数理解析研究所講究録
第1931巻 2015年 129-147

Sequence-Tripleを用いた3次元配置問題に対する局所探索法

名古屋大学大学院・工学研究科　岩澤 宏紀 (Hiroki Iwasawa)
今堀 慎治 (Shinji Imahori)
Graduate School of Engineering,
Nagoya University

概要

3次元配置問題とは，様々な大きさをした複数の直方体を互いに重ならないように容器に配置する問題の総称であり，配置制約や目的関数により様々な種類の問題を含んでいる．本研究では，直方体を配置する容器の体積の最小化を目的とする体積最小化問題を扱う．コンテナへの荷物の積み付けでは，荷物をできるだけ無駄なく配置することでより多くの荷物を運ぶことができる．また，近年ではVLSIの素子配置が3次元化してきており，ICチップの体積を小さくすることでコスト削減が実現できる．本研究で用いるSequence-Tripleは，2次元における長方形配置の表現法であるSequence-Pairを拡張した方法であり，重なりのない直方体配置を三つの順列で表すことができる．本研究ではSequence-Tripleと反復局所探索法を組み合わせ，近傍解を効率よく評価する手法を提案する．数値実験により，解を高速に評価することで従来と同程度の時間で高精度な解が得られることを示す．

1　はじめに

配置問題とは，いくつかの対象物を互いに重ならないように与えられた領域内に配置する問題であり，多くの分野に応用を持つ生産計画問題の一つである．この問題は，対象物や領域の次元，形状，配置制約，目的関数等により非常に多くのバリエーションをもつ．代表的な問題として，幾何学や組合せ最適化の分野で古くから研究されている長方形配置問題がある．長方形配置問題は集積回路設計における素子配置を決定する問題や，鉄鋼・繊維産業において，大きな鉄板や布から製品に切り分ける問題といった，実用的な問題とも密接に関わりを持つ．長方形配置問題の代表的な解法の一つにSequence-Pairがある [8]．Sequence-Pairとは，長方形名を並べた二つの順列を用いて長方形間の相対位置を定めることにより，長方形配置を表現する手法である．この手法は実現可能なあらゆる配置を表現でき，長方形を配置する母材の面積を最小化する問題に有用である．順列から配置を求めるには，単純に実装すると$O(n^2)$時間が必要であるが，様々な効率的解法 [3, 9, 11] が提案されており，$O(n \log n)$時間や，$O(n \log \log n)$時間で計算できることが知られている．一方でYamazakiらは，この手法を3次元へ拡張し，順列を三本用いて3次元配置問題に適応したSequence-Tripleを提案した [13]．3次元配置問題とは，直方体の容器に幅，高さ，奥行をもつ複数の直方体を詰め込む問題の総称であり，トラックやコンテナに荷物を詰め込む問題や，近年では集積回路の3次元配置に応用される [10]．Sequence-Tripleはあらゆる3次元配置を表現できるわけではないが，直方体を配置する容器の体積を最小化する問題に有用であり，小規模な問題においてある程度精度の良い解が得られることがわかっている．本研究では体積最小化問題を扱い，Sequence-Tripleを用いた反復局所探索法を試みる．更に，文献 [4] 及び文献 [5] の2次元の問題に対する効率的な近傍評価法を新たに3次元へ拡張する．最後に，提案法の有効性を検証するため，数値実験を通して既存手法との比較を行う．

2 3次元配置問題

3次元配置問題は，配置制約や目的関数の異なる複数の問題の総称である．本研究では体積最小化問題と呼ばれる問題を考える．体積最小化問題の入力は，直方体の容器の幅 W，高さ H，奥行 D(全て可変) 及び直方体集合 $I = \{1, 2, \ldots, n\}$ に含まれる各直方体 $i \in I$ の幅 w_i，高さ h_i，奥行 d_i である．問題の目的は，全ての直方体を互いに重ならないように一つの容器に配置し，容器の体積 WHD を最小化することである．座標の x, y, z 軸はそれぞれ容器の幅，高さ，奥行に対応するとし，x 座標の小さい方を左側，y 座標の小さい方を下側，z 座標の小さい方を前側とする．各直方体 i(の最も左，下，前) の座標を (x_i, y_i, z_i) としたとき，体積最小化問題は以下のように定義される．

目的関数　：WHD →最小化.

制約条件 1 ：各直方体 $i \in I$ は容器内に配置される．これは次の三つの不等式を満たすことと等価である．

$$0 \leq x_i \leq W - w_i, \tag{1}$$

$$0 \leq y_i \leq H - h_i, \tag{2}$$

$$0 \leq z_i \leq D - d_i. \tag{3}$$

制約条件 2 ：各直方体 $i, j \in I$ は重ならない．これは次の六つの不等式のうち一つ以上が成立することと等価である．

$$\begin{aligned} x_i + w_i \leq x_j, \qquad & x_j + w_j \leq x_i, \\ y_i + h_i \leq y_j, \qquad & y_j + h_j \leq y_i, \\ z_i + d_i \leq z_j, \qquad & z_j + d_j \leq z_i. \end{aligned} \tag{4}$$

この問題は NP 困難であることが知られており，様々な近似解法が提案されてきた [1, 2, 6, 7].

3 Sequence-Triple

Sequence-Triple とは，2次元の長方形配置問題に対する解法である Sequence-Pair[8] を3次元へ拡張した方法である．Sequence-Pair は 1996 年に Murata らにより提案された手法で，長方形名を並べた二つの順列を用いて，長方形間の相対位置関係を割り当てることで長方形配置を表現する手法である．この手法に対して，Yamazaki らは 2000 年に順列を三つ用いて3次元配置を表現する Sequence-Triple を提案した [13]．Sequence-Triple では直方体名を並べた三つの順列 $\sigma_1, \sigma_2, \sigma_3$ 内での直方体の並び順に対して上下，左右，前後の位置関係を割り当てる．直方体 i, j に着目したとき，三つの順列内での i, j の並び順によって次のルールに従い相対位置が決定される．ここで順列 σ 内で l 番目の直方体 i に対して $\sigma(l) = i$，$\sigma^{-1}(i) = l$ と表す．

1. $\{\sigma_2^{-1}(i) < \sigma_2^{-1}(j)\} \wedge \{\sigma_3^{-1}(i) > \sigma_3^{-1}(j)\}$ のとき，i は j の右に配置する．
2. $\{\sigma_1^{-1}(i) < \sigma_1^{-1}(j)\} \wedge \{\sigma_2^{-1}(i) > \sigma_2^{-1}(j)\} \wedge \{\sigma_3^{-1}(i) > \sigma_3^{-1}(j)\}$ のとき，i は j の上に配置する．
3. $\{\sigma_1^{-1}(i) > \sigma_1^{-1}(j)\} \wedge \{\sigma_2^{-1}(i) > \sigma_2^{-1}(j)\} \wedge \{\sigma_3^{-1}(i) > \sigma_3^{-1}(j)\}$ のとき，i は j の奥に配置する．

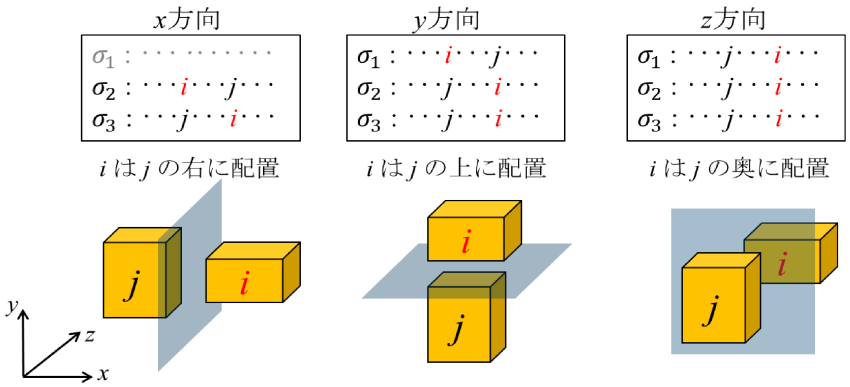

図 1: 配置ルール

図 1 に順列と配置の関係を示す.

2 次元配置を表現する Sequence-Pair では，実現可能なあらゆる配置を表現可能であったが，Sequence-Triple では表現できない 3 次元配置が存在することが知られている．文献 [13] ではこれに対して，あらゆる 3 次元配置を表現可能な手法として五つの順列における並び順によって位置関係を割り当てる Sequence-Quintuple を提案している．しかし，Sequence-Quintuple を用いると解空間が膨大となり，実用的な時間では Sequence-Triple のほうが精度の良い解が得られることが報告されている [13].

