経路と積載の同時最適化を考慮した車両配送問題の解法

○森口裕人　小野典彦　永田裕一　（徳島大学）

Solving Vehicle Routing Problems Considering Simultaneous Optimization of Routing and 3D Packing

∗Yuto Moriguchi , Norihiko Ono and Yuichi Nagata (Tokushima University)

Abstract– 3-Dimensional Loading Capacitated Vehicle Routing Problem(3L-CVRP) that considers loading of container in 3-dimensions was introduced as a model to bring VRP closer to the real problem. Approximate solution methods are often applied to the 3L-CVRP solution where it is necessary to repeatedly judge whether vehicles can load items of customers visited. In this research, we propose a packing algorithm for 3L-CVRP focusing on free space aiming at reducing computational cost. We also implement a tabu search algorithm to solve 3L-CVRP along with these packing algorithms.

Key Words: メタヒューリスティクス, 3 次元パッキング, 車両配送問題

1　序論

近年，インターネットを用いた通信販売システムの普及によって宅配便の需要は大幅に高まっている．この需要に対し，宅配便の配送計画をモデル化した組み合わせ最適化問題である車両配送問題 (Vehicle Routing Problem, VRP) についての研究は活発に行われている．

VRP をより現実問題に近づけるためのモデルとして，3 次元での荷物の積み込みを考慮した車両配送問題 (3-Dimensional Loading Capacitated Vehicle Routing Problem, 3L-CVRP) が Gendreau ら [1] によって提案されており，その後，Tarantilis ら [2]，Fuellerer ら [3]，Wenbin ら [4] などによって研究が行われている．3L-CVRP の目的は，拠点から出発する各車両がすべての顧客の需要を配送し拠点に戻るような経路の中で，車両の総移動距離が最小となる経路を求めることである．さらに，荷物の安定性や積み下ろしの利便性などの問題に対処しつつ，車両への積載を行わなければならない．3L-CVRP の実問題への適用例としては，家電製品やキッチン用品，機械部品 [3] や家具 [1] などがある．

3L-CVRP の解法にはメタヒューリスティクスを用いた近似解法がよく用いられているが，その探索の中で車両が訪問する顧客の荷物を積み込むことができるかどうかを繰り返し判断する必要がある．そのためパッキングの効率化が 3L-CVRP の解法全体の効率化に大きく関係する．本研究では，パッキングする際の容器の空き空間（Free Space）に着目した手法を提案し，パッキングの効率化を図る．

2　3L-CVRP の定義

3-Dimensional Loading Capacitated Vehicle Routing Problem（3L-CVRP）は Gendreau ら [1] によって提案された，車両配送問題と 3 次元パッキング問題の 2 つで構成される組み合わせ最適化問題である．$V = \{0, 1, \cdots, n\}$ を $n+1$ 個の頂点集合とし，0 はデポ（拠点），$1, \cdots, n$ を n 人の顧客とする．$E = (i, j)$ を各頂点 i, j を結ぶ全ての辺集合とする．$G = (V, E)$ を V と E から導出されたグラフとし，各辺 (i, j) のコストを c_{ij} とする．v を同一の条件（重量容量および積載空間の大きさが等しい）を持つ利用可能な車両とする．ここで車両数は問題ごとに指定されるものとし，全ての車両を用いて配送を行う．各車両は重量容量 D と，幅 W，高さ H，長さ L の 3 次元直方体積載空間をもつ．利用可能な直方体の積載空間を $S = W * H * L$ とする．各顧客 i は m_i 個の荷物を受け取り，荷物はそれぞれ $I_{ik}(k = 1, \cdots, m_i)$ で表す．荷物 I_{ik} はそれぞれ幅 w_{ik}，高さ h_{ik}，長さ l_{ik} の直方体であり，顧客 i が受け取る荷物の総重量は d_i で表す．顧客 i が必要とする空間の総体積は $S_i = \sum_{k=1}^{m_i} w_{ik} * h_{ik} * l_{ik}$ で表す．Fig. 1 は，顧客 6 人に対して 3 台の車両で配送を行うような経路を表している．また，Fig. 2 は Fig. 1 の車両 v_1 における実行可能な積載を表している．車両の積載空間は Fig. 2 では点線部分で表されており，原点が最も奥，最も下，最も左の点であり，座標系の第一象限に存在する．幅 W は x 軸に平行であり，高さ H は y 軸に平行であり，長さ L は z 軸に平行である．左･右，上･下，手前･奥という用語は図に示されている通りである．各荷物の位置は荷物 I_{ik} の左下奥の座標 (x_{ik}, y_{ik}, z_{ik}) で表し，座標 (x_{ik}, y_{ik}, z_{ik}) と大きさ (w_{ik}, h_{ik}, l_{ik}) の組み合わせによって荷物の積載を表現することができる．また，荷物の積み下ろしは手前（z 軸）方向に行われる．

