二段階配送計画問題に対するメタ戦略

修 士 学 位 論 文

二段階配送計画問題に対するメタ戦略

2021年度
(2022年3月)

東京海洋大学大学院
海洋科学技術研究科
海運ロジスティクス専攻

小野航生

修 士 学 位 論 文

二段階配送計画問題に対するメタ戦略

2021年度
(2022年3月)

東京海洋大学大学院
海洋科学技術研究科
海運ロジスティクス専攻

小野航生

目次

1　はじめに

現在, 日本においては宅配便取扱個数は年々増加傾向にあり, ２０２０年度の宅配便取扱個数は２０１９年度より１１.９％増加し４８億件を超えた[3]. このような状況において, 物流業界では労働者の高齢化等の理由による生産人口の減少が進んでおり, ドローン等を用いた無人機輸送を検討することが重要になると考えらえる.

配達車とドローンのような無人機を併用して配達を行うモデルの先行研究として, ハブ・スポーク配送計画問題が提案されている [5, 9, 10]. ハブ・スポーク配送計画問題はドローンが顧客点を巡回せず, 停留点から一つの顧客点に対してのみピストン配送を行うモデルである. Chang 氏と Lee 氏はドローンと配達車を用いたハブ・スポーク配送計画問題に対し, k-means 法を用いて顧客点をクラスタ分けし, クラスタごとに停留点を作成し, 各停留点からクラスタ内の顧客点にピストン配送を行う方法で配送経路のコストを最小化をするモデルを提案している [5]. このモデルは停留点を決める際に仮想区域内のいずれの場所にも停留点を作ることが可能であるという場合が想定されている. しかし, このようにして作られた停留点は山や川などの自然条件や個人の敷地といった理由で配達車が行くことが困難な場合があるため, 実際にこのモデルを適用することが難しいと予想される. 劉 楊成漢氏は各顧客点を停留点の候補として使用するモデルで配送経路のコストの最小化をするハブ・スポーク配送計画問題を提案している [10]. このモデルはすべての顧客点が停留点の候補になる場合が想定されている. しかし顧客点であっても路上駐車が禁止されている道や狭い道など, ドローンによるピストン配送が完了するまでの時間, 配達車が停車し続けることが困難である顧客点が存在する場合も想定される. これに対し, 瀧本氏らは停留点の候補点と顧客点を別々に持ったモデルを提案し, 反復局所探索法を用いて精度の高い解を得た [9].

本研究で提案する二段階配送計画問題は, 瀧本氏らが提案する停留点の候補点と顧客点を別々に持ったハブ・スポーク配送計画問題をさらに一般化したモデルである. ドローンが停留点と顧客点間をピストン配送するのではなく, 停留点を出発したドローンがいくつかの顧客点を巡回したのち, 同じ停留点に帰ってくるというモデルであり, 以下の 2 つの階層から構成される.

1 配達車はデポを出発し, いくつかの停留点を経由して, 再びデポに戻る.
2 配達車が停留点に止まっている間に, ドローンがいくつかの顧客点への配送を行い, 出発した停留点に再び戻る.

二段階配送計画問題は, このように配送車とドローンそれぞれの巡回路を求め, それらの総コストを最小化する問題である (図 1). 二段階配送計画問題は, ハブ・スポーク配送計画問題と同様に NP 困難な問題である.

二段階配送計画問題は無人機を使った配送以外にも応用が可能であり, その一つにバス停方式と呼ばれるトラックと台車を併用した配送方法がある [1]. バス停方式では, トラックで顧客に直接配達を行うのではなく, 人力で動かす台車によって配達を行うことで, トラックでは入れない細い道や, 集合住宅に対し配送を行なっている. また, 人力の台車を使うことにより, 配達によって生じる二酸化炭素の削減にもつながる.

本研究では, 配達車の容量制約, ドローンの容量制約を考慮した二段階配送計画問題に対

し, 割り当て問題を解き, 初期解を生成した. その後, ランダム多スタート法, 反復局所探索法, パスリリンキングの 3 種類のメタ戦略による解法を提案する. ランダム多スタート法において解を改善する局所探索法として異なる近傍を持つ insert-swap 近傍を用いた局所探索法, 1del-1ins 近傍を用いた局所探索法の 2 つの手法を使用する. 反復局所探索法では, 使用している停留点と使用していない停留点を入れ替える操作を kick として探索を行う. パスリリンキングでは 2 つの解を 1 つずつ近づけていく方法を提案し, また, 局所探索法を行う解を選定する基準を設けることで, パスリリンキングの高速化を行う. 計算実験を行うことでこれらの手法の有効性を確認する.