3.1 配置アルゴリズム

Sequence-Triple が表すのは直方体間の相対的な位置関係であり，位置関係を満たした上で全直方体を配置する容器の体積が最小となる配置を求める必要がある．相対位置関係を満たしつつ，容器の上下左右前後のいずれかの方向に全直方体を出来るだけ寄せたとき，容器の各方向の長さが最小の配置となる．従って，全直方体を容器のいずれかの頂点を基準に出来るだけ寄せた配置が，体積最小の配置となる．このとき各直方体の x，y，z 座標はそれぞれ独立に計算することができる．例えば容器の最も左，下，前の頂点を基準としたときの直方体 i(の最も左，下，前) の座標 (x_i,y_i,z_i) は以下で表される.

$$x_i = \begin{cases} 0 & (J_i^x = \emptyset) \\ \max_{j \in J_i^x}(x_j + w_j) & (J_i^x \neq \emptyset), \end{cases} \tag{5}$$

$$y_i = \begin{cases} 0 & (J_i^y = \emptyset) \\ \max_{j \in J_i^y}(y_j + h_j) & (J_i^y \neq \emptyset), \end{cases} \tag{6}$$

$$z_i = \begin{cases} 0 & (J_i^z = \emptyset) \\ \max_{j \in J_i^z}(z_j + d_j) & (J_i^z \neq \emptyset). \end{cases} \tag{7}$$

ただし，J_i^x は i より左にある直方体の集合，J_i^y は i より下にある直方体の集合，J_i^z は i より手前にある直方体の集合を表す．これらの座標は順列 σ_3 の前から順に座標を決定していくことで，全ての直方体の座標を $O(n^2)$ 時間で計算できる.

容器の最も右，上，奥の頂点を基準に寄せた配置の直方体 i の右，上，奥の座標 $(x_i^{(R)}, y_i^{(T)}, z_i^{(B)})$ は次のように表すことができ，順列 σ_3 の後ろから順に座標を決定していくことで $O(n^2)$ 時間で全ての座標を計算できる．ただし，ここでの座標は容器の最も右，上，奥の頂点を原点とし，x，y，z 軸は左，下，前方向を正の向きとする．図 2 に三本の順列と順列に対応した配置及び，座標軸の取り方を示す．

$$x_i^{(R)} = \begin{cases} 0 & (J_i^{x^{(R)}} = \emptyset) \\ \max_{j \in J_i^{x^{(R)}}} (x_j^{(R)} + w_j) & (J_i^{x^{(R)}} \neq \emptyset), \end{cases} \tag{8}$$

$$y_i^{(T)} = \begin{cases} 0 & (J_i^{y^{(T)}} = \emptyset) \\ \max_{j \in J_i^{y^{(T)}}} (y_j^{(T)} + h_j) & (J_i^{y^{(T)}} \neq \emptyset), \end{cases} \tag{9}$$

$$z_i^{(B)} = \begin{cases} 0 & (J_i^{z^{(B)}} = \emptyset) \\ \max_{j \in J_i^{z^{(B)}}} (z_j^{(B)} + d_j) & (J_i^{z^{(B)}} \neq \emptyset). \end{cases} \tag{10}$$

ただし，$J_i^{x^{(R)}}$ は i より右にある直方体の集合，$J_i^{y^{(T)}}$ は i より上にある直方体の集合，$J_i^{z^{(B)}}$ は i より奥にある直方体の集合を表す．

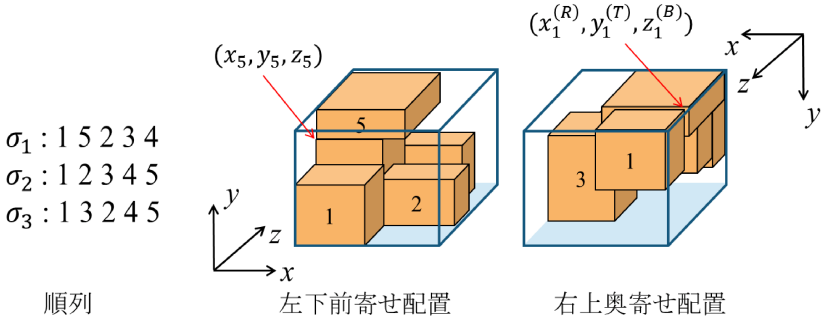

図 2: Sequence-Triple から得られる配置

文献 [3] では Sequence-Pair を用いた 2 次元の配置アルゴリズムに対し，二分探索木データ構造を用いることで $O(n \log n)$ 時間で計算する手法が提案されているが，この手法を 3 次元へ拡張することができる (詳細は付録 A に示す)．このアルゴルズムの計算量は $O(n \log^2 n)$ 時間であり，理論的には $O(n^2)$ 時間より高速である．しかし係数や定数項が大きく，複雑なデータ構造を用いているため，本研究の数値実験では $O(n^2)$ 時間で配置する手法を用いる．

3.2 クリティカルパス

Sequence-Triple から得られる配置の特性の一つにクリティカルパスがある．クリティカルパスとは，各方向において位置が制約されている直方体の連なりである．三本の順列により x 方向の関係 $(\{\sigma_2^{-1}(j) < \sigma_2^{-1}(i)\} \wedge \{\sigma_3^{-1}(j) > \sigma_3^{-1}(i)\})$ にある直方体群のうち，

$$x_i + w_i = x_j \tag{11}$$

を満たす直方体の連なりを x 方向のパスと呼ぶ．同様に y 方向，z 方向の関係にある直方体群に関してもそれぞれ

$$y_i + h_i = y_j, \tag{12}$$

$$z_i + d_i = z_j \tag{13}$$

を満たす直方体の連なりを y 方向，z 方向のパスと呼ぶ．各方向においてパスに含まれる直方体の長さの総和をそのパスの長さとしたとき，パスの長さが最長であるパスがその方向におけるクリティカルパスである．クリティカルパスに含まれる直方体は三本の順列に対応する体積最小の配置において，各方向の座標が一意に定まる．それぞれの方向には必ず一本以上のクリティカルパスが存在し，全直方体を配置する容器の大きさ W，H，D は各方向のクリティカルパスの長さに一致する．従って，クリティカルパスに含まれる直方体の位置関係を変えない限り容器の大きさが小さくなることはない．図 3 にクリティカルパスの例を示す．図 3 から各方向のクリティカルパスに含まれる直方体は，左下前と右上奥のどちらに寄せた配置においても，それぞれの方向に関して同じ位置にあることがわかる．

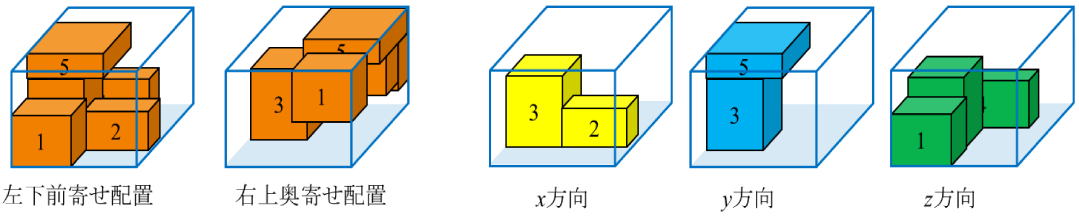

図 3: クリティカルパス

4　アルゴリズム

この章では本研究で用いるアルゴリズムについて述べる．

4.1　局所探索法

局所探索法とは，現在の解から少し変化させて得られる解の集合 (近傍) 内で，より良い解が見つかれば現在の解と置き換えるという操作を近傍内に改善解がなくなるまで繰り返す方法である．最終的に得られる解を局所最適解と呼ぶ．局所探索法の動作を定めるには，初期解の生成法，近傍の定義，解の評価法，移動戦略などを決める必要がある．次節以降では本研究における局所探索法について説明する．

4.2　近傍

Sequence-Triple を用いた局所探索法では，三つの順列を変化させることで近傍を定義する．本研究ではシフト近傍とスワップ近傍を用いる．

4.2.1 シフト近傍

シフト近傍とは，一つの直方体 i を選び，順列内で i を移動させて得られる解の集合である．シフト操作は実際の配置において，直方体 i を別の場所へ移動させる効果がある．また移動させる順列の数により，近傍のサイズを調節することができる．本研究では選択した直方体を二つの順列で移動させるダブルシフト近傍を用いる．図 4 に直方体 1 に関してダブルシフト操作を行った例を示す．ダブルシフト近傍のサイズは $O(n^3)$ である．以後，特に記述がない場合，ダブルシフト近傍のことを単にシフト近傍と記す．