さらに，現実の配送を考慮するために以下の 4 つを積載時の制約として追加する．

[条件 a] 荷物は高さに対して固定の向きを持つが，水平方向（Fig. 2 中の $w - l$ 平面上）では 90 度回転することができる．

[条件 b] 各荷物 I_{ik} は脆弱性フラグ $f_{ik}(i = 1, \cdots, n; k = 1, \cdots, m_i)$ を持ち，$f_{ik} = 1$ ならばその荷物は壊れやすく，そうでないなら 0 とする．通常の荷物を壊れやすい荷物の上に置くことはできないが，壊れやす

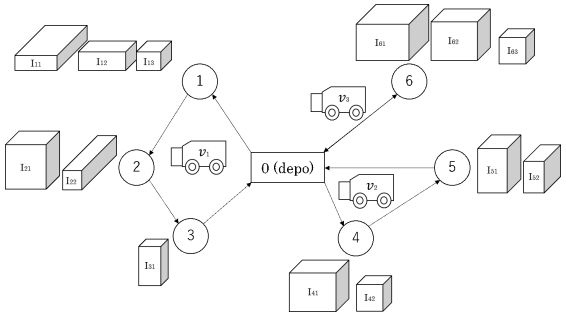

Fig. 1: An example solution of 3L-CVRP.

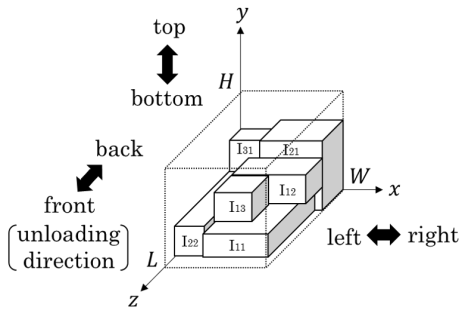

Fig. 2: The loading on v_1 in Fig. 1.

い荷物同士は積み重ねることができる．

[条件 c] 荷物 I_{ik} が他の荷物の上に置かれているとき，対応する支持領域を評価する必要がある．荷物 I_{ik} が置かれている下面の高さを $\bar{h}$ とし，$\bar{A}$（支持領域）を荷物 I_{ik} の下面と上面が高さ $\bar{h}$ にある荷物集合の重なった部分とする．支持領域が少なくとも与えられた荷物を基にした閾値 $a\%$ だけあれば，すなわち $\bar{A} \geq a * w_{ik} * l_{ik}$ が成り立つ場合にその配置は実行可能である（荷物 I_{ik} が積載空間の下面に置かれているとき，つまり $\bar{h} = 0$ の時は常に制約が満たされる）．

[条件 d] 各車両の積載は以下の LIFO 方針に従わなければならない．顧客 i を訪問したとき，積み下ろし方向（Fig. 2 中の z 軸正の向き）に平行な一連の直線運動（荷物 1 つにつき 1 回）によって，顧客が要求するすべての荷物 I_{ik} を積み下ろすことが可能でなければならない．言い換えれば，後に訪問される顧客が要求する荷物は，I_{ik} 上または I_{ik} と車両の後部（Fig. 2 における手前面）との間に配置することはできない．

3L-CVRP は，以上の制約条件を全て満たす経路の中から，総移動距離が最小のものを見つける問題である．

3 従来手法のパッキング

Wenbin ら [4] は Deepest-Bottom-Left algorithm（DBL 法）と Maximum-Touching-Area algorithm（MTA 法）を利用してパッキングを行った．ここで，DBL 法と MTA 法の概要を説明する．まず，DBL 順序および DBL 点を定義する．2 つの点 $P_1(x_1, y_1, z_1)$ と $P_2(x_2, y_2, z_2)$ において次式

$$(z_1 < z_2) \vee (z_1 = z_2 \wedge y_1 < y_2) \vee (z_1 = z_2 \wedge y_1 = y_2 \wedge x_1 < x_2) \quad (1)$$

が成り立つとき，P_1 は P_2 より DBL 順序が小さいとする．また各物体を配置可能な点集合に対し，最も DBL 順序が小さい点をその物体の DBL 点と呼ぶ．

DBL 法とは，各物体を配置可能な候補点（あるアルゴリズムによって求められる）に対し，DBL 点に配置する順序ベースな手法である．ここで候補点とは，次に配置する物体の配置可能な点 (x, y, z) の集合であるが，物体は連続的に移動できるため，そのような点は無数に存在する．そのため，その中から有限個の点を選択する必要があることに注意されたい．各物体はあらかじめ与えられた順序通りに配置していくため，配置順序を変更することにより得られる積載が異なる．また，容器および配置する物体とその配置順序が与えられると，DBL 法によって得られる積載は一意に定まる．