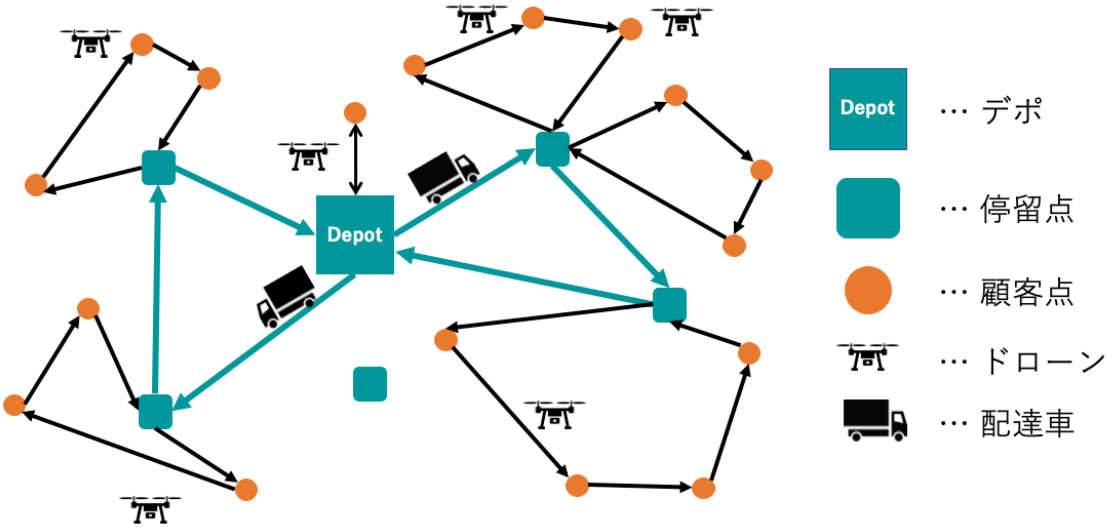

図 1: 二段階配送計画問題

2 問題定義

二段階配送計画問題を整数計画問題として定式化する. 2.1 節では集合の定義, 2.2 節では定数の定義, 2.3 節では変数の定義をする. 2.4 節では定式化を行う.

2.1 集合

$N = \{1, \ldots, n_{\mathrm{c}}\}$: 顧客の集合
$B = \{0, \ldots, n_{\mathrm{b}}\}$: 停留点およびデポの集合. 0 はデポを表す.
$V = \{1, \ldots, n_{\mathrm{v}}\}$: 配達車の集合
$P = \{1, \ldots, n_{\mathrm{p}}\}$: 各配達車に搭載されるドローンの集合

2.2 定数

c_{ij}: 点 i,j 間の移動コスト.
D_l: 顧客 l の需用量. 非負数とする.

W_{b}: 配達車の最大積載量
W_{d}: ドローンの最大積載量

2.3 変数

x_{ijk}: デポまたは停留点iとデポまたは停留点j間を配達車kが移動するとき1, そうでないとき0をとるバイナリ変数.
X_{lmkd}: 顧客点lと顧客点m間を配達車kに搭載されたドローンdが移動する場合は1, そうでないとき0をとるバイナリ変数.
$z^{(1)}_{ilkd}$: デポまたは停留点iと顧客点l間を配達車kに搭載されたドローンdが移動する場合は1, そうでない場合は0をとるバイナリ変数
$z^{(2)}_{likd}$: 顧客点lとデポまたは停留点i間を配達車kに搭載されたドローンdが移動する場合は1, そうでない場合は0をとるバイナリ変数
y_{ik}: 配達車kが停留点iを使用する場合は1, そうでない場合は0をとるバイナリ変数

2.4 定式化

一般的な配送計画問題[7]を拡張し, 二段階配送計画問題の定式化を行った.

$$\min \quad \sum_{i\in B}\sum_{j\in B}\sum_{k\in V} c_{ij}x_{ijk} + \sum_{l\in N}\sum_{m\in N}\sum_{k\in V}\sum_{d\in P} c_{lm}X_{lmkd} + \sum_{i\in B}\sum_{l\in N}\sum_{k\in V}\sum_{d\in P} c_{il}z^{(1)}_{ilkd} + \sum_{l\in N}\sum_{i\in B}\sum_{k\in V}\sum_{d\in P} c_{il}z^{(2)}_{likd} \tag{1}$$

$$\text{s.t.} \quad \sum_{m\in N}\sum_{k\in V}\sum_{d\in P} X_{mlkd} + \sum_{i\in B}\sum_{k\in V}\sum_{d\in P} z^{(1)}_{ilkd} = 1 \quad (\forall l \in N) \tag{2}$$

$$\sum_{m\in N}\sum_{k\in V}\sum_{d\in P} X_{lmkd} + \sum_{i\in B}\sum_{k\in V}\sum_{d\in P} z^{(2)}_{likd} = 1 \quad (\forall l \in N) \tag{3}$$

