4', Elkeran and Ahmed [21] proposed a guided cuckoo search and obtained the best layout of the 'shirt' as 88.96%, Sato *et al.* [41] applied the raster penetration map and achieved the best results for trousers as 90.06%. Moreover, these two algorithms are based on searching over the layout approach, which lacks scalability and efficiency for large-scale packing problems, as described in our introduction. Compared with the algorithms above, as shown in Table 3, our algorithm shows advantages both in operational efficiency and utilization on large-scale problems. Additionally, we present the final packing results in Fig. 12 and Fig. 13.

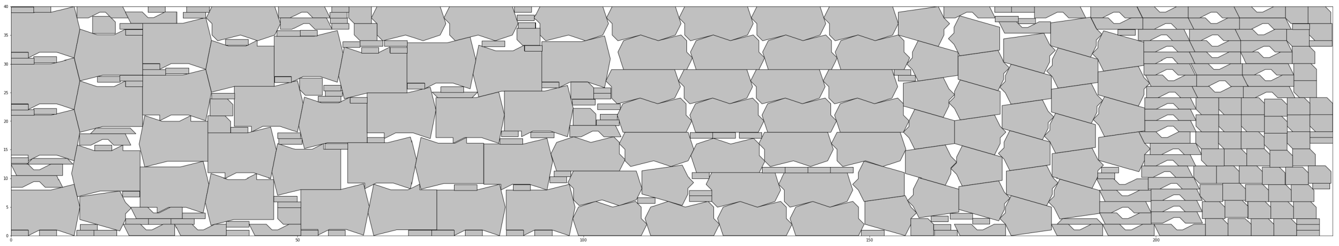


FIGURE 12. Shirts*4 (89.30%).

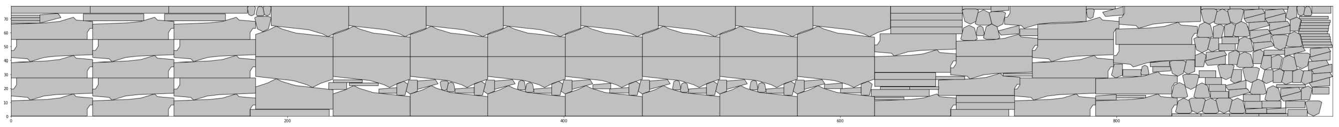


FIGURE 13. Trousers*4 (91.10)%.

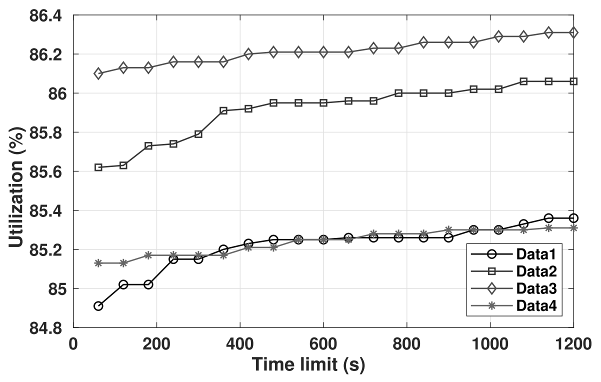


FIGURE 14. Utilization of Algorithm GAS + GA on Datasets 1-4 for different time limits.

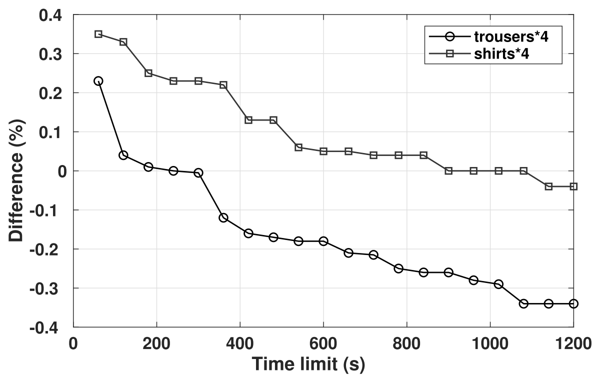


FIGURE 15. Difference between the best compaction in the literature and the GAS+GA utilization for different time limits.

C. COMPUTATIONAL PERFORMANCE DISCUSSION

In 2-dimensional irregular packing problems, most of the existing approaches have been tested on the ESICUP dataset. Most cases in the ESICUP dataset are small-scale or medium-scale and have simple shapes. However, in this paper, we focus on large-scale packing problems where a piece of cloth is usually 50 metres to 200 metres. Additionally, due to the lack of open-source code, we compared our algorithm with DeepNest, a state-of-the-art open-source solver based on bottom-left strategy and improved by deep learning [17]. In our previous numerical examples, we multiplied the number of benchmark instances and performed GAS + GA and efficiently achieved higher utilization results. In addition, we performed GAS on the data provided by the Alibaba Cloud and achieved high-utilization results, which were 3% - 4% better than those by DeepNest.