一方で MTA 法は，各物体を配置可能な候補点に対して，配置する物体と他の既配置の荷物または容器との接地面積が最大となるような位置に配置する順序ベースな手法である．また，DBL 法は DBL 順序に則って探索して最初に見つかった配置可能な点（DBL 点）を配置位置として決定するが，MTA 法ではすべての候補点で接地面積を計算した後，最大となる点を配置位置として決定する．つまり，MTA 法による配置位置は DBL 点とは限らない．そのため，MTA 法の方が DBL 法より計算コストが高い．

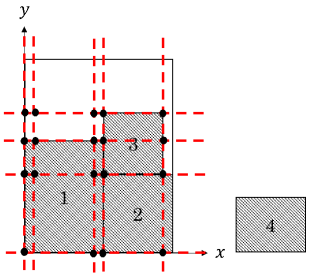

Fig. 3: The grid points of Wenbin et al.

Wenbin ら [4] のパッキングアルゴリズムにおいて，荷物を配置可能な候補点は Fig. 3 のような既配置荷物または容器と現在の荷物との格子点によって求められている．その他に候補点の列挙および各制約計算の高速化の工夫が行われているが，本稿では説明を省略する．

4 提案手法のパッキング

本論文では，荷物を配置する際に容器内の空き空間に着目した候補点探索を提案する．この手法のメリットとして，3 章で説明した格子点と比較して無駄な候補点が生成されず，効率よく候補点を探索できることが挙げられる．一方で Wenbin ら [4] の手法によって見つけられる点を発見できない場合があるというデメリットも存在する．以下で提案手法について解説する．

4.1 Free Sace の定義

Free Space とは，容器内の空間のうち現在の荷物を配置可能な最大のスペースであるとする.Free Space も荷物と同じ直方体であるため，荷物と同じように座標 (X, Y, Z) と大きさ (W, H, L) によって表現できる．一般的なパッキング（物体同士が重ならずに置ければよい）において Free Space 内に配置できるかどうかは，単に Free Space と物体との大小関係を確認する，すなわち，物体の大きさ (w, h, l) それぞれに対し，$w \leq W, h \leq H, l \leq L$ が成り立つのを確認するだけでよい．

荷物の積載を通して，Free Space は配置された荷物との重なりによって更新される．例えば 2 次元の場合，Fig. 4 のような Free Space(a) に荷物 1 を配置すると，Fig. 5 のように Free Space(a) が削除され，新たに (b), (c) が追加される．

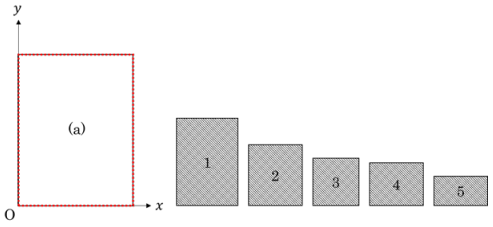

Fig. 4: A Free Space and 5 items.

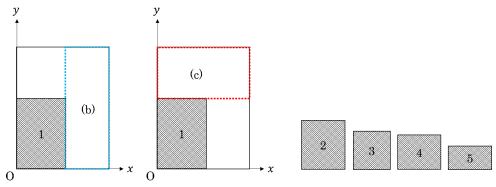

Fig. 5: Update Free Space in 2D case.

3 次元についても 2 次元と同様の流れで Free Space を更新することができる．Fig. 6 のように，ある物体 I が座標 (x_i, y_i, z_i) と大きさ (w_i, h_i, l_i) で表され，それと重なる Free Space が座標 (X_j, Y_j, Z_j) と大きさ (W_j, H_j, L_j) で表されるとき，追加される Free Space は以下の 6 つに分類できる（Fig. 7 参照）．

(i) Free Space の左側に新しい Free Space が追加される場合，すなわち $X_j < x_i$ の時，座標 (X_j, Y_j, Z_j)，大きさ $(x_i - X_j, H_j, L_j)$ の Free Space が追加

(ii) Free Space の右側に新しい Free Space が追加される場合，すなわち $x_i + w_i < X_j + W_j$ の時，座標 $(x_i + w_i, Y_j, Z_j)$，大きさ $\{X_j + W_j - (x_i + w_i), H_j, L_j\}$ の Free Space が追加

(iii) Free Space の下側に新しい Free Space が追加される場合，すなわち $Y_j < y_i$ の時，座標 (X_j, Y_j, Z_j)，大きさ $(W_j, y_i - Y_j, L_j)$ の Free Space が追加