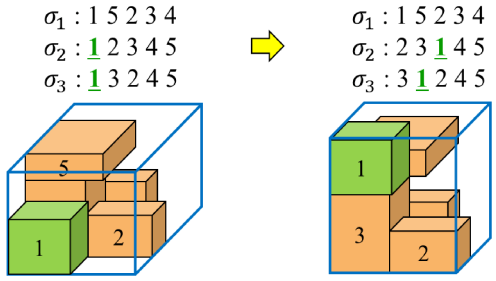

図 4: ダブルシフト操作

4.2.2 スワップ近傍

スワップ近傍とは，二つの直方体 i と j を選び，順列内で i と j を入れ替えて得られる解の集合である．スワップ近傍に関しても入れ替える順列の数により異なる近傍を定義できるが，本研究では三つの順列で入れ替えるトリプルスワップ近傍を用いる．トリプルスワップ操作は直方体 i と j に関する相対位置関係が全て入れ替わるため，実際の配置では，i と j の場所を入れ替える効果がある．図 5 に直方体 1 と 5 に関してトリプルスワップ操作を行った例を示す．トリプルスワップ近傍のサイズは $O(n^2)$ である．以後，トリプルスワップ近傍のことを単にスワップ近傍と記す．

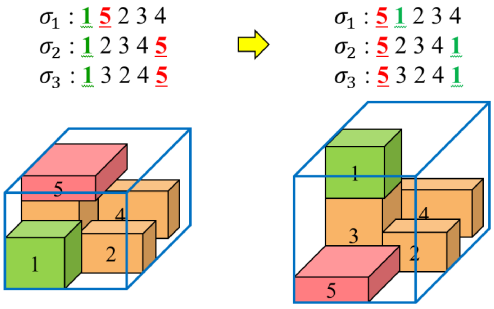

図 5: トリプルスワップ操作

これらの近傍を探索する際，クリティカルパスに含まれない直方体のみを移動しても改善されない (つまり，容器の体積が減少しない) ため，移動する直方体 (の一つ以上) をクリティカルパスに含まれる直方体に限定することで効率的な探索を行うことができる．

4.3 評価基準と移動戦略

解の評価基準には以下の二つを考える．

1. 充填率が高い解を改善解とする．

   充填率は以下の式 (14) で定義され，容器に対する直方体の割合を表す．充填率が高い解ほど直方体を配置する容器の体積 WHD が小さいことを意味する．

$$\text{充填率}\,[\%] = \frac{\text{全直方体の体積}\Sigma_{i=1}^{n} w_i h_i d_i}{\text{容器の体積}\,WHD} \times 100. \tag{14}$$

2. 充填率が同点のとき，クリティカルパスに含まれる直方体の数が少ない解を改善解とする．

   充填率の高い配置ではクリティカルパスが複数並列して存在することが多く，一つのクリティカルパスを崩しても容器の体積を小さくすることが難しい．クリティカルパスに含まれる直方体の数が減る解に移動していくことで，クリティカルパスを減らし，充填率が向上しやすい配置を作ることができる．図 6 にクリティカルパスを崩して改善する例を示す．図 6 では，わかりやすくするために 2 次元の図を用いているが，3 次元においても同じことが言える．図 6 の下の例では二回の移動に渡ってクリティカルパスを崩すことで容器の体積 (面積) を改善している．

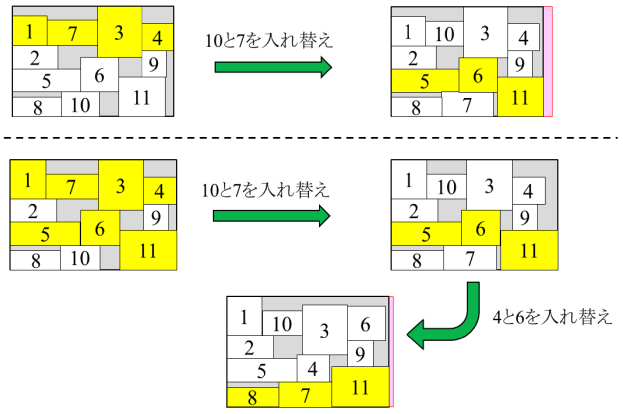

図 6: クリティカルパスを崩す操作を複数回適用して改善する例

近傍内の探索順序は，まず近傍サイズの小さいスワップ近傍を探索する．スワップ近傍では，近傍解の中で最も充填率が向上する解へ移動する最良移動戦略を用いる．スワップ近傍に改善解がなくなったら次にシフト近傍を探索する．シフト近傍ではクリティカルパスに含まれる直方体を一つ選んだ上で，その直方体を順列内で移動させて得られる全ての解を評価し，改善解が存在する場合はその中の最良の解へ移動する．この手続きをクリティカルパスに含まれる直方体に順次適用し，解の移動を繰り返す．シフト近傍に改善解がなくなったら再びスワップ近傍の探索に戻る．シフト近傍，スワップ近傍のどちらにも改善解がないとき，現在の解を局所最適解として出力する．

4.4 反復局所探索法

局所探索法は高速に計算することができるが，必ずしも満足のいく精度の解が得られるとは限らない．これに対して，多少時間はかかってもより精度の高い解を求める解法の一般的な枠組みにメタ戦略がある．メタ戦略は局所探索法の一般化と捉えることができ，多くの組合せ最適化問題に対して成功をおさめている．文献 [12] ではメタ戦略の様々な手法が紹介されている．

本研究では，メタ戦略として局所探索法を拡張した反復局所探索法を用いる．反復局所探索法とは，局所探索法で得られた局所最適解に対して，通常の近傍操作とは異なる変形を加えることで新たに初期解を生成し，繰り返し局所探索法を行う手法である．本研究で新たな初期解を生成する際は，複数回ランダムに選んだ二つの直方体に対してスワップ操作を行う．

5 近傍解の高速評価法

この章では本研究で提案する，シフト近傍，スワップ近傍の効率的な評価法について述べる．5.1 節でトリプルシフト近傍の高速評価法について述べ，5.2 節で 5.1 節の手法をダブルシフト近傍に適応させる．5.3 節でスワップ近傍の高速評価法について説明し，5.4 節でスワップ操作と同時に直方体を回転する解の高速評価法を提案する．

5.1 トリプルシフト近傍の高速評価

この節では文献 [4] で提案されている 2 次元の長方形配置問題における効率的なシフト近傍の評価法を 3 次元へ拡張する方法について述べる．この方法は一つの解を評価する際に，$O(n\log^2 n)$ 時間で全ての直方体の座標を求める代わりに，容器の各辺の長さを決定するクリティカルパスの長さのみを高速に計算する．

「直方体 i をシフトする」という操作は「i を取り除く」+「i を異なる場所に追加する」という操作に置き換えることができる．Sequence-Triple において，i を取り除く，追加するという操作を行っても i 以外の直方体間の相対位置関係は不変であり，順列から i を取り除くことで i に依存しない配置を求めることが出来る．直方体 i をシフトした (異なる場所に追加した) 後の各方向のクリティカルパスは i を含む場合と含まない場合がある．例えば x 方向に関して考えると，i を含まない場合は i が容器の幅には関与しないため，i を取り除いた配置の幅 $\tilde{W}$ がシフト後の幅 W' となる．i を含む場合は i を取り除いた配置の情報を用いることで，シフト後の i を含む x 方向のクリティカルパスの長さを計算することができる．y 方向，z 方向に関しても同様のことが言える．以下にシフト後の各方向のクリティカルパスが i を含む場合の計算方法を示す．ここで直方体 i を抜いた直方体集合を $\tilde{I}$，順列を $\tilde{\sigma}$ とする．i を取り除いた配置における直方体 j の左寄せ座標を $\tilde{x}_j$，右寄せ座標を $\tilde{x}_j^{(R)}$，下寄せ座標を $\tilde{y}_j$，上寄せ座標を $\tilde{y}_j^{(T)}$ とし，これらの値は予め $O(n\log^2 n)$ 時間で配置を求めて計算しておくとする．

- x 方向

  x 方向の位置関係は二つの順列 σ_2 と σ_3 のみで決定するので，2 次元の Sequence-Pair に対する手法を自然に用いることができる．i を $\tilde{\sigma}_1$ の α 番目，$\tilde{\sigma}_2$ の β 番目，$\tilde{\sigma}_3$ の γ 番目に追加したときの容器の左端から i の左面までのパスの長さ $\tilde{l}_{\alpha,\beta,\gamma}$，容器の右端から i の右面までの

