A GUIDED LOCAL SEARCH ALGORITHM BASED ON A FAST NEIGHBORHOOD SEARCH FOR THE IRREGULAR STRIP PACKING PROBLEM

Shunji Umetani\textsuperscript{1}, Mutsunori Yagiura\textsuperscript{2}, Takashi Imamichi\textsuperscript{3}, Shinji Imahori\textsuperscript{4}, Koji Nonobe\textsuperscript{5} and Toshihide Ibaraki\textsuperscript{6}

\textsuperscript{1}Faculty of Electro-Communications, University of Electro-Communications, Chofu, Tokyo 182-8585, Japan
\textsuperscript{2}Graduate School of Information Science, Nagoya University, Nagoya 464-8603, Japan
\textsuperscript{3}Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan
\textsuperscript{4}Graduate School of Information Science and Technology, University of Tokyo, Tokyo 113-8656, Japan
\textsuperscript{5}Faculty of Engineering, Hosei University, Koganei, Tokyo 184-8584, Japan
\textsuperscript{6}School of Science and Technology, Kwansei Gakuin University, Sanda, Hyogo 669-1337, Japan

1 umetani@se.uec.ac.jp, 2 yagiura@nagoya-u.jp, 3 ima@amp.i.kyoto-u.ac.jp, 4 imahori@simplex.t.u-tokyo.ac.jp
5 nonobe@k.hosei.ac.jp, 6 ibaraki@ksc.kwansei.ac.jp

Abstract

The irregular strip packing problem asks to place a set of polygons within a rectangular strip of fixed height without overlap, so as to minimize the strip width required. We consider an overlap minimization problem which minimizes the amount of overlap penalty for all pairs of polygons within a given bound of strip width. We propose a local search algorithm which translates a polygon in horizontal and vertical directions iteratively, and incorporate it in metaheuristic approaches called the iterated local search and the guided local search. Computational results show that our algorithm is competitive with other existing algorithms.

Keywords: Cutting, Packing, Guided local search, Irregular strip packing, Nesting.

1. Introduction

Many cutting and packing problems appear in various industries such as wood, textile, sheet metal, plastics, glass and leather. The irregular strip packing problem is one of cutting and packing problems that asks to place a set of polygons within a rectangular strip of fixed height without overlap, so as to minimize the strip width required (Fig. 1). The problem is known to be NP-hard even without rotation. The problem is much harder than the rectangle strip packing problem because of a burden of geometrical computation. However, based on rapid development of computing power and theory of computational geometry, many approaches have been developed in recent years.

Albano and Sapuppo (1980) proposed a bottom-left heuristic algorithm that places polygons one by one at the bottommost and leftmost position according to a sequence of input polygons, and used to a tree search to obtain a good sequence. Gomes and Oliveira (2002) used a local search algorithm to find a good sequence for the bottom-left heuristics. Li and Milenkovic (1995) proposed separation and compaction algorithms based on linear programming that reduce the strip width and the amount of overlap, respectively. Bennell and Dowsland (2001), and Gomes and Oliveira (2006) developed hybrid approaches of the bottom-left heuristic and linear programming to obtain good solutions. Burke et al. (2006) proposed a new bottom-left fill heuristic algorithm which can place shapes that include circular arcs and holes, and incorporated it with tabu search.

Egeblad et al. (2006) considered an overlap minimization problem which minimizes the amount of overlap penalty for all pairs of polygons within a given bound of strip width. Here, they used the area of intersection for a pair of polygons as the overlap penalty. For the problem, they proposed an efficient neighborhood search that finds a position with the minimum total overlap penalty in its neighborhood, and incorporate it with the guided local search (Voudouris and Tsang, 1999).

In this paper, we first propose another overlap minimization problem, in which we use an approximate penetration depth for a pair of polygons as the overlap penalty. We propose an efficient implementation of the neighborhood search...
by utilizing a data structure called the no-fit polygon (NFP). Based on this, we propose a local search algorithm which repeats translating a polygon in horizontal and vertical directions until no better position is found in either direction, and incorporate it with the iterated local search and a variant of guided local search called the weighting method (Selman and Kautz, 1993).