(iv) Free Space の上側に新しい Free Space が追加される場合，すなわち $y_i + h_i < Y_j + h_j$ の時，座標 $(X_j, y_i + h_i, Z_j)$，大きさ $\{W_j, Y_j + H_j - (y_i + h_i), L_j\}$ の Free Space が追加

(v) Free Space の奥側に新しい Free Space が追加される場合，すなわち $Z_j < z_i$ の時，座標 (X_j, Y_j, Z_j)，大きさ $(W_j, H_j, z_i - Z_j)$ の Free Space が追加

(vi) Free Space の手前側に新しい Free Space が追加される場合，すなわち $z_i + l_i < Z_j + L_j$ の時，座標 $(X_j, Y_j, z_i - l_i)$，大きさ $\{W_j, H_j, Z_j + L_j - (z_i + l_i)\}$ の Free Space が追加

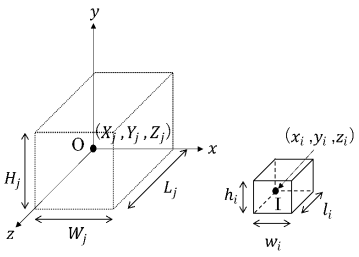

Fig. 6: A Free Space and an item in 3D case.

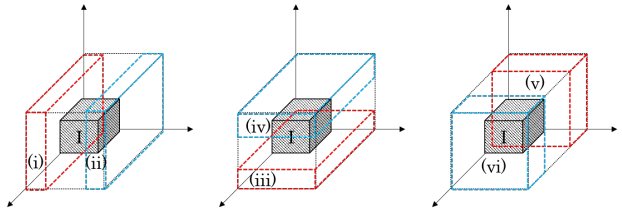

Fig. 7: Generate Free Space in 3D case.

4.2 Free Space と DBL 法の組み合わせ

4.1 節のアイディアを元に，DBL 法と組み合わせた 3L-CVRP のためのパッキングアルゴリズム（以降 FS-DBL 法と呼ぶ）を提案する．アルゴリズムの流れは Algorithm1 の通りである．

ある車両 1 台において，m 個の荷物 $I_k (k = 1, \cdots, m)$ の積載を行うとする．まず，各荷物の積載順序を順列 $\sigma_o (o = 1, \cdots, m)$ とする．積載順序の決定方法は 4.4 節で述べる．既配置荷物集合を $PlacedItem$，その積載長を λ とし，最初の $PlacedItem = NULL$，$\lambda = 0$ とする（line1,2）．また，Free Space の集合を FS とし，最初の $FS = \{Container\}$ とする（line1,2）．ここで，$Container$ は大きさ (W, H, ∞) の直方体である．FS 内では Free Space は座標の DBL 順にソートされる．

順列 σ_o に従って荷物 I_k を選択し，配置を試みる（line3,4）．荷物 I_k は 2 章の条件 (a) の条件のように回転できるため，各 I_k に対して，大きさ $(w_I k, h_I k, l_I k)$ と $(l_I k, h_I k, w_I k)$ の 2 つが試されることとなる（line7）．集合 FS 内を順に見ていき，I_k を配置することが可能な Free Space（すべての次元において I_k よりも大きい Free Space に対し）があれば，その Free Space の座標に I_k の配置を試みる（line5,6）．つまり，この手法において荷物を配置する候補点は Free Space の座標集合である．Free Space は座標の DBL 順にソートされているため，最初に見つかった Free Space が I_k の DBL 点である．

空間的に配置可能な場合（line8），2 章 (b)〜(d) で定義した積載制約を確認する（line9 -14）．まず，すべての既配置荷物 $P_j \in PlacedItem (j = 0, \cdots, \leq m)$ に対し，I_k の上面と P_j の底面が接するならばその面積 $Area$ を計算する（line10）．$Area > 0$ ならば，I_k と P_j の脆弱性 f_{I_k} と f_{P_j} を確認し，$f_{I_k} = 0$ かつ $f_{P_j} = 1$ の時はその点において脆弱性条件を満たしていないため，次の候補点に移る（line12）．脆弱性条件を満たしているならば，$Area$ の値を支持領域 $\bar{A}$ に加算する．

次に，LIFO 制約を確認する（line10）．I_k を受け取る顧客が P_j を受け取る顧客よりも先に訪問される場合，I_k を上または手前に平行移動した際に P_j と重なる場合は LIFO 制約を満たさないものとして，次の候補点に移る．同様に，P_j を受け取る顧客が I_k を受け取る顧客よりも先に訪問される場合，P_j を上または手前に平行移動した際に I_k と重なる場合は LIFO 制約を満たさないものとして，次の候補点に移る（line12）．

p 個の $PlacedItem$ すべてに対して制約確認を行った後，支持領域 $\bar{A}$ を確認する（line15）．$\bar{A} \geq a w_{I_k} l_{I_k}$（$a$ はあらかじめ設定された $0 \leq a \leq 1$ のパラメータ，2 章の条件 c 参照）を満たす場合にのみ支持領域を満たすものとする．条件を満たさない場合には，次の候補点に移る（line19）．