パスの長さ $\tilde{r}_{\alpha,\beta,\gamma}$ は次のように書ける．

$$\tilde{l}_{\alpha,\beta,\gamma} = \begin{cases} 0 & (\beta = n \text{ or } \gamma = 1), \\ \tilde{x}_{\tilde{\sigma}_2(\beta)} + w_{\tilde{\sigma}_2(\beta)} & (\tilde{\sigma}_2(\beta) = \tilde{\sigma}_3(\gamma - 1)), \\ \max\{\tilde{l}_{\alpha,\beta+1,\gamma}, \tilde{l}_{\alpha,\beta,\gamma-1}\} & (\tilde{\sigma}_2(\beta) \neq \tilde{\sigma}_3(\gamma - 1)). \end{cases} \tag{15}$$

$$\tilde{r}_{\alpha,\beta,\gamma} = \begin{cases} 0 & (\beta = 1 \text{ or } \gamma = n), \\ \tilde{x}^{(R)}_{\tilde{\sigma}_3(\gamma)} + w_{\tilde{\sigma}_3(\gamma)} & (\tilde{\sigma}_2(\beta - 1) = \tilde{\sigma}_3(\gamma)), \\ \max\{\tilde{r}_{\alpha,\beta-1,\gamma}, \tilde{r}_{\alpha,\beta,\gamma+1}\} & (\tilde{\sigma}_2(\beta - 1) \neq \tilde{\sigma}_3(\gamma)). \end{cases} \tag{16}$$

$\tilde{l}_{\alpha,\beta,\gamma}$, $\tilde{r}_{\alpha,\beta,\gamma}$ は動的計画法に基づいて計算でき，計算は一つ当たり $O(1)$ 時間で可能である．直方体 i が x 方向のクリティパスに含まれるとき，クリティカルパスの長さは $\tilde{l}_{\alpha,\beta,\gamma} + w_i + \tilde{r}_{\alpha,\beta,\gamma}$ となる．従ってシフト後の幅 W' は以下の式で計算できる．

$$W' = \begin{cases} \tilde{W} & (\tilde{W} > \tilde{l}_{\alpha,\beta,\gamma} + w_i + \tilde{r}_{\alpha,\beta,\gamma}), \\ \tilde{l}_{\alpha,\beta,\gamma} + w_i + \tilde{r}_{\alpha,\beta,\gamma} & (\tilde{W} \leq \tilde{l}_{\alpha,\beta,\gamma} + w_i + \tilde{r}_{\alpha,\beta,\gamma}). \end{cases} \tag{17}$$

$W' = \tilde{l}_{\alpha,\beta,\gamma} + w_i + \tilde{r}_{\alpha,\beta,\gamma}$ となるとき，直方体 i が x 方向のクリティカルパスに含まれることを意味する．従ってこの評価法を用いることで，4.3 節で述べたクリティカルパスに含まれる直方体の数が減る解の評価をすることも可能である．全体の計算量は直方体 i を取り除いた配置を求めるのに $O(n \log^2 n)$ 時間，解の評価に $O(n^3)$ 時間がかかり，全近傍解の評価は $O(n^3)$ 時間で可能である．従って解一つ当たりの評価時間は $O(1)$ となり，既存の評価法 $O(n^2)$ と比較して $O(n^2)$ 倍高速である．

- y 方向

y 方向の位置関係は三つの順列により決定される．容器の下端から i の下面までのパスの長さ $\tilde{b}_{\alpha,\beta,\gamma}$，容器の上端から i の上面までのパスの長さ $\tilde{t}_{\alpha,\beta,\gamma}$ は以下の式で表すことができる．

$$\tilde{b}_{\alpha,\beta,\gamma} = \begin{cases} 0 & (\alpha = n \text{ or } \beta = 1 \text{ or } \gamma = 1), \\ \tilde{y}_{\tilde{\sigma}_1(\alpha)} + h_{\tilde{\sigma}_1(\alpha)} & (\tilde{\sigma}_1(\alpha) = \tilde{\sigma}_2(\beta - 1) = \tilde{\sigma}_3(\gamma - 1)), \\ \max\{\tilde{b}_{\alpha+1,\beta,\gamma}, \tilde{b}_{\alpha,\beta-1,\gamma}, \tilde{b}_{\alpha,\beta,\gamma-1}\} & (\text{それ以外}). \end{cases} \tag{18}$$

$$\tilde{t}_{\alpha,\beta,\gamma} = \begin{cases} 0 & (\alpha = 1 \text{ or } \beta = n \text{ or } \gamma = n), \\ \tilde{y}^{(T)}_{\tilde{\sigma}_2(\beta)} + h_{\tilde{\sigma}_2(\beta)} & (\tilde{\sigma}_1(\alpha - 1) = \tilde{\sigma}_2(\beta) = \tilde{\sigma}_3(\gamma)), \\ \max\{\tilde{t}_{\alpha-1,\beta,\gamma}, \tilde{t}_{\alpha,\beta+1,\gamma}, \tilde{t}_{\alpha,\beta,\gamma+1}\} & (\text{それ以外}). \end{cases} \tag{19}$$

x 方向と同様に，i が y 方向のクリティパスに含まれるとき，クリティカルパスの長さは $\tilde{b}_{\alpha,\beta,\gamma} + h_i + \tilde{t}_{\alpha,\beta,\gamma}$ となる．

- z 方向

z 方向に関しても y 方向と同様の考え方を用いることで計算できる．

5.2 ダブルシフト近傍の高速評価法

前節の評価法を用いることで解一つ当たり $O(1)$ 時間で計算できるが，トリプルシフト近傍のサイズはシフト操作を行う直方体を一つ選んだ上で $O(n^3)$ である．これを反復して探索するには計算量が大きいため，本研究ではダブルシフト近傍を用いる．ダブルシフト近傍の評価に関しても基本的には前節と同じ考え方を用いることができるが，一つの順列が変化しないため直方体 i がシフト操作後にクリティカルパスに含まれる場合の計算方法が少し異なる．以下に順列 σ_3 を固定し，i を $\tilde{\sigma}_1$ の α 番目，$\tilde{\sigma}_2$ の β 番目に追加したときの計算方法を示す．ここで σ_3 において i より前にある直方体集合を $J^+_{\sigma_3,i}$，後ろにある直方体集合を $J^-_{\sigma_3,i}$ と定義すると $J^+_{\sigma_3,i}$，$J^-_{\sigma_3,i}$ は以下のように書ける．

$$J^+_{\sigma_3,i} = \{j \in I \mid \sigma_3^{-1}(j) < \sigma_3^{-1}(i)\}, \tag{20}$$

$$J^-_{\sigma_3,i} = \{j \in I \mid \sigma_3^{-1}(j) > \sigma_3^{-1}(i)\}. \tag{21}$$

- x 方向

  容器の左端から i の左面までのパスの長さ $\tilde{l}^{\sigma_3}_{\alpha,\beta}$，容器の右端から i の右面までのパスの長さ $\tilde{r}^{\sigma_3}_{\alpha,\beta}$ は次のように書ける．

$$\tilde{l}^{\sigma_3}_{\alpha,\beta} = \begin{cases} 0 & (\beta = n), \\ \max\{\tilde{l}^{\sigma_3}_{\alpha,\beta+1}, \tilde{x}_{\tilde{\sigma}_2(\beta)} + w_{\tilde{\sigma}_2(\beta)}\} & (\tilde{\sigma}_2(\beta) \in J^+_{\sigma_3,i}), \\ \tilde{l}^{\sigma_3}_{\alpha,\beta+1} & (\tilde{\sigma}_2(\beta) \notin J^+_{\sigma_3,i}). \end{cases} \tag{22}$$

$$\tilde{r}^{\sigma_3}_{\alpha,\beta} = \begin{cases} 0 & (\beta = 1), \\ \max\{\tilde{r}^{\sigma_3}_{\alpha,\beta-1}, \tilde{x}^{(R)}_{\tilde{\sigma}_2(\beta-1)} + w_{\tilde{\sigma}_2(\beta-1)}\} & (\tilde{\sigma}_2(\beta-1) \in J^-_{\sigma_3,i}), \\ \tilde{r}^{\sigma_3}_{\alpha,\beta-1} & (\tilde{\sigma}_2(\beta-1) \notin J^-_{\sigma_3,i}). \end{cases} \tag{23}$$

  $\tilde{l}^{\sigma_3}_{\alpha,\beta}$，$\tilde{r}^{\sigma_3}_{\alpha,\beta}$ についてもトリプルシフト近傍の高速評価法と同様に，動的計画法に基づいて計算ができるので，解一つ当たり $O(1)$ 時間で評価できる．

