# ガラス産業における板取り問題の最適化

喜田　豪，塔ノ上　亮太，鈴木　一弘，今堀　慎治

窓ガラスサイズのガラス板を生産する工程には，大板から既定のサイズの製品を切り出す作業がある．大板にさまざまなサイズの製品配置を行う最適化は，板取り問題と呼ばれ，NP 困難であるのに加えて，ガラス特有に求められるギロチンカット制約を含む必要がある．現場のタクトタイムの関係上，板取り問題にかけられる時間はとても短いため，歩留まりの良い解を高速に計算する必要がある．また，複雑な配置はガラスを切り出す際の不良発生のリスクが高まるため，板・切線の配置はシンプルなものが求められる．本稿では，(1) 板の配置，(2) 配置から切線解釈を行う後処理に関するアルゴリズム開発事例を紹介する．

キーワード：板取り問題，ギロチンカット制約，BL 法，貪欲法，ビンパッキング，メタヒューリスティクス

## 1.　はじめに

ガラスや鉄鋼，木材などの素材加工業では，板状の母材から実際の製品サイズ・形状に切り出す工程がある．与えられた数量のアイテムをできるだけ少ない母材から切り出す問題は「板取り問題」と呼ばれ，NP 困難であることが知られている [1]．住宅用のガラス板は，寸法が mm 単位で異なるため，多種多様な寸法のアイテムを同時に母材上に配置する必要があり，その最適化問題は容易に解くことができないことが想像される．

ガラス板の切断は図 1 のように行われる．母材となる大きいガラス板（素板）に切断部で切線と呼ばれる線状の傷をつける．この切線に沿って，ガラスを折ることでほしいサイズのガラス板を切り出していく．ガラスは，紙のようにハサミで切断することができないため，切断したいガラス板の端から端まで切線が通る必要がある．これは，ギロチンカット制約と呼ばれ，ガラス・鉄鋼業などにおいては，必須の制約である．ほかにも，素材の性質上守らなければならない制約条件を満たしながら，アイテムの配置をしなければならない．

本稿では，ガラス産業における板取り問題において，上記の制約条件を満たした最適化アルゴリズムとその性能評価について報告をする．

きだ　たけし，とうのうえ　りょうた
AGC 株式会社 先端基盤研究所
〒 230–0045 横浜市鶴見区末広町 1–1
すずき　かずひろ
AGC 株式会社 化学品カンパニー
戦略本部基盤技術開発部外部連携推進室
〒 290–8566 千葉県市原市五井海岸 10 番地
いまほり　しんじ
中央大学理工学部情報工学科
〒 112–8551 東京都文京区春日 1–13–27

## 2.　制約条件

今回取り扱う板取り問題は，ただアイテムが重ならないように配置するだけではない．特殊な制約条件について以下に記述する．

### 2.1　ギロチンカット制約

建材用ガラスの切断工程は，図 1 のように切断部で自動的に切線を入れる工程と，板折部で切線に沿って人力もしくは機械でガラスを折る作業からなる．ガラスを折る際に切線の延長線上に別の製品が配置されていると，その製品を破壊してしまう．これを避けるために，ギロチンカットと呼ばれる制約条件が課せられる．具体例を図 2 で示す．図 2(a) のような配置では，適切な順序で折ると端から端まで切線を通すことができ，すべての製品を切り出すことができる．一方，図 2(b) の配置では，切線が端から端まで通っていないので，どのような順序でもいずれかの製品を破壊することになってしまう．この制約はガラス産業だけでなく，鉄鋼産業などさまざまな素材の切り出し問題において適用されるため，過去に多くの研究がなされている [2, 3]．