制約確認をすべて満たした候補点は実行可能な点であるため，これを配置位置として決定し（line17），I_k を $PlacedItem$ 内に追加する（line24）．同時に積載長 λ の値も更新される．また，I_k の配置によって FS が更新される（line25）．更新方法は 4.1 節で述べている．

すべての荷物の配置が行われる，またはどこにも配置できない荷物が存在する場合アルゴリズムは終了し，積載長 λ を返す（line30）．すべての荷物を配置できなかったときは，$\lambda = 2L$（車両の長さの 2 倍の値）とす

Algorithm 1 FS-DBL algorithm

```
1: Let FS be the Free Spaces
   Let PI be the placed items
   Let BN be the best neighborhood operation
2: FS = {Container}
   PI = NULL
   σ_o = Change_Item_Order()
   λ = 0
3: for c_1 = 1 to num of item do
4:   select item I_k ∈ σ_o
5:   for c_2 = 1 to num of Free Space do
6:     select Free Space ∈ FS
7:     for each orientation of item I_k do
8:       if I_k is possible to put in Free Space then
9:         for c_3 = 1 to num of Placed Item do
10:          Check fragility and LIFO constraint
             Calculate Supporting Area
11:          if constraint is not satisfied then
12:            goto line 6 to try next Free Space
13:          end if
14:        end for
15:        Check Supporting Area constraint.
16:        if All constraint is satisfied then
17:          goto 22 to place item
18:        else
19:          goto line 6 to try next Free Space
20:        end if
21:      end if
22:    end for
23:  end for
24:  Place item I_k and add to PI
25:  Update λ and FS
26: end for
27: if There are the items can't be placed then
28:   λ = 2L
29: end if
30: return λ
```

る（line27,28）.

4.3 Free Space と MTA 法の組み合わせ

4.1 節のアイディアを元に，MTA 法と組み合わせた 3L-CVRP のためのパッキングアルゴリズム（以降 FS-MTA 法と呼ぶ）を提案する．基本的な流れは FS-DBL 法と同様であるが，候補点の数を (i) 物体 I が Free Space と左面・下面・奥面の 3 つで接するような点，(ii) 物体 I が Free Space と右面・下面・奥面の 3 つで接するような点，(iii) 物体 I が Free Space と左面・下面・手前面の 3 つで接するような点，(iv) 物体 I が Free Space と右面・下面・手前面の 3 つで接するような点の 4 つ（Fig. 8 参照）に増やしている．これは，MTA 法では DBL 順序を厳密に守る必要はなく，Free Space の各面は何らかの面と接している場合が多いからである．安定性制約を考慮すると，Free Space 内ではこの 4 つの候補点だけで十分に MTA 法に適用できると考える．また，MTA 法において必要な接地面積の計算は，制約計算の時に同時に計算される．

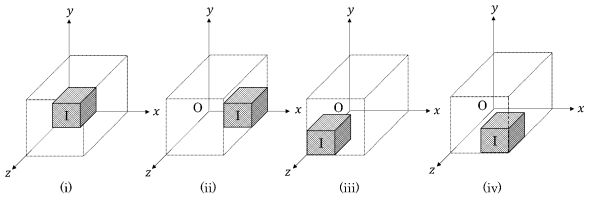

Fig. 8: Candidate positions in 3D Free Space for FS-MTA algorithm.

4.4 積載順序探索

Wenbin ら [4] は初期積載順序として，(i) 顧客の配送順，(ii) 脆弱性条件順，(iii) 荷物の体積順にソートしたものを用いている．また，パッキングが行われた後に積載長が違反しているならば，この順序のうち 2 つをランダムに入れ替えて再度パッキングを行うことを繰り返し，実行可能解となる積載順序の局所探索を行っている．Wenbin らの局所探索は積載順序の入れ替えを K 回繰り返し，それぞれの順序で DBL 法を行う．積載順序は初期順序から局所探索を行うことによって得られる．積載順序のいずれかがすべての荷物を詰め込める実行可能な積載計画へつながる場合，局所探索は直ちに停止する．DBL 法で実行可能解を見つけられなければ，MTA 法についても同様に積載順序の入れ替えを K 回繰り返し，それぞれの順序で MTA 法を行う．積載順序の局所探索を行う上での近傍オペレータは，ランダムに選択された荷物のペアを交換する，という操作のみである．この局所探索において評価関数は用いず，選択されたペアに対して必ず交換操作が行われ，次の解へ移動する．K の値はパッキングの難しさに基づいて決定される．Wenbin らの実装では，積載空間の体積に対する荷物の総体積の割合によって，積載の困難さを推定する．r = 荷物の総体積/積載空間の体積 とし，$r \leq 0.6$ のとき $K = 150$，そうでなければ $K = max\{5, 75(1-r)\}$ とする．