- y 方向

  容器の下端から i の下面までのパスの長さ $\tilde{b}^{\sigma_3}_{\alpha,\beta}$，容器の上端から i の上面までのパスの長さ $\tilde{t}^{\sigma_3}_{\alpha,\beta}$ は以下のように書ける．

$$\tilde{b}^{\sigma_3}_{\alpha,\beta} = \begin{cases} 0 & (\alpha = n \text{ or } \beta = 1), \\ \max\{\tilde{b}^{\sigma_3}_{\alpha+1,\beta}, \tilde{b}^{\sigma_3}_{\alpha,\beta-1}, \tilde{y}_{\tilde{\sigma}_1(\alpha)} + h_{\tilde{\sigma}_1(\alpha)}\} & (\{\tilde{\sigma}_1(\alpha) \in J^-_{\sigma_3,i}\} \wedge \{\tilde{\sigma}_1(\alpha) = \tilde{\sigma}_2(\beta-1)\}), \\ \max\{\tilde{b}^{\sigma_3}_{\alpha+1,\beta}, \tilde{b}^{\sigma_3}_{\alpha,\beta-1}\} & (\{\tilde{\sigma}_1(\alpha) \notin J^-_{\sigma_3,i}\} \vee \{\tilde{\sigma}_1(\alpha) \neq \tilde{\sigma}_2(\beta-1)\}). \end{cases} \tag{24}$$

$$\tilde{t}^{\sigma_3}_{\alpha,\beta} = \begin{cases} 0 & (\alpha = 1 \text{ or } \beta = n), \\ \max\{\tilde{t}^{\sigma_3}_{\alpha-1,\beta}, \tilde{t}^{\sigma_3}_{\alpha,\beta+1}, \tilde{y}^{(T)}_{\tilde{\sigma}_2(\beta)} + h_{\tilde{\sigma}_2(\beta)}\} & (\{\tilde{\sigma}_2(\beta) \in J^+_{\sigma_3,i}\} \wedge \{\tilde{\sigma}_1(\alpha-1) = \tilde{\sigma}_2(\beta)\}), \\ \max\{\tilde{t}^{\sigma_3}_{\alpha-1,\beta}, \tilde{t}^{\sigma_3}_{\alpha,\beta+1}\} & (\{\tilde{\sigma}_2(\beta) \notin J^+_{\sigma_3,i}\} \vee \{\tilde{\sigma}_1(\alpha-1) \neq \tilde{\sigma}_2(\beta)\}). \end{cases} \tag{25}$$

- z 方向

  z 方向に関しても y 方向と同様の考え方を用いることで計算できる．

順列 σ_1，σ_2 を固定する場合や，シングルシフト近傍に関しても同じような変更を加えることで高速に評価できる．

5.3 スワップ近傍の高速評価

この節では著者らが文献 [5] で提案した，2次元におけるスワップ近傍の高速評価法を3次元に拡張する方法について述べる．スワップ近傍に関してもシフト近傍の高速評価法と同様に，直方体の座標ではなくクリティカルパスの長さを計算することで解を評価する．シフト操作では一つの直方体に関する位置関係のみが変わるため，一つの直方体を取り除いた配置を求めておくことで，高速に評価することができた．一方，スワップ操作では二つの直方体に関する位置関係が変わるため，直方体を一つ取り除いた配置だけではシフト近傍と同様の高速評価はできない．スワップ操作を行う直方体 i と j を取り除いた配置を求めておけば動的計画法による計算はできるが，$O(n^2)$ 個の解に対し $O(n^2)$ 個の配置を求めているため，計算効率は通常の評価法と変わらない．そこでスワップ近傍の高速評価法では，スワップ操作前後のクリティカルパスの変化と直方体の各辺の大小関係に着目する．

スワップ操作を行う直方体 i と j に着目すると，スワップ操作後の各方向のクリティカルパスはそれぞれ以下の4つの場合のいずれかの状態となる．

1. 直方体 i，j がどちらもクリティカルパスに含まれていない．
2. 直方体 i のみクリティカルパスに含まれている．
3. 直方体 j のみクリティカルパスに含まれている．
4. 直方体 i，j がどちらもクリティカルパスに含まれている．

実際にどの状態となるかは配置を求めなければわからないので，スワップ操作後に上記の1から4のうちどの状態へ変化したかを仮定する．仮定したそれぞれの状況におけるクリティカルパスの長さを，直方体の各辺の大小関係と，直方体を一つ取り除いた配置の情報を用いて計算する．計算した中で最長のものが実際のクリティカルパスの長さとなっており，容器の大きさを求めることができる．

以下に x 方向に関する評価法を示す．ここで $w_i > w_j$ であり，スワップ操作前の容器の幅を W，スワップ操作前の配置における直方体 j の左寄せ座標を x_j，右寄せ座標を $x_j^{(R)}$，直方体 i を取り除いた配置の幅を $\tilde{W}$，i を取り除いた配置における直方体 j の左寄せ座標を $\tilde{x}_j$，右寄せ座標を $\tilde{x}_j^{(R)}$ とし，スワップ操作後の幅を W' とする．$w_i = w_j$ のときはスワップ操作の前後で容器の幅は変化しない．また，i と j が x 方向の関係にあるとき ($\{\sigma_2^{-1}(i) < \sigma_2^{-1}(j)\} \wedge \{\sigma_3^{-1}(i) > \sigma_3^{-1}(j)\}$) とそうでないときで，同じパスに含まれるか否かが変わるため，評価法も少し異なる．

- i と j が x 方向の関係にないとき

  1. スワップ操作前に i，j がともに x 方向のクリティカルパス上にないとき．
     i) スワップ操作後に i，j がともに x 方向のクリティカルパス上にないとき．
     スワップ操作前後で i，j は幅 W' の決定に関与しないので
     スワップ操作後の幅 W' は
     $$W' = W.$$
     ii) スワップ操作後に i のみ x 方向のクリティカルパス上にあるとき．
     このとき j を抜いた場所に i を入れたときの幅が W' となるので
     $$W' = x_j + w_i + x_j^{(R)}.$$

  $w_i > w_j$ より，j のみ，又は i と j がともにクリティカルパスに含まれることはない．
  従って $W' = \max\{W, x_j + w_i + x_j^{(R)}\}$ となる．

2. スワップ操作前に i のみ x 方向のクリティカルパス上にあるとき.

   i) スワップ操作後に i, j がともに x 方向のクリティカルパス上にないとき.
   このとき i, j は幅 W' の決定に関与しないので, W' は i を取り除いた配置の幅 $\tilde{W}$ により決定する. ここで j を取り除かない理由は, もし i と j を取り除いた配置の幅が $\tilde{W}$ より小さければ, スワップ操作後に i と j のうち少なくとも一つはクリティカルパスに含まれるためである. 従って
   $$W' = \tilde{W}.$$

   ii) スワップ操作後に i のみ x 方向のクリティカルパス上にあるとき.
   このとき j を抜いた場所に i を入れたときの幅より,
   $$W' = x_j + w_i + x_j^{(R)}.$$

   iii) スワップ操作後に j のみ y 方向のクリティカルパス上にあるとき.
   このとき i を抜いた場所に j を入れたときの幅より,
   $$W' = W - w_i + w_j.$$

   iv) スワップ操作後に i, j がともに x 方向のクリティカルパス上にあるとき.
   このとき, ii) かつ iii) となるので,
   $$W' = W - w_i + w_j = x_j + w_i + x_j^{(R)}.$$

   以上より $W' = \max\{\tilde{W}, x_j + w_i + x_j^{(R)}, W - w_i + w_j\}$ となる.

3. スワップ操作前に j のみ x 方向のクリティカルパス上にあるとき.
   このとき $w_i > w_j$ より j を抜いて i を入れたときの幅が W' となる.
   従って $W' = W + w_i - w_j$ となる.

4. スワップ操作前に i, j がともに x 方向のクリティカルパス上にあるとき.
   この場合も 3. と同様で $w_i > w_j$ より j を抜いて i を入れたときの幅が W' となる.
   従って $W' = W + w_i - w_j$ となる.

- i と j が x 方向の関係にあるとき

1. スワップ操作前に i, j がともに x 方向のクリティカルパス上にないとき.

   i) スワップ操作後に i, j がともに x 方向のクリティカルパス上にないとき.
   このとき, スワップ操作前後で i, j は幅 W, W' の決定に関与しないので
   スワップ操作後の幅 W' は
   $$W' = W.$$

   ii) スワップ操作後に i のみ x 方向のクリティカルパス上にあるとき.
   このとき i を抜いた配置において j を抜いた場所に i を入れたときの幅が W' となる. i を抜くのは x_j, $x_j^{(R)}$ が i に依存している可能性があるためである. 従って
   $$W' = \tilde{x}_j + w_i + \tilde{x}_j^{(R)}.$$

   $w_i > w_j$ より, j のみ, 又は i と j がともにクリティカルパスに含まれることはない.
   従って $W' = \max\{W, \tilde{x}_j + w_i + \tilde{x}_j^{(R)}\}$ となる.