In the following section, we formulate the irregular strip packing problem and the overlap minimization problem. In Section 4, we describe the local search algorithm with details concerning its implementation. Finally, Section 5 presents the computational experiments and some concluding remarks.

2. Formulation

The problem is described as follows: We are given a set of small polygons\(^1\) \(\mathcal{P} = \{P_1, P_2, \ldots, P_n\}\) and a rectangular strip \(R = R(W)\) of height \(H\) and width \(W\) called the stock sheet, where \(R\) is a given constant and \(W\) is a unit variable. Each polygon \(P_i\) in \(\mathcal{P}\) has a set of modes \(M_i = \{1, 2, \ldots, m_i\}\), which specifies its configuration except for its position, e.g., reflection, rotation by a given degree. We denote polygon \(P_i\) in \(\mathcal{P}\) specified by a mode \(k_i \in M_i\) by \(P(k_i)\).

For convenience, we consider that each of polygons \(P_i \in \mathcal{P}\) represents its inner region without its boundary, and the strip \(R\) represents its inner region including its boundary. Let \(\mathcal{R} = R(W)\) be the complement (i.e., the outer region) of the strip \(R\). We describe translations of polygons by Minkowski sums (de Berg et al., 1998), i.e., we denote the translation of polygon \(P_i \in \mathcal{P}\) by a translation vector \(t_i = (x_i, y_i)\) by \(P_i \oplus t_i = \{p + t_i | p \in P_i\}\). We describe a solution of this problem by positions \(t = (t_1, \ldots, t_n)\) and modes \(k = (k_1, \ldots, k_n)\) of all polygons \(P_i \in \mathcal{P}\). Note that the minimum width \(W\) is the x-coordinate interval between the leftmost and rightmost points of the polygons placed by \((t, k)\). The irregular strip packing problem is formally described as follows:

\[
\begin{align*}
\text{minimize} & \quad W \\
\text{subject to} & \quad (P(k_i) \oplus t_i) \cap (P(j_k) \oplus t_j) = \emptyset, (1 \leq i < j \leq n), \\
& \quad (P(k_i) \oplus t_i) \cap \mathcal{R}(W) = \emptyset, (1 \leq i \leq n), \\
& \quad W \in \mathbb{R}_+, \\
& \quad t_i \in \mathbb{R}^2, (1 \leq i \leq n), \\
& \quad k_i \in M_i, (1 \leq i \leq n).
\end{align*}
\]

The first constraint ensures that no pair of polygons overlaps, and the second constraint ensures that no polygon protrudes from the strip.

We now consider a variant of the irregular strip packing problem called the overlap minimization problem that minimizes the amount of the overlap penalty \(f_{ij}(t, k)\) for all pairs of polygons \(P_i, P_j \in \mathcal{P}\), while constraining the strip width \(W\) within a bound \(W_{UB}\) given by users.

\[
\begin{align*}
\text{minimize} & \quad F(t, k) = \sum_{1 \leq i < j \leq n} f_{ij}(t, k) \\
\text{subject to} & \quad (P(k_i) \oplus t_i) \cap \mathcal{R}(W_{UB}) = \emptyset, (1 \leq i \leq n), \\
& \quad t_i \in \mathbb{R}^2, (1 \leq i \leq n), \\
& \quad k_i \in M_i, (1 \leq i \leq n).
\end{align*}
\]