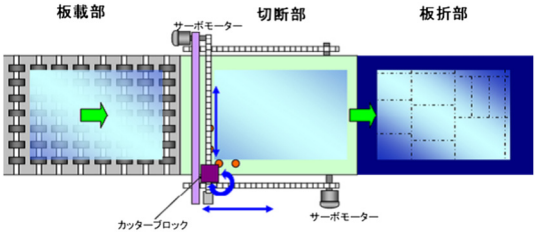



**図 1**　ガラス板の切断フロー

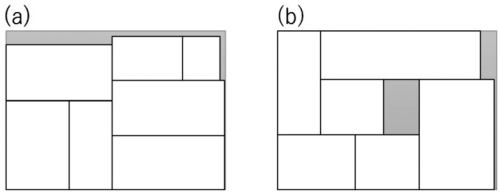



**図 2**　ギロチンカット制約

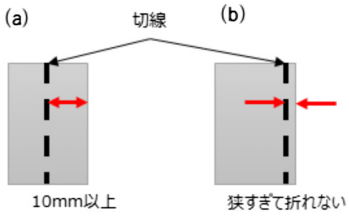



**図 3**　折り代確保制約

### 2.2　折り代確保制約

板を切線に従って折る際に，切線から余材側の端までの距離を折り代と呼ぶ．板を切る作業時，この折り代が短すぎると製品側に欠けが発生するなどのダメージを与えてしまうリスクがある．そこで図 3 に示すように，一定以上の折り代を確保する制約条件を適用し，欠け発生リスクを低減している．確保する折り代は板の厚みに関係するが，おおよそ 10〜15 mm 程度の長さが必要とされている．本値は，入力データとして与えられる．一方で折り代が長すぎると余材の面積が増加するため，高い歩留まりを得るには余材面積を最小化することが望ましい．

### 2.3　欠点回避制約

配置する素板には，気泡などの欠点が存在する場合がある．この場合，すべての欠点を避けて製品を配置する必要がある．欠点は長方形として扱い，位置とサイズは入力データによって与えられる．切線が欠点を通ることは問題視されない．

### 2.4　折深さ制約

ガラスの板取り問題においては，各切線について折深さを考慮する．この折深さは，切り出しの複雑さを示すものになる．具体例を図 4 に示す．素板を縦に貫通する線を深さ 1 と定義する．深さ 2 は折深さ 1 の線を切った後にはじめて切断可能になる切線である．深さが 1 増えるごとに，切線の方向は縦→横→縦→…のように切り替わる．理想的には折深さ制限を設けずに配置した方が高歩留まりを期待できるが，設備仕様や作業時間を考慮して深さに制限を加えている．本制約を守るために，製品配置や切線の入れ方を決定する際に，