However, the lack of open-source code makes it difficult to determine the quality of our algorithm and compare it with other approaches, which were tested with different running environments and computational capacities. To enhance the comparison analysis of our algorithm, a computational performance study of GAS + GA was conducted, and the numerical results are presented in this subsection. During the execution of GAS + GA on all our test examples, the utilization of the solutions was sampled at each 60-second interval. In addition, we ran the first stage in GAS + GA for 10 generations and then terminated the second stage in GAS + GA when the running time exceeded 1, 200. This procedure is equivalent to setting stricter time limits on GAS + GA. Since datasets 1-4 have not been tested by existing works, we present the changes in the utilization rate in Fig14 to illustrate the detailed performance of GAS + GA. In addition, because trousers*4 and shirts*4 have not been tested by existing efficient approaches, we present the difference between GAS+GA and the best-in-literature utilization rates for trousers and shirts in the ESICUP dataset. The difference is presented in Fig. 15.

The testing results in Fig. 14 and Fig. 15 show similar tendencies on all testing examples. It can be noted that the utilization increases more intensely in the first stage of GAS+GA, which coincides with our discussion in Section IV-B and illustrates the high efficiency of our two-stage algorithm. This provides further evidence that a reduced time limit does not greatly impact the results for these cases.

VI. CONCLUSION

The greedy adaptive search algorithm (GAS) is designed to solve large-scale fabric packing problems. Inspired by TOPOS, we construct a dynamically adjusted evaluation function as well as a restricted local search strategy. In addition, we proposed a two-stage genetic algorithm, searching for both proper parameters and input sequences.

In our algorithm, the overlap is detected by the no-fit polygon, which is computed in parallel in our preprocessing step. Because the contours of the parts in the fabric packing problem can have a large number of vertices, we introduce two algorithms to simplify the contours and thus accelerate the geometric computation in the preprocessing stage. As illustrated in Table 1, our proposed approaches reduce approximately 50% of the vertices.

Then, we tested two sets of instances by using the proposed GAS + GA. We first compare GAS + GA with GAS + GA ($\alpha = 1$), GAS and DeepNest on our test problems. The results illustrate that GAS + GA is more efficient and stable, and our proposed restricted local search with the depth strategy results in higher utilization results than DeepNest, even in one pass of the algorithm. In addition, tests using benchmark cases show that GAS + GA yields competitive solutions, producing the best solution in the literature on our test instances, which were generated from shirts and trousers in the ESICUP dataset.

In conclusion, on large-scale packing problems, our algorithm is significantly better than existing open-source software and papers. The GAS shows innovation in both the evaluation function and the idea of the penalty in restricted local search. The potential of applying the GAS + GA to large-scale packing problems in the fabric industry emerges, and we will continuously work on developing efficient algorithms on large-scale packing problems.

REFERENCES

[1] P. K. Agarwal, E. Flato, and D. Halperin, "Polygon decomposition for efficient construction of minkowski sums," *Comput. Geometry*, vol. 21, nos. 1–2, pp. 39–61, Jan. 2002.

[2] R. Alvarez-Valdes, A. Martinez, and J. M. Tamarit, "A branch & bound algorithm for cutting and packing irregularly shaped pieces," *Int. J. Prod. Econ.*, vol. 145, no. 2, pp. 463–477, Oct. 2013.

[3] R. Carl, Jr., "An approach to the two dimensional irregular cutting stock problem," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 1966.

[4] T. Asano, A. Hernández-Barrera, and S. C. Nandy, "Translating a convex polyhedron over monotone polyhedra," *Comput. Geometry*, vol. 23, no. 3, pp. 257–269, Nov. 2002.

[5] J. A. Bennell and K. A. Dowsland, "A tabu thresholding implementation for the irregular stock cutting problem," *Int. J. Prod. Res.*, vol. 37, no. 18, pp. 4259–4275, Dec. 1999.

[6] J. A. Bennell and K. A. Dowsland, "Hybridising tabu search with optimisation techniques for irregular stock cutting," *Manage. Sci.*, vol. 47, no. 8, pp. 1160–1172, Aug. 2001.

[7] J. A. Bennell and X. Song, "A comprehensive and robust procedure for obtaining the nofit polygon using minkowski sums," *Comput. Oper. Res.*, vol. 35, no. 1, pp. 267–281, Jan. 2008.