Egeblad et al. (2006) used the area of intersection of polygons \(P_i, P_j \in \mathcal{P}\) as the overlap penalty \(f_{ij}(t, k)\). In this paper, we use an approximate penetration depth of a pair of overlapping polygons \(P_i\) and \(P_j\) instead. We define the approximate penetration depth \(f_{ij}(t, k, v)\) of a pair of overlapping polygons \(P_i\) and \(P_j\) as the minimum translational distance in a given direction \(v = (v_x, v_y) \in \mathbb{R}^2\) to separate them. If a pair of polygons do not overlap, their approximate penetration depth is zero. The approximate penetration depth is formally described as:

\[
f_{ij}(t, k, v) = \min \{|z| \mid (P(k_i) \oplus t_i) \cap (P(k_j) \oplus t_j + zv) = \emptyset, z \in \mathbb{R}\},
\]

and the overlap penalty for a pair of polygons \(P_i\) and \(P_j\) is accordingly given by

\[
f_{ij}(t, k) = \min \{f_{ij}(t, k, v) \mid v \in \{e_x, e_y\}\},
\]

where \(e_x\) (resp., \(e_y\)) is a unit vector of horizontal (resp., vertical) direction.

3. No-fit polygon

For fast computation of the overlap penalty \(f_{ij}(\cdot)\), we introduce a data structure called the no-fit polygon (NFP), which is often used in the irregular strip packing problem. The no-fit polygon NFP\((P_i, P_j)\) for a pair of polygons \(P_i\) and \(P_j\) is defined by

\[
\text{NFP}(P_i, P_j) = P_i \oplus (-P_j) = \{p - q | p \in P_i, q \in P_j\}.
\]

We can easily check whether two polygons \(P_i\) and \(P_j\) overlap or not, by simply checking whether the reference point of \(P_i\) is inside NFP\((P_i, P_j)\) or not.

![Fig. 2 An example of no-fit polygon (NFP)](image)

When \(P_i\) and \(P_j\) are both convex, we can compute NFP\((P_i, P_j)\) by the following simple procedure. We first put
the reference point of \( P_i \) at the origin, and slide \( P_j \) around \( P_i \)
keeping in touch with \( P_i \). The NFP(\( P_i, P_j \)) is the inner region of
the trajectory drawn by the reference point of \( P_j \) (Fig. 2).

We can also check whether a polygon \( P_i \in \mathcal{P} \) protrudes
from the strip \( R \) by NFP(\( R, P_i \)), i.e., a polygon \( P_i \) protrudes from
\( R \) if and only if the reference point of \( P_i \) is inside NFP(\( R, P_i \)).

Although it takes \( O(p_1^2 p_2^2) \) time to compute an NFP of
two non-convex polygons with \( p_1 \) and \( p_2 \) edges (de Berg et al.,
1998) in the worst case, practical algorithms to compute it have been proposed, e.g., by Bennell et al. (2001) and
Burke et al. (2006).

4. The local search algorithm

4.1 Overview of the algorithm

The local search (LS) is a basic component of metaheuristics,
which starts from an initial solution and repeatedly replaces
the current solution with a better solution in its neighborhood
until no better solution is found in the neighborhood.

The outline of the local search (LS) algorithm for the overlap
minimization problem is given as follows. LS first generates an initial solution \( (t, k) \) by random placement policy,
i.e., it selects modes of all polygons randomly, and places them at random positions within the strip \( R(W_{UB}) \). Next, the neighborhood \( NB(t, k) \) of the current solution \( (t, k) \) is defined as the set of solutions obtained by applying the following steps for a polygon \( P_i \in \mathcal{P} \). LS selects a mode \( k_i \in \mathcal{M}_i \) of a polygon \( P_i \in \mathcal{P} \), and repeatedly translates it in horizontal and vertical directions alternately until no better position is found in either direction. If a neighbor solution \( (t', k') \in NB(t, k) \) satisfies \( F(t', k') < F(t, k) \), LS replaces the current solution \( (t, k) \) with the neighbor solution \( (t', k') \); otherwise LS finds another solution in the neighborhood \( NB(t, k) \). Finally, if no overlapping polygon exists or no better solution is found in the neighborhood \( NB(t, k) \), LS outputs the current solution \( (t, k) \) as a locally optimal solution and halts.