本手法でも同様の方法を用いて積載順序を変更しながら，FS-DBL アルゴリズムおよび FS-MTA アルゴリズムを繰り返し呼び出すことによって実行可能な積載を探索している．

さらに，Random Partial Neighborhood Search (RPNS)[5] を用いた順序探索手法を新たに提案し，二つの積載順序探索の手法を比較する．RPNS は部分探索のアイディアを導入した局所探索手法である．各反復においてすべての近傍を探索するのではなく，事前に決めておいた近傍の一部を用いることで計算量を削減しようという考え方である．このアイディアは計算量の削減のほかに探索の多様性を持たせるという効果もあるため，適切な部分近傍のサイズを定めることで探索の集中化と多様化のバランスをとることができる．

RPNS を用いたパッキングアルゴリズム全体の流れを説明する．FS-DBL 法に適用したものを Algorithm2 に示す．MTA 法についても同様に処理が行われる．まず (i) 顧客の配送順，(ii) 脆弱性条件順，(iii) 荷物の体積順にソートしたものを初期積載順序とし，積載長 λ および RPNS による部分近傍のサイズ $NumOfSwap$ を定める（line1）．初期積載順序に対し，DBL 法での積載を試す（line2）．実行不可能解ならば RPNS による積載順序探索を行う（line3 -23）．まず，σ_o の中から I_a，I_b を

Algorithm 2 packing algorithm using RPNS

1: $\sigma_o = Sort\_Items()$
$\lambda = 0$
$Num\ Of\ Swap$
$= \frac{1}{2}\{2m - 1 - \sqrt{4m(m-1)(1-ratio+1)}\}$
2: $\lambda = FS - DBL(\sigma_o)$
3: **if** $\lambda > L$ **then**
4: **for** $iteration = 1$ to $Maxiter$ **do**
5: **for** $c_1 = 1$ to $Num\ Of\ Swap$ **do**
6: select item $I_a \in \sigma_o$
7: **for** $c_2 = 1$ to $num\ of\ item$ **do**
8: select item $I_b \in \sigma_o$
9: $\acute{\sigma_o} \leftarrow$ Change order σ_o using $Swap(I_a, I_b)$
10: $\acute{\lambda} = FS - DBL(\acute{\sigma_o})$
11: **if** $\acute{\lambda} \leq L$ **then**
12: $\lambda = \acute{\lambda}$
13: go to line 24
14: **else if** $\acute{\lambda} < \lambda_{best}$ **then**
15: $\lambda_{best} = \acute{\lambda}$
16: $\sigma_{best} = \acute{\sigma_o}$
17: **end if**
18: **end for**
19: **end for**
20: $\sigma_o = \sigma_{best}$
21: $\lambda = \lambda_{best}$
22: **end for**
23: **end if**
24: **return** λ

選択する．このとき，I_a は σ_o の中から $NumOfSwap$ 個選ばれる（line5）．荷物ペア (I_a, I_b) の積載順序を交換した順序を $\acute{\sigma_o}$ とし，この順序 $\acute{\sigma_o}$ に対して DBL 法での積載を試し，その積載長を $\acute{\lambda}$ とする（line10）．$\acute{\lambda} < L$ すなわち実行可能解の時 $\lambda = \acute{\lambda}$ として探索を終了する（line12,13）．また $\acute{\lambda}$ が現在のイテレーション中の最良積載長 λ_{best} よりも短いならば λ_{best} とそれを得られた順序 σ_{best} を保存する（line15,16）．実行可能解が見つかるか指定イテレーションに達した時，現在見つかっている最良の積載長 λ_{best} を返り値とし，アルゴリズムは終了する（line24）．