[8] J. A. Bennell and X. Song, "A beam search implementation for the irregular shape packing problem," *J. Heuristics*, vol. 16, no. 2, pp. 167–188, Apr. 2010.

[9] J. Błażewicz, P. Hawryluk, and R. Walkowiak, "Using a tabu search approach for solving the two-dimensional irregular cutting problem," *Ann. Oper. Res.*, vol. 41, no. 4, pp. 313–325, Dec. 1993.

[10] E. Burke, R. Hellier, G. Kendall, and G. Whitwell, "A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem," *Oper. Res.*, vol. 54, no. 3, pp. 587–601, Jun. 2006.

[11] E. K. Burke, R. S. R. Hellier, G. Kendall, and G. Whitwell, "Complete and robust no-fit polygon generation for the irregular stock cutting problem," *Eur. J. Oper. Res.*, vol. 179, no. 1, pp. 27–49, May 2007.

[12] M. A. Carravilla, C. Ribeiro, and J. F. Oliveira, "Solving nesting problems with non-convex polygons by constraint logic programming," *Int. Trans. Oper. Res.*, vol. 10, no. 6, pp. 651–663, Nov. 2003.

[13] M. Chen, K. Li, D. Zhang, L. Zheng, and X. Fu "Hierarchicalsearch-embedded hybrid heuristic algorithm for two-dimensional strip packing-problem," *IEEE Access*, vol. 7, pp. 179086–179103, 2019.

[14] N. Chernov, Y. Stoyan, T. Romanova, and A. Pankratov, "Phi-functions for 2D objects formed by line segments and circular arcs," *Adv. Oper. Res.*, vol. 2012, pp. 1–26, May 2012.

[15] L. H. Cherri, L. R. Mundim, M. Andretta, F. M. B. Toledo, J. F. Oliveira, and M. A. Carravilla, "Robust mixed-integer linear programming models for the irregular strip packing problem," *Eur. J. Oper. Res.*, vol. 253, no. 3, pp. 570–583, Sep. 2016.

[16] K. Daoden and T. Thaiupathump "Applying shuffled frog leaping algorithm and bottom left fill algorithm in rectangular packing problem," in *Proc. 7th IEEE Int. Conf. Electron. Inf. Emergency Commun. (ICEIEC)*, Jul. 2017, pp. 136–139.

[17] (Jan. 18, 2020). *Deepnest*. [Online]. Available: https://deepnest.io/

[18] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica, Int. J. Geographic Inf. Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[19] K. A. Dowsland, W. B. Dowsland, and J. A. Bennell, "Jostling for position: Local improvement for irregular cutting patterns," *J. Oper. Res. Soc.*, vol. 49, no. 6, pp. 647–658, 1998.

[20] J. Egeblad, B. K. Nielsen, and A. Odgaard, "Fast neighborhood search for two- and three-dimensional nesting problems," *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1249–1266, Dec. 2007.

[21] A. Elkeran, "A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering," *Eur. J. Oper. Res.*, vol. 231, no. 3, pp. 757–769, Dec. 2013.

[22] M. Fischetti and I. Luzzi, "Mixed-integer programming models for nesting problems," *J. Heuristics*, vol. 15, no. 3, pp. 201–226, Jun. 2009.

[23] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto, "Optimal packing and covering in the plane are NP-complete," *Inf. Process. Lett.*, vol. 12, no. 3, pp. 133–137, Jun. 1981.

[24] A. M. Gomes and J. F. Oliveira, "A 2-exchange heuristic for nesting problems," *Eur. J. Oper. Res.*, vol. 141, no. 2, pp. 359–370, Sep. 2002.

[25] A. M. Gomes and J. F. Oliveira, "Solving irregular strip packing problems by hybridising simulated annealing and linear programming," *Eur. J. Oper. Res.*, vol. 171, no. 3, pp. 811–829, Jun. 2006.

[26] B. Guo, Z. Liang, Q. Peng, Y. Li, and F. Wu "Irregular packing based on principal component analysis methodology," *IEEE Access*, vol. 6, pp. 62675–62686, 2018.

[27] Y. Hu, S. Fukatsu, H. Hashimoto, S. Imahori, and M. Yagiura, "Efficient overlap detection and construction algorithms for the bitmap shape packing problem," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage.*, Jan. 2014, pp. 1056–1060.

[28] T. Imamichi and H. Nagamochi, "A multi-sphere scheme for 2D and 3D packing problems," in *Proc. Int. Workshop Eng. Stochastic Local Search Algorithms*. Cham, Switzerland: Springer, 2007, pp. 207–211.