Local search

Step1: Select a mode \( k_i \in \mathcal{M}_i \) and place a position \( t_i \), within
the strip \( R(W_{UB}) \) randomly for each polygon \( P_i \in \mathcal{P} \), to construct an initial solution \( (t, k) \).

Step2: Select a polygon \( P_i \in \mathcal{P} \) and its mode \( k_i \in \mathcal{M}_i \), and repeats horizontal and vertical translations alternately until no better position is found in either direction, to obtain a neighbor solution \( (t', k') \).

Step3: If \( F(t', k') < F(t, k) \) holds, set \( (t, k) \leftarrow (t', k') \).

Step4: If \( F(t, k) = 0 \) (i.e., no overlapping polygon exists) holds or no better solution is found in the neighborhood \( NB(t, k) \), output \( (t, k) \) and halt; otherwise return to Step2.

4.2 Fast neighborhood search

For each translation in the neighborhood search, LS finds a position \( t_i' = t_i + z_i'v \ (v \in \{e_x, e_y\}) \) of a given polygon \( P_i \in \mathcal{P} \) that minimizes the following total overlap penalty function of \( P_i \):

\[
F_i(t_i, k_i, v) = \sum_{P_j \in \mathcal{P}, j \neq i} f_{ij}(t_i, k_i, v).
\]

Fig. 3 illustrates how to compute a position \( z \) of \( P_i \) with the minimum total overlap \( F_i(t_i, k_i, v) \), where we assume that the polygon \( P_i \) moves in horizontal direction, and both \( P_i \) and \( P_j \) are convex. We first detect the set of overlapping posi-

![Fig. 3 An example of the neighborhood search](image)

tions as an interval \((z_{ij}^{\text{min}}, z_{ij}^{\text{max}})\) of \( P_i \) and \( P_j \) by computing the intersection between the horizontal line \( L \) through the reference point of \( P_i \) and NFP(\( P_j, P_i \)), where we note that NFP(\( P_j, P_i \)) is convex if both \( P_j \) and \( P_i \) are convex. From this, the overlap penalty function for the pair of \( P_i \) and \( P_j \) is computed as follows:

\[
f_{ij}(t_i, k_i, v) = \begin{cases} 0 & \text{if } z \leq z_{ij}^{\text{min}} \text{ or } z \geq z_{ij}^{\text{max}} \\ \min\{z - z_{ij}^{\text{min}}, z_{ij}^{\text{max}} - z\} & (z_{ij}^{\text{min}} < z < z_{ij}^{\text{max}}) \end{cases}
\]

We accordingly sum up overlap penalty functions \( f_{ij}(t_i, k_i, v) \) for all \( P_j \in \mathcal{P} \ (j \neq i) \), and find a position \( z^* \) minimizing the total overlap penalty \( F_i(t_i, k_i, v) \). If LS finds several positions with the minimum total overlap penalty, we choose the nearest position from the current position. In case of non-convex polygons, we can compute the total overlap penalty \( F_i(t_i, k_i, v) \) by decomposing them into convex elements, i.e., we detect the sets of overlapping positions as intervals for all pairs of convex elements and merge them. We note that our algorithm uses only simple operations instead of difficult geometrical computations such as Minkowski sums of non-convex polygons.

To facilitate the neighborhood search, we detect possibly overlapping polygons \( P_j \in \mathcal{P} \) of \( P_i \) before computing the overlap penalty function \( f_{ij}(t_i, k_i, v) \). We first project each polygon onto the y-axis (resp., x-axis) when the polygon \( P_i \)
moves in the horizontal (resp., vertical) direction (Fig. 4). We then check the intersection of two intervals \((y_i^\min, y_i^\max)\) and \((y_j^\min, y_j^\max)\), which are the projections of two polygons \(P_i\) and \(P_j\) onto the y-axis. If \((y_i^\min, y_i^\max) \cap (y_j^\min, y_j^\max) = \emptyset\) holds, we can skip the computation of the overlap penalty \(f_{ij}(t, k)\) when \(P_i\) moves in the horizontal direction.