$$\sum_{l\in N} z^{(1)}_{ilkd} = \sum_{l\in N} z^{(2)}_{likd} = y_{ik} \quad (\forall i \in B, \forall k \in V, \forall d \in P) \tag{4}$$

$$\sum_{l\in}\sum_{m\in N\setminus S} X_{lmkd} + \sum_{l\in S}\sum_{i\in B} z^{(2)}_{likd} \geq 1 \quad (\forall S \subseteq N, \forall k \in V, \forall d \in P) \tag{5}$$

$$\sum_{j\in B} x_{0jk} = 1 \quad (\forall k \in V) \tag{6}$$

$$\sum_{j\in B} x_{ijk} = \sum_{j\in B} x_{jik} = y_{ik} \quad (\forall i \in B, \forall k \in V) \tag{7}$$

$$\sum_{i\in S}\sum_{j\in B\setminus S} x_{ijk} \geq y_{hk} \quad (\forall S \subseteq B \setminus \{0\}, \forall h \in S, \forall k \in V) \tag{8}$$

$$\sum_{l\in N}\sum_{m\in N}\sum_{d\in P} D_m X_{lmkd} + \sum_{i\in B}\sum_{l\in N}\sum_{d\in P} D_l z^{(1)}_{ilkd} \leq W_{\mathrm{b}} \quad (\forall k \in V) \tag{9}$$

$$\sum_{l\in N}\sum_{m\in N} D_m X_{lmkd} + \sum_{i\in B}\sum_{l\in N} D_l z^{(1)}_{ilkd} \leq W_{\mathrm{d}} \quad (\forall k \in V, \forall d \in P) \tag{10}$$

$$x_{ijk} \in \{0,1\} \quad (\forall i \in B, \forall j \in B, \forall k \in V) \tag{11}$$

$$X_{lmkd} \in \{0,1\} \quad (\forall l \in N, \forall m \in N, \forall k \in V, \forall d \in P) \tag{12}$$

$$z^{(1)}_{ilkd} \in \{0,1\} \quad (\forall i \in B, \forall l \in N, \forall k \in V, \forall d \in P) \tag{13}$$

$$z^{(2)}_{likd} \in \{0,1\} \quad (\forall l \in N, \forall i \in B, \forall k \in V, \forall d \in P) \tag{14}$$

$$y_{ik} \in \{0,1\} \quad (\forall i \in B, \forall k \in V) \tag{15}$$

目的関数(1)は配達車とドローンの総移動距離である. 式(2)は顧客点には必ず1つの停留点または顧客点から1台のドローンが入ることを示す制約である. 式(3)は顧客点から必ず1つの停留点または顧客点に1台のドローンが出ていくことを示す制約である. 式(4)はある停留点iから出ていくドローンとある停留点iに入ってくるドローンが同じものであるという制約である. 式(5)はドローンの部分閉路除去制約である. 式(6)は配達車は必ずデポからいずれかの停留点に行かなければならないという制約である. 式(7)は停留点iを使用するならば、停留点iに入る配達車と停留点iから出ていく配達車は1台であるという制約である. 式(8)は配達車の部分閉路を除去するための制約である. 式(9)は配達車の容量制約である. 式(10)はドローンの容量制約である. 式(11), 式(12), 式(13), 式(14), 式(15)はバイナリ制約である.

3　初期解生成

二段階配送計画問題の解を決定するには

- どの配達車にどの停留点を割り当てるか
- どの停留点にどのドローンを割り当てるか
- どのドローンにどの顧客点を割り当てるか

の3つの割り当ておよび,

- 配達車のルート
- ドローンのルート

の決定が必要である. ただし, 配達車とドローンのルートはそれぞれ容量制約を満たさなければならない. 3.1節では割り当て問題の定式化を行い, 3.2節では割り当て問題の結果から, 初期解を生成する方法について述べる.

3.1　割り当て問題

3つの割り当てを決定するために以下の割り当て問題を解く.

顧客点をk-means法を用いて, 配達車の台数と同じクラスタ数になるようにクラスタリングし, それぞれのクラスタの重心を種点とした. $B_{\text{use}}(\subseteq B)$は停留点集合から停留点をランダムに選び作成した. 割り当て問題の定式化は以下のように行った.

3.1.1　集合

$C = \{1, \ldots, n_{\text{c}}\}$: 顧客の集合
$B_{\text{use}} = \{0, \ldots, n'_{\text{b}}\}$: 今回使用する停留点およびデポの集合. 0はデポを表す.
$V = \{1, \ldots, n_{\text{v}}\}$: 配達車の集合
$P = \{1, \ldots, n_{\text{p}}\}$: 各配達車に搭載されるドローンの集合
$\text{Seed} = \{1, \ldots, n_{\text{s}}\}$: 種点の集合

3.1.2　定数

c_{ij}: 点i,j間の移動コスト.
D_l: 顧客lの需用量. 非負数とする.
W_{b}: 配達車の最大積載量
W_{d}: ドローンの最大積載量

3.1.3　変数

x_{ijp}: 顧客iを停留点jから出発するドローンpに割り当てるとき1, そうでないとき0をとるバイナリ変数.

y_{ls}: 停留点lを種点sに割り当てるとき1, そうでないとき0をとるバイナリ変数.
z_{is}: 顧客iを種点sに割り当てるとき1, そうでないとき0をとるバイナリ変数.