2. スワップ操作前に i のみ x 方向のクリティカルパス上にあるとき.

   i) スワップ操作後に i, j がともに x 方向のクリティカルパス上にないとき.
   このとき i, j は幅 W' の決定に関与しないので, W' は i を取り除いた配置の幅 $\tilde{W}$

により決定する．ここでjを取り除かない理由は，もしiとjを取り除いた配置の幅が$\tilde{W}$より小さければ，スワップ操作後にiとjのうち少なくとも一つはクリティカルパスに含まれるためである．従って

$$W' = \tilde{W}.$$

ii) スワップ操作後にiのみx方向のクリティカルパス上にあるとき．
このときiを抜いた配置でjを抜いた場所にiを入れたときの幅となる．iを抜くのは$x_j, x_j^{(R)}$がiに依存している可能性があるためである．従って

$$W' = \tilde{x}_j + w_i + \tilde{x}_j^{(R)}.$$

iii) スワップ操作後にjのみx方向のクリティカルパス上にあるとき．
このときiを抜いた場所にjを入れたときの幅がW'となる．ここでiを抜かない理由は，x_i，$x_i^{(R)}$はiを抜いても変わらないためである．従って

$$W' = W - w_i + w_j.$$

iv) スワップ操作後にi，jがともにx方向のクリティカルパス上にあるとき．

* i，jが同じパス上にあるとき
このとき元の配置でjを抜いてiを入れたときの幅がW'となる．ただし，x_i，$x_i^{(R)}$はiに依存しており，$w_i - w_j$だけ小さくなるので
$$W' = x_j + w_i + x_j^{(R)} - (w_i - w_j) = x_j + w_j + x_j^{(R)}.$$
* i，jが異なるパス上にあるとき
このとき，ii) かつ iii) となるので，
$$W' = W - w_i + w_j = \tilde{x}_j + w_i + \tilde{x}_j^{(R)}.$$

以上より$W' = \max\{\tilde{W}, \tilde{x}_j + w_i + \tilde{x}_j^{(R)}, W - w_i + w_j, x_j + w_j + x_j^{(R)}\}$となる．

3. スワップ操作前にjのみx方向のクリティカルパス上にあるとき．
このときiはx方向のクリティカルパス上にないのでx_j，$x_j^{(R)}$はiに依存しない．従って$w_i > w_j$よりjを抜いてiを入れたときの幅がW'となるので

$$W' = W + w_i - w_j.$$

4. スワップ操作前にi，jがともにx方向のクリティカルパス上にあるとき．

– iとjが同じパス上にあるとき．

i) スワップ操作後にi，jがともにx方向のクリティカルパス上にあるとき．
このときjを抜いてiを入れたときの幅がW'となる．ただし，x_i，$x_i^{(R)}$はiに依存しており，$w_i - w_j$だけ小さくなるので
$$W' = x_j + w_i + x_j^{(R)} - (w_i - w_j) = W.$$

ii) スワップ操作後にiのみx方向のクリティカルパス上にあるとき．
x_j，$x_j^{(R)}$はiに依存している可能性があるため，iを抜いた配置でjを抜いた場所にiを入れたときの幅がW'となる．従って
$$W' = \tilde{x}_j + w_i + \tilde{x}_j^{(R)}.$$

– iとjが異なるパス上にあるとき．
このとき$w_i > w_j$よりjを抜いてiを入れたときの幅がW'となる．異なるパス上にあるため，x_j，$x_j^{(R)}$はiに依存しない．従って$x_j = \tilde{x}_j$，$x_j^{(R)} = \tilde{x}_j^{(R)}$であり，

$$W' = x_j + w_i + x_j^{(R)} = \tilde{x}_j + w_i + \tilde{x}_j^{(R)}.$$

i, j がどのクリティカルパス上にあるかは分からないが，異なるクリティカルパス上にあるとき $W' > W$ であり，$W' = \max\{W, \tilde{x}_j + w_i + \tilde{x}_j^{(R)}\}$ となる．

直方体の大小関係と直方体 i, j が同じパスに含まれるかどうかを考慮することで y 方向，z 方向に関しても同様に評価することができる．いずれの状況においても，スワップ操作前の配置と $w_i > w_j$ を満たす直方体 i を一つ取り除いた配置を求めておくことで，近傍解を一つ当たり $O(1)$ 時間で計算可能である．全体の計算量は n 通りの直方体を取り除いた配置を求めるのに $O(n^2 \log^2 n)$ 時間，解の評価に $O(n^2)$ 時間かかり，全近傍解の評価は $O(n^2 \log^2 n)$ 時間で可能である．従って解一つ当りの評価時間は $O(\log^2 n)$ 時間となり，既存の評価法 $O(n^2)$ 時間と比較して $O(\frac{n^2}{\log^2 n})$ 倍高速である．この評価法では w_i，w_j といった直方体の各辺の長さがスワップ操作の前後で変化しないことを利用している．従って，各辺の長さが変化する回転操作とスワップ操作を同時に行う解は評価できない．そこで，スワップ操作と同時に直方体を回転させる解を評価する手法を次節で提案する．

5.4 スワップ操作と回転操作を同時に行う解の高速評価法

前節で提案したスワップ近傍の高速評価法は従来法と比較して $O(\frac{n^2}{\log^2 n})$ 倍高速であるが，スワップ操作と同時に直方体を回転させる解を評価することができない場合がある．そこで，本節では回転を考慮したスワップ近傍を $O(n^2 \log^2 n)$ 時間で評価する手法を提案する．

解の評価をできない場合があると述べたが，具体的には直方体 i と j に関してスワップ操作を行う際に，(i) スワップ操作前の配置において i と j の少なくとも一方がクリティカルパスに含まれており，スワップ操作後にどちらの直方体もクリティカルパスに含まれない場合のクリティカルパスの長さ，(ii) スワップ操作後に i と j が同じクリティカルパス上にあるときのクリティカルパスの長さ，を評価できない．一つ目の状況を本質的に解決することは困難なため，スワップ操作と回転操作を同時に行った後の容器の体積を正確に評価することはできない場合がある．このため，いずれかのクリティカルパスの長さが大きくなる改善解については本節の手法で発見することができず，すべての方向において，クリティカルパスの長さが減少するかクリティカルパスに含まれる直方体の数が減少する改善解を探索する．

二つ目の状況におけるクリティカルパスの長さの評価には，直方体 i に対して，i を取り除いた配置と，i を “クリティカルパスに含まれるまで拡大” した配置の情報を用いる．直方体 i をクリティカルパスに含まれるまで拡大するとは，i の幅を $W - x_i - x_i^{(R)}$，高さを $H - y_i - y_i^{(T)}$，奥行を $D - z_i - z_i^{(B)}$ とすることである．この二つの配置を求めておくことで，近傍解一つ当たり，$O(1)$ 時間でクリティカルパスの長さを評価する．直方体 i と j に関してスワップ操作を行うときの x 方向に関する評価法を以下に述べる．y 方向，z 方向に関しても x 方向と同様に評価することができる．

まず，これから使用する用語を定義する．スワップ操作前の容器の幅を W，直方体 i の幅を w_i，スワップ操作前の配置における i の左寄せ座標を x_i，右寄せ座標を $x_i^{(R)}$ とする．スワップ操作後の直方体 i の幅を w_i' (直方体を回転する場合 $w_i' = w_i$ とは限らない) とする．また，“スワップ操作前の i の最大幅 e_i” を $e_i = W - x_i - x_i^{(R)}$ と定義する．図 7 に直方体 3 の最大幅 e_3 を示す．図 7 ではわかりやすくするため 2 次元の配置を用いている．最大幅はクリティカルパスに含まれるか否かの境界値になっており，直方体 3 が配置されている場所では，最大幅 e_3 未満の幅を持つ直方体であればクリティカルパスに含まれないことがわかる．

次に，$e_i \geq w'_j$ を満たす直方体 i と j に関してスワップ操作を行うときの，スワップ操作後のクリティカルパスの長さを考える．($e_i < w'_j$ かつ $e_j < w'_i$ のとき，x 方向のクリティカルパスの長さが必ず増加するが，本手法ではこのような解の評価は行わない．) $e'_{j,i}$ を，直方体 i のあった位置に直方体 j を回転させて配置したときの，直方体 j のあった位置における最大幅（ただし，元の配置のクリティカルパスの長さを基準）とする．$e'_{j,i}$ の値がわかれば，w'_i と $e'_{j,i}$ を用いることで，スワップ操作後のクリティカルパスの長さが求まる．図 8 に直方体 5 と 3 に関してスワップ操作を行うときの最大幅 $e'_{3,5}$ を示す．