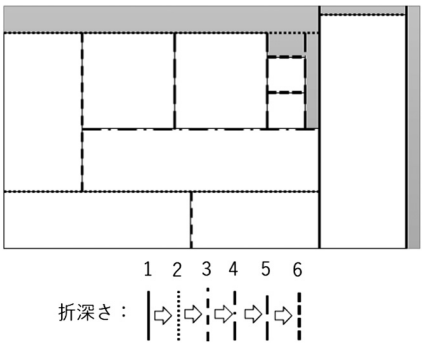



**図 4**　折深さのカウント例

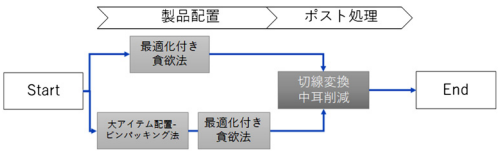



**図 5**　アルゴリズムフロー

折深さが既定値を超えないように考慮する必要がある．

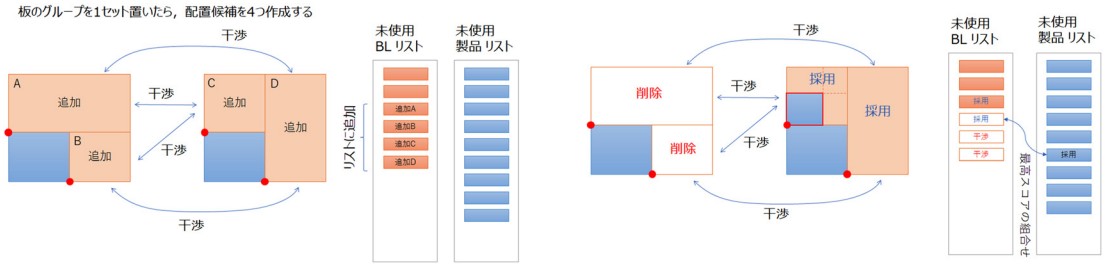
## 3.　アルゴリズム開発

開発したエンジンは，最適化付き貪欲法，大アイテム配置-ビンパッキング法による製品配置と，ポスト処理としての切線変換・中耳削減の 3 種類のアルゴリズムで構成される．全体の流れを図 5 に示す．本エンジンに求められる性能としては 2 節に示した制約と共に，計算時間にも厳しい制限が課せられている．

### 3.1　最適化付き貪欲法

#### 3.1.1　アルゴリズム概要

本アルゴリズムは，板を左下 (Bottom Left: BL) から詰めていき，最も良い形を選定する．配置の是非は 1 回ごとに評価関数で決めている．製品と候補領域（BL 領域）の最も良い組合せを貪欲的に選び続ける．最初の BL 領域は素板 1 枚分に相当する．候補となる BL 領域と製品をそれぞれの未使用リストに追加しておく．次に任意の BL 領域に対して，任意の製品を選び，配置を行う．この状態を後述する評価関数で評価して，最も良い BL 領域と製品の組合せを選ぶ．選定した組合せに基づいて図 6(a) に示すように，次の BL 領域を作り未使用リストに追加する．このとき，次の BL 領域の候補は 2 種類のパターン（A・B と C・D のセット）がある．この 2 種類の BL 領域は，どちらかが採用されたときには BL 領域の候補から削除をする必要がある．どちらか一方のパターンしか取れない状況になっているものを干渉と呼んでいる．図 6(b) に示す




(a)領域の追加　　(b)干渉領域の削除

**図 6**　配置領域の操作方法

ように次に埋めるべき BL 領域が C の領域に決まったときに，採用された領域と共に干渉する A・B の BL 領域も未使用リストから削除する．また採用された製品も未使用リストから削除する．候補となる BL 領域がなくなったら，制約で許される限り新しい素板に相当する BL 領域を未使用リストに追加して配置を繰り返す．

### 3.1.2　評価関数とその最適化

BL 領域と製品の組合せを評価する項目の概念図を図 7 に示す．今回の評価関数は BL 領域に対して占有率が高かったり，エッジや欠点に近かったり，製品と余白のアスペクト方向の相性によって高スコアとなるように設計しており，主に七つの項目で評価し，式 (1) のように線形和を計算する．

$$\text{評価関数} = \sum_{i=1}^{7} w_i \times value_i \qquad (1)$$

ここで，$w_i$ は各項目にかける重み，$value_i$ は各評価項目の値を示す．ただし，素板のサイズや製品のサイズ，枚数によって最適な重みは変わることが予想される．そこで本アルゴリズムでは，計算時間が許す限り配置を繰り返して評価関数の重み $w_i$ を粒子群最適化（Particle Swarm Optimization: PSO）法で最適化している．PSO 法を用いる際の評価尺度は全体の歩留まりとしている．

## 3.2　大アイテム配置-ビンパッキング法

サイズの大きい板が残ってくると，歩留まりが低いパッキングしかできなくなる．ここでは，大きい板を優先して，パッキングを行っていく．また，単独で大きい板があればよいが，大きめの板同士を組み合わせて，大きい板のセットを作ることにより，大きい板を模倣する戦略も取る．大きい板がない場合は，3.1 節のアルゴリズムのみで計算を行う．3.1 節のアルゴリズムの前に本アルゴリズムを前処理機能として働かせ