3.1.4 割り当て問題の定式化

$$\min \quad \sum_{i\in C}\sum_{j\in B_{\text{use}}}\sum_{p\in P} c_{ij}x_{ijp} + \sum_{j\in B_{\text{use}}}\sum_{s\in \text{Seed}} c_{js}y_{js} \tag{16}$$

$$\text{s.t.} \quad \sum_{j\in B_{\text{use}}}\sum_{p\in P} x_{ijp} = 1 \quad (\forall i \in C) \tag{17}$$

$$\sum_{s\in \text{Seed}} y_{js} \leq 1 \quad (\forall j \in B_{\text{use}}) \tag{18}$$

$$\sum_{p\in P} x_{ijp} \leq \sum_{s\in \text{Seed}} y_{js} \quad (\forall i \in C, \forall j \in B_{\text{use}}) \tag{19}$$

$$\sum_{p\in P} x_{ijp} + y_{js} \leq z_{is} + 1 \quad (\forall i \in C, \forall j \in B, \forall s \in \text{Seed}) \tag{20}$$

$$\sum_{s\in seed} z_{is} = 1 \quad (\forall i \in C) \tag{21}$$

$$\sum_{j\in B} y_{js} \geq z_{is} \quad (\forall i \in C, \forall s \in Seed) \tag{22}$$

$$\sum_{i\in C} x_{ijp} D_i \leq W_{\text{d}} \quad (\forall j \in B_{\text{use}}, \forall p \in P) \tag{23}$$

$$\sum_{i\in C} z_{is} D_i \leq W_{\text{b}} \quad (\forall s \in \text{Seed}) \tag{24}$$

$$x_{ijp} \in \{0,1\} \quad (i \in C, j \in B_{\text{use}}, p \in P) \tag{25}$$

$$y_{bs} \in \{0,1\} \quad (b \in B_{\text{use}}, s \in Seed) \tag{26}$$

$$z_{is} \in \{0,1\} \quad (i \in C, s \in Seed) \tag{27}$$

目的関数(16)は顧客点を停留点から出発するドローンに割り当てるコストと、停留点を種点に割り当てるコストの合計を表す. 式(17)は顧客点が必ず1つの「ある停留点から出発するドローン」に割り当てられることを示す制約である. 式(18)は停留点は必ず1つ以下の種点に割り当てられることを示す制約である. 式(19)は顧客点をある停留点から出発するドローンに割り当てる場合、その停留点はいずれかの種点に割り当てられるという制約である. 式(20)は顧客iが停留点jに割り当てられ、停留点jが種点sに割り当てられている時, 顧客点iは種点sに割り当てられるという制約である. 式(21)は顧客は必ず1つの種点に割り当てられるという制約である. 式(22)はいずれかの顧客が種点sに割り当てられている時, その種点に割り当てられた停留点が存在しなくてはならないという制約である. 式(23)はドローンの容量制約である. 式(24)は配達車の容量制約である. 式(25), 式(26), 式(27)はバイナリ制約である.

3.2 初期解生成法

節 3.1 の割り当て問題を MIP ソルバの GUROBI[2] を使用して求解する. 割り当て問題を解くことで, 図 3.2.1 に示すように, 顧客点をどのドローンに割り当てるか, ドローンをどの停留点に割り当てるか, 停留点をどの種点に割り当てるかを決定することができる.