The computation time of the algorithm is dominated by sorting the event points of the total overlap penalty function \(F(t, k, v)\). Since Egeblad et al. (2006) use the area of intersection of polygons \(P_i\) and \(P_j\) as the overlap penalty function \(f_{ij}(t, k, v)\), the number of event points of their total overlap penalty function \(F(t, k, v)\) of \(P_i\) is the product of the numbers of edges of the polygon \(P_i\) and the other polygons \(P_j \in \mathcal{P} \setminus \{P_j\}\). The other hand, since the number of event points of our overlap penalty function \(f_{ij}(t, k, v)\) is always three when two polygons \(P_i\) and \(P_j\) are both convex, the number of event points of our total overlap penalty function \(F(t, k, v)\) of \(P_i\) is at most three times as the product of the numbers of convex elements in the polygon \(P_i\) and the other polygons \(P_j \in \mathcal{P} \setminus \{P_j\}\). This implies that the number of event points of our overlap penalty function becomes much smaller than those of Egeblad’s overlap penalty function, when all polygons are convex or possible to divide a few convex elements.

4.3 Metaheuristics

It is often the case that local search (LS) alone may not attain a sufficiently good solution. To improve the situation, many variants of simple LS have been developed, and their frameworks are called metaheuristics. The iterated local search (ILS) and guided local search (GLS) are representative metaheuristic approaches, which are simple but are known to be quite effective (Glover and Kochenberger, 2003). ILS repeats LS from different initial solutions generated by perturbing the best solution so far. GLS repeats on adaptive evaluation function which is adaptively modified to resume the search from the previous locally optimal solution. For the overlap minimization problem, we develop a hybrid approach of ILS and GLS, called the iterated guided local search (IGLS).

We introduce a variant of GLS called the weighting method which has been proposed by Selman and Kautz (1993) for the satisﬁability problem (SAT). Based on preliminary computational experiments, we adopt a modiﬁed overlap penalty function for a pair of polygons \(P_i, P_j \in \mathcal{P}\) as follows:

\[
\tilde{f}_{ij}(t, k) = w_{ij} \cdot f_{ij}(t, k).
\]

where \(w_{ij}\) is the penalty weight for a pair of polygons \(P_i\) and \(P_j\). We also adopt the amount of the modified overlap penalty as follows:

\[
\tilde{F}(t, k) = \sum_{1 \leq i < j \leq n} \tilde{f}_{ij}(t, k).
\]

The penalty weights \(w_{ij}\) are adaptively modiﬁed for every LS, i.e., if two polygons \(P_i(k_i) \oplus t_i\) and \(P_j(k_j) \oplus t_j\) overlap in the last locally optimal solution \((t, k)\), GLS increases the penalty weight \(w_{ij}\) by one.

ILS starts from the first initial solution generated by random placement as same as the simple LS, and then the subsequent initial solutions are taken from their last locally optimal solutions except for \(n\) iterations of LS. In case of \(n\) iterations of LS, the next initial solution is generated by swapping the positions \(\tilde{t}_i\) and \(\tilde{t}_j\) of two polygons \(P_i, P_j \in \mathcal{P}\) \((i \neq j)\) of the best solution \((\tilde{t}^*, \tilde{k}^*)\) on the adaptive evaluation function \(\tilde{F}(\cdot)\) obtained so far, where \(P_i\) and \(P_j\) are randomly selected from \(\mathcal{P}\). If the new position \(t_i = \tilde{t}_i\) (resp., \(t_j = \tilde{t}_j\)) of the polygon \(P_i\) (resp., \(P_j\)) is infeasible, ILS selects another polygon randomly.

The outline of the iterated guided local search (IGLS) is given as follows. Here, \(iter\) and \(kick\) denote the current number of iterations of restarting LS from the last improvement and the last perturbation of the initial solution, respectively. \(\max iter\) (an input parameter given by users) speciﬁes the upper bound on \(iter\).