[29] T. Imamichi, M. Yagiura, and H. Nagamochi, "An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem," *Discrete Optim.*, vol. 6, no. 4, pp. 345–361, Nov. 2009.

[30] J. Kallrath, "Cutting circles and polygons from area-minimizing rectangles," *J. Global Optim.*, vol. 43, nos. 2–3, pp. 299–328, Mar. 2009.

[31] I. Kierkosz and M. Łuczak, "A one-pass heuristic for nesting problems," *Oper. Res. Decis.*, vol. 29, no. 1, pp. 37–60, 2019.

[32] A. A. S. Leao, F. M. B. Toledo, J. F. Oliveira, and M. A. Carravilla, "A semi-continuous MIP model for the irregular strip packing problem," *Int. J. Prod. Res.*, vol. 54, no. 3, pp. 712–721, Feb. 2016.

[33] A. A. S. Leao, F. M. B. Toledo, J. F. Oliveira, M. A. Carravilla, and R. Alvarez-Valdés, "Irregular packing problems: A review of mathematical models," *Eur. J. Oper. Res.*, vol. 282, no. 3, pp. 803–822, May 2020.

[34] L. R. Mundim, M. Andretta, M. A. Carravilla, and J. F. Oliveira, "A general heuristic for two-dimensional nesting problems with limited-size containers," *Int. J. Prod. Res.*, vol. 56, nos. 1–2, pp. 709–732, Jan. 2018.

[35] J. F. Oliveira, A. M. Gomes, and J. S. Ferreira, "TOPOS—A new constructive algorithm for nesting problems," *OR Spektrum*, vol. 22, no. 2, p. 263, 2000.

[36] P. R. Pinheiro, B. Amaro, Jr., and R. D. Saraiva, "A random-key genetic algorithm for solving the nesting problem," *Int. J. Comput. Integr. Manuf.*, vol. 29, no. 11, pp. 1159–1165, Nov. 2016.

[37] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Comput. Graph. Image Process.*, vol. 1, no. 3, pp. 244–256, Nov. 1972.

[38] C. Ribeiro and M. A. Carravilla, "A global constraint for nesting problems," in *Proc. Int. Conf. Integr. Artif. Intell. (AI) Oper. Res. (OR) Techn. Constraint Program.* Cham, Switzerland: Springer, 2004, pp. 256–270.

[39] C. Ribeiro, M. A. Carravilla, and J. F. Oliveira, "Applying constraint logic programming to the resolution of nesting problems," *Pesquisa Operacional*, vol. 19, no. 2, pp. 239–247, 1999.

[40] M. O. Rodrigues and F. M. B. Toledo, "A clique covering MIP model for the irregular strip packing problem," *Comput. Oper. Res.*, vol. 87, pp. 221–234, Nov. 2017.

[41] A. K. Sato, T. C. Martins, A. M. Gomes, and M. S. G. Tsuzuki, "Raster penetration map applied to the irregular packing problem," *Eur. J. Oper. Res.*, vol. 279, no. 2, pp. 657–671, Dec. 2019.

[42] G. Scheithauer and J. Terno, "Modeling of packing problems," *Optimization*, vol. 28, no. 1, pp. 63–84, Jan. 1993.

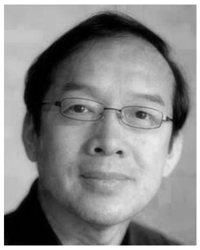
[43] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994.

[44] Y. Stoyan, T. Romanova, A. Pankratov, and A. Chugay, "Optimized object packings using Quasi-Phi-functions," in *Optimized Packings With Applications*. Cham, Switzerland: Springer, 2015, pp. 265–293.

[45] Q. Yang, "No fit polygon for nesting problem solving with hybridizing ant algorithms," *J. Softw. Eng. Appl.*, vol. 7, no. 5, pp. 433–439, 2014.

XIAOYIN HU received the B.S. degree from Zhejiang University, China, in 2016. She is currently pursuing the Ph.D. degree with the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China. Her research interests include combinatorial optimization problem and complexity analysis of algorithm.

JIANSHU LI received the B.S. degree in economics from the University of International Business and Economics, Beijing, China, in 2015. He is currently pursuing the Ph.D. degree in mathematics from the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing. His research interests include combinatorial optimization and computational complexity.



JINCHUAN CUI received the B.S. degree from Zhejiang University, China. He is currently a Professor at the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China. His research interests include combinatorial optimization problems, complexity analysis of algorithm, and resource allocation problems. He received the Prize for OR in Development in IFORS 1996.