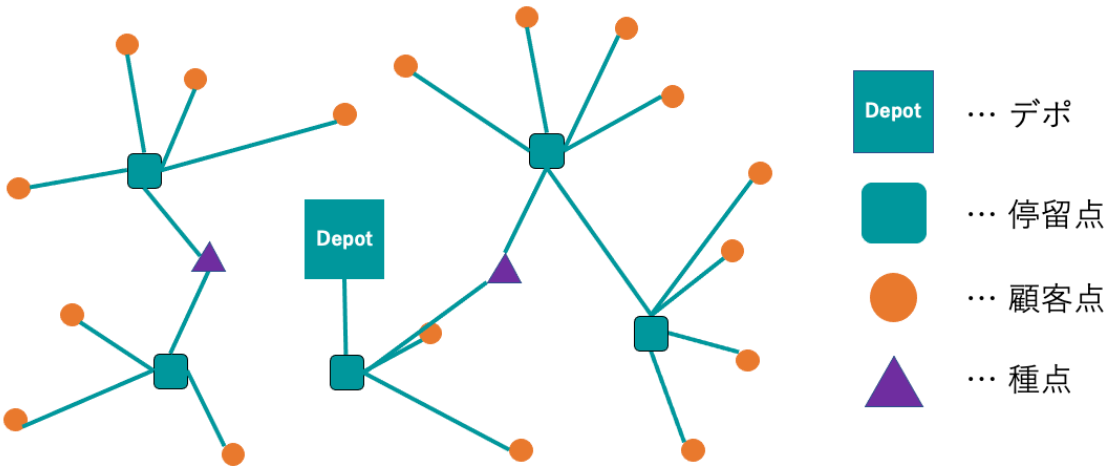

図 3.2.1: 割り当て問題の求解

割り当て問題を求解した後,

- デポを起点に, 各種点に割り当てられている停留点を巡回するルート
- 各停留点から出発する各ドローンに割り当てられている顧客点を巡回するルート

をそれぞれを巡回セールスマン問題として求解することで, 図 3.2.2 に示すような実行可能解を生成することができる.

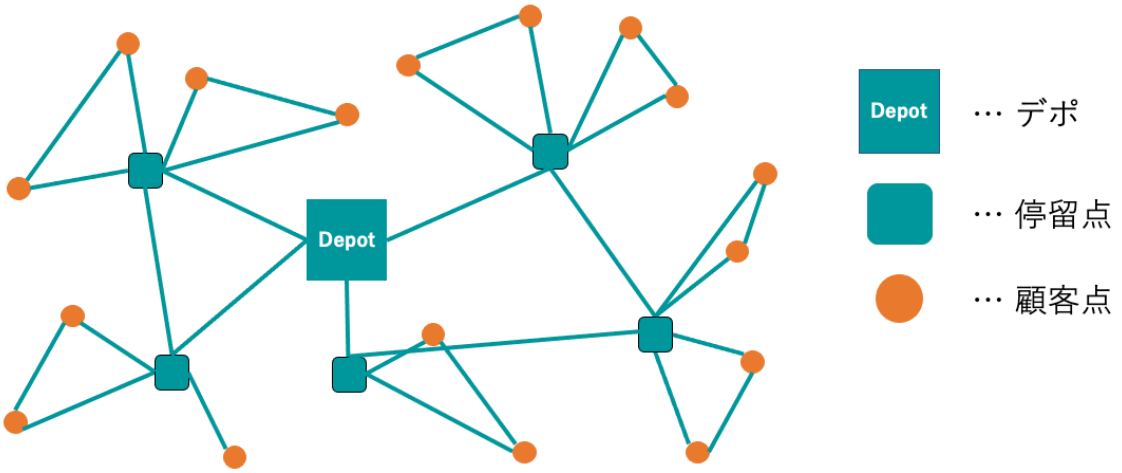

図 3.2.2: 初期解生成

この方法で初期解を生成する場合, B_{use} を変更することによって, 様々な初期解を生成す

ることが可能になる.

4 局所探索法

局所探索法とは, 初期解に対し, その近傍内により良い解があった場合, その解に移動するということを繰す手法である. この際, 近傍内に改善解が存在しない場合, その解を出力する. NP 困難の問題に対しては, 一般的に現実的な計算時間内で最適解を求解することが困難である. そのような問題に対し, 局所探索法を適用することは有効な手段となる [4]. 本研究では, insert-swap 近傍を用いた局所探索法と 1del-1ins 近傍を用いた局所探索法の 2 つの局所探索法を使用している. 本章では 2 つの局所探索法, insert-swap 近傍を用いた局所探索法と 1del-1ins 近傍を用いた局所探索法のアルゴリズムを説明をする.

4.1 解の評価値

局所探索法において解を評価する際, 実行可能解のみに限定して探索を行うと探索空間が狭くなってしまい, 有効な探索を行えない場合がある. そこで本研究では局所探索法を行う際, 実行不可能な解に対しては評価値にペナルティを加えて評価した. ペナルティは以下のように計算した.

4.1.1 配達車の容量制約を破った場合

ϕ_k を配達車 $k(k \in V)$ のペナルティとする. α は問題例によって変更するハイパーパラメータとした. ϕ_k は以下のように決定する.