Iterated guided local search

Step1: Set \(iter \leftarrow 0\) and \(kick \leftarrow 0\), and initialize \(w_{ij}\) for all pairs of \(P_i, P_j \in \mathcal{P}\). Construct the ﬁrst initial solution \((t, k)\) by random placement, and set \((\tilde{t}^*, \tilde{k}^*) \leftarrow (t, k)\) and \((\tilde{t}^*, \tilde{k}^*) \leftarrow (t, k)\).

Step2: If \(kick \geq n\) holds, apply a random perturbation to the best solution \((\tilde{t}^*, \tilde{k}^*)\) on the adaptive evaluation function \(\tilde{F}(\cdot)\), to obtain the next initial solution \((t, k)\), and set \(kick \leftarrow 0\).

Step3: Start LS on the adaptive evaluation function \(\tilde{F}(\cdot)\) from the initial solution \((t, k)\), to obtain a locally optimal solution \((t^*, k^*)\).

Step4: Modify the penalty weights \(w_{ij}\) for all pairs of \(P_i, P_j \in \mathcal{P}\). If \(\tilde{F}(t^*, k^*) < \tilde{F}(\tilde{t}^*, \tilde{k}^*)\) holds, set \((\tilde{t}^*, \tilde{k}^*) \leftarrow (t^*, k^*)\).

Step5: If \(\tilde{F}(t^*, k^*) < F(t^*, k^*)\) holds, set \((t^*, k^*) \leftarrow (t^*, k^*)\) and \(iter \leftarrow 0\), and return to Step2.

Step6: If \(iter \geq \max iter\) holds, output \((t^*, k^*)\) and halt; otherwise set \(iter \leftarrow iter + 1\) and \(kick \leftarrow kick + 1\), and return to Step2.
5. Computational experiments

We conducted computational experiments for six well known benchmark instances (Table 1), which can be downloaded from the ESICUP website.2

Table 1 The benchmark instances of the irregular strip packing problem

<table>
<thead>
<tr>
<th>Instance</th>
<th>NDS</th>
<th>NTP</th>
<th>NAV</th>
<th>Degrees</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blaz1</td>
<td>7</td>
<td>28</td>
<td>6.29</td>
<td>0,180</td>
<td>15</td>
</tr>
<tr>
<td>Shapes0</td>
<td>4</td>
<td>43</td>
<td>8.75</td>
<td>0,180</td>
<td>40</td>
</tr>
<tr>
<td>Shapes1</td>
<td>4</td>
<td>43</td>
<td>8.75</td>
<td>0,180</td>
<td>40</td>
</tr>
<tr>
<td>Shirts</td>
<td>8</td>
<td>99</td>
<td>6.63</td>
<td>0,180</td>
<td>40</td>
</tr>
<tr>
<td>Swim</td>
<td>10</td>
<td>48</td>
<td>21.90</td>
<td>0,180</td>
<td>5752</td>
</tr>
<tr>
<td>Trousers</td>
<td>17</td>
<td>64</td>
<td>5.06</td>
<td>0,180</td>
<td>79</td>
</tr>
</tbody>
</table>

NDS: The number of different shapes
NTP: The total number of polygons
NAV: The average number of vertices of different shapes

Table 2 shows that comparison of our iterated guided local search (IGLS) with three existing algorithms, 2DNEST (Egeblad et al., 2006), SAHA (Gomes and Oliveira, 2006), and BLF-tabu (Burke et al.), in their best efficiency and computation time in seconds.