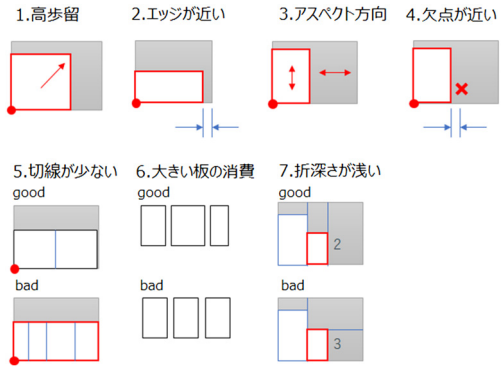



**図 7**　評価関数の重みのつけ方

たものを 3.1 節のアルゴリズム単体と並列させることにより，大きい板が多数存在する問題においても，高歩留まりを得やすくしている．

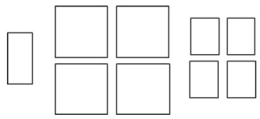
### 3.2.1　大きい板の作成

配置の後半において，面積が大きい板が数多く残っている場合，歩留まりを大きく下げる要因になることが多い．序盤の素板の歩留まり向上を優先させて，小回りの効く面積が小さい板が初期に優先的に使用され，面積の大きい板が多数使われずに最後まで残り，終盤の素板の歩留まりが悪くなることはよく起こる．そこで，図 8 で示すように，事前にサイズの大きな類似した製品板を結合して，一つの大きなセットを作って配置する戦略をとっている．大きなセットは，一定の面積より大きい板を対象に縦もしくは横に結合して作成した．

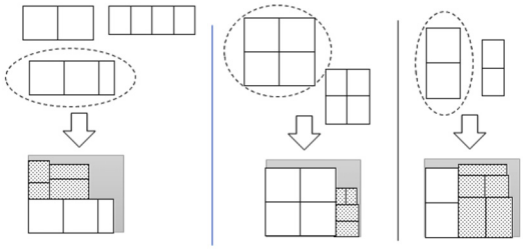
### 3.2.2　大きなセットと余白部の配置

前節で作った大きなセットのみで，歩留まり 90%を超えるパッキングができれば，その素板のパッキングは完成とする．しかし，大きいセットのみでなかなか高歩留まりを得るのは難しい．通常の手順として，図 8(b) に示すように，縦もしくは横方向や段積み基準として作成した大きいセットのそれぞれで一番大きいものを選び，素板原点 (BL) に配置する．配置すると，右側



(a) 大きい板候補



(b) 候補の板を組み合わせ，一番大きい板を選び，残りを埋める．その中で一番歩留まりがよいものを採用

**図 8**　大アイテム配置-ビンパッキングの概要

と上側に余白ができる．それぞれの余白において，ビンパッキングとナップサック問題を段階的に解いて埋めていく．1 段階目は，余白の水平方向の寸法を基準にし，高さが同じ未使用製品をアイテムとして，ビンパッキング問題として解く．2 段階目は，1 段階目で選ばれた製品をアイテムとして，余白の垂直方向の寸法を基準にナップサック問題を解き，余白に入るアイテムを選ぶ．最終的に，歩留まりが良いものを採用する．

### 3.3　後処理

3.1 節，3.2 節のアルゴリズムでは，製品が割り当てられず廃棄される領域の総面積が最小になるように，板と切線の配置を最適化する問題を解いている．一方で，素板切断工程における別のコスト要因として，以下の二つのことを考える必要がある．

- 切線の修正作業や折り作業の複雑化によるタクトタイム増大
- 作業ミスや運搬時の割れによるロス増大

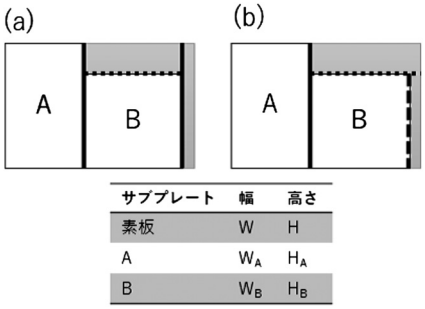
できるだけ単純で折りやすく，再現性のある切線パターンにすることにより，作業ミスが減り，タクトタイムも短縮されることが期待できる．そこで，次のルールに従うよう切線と製品配置の修正を行うアルゴリズムを開発した．