$\sum_{l\in N}\sum_{m\in N}\sum_{d\in P} D_m X_{lmkd} + \sum_{i\in B}\sum_{l\in N}\sum_{d\in P} D_l z^{(1)}_{ilkd} \leq W_{\mathrm{b}}$ のとき : $\phi_k = 0$
$\sum_{l\in N}\sum_{m\in N}\sum_{d\in P} D_m X_{lmkd} + \sum_{i\in B}\sum_{l\in N}\sum_{d\in P} D_l z^{(1)}_{ilkd} > W_{\mathrm{b}}$ のとき :
$\phi_k = (\sum_{l\in N}\sum_{m\in N}\sum_{d\in P} D_m X_{lmkd} + \sum_{i\in B}\sum_{l\in N}\sum_{d\in P} D_l z^{(1)}_{ilkd} - W_{\mathrm{b}})\alpha$

4.1.2 ドローンの容量制約を破った場合

ψ_d をドローン $d(d \in P)$ のペナルティとする. β は問題例によって変更するハイパーパラメータとした. ψ_d は以下のように決定する.

$\sum_{l\in N}\sum_{m\in N} D_m X_{lmkd} + \sum_{i\in B}\sum_{l\in N} D_l z^{(1)}_{ilkd} \leq W_{\mathrm{d}}$ のとき : $\psi_d = 0$
$\sum_{l\in N}\sum_{m\in N} D_m X_{lmkd} + \sum_{i\in B}\sum_{l\in N} D_l z^{(1)}_{ilkd} > W_{\mathrm{d}}$ のとき :
$\psi_d = (\sum_{l\in N}\sum_{m\in N} D_m X_{lmkd} + \sum_{i\in B}\sum_{l\in N} D_l z^{(1)}_{ilkd} - W_{\mathrm{d}})\beta$

4.2 insert-swap 近傍を用いた局所探索法

insert-swap 近傍を用いた局所探索法では insert 近傍を用いた局所探索法と, swap 近傍を用いた局所探索法を組み合わせることにより局所探索法を行う. この節では insert 近傍を用い

た局所探索法と, swap近傍を用いた局所探索法のアルゴリズムについてそれぞれ説明する.

4.2.1 insert 近傍を用いた局所探索法

現在の解を$\sigma_{current}$として以下の操作を行う.

step:1 $\sigma_{current}$の顧客点を1つ選択し, 現在とは別のドローンの巡回路に挿入する. これをσ_{new}として評価する. この際, 配達車やドローンの容量制約を違反していた場合, ペナルティをコストに加算して解を評価する.

step:2 σ_{new}の評価値が, $\sigma_{current}$よりも良い評価値だった場合, $\sigma_{current} \leftarrow \sigma_{new}$とする. そうでなければ, 何もしない.

step1, step2の操作を改善がなくなるまで繰り返し, 改善がなくなったとき$\sigma_{current}$を出力する.

4.2.2 swap 近傍を用いた局所探索法

現在の解を$\sigma_{current}$として以下の操作を行う.

step:1 $\sigma_{current}$の顧客点を2つ選択し, 顧客A, 顧客Bとする.

step:2 顧客Aを顧客Bが所属していたドローンの順回路に挿入する. 同じように, 顧客Bを顧客Aが所属していたドローンの順回路に挿入する. 新しくできた解をσ_{new}とし, 評価する. この際, 配達車やドローンの容量制約を違反していた場合, ペナルティをコストに加算して解を評価する.

step:3 σ_{new}の評価値が, $\sigma_{current}$よりも良い評価値だった場合, σ_{new}を出力する. そうでなければ, step1に戻る. このとき, すでに全ての顧客の組み合わせに対しstep1, step2の操作を行っていた場合, 操作を終了し$\sigma_{current}$を出力する.

4.2.3 insert-swap 近傍を用いた局所探索法

現在の解を$\sigma_{current}$として以下の操作を行う.

step:1 $\sigma_{current}$に対して, insert近傍を用いた局所探索法を行う.

step:2 step1で求めたinsert近傍の局所最適解に対して, swap近傍を用いた局所探索法による改善を行う. このswap近傍を用いた局所探索法によって解を改善することができた場合, その解を$\sigma_{current}$としてstep1に戻る. そうでない場合, 操作を終了し, 解を出力する.

4.3 1del-1ins 近傍を用いた局所探索法

1del-1ins近傍を用いた局所探索法は, 現在の解が使用している停留点を1つ削除する1del操作, 現在の解が使用していない停留点を1つ挿入する1ins操作を繰り返し行う局所探索法である. この節では1del操作, 1ins操作及び, 1del-1ins近傍を用いた局所探索法のアルゴリズムについてそれぞれ説明する.