5 経路最適化のためのタブー探索

5.1 初期解生成

初期解生成は，Wenbin ら [4] の手法をベースとしている．まず，1 つの経路ごとに 1 人の顧客となる経路を作成し，それ以上の経路の結合が行えなくなるまで 2 つの経路を 1 つに結合することを繰り返す．各反復では，総移動距離が最も短くなるような 2 つの経路の組が選択され，結合される．1 段階目の結合では，後に得られる経路が実行可能であるならば，つまりすべての荷物の合計重量が車両の重量容量を超えず，積載制約に違反することなくすべての荷物を車両に積み込むことができる場合にのみ結合は受け入れられる．1 段階目の結合の後，経路数が利用可能な車両の数を超えているならば，次の段階の結合を開始する．2 段階目の結合では，経路内の荷物の総重量が車両の容量を超え，積載の長さが車両の長さを超えることを許す．ただし，先行手法では総移動距離が最も短くなるような 2 つの経路の組が選択されていたが，本手法では積載長の違反量が最も少なくなるものを選択し，違反量が同じなら総移動距離が最も短くなるものが選ばれるようにした．2 段階目の結合は，経路数が利用可能な車両数と一致した場合に終了する

5.2 近傍オペレータ

近傍オペレータには VRP において代表的な挿入近傍，交換近傍，2-opt 近傍の 3 つを用いる．挿入近傍は，現在の経路から 2 人の顧客を選択し，ある顧客の訪問順序をもう一人の顧客の後ろに移動して得られる経路集合である．交換近傍は，現在の経路から 2 人の顧客を選択し，それぞれの配送順序を入れ替えることによって得られる経路集合である．2-opt 近傍は経路に含まれる 2 辺を切り，繋ぎ変えることによって得られる経路集合である．2-opt 近傍では，繋ぎ変える際に配送順序が逆になる顧客が存在するパターンがある．これらの近傍オペレータは同一経路内だけではなく，2 経路間でも行われる．また近傍オペレータによっては経路数を増減することができるが，本手法ではそのような操作は行わないものとする．

5.3 タブー探索

4.1 節で得られた初期解をもとにタブー探索による総移動距離の最小化を行う．提案手法においては実行不可能解への移動を常に許容して探索を行う．つまり，重量容量制約を超えるような荷物を配送する経路や，車両内にすべての荷物を積み込むことができない経路も許容される．経路の変更には 5.2 節の近傍オペレータを用いる．ある顧客 i が選択されると，i から近い順に N_{near} 人（$N_{near} < n : n$ は顧客数）の顧客 $j(j \neq i)$ が i のペア候補となり，N_{near} 通りのペア (i, j) が生成される．これを 1 イテレーションにつき N_{max} 人 $(N_{max} \leq n)$，すなわち $N_{max} * N_{near}$ 通りの顧客ペアが生成される．それぞれの顧客ペアに対し挿入近傍・交換近傍・2-opt 近傍が試される．挿入近傍では i を j の後に挿入する近傍と，j を i の後に挿入する近傍を試す．また，ペア (i, j) の組み合わせによっては，経路数の増減をしてしまうため近傍操作が行われない場合がある．近傍操作による変更後の経路は以下の評価値で評価される．

$$objectivevalue = (routelength) + \alpha(excessweight) + \beta(excesslength) \quad (2)$$

ここで，$routelength$ は総移動距離，$excessweight$ はすべての経路の重量制約違反量の合計，$excesslength$ はすべての経路の積載制約違反量の合計，α,β は各違反量に対する重みづけである．積載制約の違反量は 4 章のパッキングアルゴリズムによって得られる．近傍操作によって積載が変更される（経路内の訪問順序が変わる）経路のみ，パッキングアルゴリズムによって違反量を再計算する．しかし，経路内の荷物の総体積 v_{sum} が車両の体積 V を超える場合は，配置できないことが自明であるためパッキングアルゴリズムは呼び出されず，その経路の違反量は $2L * v_{sum}/V$ とする．1 イテレーションあたり最大 $(N_{max} * N_{near} * 4)$ 通りの近傍解が生成され，それら全てを (2) 式で評価する．近傍解の中で評価値が最小となるものが選択され解の移動が行われる．しかし，タブーリスト内に存在する近

傍解は選択できない．ただし，現在の最良解を更新できる場合にのみタブーリストを無視して選択することができる．タブーリストには選択された顧客ペアが保存されており，同じ顧客ペアによる近傍操作が繰り返せないようになっている．タブーリストは挿入・交換近傍用と 2-opt 近傍用の 2 種類を用いている．

近傍解が選択され，解の移動が行われると同時にタブーリストが更新される．また，移動した解の総移動距離が最良解よりも短く，実行可能解である場合には最良解を更新する．これを指定イテレーションに達するまで繰り返すとタブー探索は終了し，最良解となる経路およびそれぞれの経路における積載を返す．