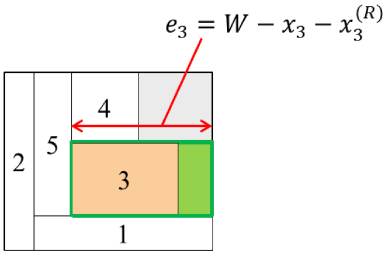

図 7: 直方体 3 の最大幅 e_3

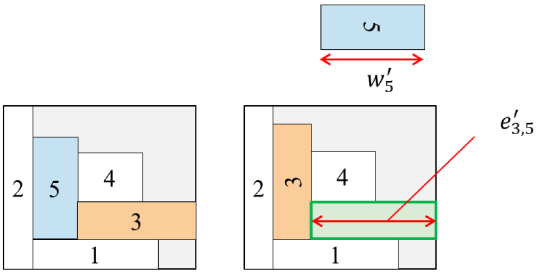

図 8: 直方体 5 と 3 におけるスワップ操作後の最大幅 $e'_{3,5}$

スワップ操作後の最大幅 $e'_{j,i}$ の計算方法について述べる．まず，直方体 i を取り除いた配置と i をクリティカルパスに含まれるまで拡大した配置をそれぞれ $O(n \log^2 n)$ 時間で求めておく．i を取り除いた配置における j の左寄せ座標を $\hat{x}_{j,i}$，右寄せ座標を $\hat{x}^{(R)}_{j,i}$ とし，i を拡大した配置の j の左寄せ座標を $\check{x}_{j,i}$，右寄せ座標を $\check{x}^{(R)}_{j,i}$ とする．このときスワップ操作後の i の最大幅 $e'_{j,i}$ の上限 $\hat{e}_{i,j}$ と下限 $\check{e}_{i,j}$ は次のように表せる．

$$\hat{e}_{j,i} = W - \hat{x}_{j,i} - \hat{x}^{(R)}_{j,i}, \tag{26}$$

$$\check{e}_{j,i} = W - \check{x}_{j,i} - \check{x}^{(R)}_{j,i}. \tag{27}$$

直方体 j ($e_i \geq w'_j$) を直方体 i のあった位置へ配置したとき，$e_i - w'_j$ だけの幅領域が余る．この余った幅領域を全て i が利用できる場合，i のスワップ操作後の最大幅は $e'_{j,i} = \check{e}_{j,i} + e_i - w'_j$ となる．$e'_{j,i}$ は $\hat{e}_{j,i}$ を超えることはないので実際の最大幅は以下の式で表される．

$$e'_{j,i} = \min\{\hat{e}_{j,i}, \check{e}_{j,i} + e_i - w'_j\}. \tag{28}$$

式 (26)–(28) は i を取り除いた配置と拡大した配置を求めておくことで $O(1)$ 時間で計算できる．全体の計算量は n 通りの直方体を取り除いた配置と拡大した配置を求めるのに $O(n^2 \log^2 n)$ 時間，解の評価に $O(n^2)$ 時間がかかり，全近傍解の評価は $O(n^2 \log^2 n)$ 時間で可能である．

6 数値実験

計算機実験により，既存解法と提案手法の比較を行った．計算は全てワークステーション (CPU: Intel Xeon E5-2690 (2.90GHz)，メモリ: 64GB) で行い，各辺の長さをランダムに決定した直方体数 30，100，300 の問題例を用いた．以下の三つの手法の比較を行った結果を表 1 に示す．それぞれの手法に対して表 1 に示す時間まで反復局所探索法を繰り返す実験を 10 回ずつ行った．表 1 中の値は充填率を表し，10 回の試行の平均値を示す．

1. 従来法
   シフト近傍，スワップ近傍ともに解一つ当たり $O(n^2)$ 時間で配置を求めて評価する．シフト近傍に関しては，シフト操作と直方体の回転操作を同時に行う解の評価も行う．移動戦略は改善解が見つかったら即座に移動する，即時移動戦略を用いる．

2. 提案法 1
   シフト近傍は 5.2 節で提案した手法を，スワップ近傍は 5.3 節で提案した手法を用いる．シフト近傍に関しては，シフト操作と直方体の回転操作を同時に行う解の評価も行う．移動戦略は 4.3 節のものを用いる．

3. 提案法 2
   提案法 1 に加え，5.4 節で提案した手法を用いてスワップ操作と回転操作を同時に行う解の評価を行う．

表 1: 実験結果

時間	直方体数 30			直方体数 100			直方体数 300		
[秒]	従来法	提案法 1	提案法 2	従来法	提案法 1	提案法 2	従来法	提案法 1	提案法 2
10	76.22	86.90	87.91	19.69	81.05	84.68	7.09	8.45	8.46
100	81.52	88.21	89.56	39.87	84.94	88.64	10.27	14.53	14.53
1000	85.71	89.03	90.43	67.16	86.39	90.38	18.35	78.21	81.65
10000	88.52	90.26	91.25	81.09	88.03	91.03	24.76	82.60	86.24

表 1 より，いずれの問題例に対しても従来法より提案法のほうが同じ計算時間において精度の良い解が得られることが確認された．特に規模の大きい問題例ほどこの傾向は顕著に見られた．また，提案法 2 が提案法 1 より良い解が得られていることから，スワップ操作と回転操作を同時に行う解の探索が有効であることが確認できた．図 9 に実験で最終的に得られた解の一部を示す．図 9 の上側の配置は従来法による結果のうち，最も充填率の高かった配置を示し，下側の配置は提案法 2 の結果のうち最も充填率の高かった配置を示す．図中の数字はそれぞれの配置の充填率を示す．

7 まとめ

本研究では，3 次元配置問題の体積最小化問題に対して，Sequence-Triple を用いた局所探索法の検討を行った．シフト近傍に関して文献 [4] の手法を 3 次元に拡張することで近傍解一つ当たり $O(1)$ 時間で評価する手法を提案した．また，スワップ近傍に関して文献 [5] の手法を 3 次元に拡張することで近傍解一つ当たり $O(\log^2 n)$ 時間で評価する手法を提案した．更にスワップ近傍に関して，スワップ操作を行うと同時に直方体を回転する解を一つ当たり $O(\log^2 n)$ 時間で評価する手法を提案した．

数値実験を行った結果，従来手法と比較して，多数の解を高速に評価することができ，大幅に解の精度が向上することを確認した．Sequence-Triple では従来扱いづらかった直方体数数百規模の問題例でもある程度良い解が得られており，提案手法の有効性が示された．スワップ操作と回

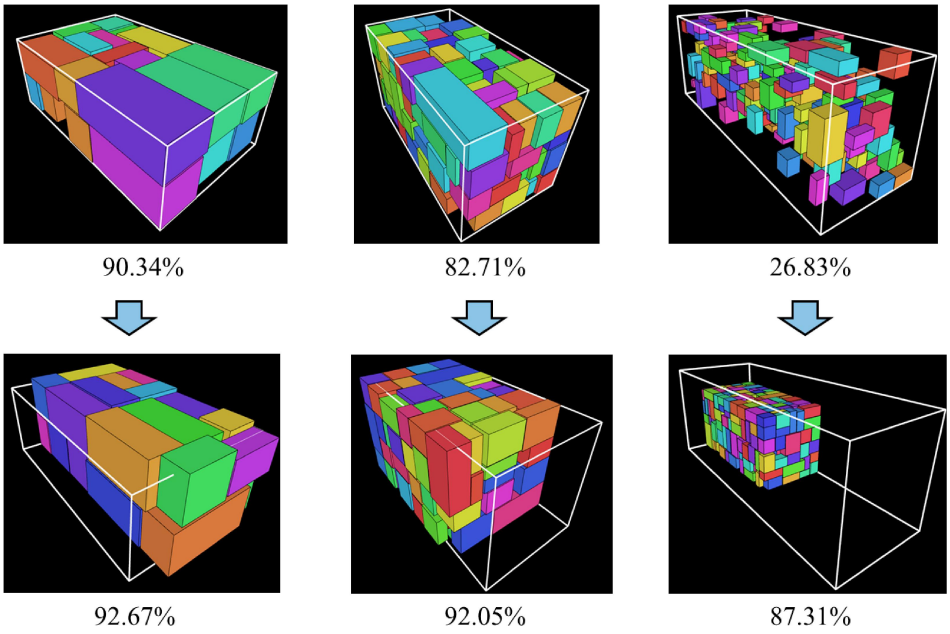

図 9: 従来法と提案法による配置図の比較

転操作を同時に行う解の高速評価法は，近傍内の全ての改善解を見つけることはできないが，数値実験より精度に関して十分効果があることが確認できた．

今回，近傍探索の効率化を行ったが，Sequence-Triple から配置を高速に計算することができれば，さらなる効率化が期待できる．2 次元の Sequene-Pair に関しては配置を求める様々な効率化手法が提案されており，3 次元に関しても今後の課題である．また，今回は直方体を積み上げたときの安定性や，配置場所などは考慮しなかったが，より現実問題に近い制約条件を取り入れることも今後考えていきたい．

参考文献

[1] A. Bortfeldt, D. Mack, A heuristic for the three-dimensional strip packing problem, European Journal of Operational Research, Vol. 183, pp. 1267–1279 (2007).