4.3.1 1del 操作

現在の解を$\sigma_{current}$とする. $\sigma_{current}$で使用している停留点の集合をB'_{use}に代入し, 以下の操作を行う.

step:1 停留点$b(\in B'_{\mathrm{use}})$をランダムに選び, $\sigma_{current}$の配達車のルートからbを取り除く. $B'_{\mathrm{use}} \leftarrow B'_{\mathrm{use}} \setminus \{b\}$とする.

step:2 bから運ばれていた顧客点を他のドローンのルートに挿入する. 新しくできた解をσ_{new}とし, ペナルティをコストに加算して評価する. σ_{new}が$\sigma_{current}$よりも良い評価値ならば, $\sigma_{current} \leftarrow \sigma_{new}$とする. そうでなければ, 何もしない.

step:2 $B'_{\mathrm{use}} = \emptyset$ならば操作を終了する. そうでなければstep1に戻る

4.3.2 1ins 操作

現在の解を$\sigma_{current}$とする. $\sigma_{current}$で使用していない停留点の集合をB'_{notuse}に代入し, 以下の操作を行う.

step:1 停留点$b(\in B'_{\mathrm{notuse}})$をランダムに選び, $\sigma_{current}$の配達車のルートにコストが最も小さくなるように挿入する. $B'_{\mathrm{notuse}} \leftarrow B'_{\mathrm{notuse}} \setminus \{b\}$とする.

step:2 step1の解に対しinsert-swap近傍を用いた局所探索法による改善を行い, 新しくできた解をσ_{new}とし, ペナルティをコストに加算して評価する. σ_{new}が$\sigma_{current}$よりも良い評価値ならば, $\sigma_{current} \leftarrow \sigma_{new}$とする. そうでなければ, 何もしない.

step:3 $B'_{\mathrm{use}} = \emptyset$ならば操作を終了する. そうでなければstep1に戻る

4.3.3 1del-1ins 近傍を用いた局所探索法

現在の解を$\sigma_{current}$として以下の操作を行う.

step:1 $\sigma_{current}$に対して, 1del操作を行う.

step:2 step1の解に対して1ins操作を行ったものをσ_{new}とする.

step:3 step1またはstep2で改善があった場合, $\sigma_{current} \leftarrow \sigma_{new}$として, step1に戻る. そうでない場合, $\sigma_{current}$を出力する.

5 メタ戦略

本研究では二段階配送計画問題に対して, ランダム多スタート法, 反復局所探索法, パスリリンキングの3つのメタ戦略での求解を行った.

5.1 ランダム多スタート法

複数の初期解から局所探索法を実行し, その結果として得られた局所最適解の中で, 最良のものを出力するアルゴリズムを多スタート局所探索法と呼ぶ. 初期解をランダムに生成する場合をとくにランダム多スタート法と呼ぶ[8]. この節では2つの局所探索法を用いて2種類のランダム多スタート法を提案する.

5.1.1 ランダム多スタート法-insert-swap

step:1 停留点集合Bからランダムに停留点を選び, B_{use}を作成する. この際, 選ぶ停留点の個数は上限値と下限値をハイパーパラメータとしてランダムに決定する.

step:2 step1で作成したB_{use}を使用して3.2節の手順で生成した初期解に対し, insert-swap近傍を用いた局所探索法での改善を行い, 実行可能解を生成する.

step1〜step2の操作を制限時間内で繰り返し, 最も結果の良かった解を出力する.

5.1.2 ランダム多スタート法-1del-1ins

step:1 停留点集合Bからランダムに停留点を選び, B_{use}を作成する. この際, 選ぶ停留点の個数は上限値と下限値をハイパーパラメータとしてランダムに決定する.

step:2 step:1で作成したB_{use}を使用して3.2節の手順で生成した初期解に対し, 1del-1ins近傍を用いた局所探索法での改善を行い, 実行可能解を生成する.

step1〜step2の操作を制限時間内で繰り返し, 最も結果の良かった解を出力する.

5.2 反復局所探索法

通常の局所探索法では局所最適解に到達した後, 新しい解に移動することはない. 局所最適解に”kick”という操作をし, 解を変形させた後, 局所探索法を行うという操作を繰り返すことにより, より広い解空間を探索できるようになる. これを反復局所探索法と呼ぶ[6].

5.2.1 kick

本研究では先行研究[9]を参考にkickを以下のように行った.

step:1 停留点b_1を, 現在使用している停留点の中からランダムに１つ選ぶ

step:2 停留点b_2を,「現在使用していない停留点」かつ「b_1から近い停留点」の中からランダムに１つ選ぶ. 該当する停留点がひとつもない場合、step1に戻る.

step:3 停留点b_1を現在の配達車のルートから削除し, b_1があった位置にb_2を入れる. b_1から配達していたドローンのルートを全てb_2に付け替える.

step1〜step3までの操作を、kick-num回行う. kick-numは上限値と下限値をハイパーパラメーターとして, ランダムに決定する. 本研究では上限値を3, 下限値を1とした.「b_1から近い停留点」については全ての停留点をb_1に近い順にソートしたリストを作成し, そのリストの上位10個の停留点という基準を設定した.