Table 2 Computational results of IGLS and other existing algorithms in the best length and efficiency

<table>
<thead>
<tr>
<th>Instance</th>
<th>IGLS</th>
<th>2DNEST</th>
<th>SAHA</th>
<th>BLF-tabu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blaz1</td>
<td>26.67</td>
<td>26.60</td>
<td>25.84</td>
<td>†27.20</td>
</tr>
<tr>
<td>Shapes0</td>
<td>60.45</td>
<td>59.47</td>
<td>60.0</td>
<td>65.00</td>
</tr>
<tr>
<td>Shapes1</td>
<td>55.42</td>
<td>54.04</td>
<td>56.0</td>
<td>†58.40</td>
</tr>
<tr>
<td>Shirts</td>
<td>63.53</td>
<td>62.55</td>
<td>62.22</td>
<td>63.00</td>
</tr>
<tr>
<td>Swim</td>
<td>6319.70</td>
<td>6184.37</td>
<td>5948.37</td>
<td>6462.40</td>
</tr>
<tr>
<td>Trousers</td>
<td>250.35</td>
<td>242.44</td>
<td>242.11</td>
<td>243.40</td>
</tr>
</tbody>
</table>

† They used a hill-climbing algorithm instead of the tabu search.

Table 3 Computational time of IGLS and other existing algorithms (in seconds)

<table>
<thead>
<tr>
<th>Instance</th>
<th>IGLS</th>
<th>2DNEST</th>
<th>SAHA</th>
<th>BLF-tabu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blaz1</td>
<td>174.6</td>
<td>600</td>
<td>2257</td>
<td>†501.91</td>
</tr>
<tr>
<td>Shapes0</td>
<td>40.7</td>
<td>600</td>
<td>3914</td>
<td>1515.49</td>
</tr>
<tr>
<td>Shapes1</td>
<td>66.9</td>
<td>600</td>
<td>10314</td>
<td>†1810.14</td>
</tr>
<tr>
<td>Shirts</td>
<td>56.2</td>
<td>600</td>
<td>10391</td>
<td>806.5</td>
</tr>
<tr>
<td>Swim</td>
<td>243.5</td>
<td>600</td>
<td>6937</td>
<td>607.37</td>
</tr>
<tr>
<td>Trousers</td>
<td>105.7</td>
<td>600</td>
<td>8588</td>
<td>3611.99</td>
</tr>
</tbody>
</table>

† They used a hill-climbing algorithm instead of the tabu search.

Table 2 and 3 show that comparison of our iterated guided local search (IGLS) with three existing algorithms, 2DNEST (Egeblad et al., 2006), SAHA (Gomes and Oliveira, 2006), and BLF-tabu (Burke et al.), in their best efficiency and computation time in seconds.

SAHA shows the average computation time, the column of 2DNEST shows the time limit of its computation, and the column BLF-tabu shows the computation time to find the best solution in the run found it.

SAHA shows the best results in the four algorithms; however, it spends much more computation time than the other algorithms. IGLS shows better results than those of BLF-tabu in efficiency and computation time except for Shirts and Trousers; however, it shows worse results than 2DNEST even taking account for the computation time. We note that it is not precise comparison of IGLS and the other algorithms, since IGLS solves the overlap minimization problem while the other algorithms solve the irregular strip packing problem. Nevertheless, IGLS finds no overlapping solution within short time for closely best efficiency in the literature. It is the future study to develop an efficient algorithm that minimizes the required strip length without any overlapping polygon.

Fig. 5–9 show the best solutions for five benchmark instances, Blaz1, Shapes1, Shirts, Swim and Trousers.

---

2http://www.apdio.pt/esicup/
6. Conclusions

In this paper, we present a local search algorithm for the irregular strip packing problem. We consider an overlap minimization problem which minimizes the amount of overlap penalty for all pairs of polygons within a given bound of strip length. We propose a fast neighborhood search which alternately translates a polygon in horizontal and vertical directions, so as to minimize the overlap penalty with the polygon. The neighborhood search can quickly compute a new position of the translating polygon by the projection checking and the no-fit polygon. We incorporate it in the iterated local search and the guided local search approaches. The computational results show that our algorithm attains competitive results to the best results previously published within shorter computation time.

Fig. 7 The best solution for the “Shirt” instance

Fig. 8 The best solution for the “Swim” instance

Fig. 9 The best solution for the “Trousers” instance

References