6 提案手法の性能評価実験

4 章および 5 章で述べた提案手法をもとにアルゴリズムを実装した．また，比較のために Wenbin ら [4] のパッキングアルゴリズムにおいて高速化の工夫を行っていないものを実装した．各アルゴリズムは C++でコーディングし，g++コンパイラを使用してコンパイルおよび最適化した．実装したアルゴリズムの性能を評価するために，Gendreau ら [1] が提案した 27 個のベンチマーク問題（G27 と呼ぶ）を用いる．この問題は以下のホームページ http://www.or.deis.unibo.it/research_pages/ORinstances/ORinstances.html で公開されている．この問題に対し，提案手法で積載順序探索をランダムな交換のみにした実装（FS-TS），提案手法で積載順序探索を RPNS とした実装（FS-TS-RPNS），比較用に実装した Wenbin らの手法の単純な実装（DMTS-simple），および先行研究の実装 TS[1]，GTS[2]，ACO[3]，DMTS[4] で比較を行う．実装したプログラムは，OS が Ubuntsu16.04，CPU が Intel Core i7-4790@3.60GHz の Linux PC で実行した．実験 1 として，提案手法 FS-TS，FS-TS-RPNS と DMTS-simple による比較を行う．実験 2 として，提案手法 FS-TS-RPNS と先行手法 TS[1]，GTS[2]，ACO[3]，DMTS[4] の比較を行う．

6.1 実験 1

実験 1 の結果を Table1 に示す．z は総移動距離，t(s) は最良解が得られた時間，tt(s) は総実行時間を表しており，各数値は 10 試行の平均値である．Table 1 において，最も総移動距離の短い解を見つけられたのは DMTS-Simple であった．これは，提案手法におけるパッキングにおいて，Wenbin らが見つけることのできたパッキングを見つけられず，実行可能な経路の選択肢が狭まっていることが原因であると考える．一方で，本提案手法における狙いである探索の効率化は効果的で，計算時間を 10 倍以上削減できている．また Table 2 に，実際にパッキング中に探索した候補点の数を示す．この表から探索した候補点の数を大幅に削減できていることが分かる．

	z	t(s)	tt(s)
FS-TS-RPNS	980.15	1735.20	2450.18
FS-TS	983.55	1711.34	2361.26
DMTS-simple	973.34	24012.03	34714.16

Table 1: Performance of proposed method.

	DBL 法 1 回あたりの探索した候補点の数	MTA 法 1 回あたりの探索した候補点の数
FS-TS-RPNS	148	230
FS-TS	109	150
DMTS-simple	708	3729

Table 2: Number of candidate points.

6.2 実験 2

実験 2 の結果を Table3 に示す．この結果から提案手法は先行手法 TS[1]，GTS[2] よりは優れているが，ACO[3]，DMTS[4] には劣っていることが分かる．この理由として，より良い解を見つけるために必要な候補点も削減してしまっている可能性が挙げられる．また，Wenbin らのように問題の特徴に合わせた高速化の工夫を取り入れ，さらなる改善を行う必要がある．

	z	t(s)	tt(s)
FS-TS-RPNS	978.59	1693.8	2464.8
TS[1]	1042.26	2058.9	4200.0
GTS[2]	997.18	1471.6	2415.9
ACO[3]	966.67	1746.5	1793.1
DMTS[4]	962.08	1419.9	2252.2

Table 3: Comparison with previous research.

7 まとめと今後の課題

3L-CVRP は，VRP をより現実問題に近づけるためのモデルとして，3 次元での荷物の積み込みを考慮したものであり，2 つの NP 困難な問題の組み合わせである．本研究ではこの問題に対し，従来手法とは違ったアプローチによるパッキングを行うことにより，解法の効率化を図った．提案した手法に対してアルゴリズムを実装し比較実験を行ったところ，狙い通りの効率化を行うことができ，いくつかの手法には勝る結果となった．しかし，より効率的な手法には劣っていたため，さらなる効率化の工夫が必要である．

今後の課題として，パッキングのさらなる改善が必要である．さらに，現実世界で扱うには需要に応じた様々な条件を取り扱うことのできる枠組みや実用に向けたユーザーインターフェースの構築も必要である．

参考文献

1) Gendreau M, Iori M, Laporte G, Martello S : A tabu search algorithm for routing and container loading problem, Transportation Science, 40, 342/350 (2006)

2) Tarantilis C.D, Zachariadis E.E, Kiranoudis C.T : A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container loading problem, IEEE Transactions on Intelligent Transportation Systems, 10, 255/271 (2009)

3) Fuellerer G, Doerner K.F, Hartl R.F, Iori M : Metaheuristic for vehicle routing problems with three-dimensional loading constraints, European Journal of Operation Research, 201, 751/759 (2010)

4) Wenbin Z, Hu Q, Andrew L, Lei W : A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3L-CVRP, Computers & Operations Research, 39, 2178/2195 (2012)

5) Yuichi Nagata, Isao Ono : Random Partial Neighborhood Search for University Course Timetabling Problem, Parallel Problem Solving from Nature, PPSN XIII, 782/791 (2014)