#### 3.3.1　切線ルール

切線を決定する際に考慮することを以下に列挙する．

1) 余白領域を折りやすくすること
2) 素板の端から伸び，素板を貫通しない切線を禁止
3) 充填率が低い場合は，原点の対角側の余白領域面積を最大化すること

1) は，余白領域が細かく切断されると折り失敗のリスクが高まるため，それを防止するためのルールである．基本的に余白部の切線方向の長さが短く，折り代長さが長いほど折り失敗のリスクが低下することがわかっているので，折り作業を行う際の折りにくさの指標 Difficulty ($Diff$) を，以下のように定義した．




| サブプレート | 幅 | 高さ |
|---|---|---|
| 素板 | W | H |
| A | $W_A$ | $H_A$ |
| B | $W_B$ | $H_B$ |

**図 9**　2 枚の製品で構成するサブプレートの例と考えられる切線パターン

$$Diff = \frac{\text{切線方向の長さ}}{\text{折り代の長さ}}$$

$Diff$ が小さいと折りやすいので，サブプレート（製品板が配置されている素板より小さい領域）単位で最も折りやすく切線を決定する最適化問題を，この $Diff$ の総和の最小化問題として定義した．例として図 9 のようなサブプレートの切線を決めたい．図 9 で B のアイテムを切り出すためには，図 9(a) と図 9(b) の 2 種類の切線パターンが考えられるが，$Diff$ を比較すると，

$$Diff_{(a)} = \frac{H}{W - W_A - W_B} + \frac{W_B}{H - H_B} \quad (2)$$

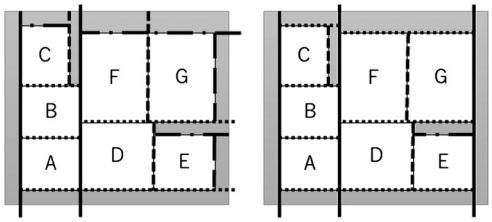
$$Diff_{(b)} = \frac{H_B}{W - W_A - W_B} + \frac{W - W_A}{H - H_B} \quad (3)$$

となり，$Diff_{(a)}$ と $Diff_{(b)}$ の差は，

$$\begin{aligned} Diff_{(a)} - Diff_{(b)} &= \frac{H - H_B}{W - W_A - W_B} - \frac{W - W_A - W_B}{H - H_B} \\ &\propto (H - H_B)^2 - (W - W_A - W_B)^2 \end{aligned} \quad (4)$$

である．$H - H_B$，$W - W_A - W_B$ はそれぞれ折り代の長さなので，折りやすさを考慮する場合はこれらを比較して選択すれば良い．このケースでは，$H - H_B > W - W_A - W_B$ なので $Diff_{(a)} > Diff_{(b)}$ となり，(b) の配置の方が折りやすいと判断できる．

2) は，搬送時の振動による割れを防ぐために導入されたルールである．割れは素板の端部に掛かっている切線から発生し，端から伸びる切線が途中で止まっていると，製品部分まで割れてしまうことがあ