[2] J. Egeblad, D. Pisinger, Heuristic approaches for the two- and three-dimensional knapsack packing problem, Computers & Operations Research, Vol. 36, pp. 1026–1049 (2009).

[3] S. Imahori, M. Yagiura and T. Ibaraki, Local search algorithms for the rectangle packing problem with general spatial costs, Mathematical Programming, Vol. 97, pp. 543–569 (2003).

[4] S. Imahori, M. Yagiura and T. Ibaraki, Improved local search algorithms for the rectangle packing problem with general spatial costs, European Journal of Operational Research, Vol. 167, pp. 48–67 (2005).

[5] 岩澤宏紀, 今堀慎治, 順列対を用いた長方形配置問題に対する局所探索法–近傍探索の改良–, スケジューリング・シンポジウム 2013 講演論文集, pp. 107–112 (2013).

[6] 川島大貴, 田中勇真, 今堀慎治, 柳浦睦憲, 3 次元箱詰め問題に対する構築型解法の効率的実現法, 第 9 回情報科学技術フォーラム（FIT2010）, 講演論文集 (第 1 分冊), pp. 31-38 (2010).

[7] 川島大貴, 田中勇真, 今堀慎治, 柳浦睦憲, 3 次元パッキング問題に対する best-fit 法の効率的実現法, 第 10 回情報科学技術フォーラム（FIT2011）, 講演論文集 (第 1 分冊), pp. 29–36 (2011).

[8] H. Murata, K. Fujiyoshi, S. Nakatake and Y. Kajitani, VLSI module placement based on rectangle-packing by the sequence-pair, IEEE Transactions on Computer Aided Design, Vol. 15, pp. 1518–1524 (1996).

[9] D. Pisinger, Denser packings obtained in $O(n \log \log n)$ time, INFORMS Journal on Computing, Vol. 19, pp. 395–405 (2007).

[10] Y. Sheng, A. Takahashi and S. Ueno, 2-stage simulated annealing with crossover operator for 3D-packing volume minimization, SASIMI 2012 Proceedings, pp. 227–232 (2012).

[11] X. Tang, D. F. Wong, FAST-SP: a fast algorithm for block placement based on sequence pair, Asia and South Pacific Design Automation Conference, pp. 521–526 (2001).

[12] 柳浦睦憲, 茨木俊秀, 組合せ最適化–メタ戦略を中心として–, 朝倉書店 (2001).

[13] H. Yamazaki, K. Sakanushi, S. Nakatake and Y. Kajitani, The 3D-packing by meta data structure and packing heuristics, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E83-A, pp. 639–645 (2000).

付録 A　順列から $O(n \log^2 n)$ 時間で配置を求める手法

3.1 節では $O(n^2)$ 時間で三つの順列から配置を求める手法を紹介した．ここでは $O(n \log^2 n)$ 時間で配置を計算する手法を提案する．以下では，8 個の直方体集合 $\{1,2,3,4,5,6,7,8\}$ と順列 $\sigma_1 = \langle 8,7,1,5,2,4,3,6 \rangle$，$\sigma_2 = \langle 3,4,1,7,8,6,2,5 \rangle$，$\sigma_3 = \langle 4,5,1,8,6,2,3,7 \rangle$ が与えられたとき，z 座標を計算する例を用いて配置アルゴリズムを説明する．直方体数 n が 2 の冪でない場合は，n を超える最小の 2 の累乗数 m $(< 2n)$ を考えれば良い．

z 座標は 3.1 節の式 (7) で表され，順列 σ_3 の前から順に座標を決定していくことができる．σ_3 で l 番目の直方体 i $(= \sigma_3(l))$ に対して，$\sigma_1^{-1}(j) < \sigma_1^{-1}(i)$，$\sigma_2^{-1}(j) < \sigma_2^{-1}(i)$，$\sigma_3^{-1}(j) < l$ を満たす直方体 j のうち，$z_j + d_j$ が最大のものが i の z 座標となる．j の候補を一つ一つ計算する場合，直方体 i の座標決定に $O(n)$ 時間がかかるが，図 10 に示したデータ構造を用いることで $O(\log^2 n)$ 時間で求めることができる．

図 10 の各データ構造の横軸は順列 σ_1 に，縦軸は順列 σ_2 に対応しており，四角で仕切られた各要素にはその集合に含まれる直方体の最大奥座標 $(z_j + d_j)$ を記憶する．例えば図 10 の要素 A は，順列 σ_1 で 2 より前 (8,7,1,5)，順列 σ_2 で 8 より前 (3,4,1,7) にある直方体 $j \in \{1,7\}$ のうち $z_j + d_j$ が最大のものを記憶する．ただし，ここで対象とする直方体は $\sigma_3^{-1}(j) < l$ を満たす直方体 j のみとする．配置を求める際には，l を 1 から n まで順に増やしながら直方体 $\sigma_3(l)$ の座標を決定し，図 10 に示すデータ構造の更新を行う．

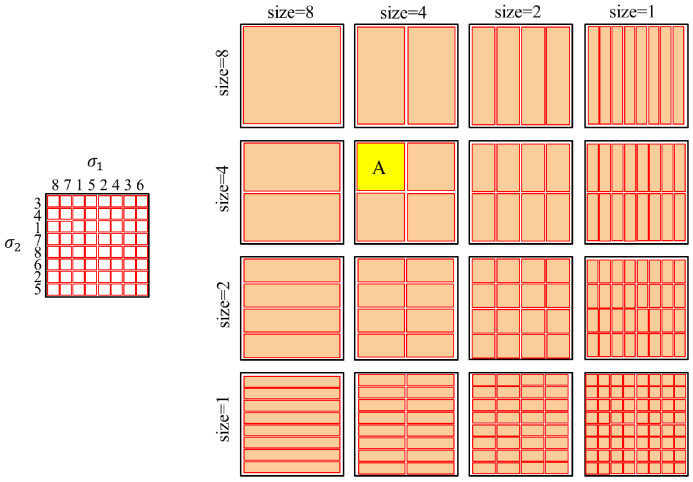

図 10: データ構造

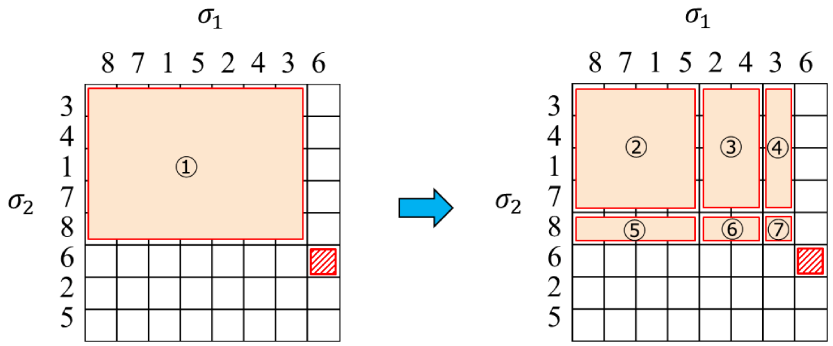

図 11: 直方体 6 の座標決定

直方体 6 の z 座標を決定する際，σ_1 で 6 より前 (8,7,1,5,2,4,3)，σ_2 で 6 より前 (3,4,1,7,8) の直方体のうち，既に座標が決まっている (σ_3 で 6 より前) 直方体 j に注目して $\max(z_j + d_j)$ を求める．σ_3 の前から順に座標を決定していくため，データ構造の各要素には，$\sigma_3^{-1}(j) < l$ を満たす直方体 j の最大奥座標が記憶されている．σ_1 と σ_2 で 6 より前の直方体 (図 11 の領域①) のうち $z_j + d_j$ が最大のものがわかれば良い．図 11 において領域①の全ての要素を一つずつ計算すると $O(n^2)$ 時間かかるが，データ構造の要素②から⑦を用いることで $O(\log^2 n)$ 時間で評価することができる．直方体 6 の座標決定後のデータ構造の更新は，直方体 6 が含まれる要素のみ更新すれば良いので $O(\log^2 n)$ 時間で可能である．よって，全直方体の座標の決定を $O(n \log^2 n)$ 時間でできる．

ここで，データ構造の構築及び初期化に要する時間を評価する．図 10 に示したデータ構造の要素数は $(2n-1)^2$ であり，この構築はアルゴリズム実行の当初に $O(n^2)$ 時間かけて一度だけ行う．提案法では，順列から配置の計算を多数行い，各配置計算のためにデータ構造の初期化 (全ての要素の値を 0 とする) が必要である．このとき，全ての要素にアクセスすると $O(n^2)$ 時間を要するため，前回の配置計算で値の更新を行った要素を全て記憶しておき，それらの要素のみ初期化する．この場合，データ構造の初期化に要する時間は配置計算の時間以下となる．