(a) 折りやすさ優先時　(b) 搬送割れ防止優先時

**図 10**　切線の入れ方

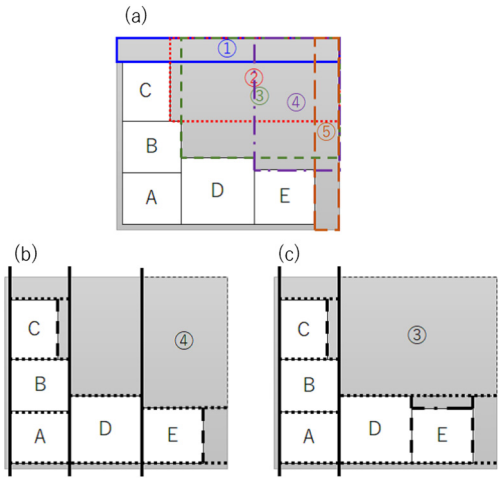
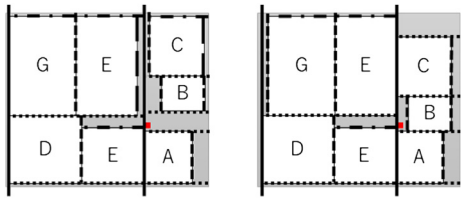



**図 11**　余白部分の取り方

る．これを防ぐために，端から伸びる貫通しない（深さ 2 以上の）切線をできるだけ少なくするような切線ルールを加えた．まず，1) のルールにより折りやすさのみを考慮して切線を決定すると，図 10(a) のような切線となる．折りやすさを優先する場合，余白領域をできるだけ通過させて細切れにするため，端にかかる深さ 2 以上の切線が 6 本出現している．一方で，2) を考慮すると図 10(b) のような切線となり，ここでは割れリスクのある切線は 1 本も出現しない．

3) は，特に最終素板においては製品の充填率が低くなるため，その場合は余白領域を再利用したい．配置最適化のアルゴリズムでは欠点がない場合は原点側に製品が偏るので，右上の領域に余白が大きく発生する．この性質を利用して，「原点と対角側の頂点を含む余白領域を最大化」する問題として定義し，全探索による最適化を行った．

たとえば，図 11(a) のような製品配置の場合，候補となる余白領域は①〜⑤の 5 種類が存在する．面積の大きい③・②・④・①・⑤の順にダミー製品板を割り当てて切線最適化を適用し，実行可能な解が得られたらその時点で計算を終了する．何も処理せずに折りやすいような切線を決定すると図 11(b) のように④の余白領域ができるが，余白最大化を行うと，図 11(c) のように③の領域を余白として，再利用性を大きく向上できることがわかる．




(a) 中耳削減前　(b) 中耳削減後

**図 12**　配置修正（■は欠点）

### 3.3.2　製品配置修正ルール

3.1 節，3.2 節のアルゴリズムでは，折り代確保制約を満たした配置が得られるが，余材が素板の中心部などに存在すると，切線が複雑になったり，作業者が作業しづらい（このような余材を中耳と呼ぶ）．そこで，与えられた配置データから，①中耳の有無の判定，②中耳の移動の可否判定，③中耳を端に移動，④切線の修正という手順で中耳削減操作を行った．中耳にはさまざまな発生パターンがあり，そのほかの制約条件によっても対処法が変わるため，ルールベースでパターンごとに対処を行うよう実装した．図 12 に中耳削減パターンの一例と修正後の配置を示す．

## 4.　評価

今回開発したアルゴリズムと，実稼働している既存のアルゴリズムとの性能比較を実施した．

### 4.1　評価データ

評価データとして，25 インスタンス分のデータを用いた．各インスタンスの計算条件と最適化結果のスコアを表 1 に示す．すべて実際に社内の現場で板取り計算をしたときの条件である．今回検証した計算環境は以下のとおりである．

- OS: Windows 10 Pro 64 bit
- CPU: Intel® Core™ i7-8550U
- メモリ：16.0 GB
- 並列スレッド数：4
- 開発言語：python and C++

表 1 中の *score* は充填率を表し，式 (5), (6) のように定義する．

未配置の製品が残っている場合：

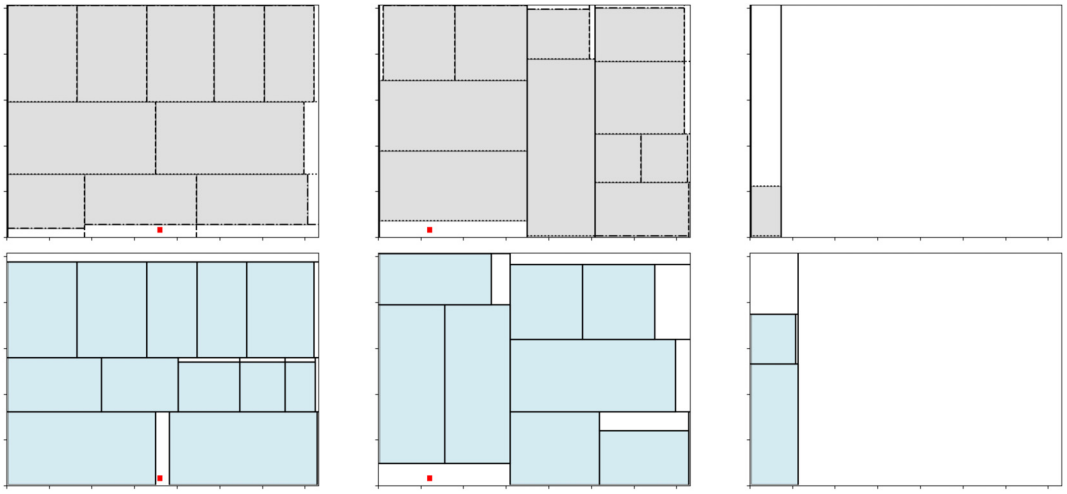


**図 13**　検証データ No.3 の板配置（上段：開発，下段：既存）
■は欠点

$$score = \frac{\text{配置した製品の面積}}{n \times \text{素板面積}} \tag{5}$$

製品をすべて素板に配置した場合：

$$score = \frac{\text{配置した製品の面積}}{(n-1) \times \text{素板面積} + \text{最終板に配置した製品面積}} \tag{6}$$

ここで，$n$ は素板の総枚数である．素板の枚数が十分でないとき，すべての素板に配置しても未配置の製品が残る場合がある．このとき，平均歩留まりは式 (5) のように定義できる．一方で素板の枚数が十分大きいとき，製品をすべて素板に配置することができる．この場合，最終素板の充填率は低くなるが，一般に最終素板の余白は再利用されるため，歩留まり低下に直接つながるわけではない．そこで，最終素板のみ余白をすべて再利用できるという前提に基づき，式 (6) のように定義した．

図 13 に表 1 のインスタンス番号 No. 3 の配置結果を示す．本インスタンスは，素板内に欠点が存在しているものである．開発アルゴリズムも既存アルゴリズムも欠点をしっかりと避けて，アイテムを配置できていることがわかる．最終素板に着目すると，今回開発したアルゴリズムの方が，小さい製品を一つだけ配置しており，既存に比べて良い配置ができていることがわかる．表 1 では *score* のより高い結果を太文字で示している．開発したアルゴリズムは既存のアルゴリズムに対し，多くの問題で同等以上のスコアを得ることができた．特に折り深さ制限が 6 のインスタンスでは，

**表 1**　検証データによる評価結果

| No. | 折深さ | 欠点 | 開発 *score* | 既存 *score* |
|---|---|---|---|---|
| 1 | 6 | | **0.949** | 0.944 |
| 2 | 6 | | **0.946** | 0.93 |
| 3 | 6 | ○ | **0.923** | 0.884 |
| 4 | 6 | | 1 | 1 |
| 5 | 6 | ○ | 0.922 | 0.922 |
| 6 | 6 | | 0.925 | 0.925 |
| 7 | 6 | ○ | **0.882** | 0.851 |
| 8 | 6 | | 1 | 1 |
| 9 | 6 | ○ | **0.905** | 0.887 |
| 10 | 6 | | **0.94** | 0.891 |
| 11 | 6 | ○ | **0.954** | 0.924 |
| 12 | 6 | ○ | **0.906** | 0.886 |
| 13 | 6 | | 1 | 1 |
| 14 | 6 | | **0.883** | 0.857 |
| 15 | 6 | ○ | **0.944** | 0.921 |
| 16 | 4 | ○ | 0.904 | **0.923** |
| 17 | 4 | ○ | 0.84 | **0.909** |
| 18 | 4 | | 0.868 | 0.868 |
| 19 | 4 | | 0.881 | 0.881 |
| 20 | 4 | | 0.869 | **0.902** |
| 21 | 6 | ○ | **0.925** | 0.915 |
| 22 | 6 | | 1 | 1 |
| 23 | 6 | | 1 | 1 |
| 24 | 6 | | 0.746 | 0.746 |
| 25 | 6 | | 0.834 | **0.862** |

ほとんどのケースで同等以上の性能を示した．一方，深さ制限が 4 のインスタンスでは，すべてのインスタンスで既存のアルゴリズムと同等以下の性能であった．深さ上限が大きいとき，すなわち解の探索空間が広いときに開発アルゴリズムはより高性能を示し，逆に探

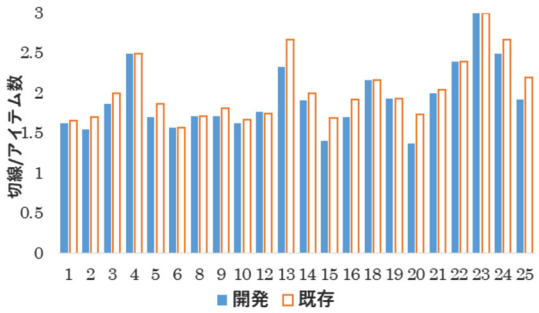



**図 14**　配置したアイテム 1 枚当たりの切線の数

索空間が狭い問題では高性能を得にくい性質があると考えられる．

図 14 に切線の総数を配置したアイテム数で割ったアイテム 1 枚当たりの切線の数を比較したグラフを示す．各インスタンスで，開発アルゴリズムの方が既存アルゴリズムと比べて小さくなっているところが多くある．折りやすさや最終板の余白を大きく取るといった作業者が作業しやすいように切線の配置をしてきたが，中耳削減も加わることで，既存アルゴリズムより配置アイテム数に対する切線の数も改善ができた．切線の数の削減は，作業時間に関係してくるため，歩留まり性能を維持しながら，作業の効率化の手助けも行えるアルゴリズムが開発できたと思われる．

## 5.　結言

NP 困難に分類される板取り問題において，ギロチンカット制約や欠点避けのみならず，切り機依存による現場の細かい制約を満たしたエンジン開発を行った．①評価関数の重み最適化を内部ルーチンとした貪欲法と，②サイズの大きい板をグループ化して優先配置する手法を並列計算し，歩留まりの高い結果を選択する方式を採用した．また，作業性を考慮して製品や切線の配置を修正するポスト処理機能を開発した．

サンプルデータによるベンチマークテストを行い，開発アルゴリズムは大部分のインスタンスで既存アルゴリズム以上の歩留まりを得た．切線の総数の削減も行え，タクトタイムの削減を期待できる．一方で，折深さが 4 以下という条件では既存に劣る結果が得られることもあった．改善のためにはより狭い範囲を隈なく探索するアルゴリズムが望ましいと考えられる．

## 参考文献

[1] M. Iori, V. L. de Lima, S. Martello, F. K. Miyazawa and M. Monaci, “Exact solution techniques for two-dimensional cutting and packing,” *European Journal of Operational Research*, **289**, pp. 399–415, 2021.
[2] R. Alvarez-Valdes, R. Martí, A. Parajón and J. M. Tamarit, “GRASP and path relinking for the two-dimensional two-stage cutting-stock problem,” *Informs Journal on Computing*, **19**, pp. 261–272, 2007.
[3] J. Puchinger and G. R. Raidl, “Models and algorithms for three-stage two-dimensional bin packing,” *European Journal of Operational Research*, **183**, pp. 1304–1327, 2007.