5.2.2 反復局所探索法のアルゴリズム

本研究では反復局所探索法を以下の手順で行なった.

step:1 $B_{\text{use}} = B$として3節の手順で初期解を生成した後, 1del-1ins近傍を用いた局所探索法により改善を行なったものを$\sigma_{current}$とする

step:2 $\sigma_{current}$に対し, kickを行い, 1del-1ins近傍を用いた局所探索法による改善を行ったものをσ_{kick}とする.

step:3 $\sigma_{current}$ と σ_{kick} のうち, どちらか評価値が良かった方を, $\sigma_{current}$ として step2 に戻る.

step2〜step3 を制限時間内で繰り返す.

5.3 パスリリンキング

パスリリンキングは一般的に以下のような手順で行う.

step:1 reference-set という解の集合を作成する.
step:2 reference-set から解を 2 つ選び, 一方の解をもう一方の解に 1 つずつ近づけていく.
step:3 step2 の過程で生成された解の集合に対し, それぞれ局所探索を行い評価値が良かったものを reference-set に加える. 新たに reference-set に加えた解の個数分だけ, 評価値の悪い解を reference-set から削除する.

step2〜step3 を制限時間内で繰り返す.

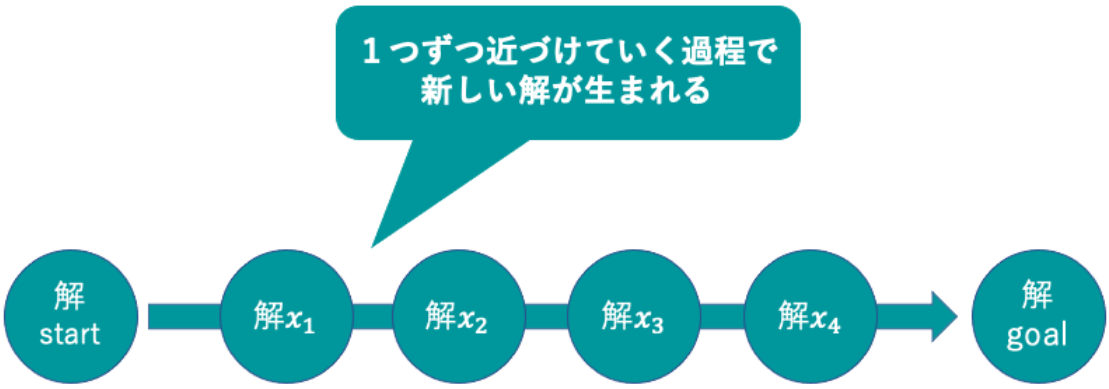

図 5.3: パスリリンキング

この節では,

- reference-set の作成方法
- 解の近づけ方
- パスリリンキングのアルゴリズム

について述べる.

5.3.1 reference-set の作成方法

まず, 空のリスト reference-set を作成し, 以下の手順により要素を追加した.

step:1 停留点集合 B からランダムに停留点を選び, B_{use} を作成する. この際, 選ぶ停留点の個数は上限値と下限値をハイパーパラメータとしてランダムに決定する.
step:2 B_{use} を使用して 3.2 節の手順で生成した初期解に対し, insert-swap 近傍を用いた局所探索法での改善を行い, 実行可能解を生成する. この実行可能解を reference-set に追加する.

step1〜step2 を reference-set-num 回繰り返えす.
reference-set-num は問題の大きさによって設定するハイパーパラメータとした.

5.3.2 解の近づけ方

パスリリンキングでは一方の解をもう一方の解に一つずつ近づけていく必要がある. 本研究では以下のような手順で行なった.

step:1 2つの解をそれぞれσ_{start}, σ_{goal}とする. σ_{start}で使用していて, σ_{goal}で使用していない停留点の集合をB_{out}とし, σ_{start}で使用しておらず, σ_{goal}で使用している停留点の集合をB_{in}とする. $B_{\text{out}} \cup B_{\text{in}} = B_{\text{yet}}$とする.

step:2 B_{yet}から停留点bをランダムに1つ選ぶ. $b \in B_{\text{in}}$だった場合, **操作 b-in** を行う. $b \in B_{\text{out}}$だった場合, **操作 b-out** を行う.

step:3 $B_{\text{yet}} = B_{\text{yet}} \setminus \{b\}$とし, $B_{\text{yet}} = \emptyset$であれば操作を終了する. そうでなければstep2に戻る.

b-in 操作, b-out 操作は以下の様に行う.

5.3.2.1 b-in 操作

現在の解を$\sigma_{cuerrent}$とする. 現在の解にbを追加する. このとき, 新しく生成する解の配達車のルート, ドローンのルートは以下のように決定する.

配達車のルートの作成方法

bを$\sigma_{cuerrent}$の全ての配達車のルートに対して挿入し, そのルートの評価を行う. その中で最も評価の良いルートに挿入する. この際, 配達車の容量制約をペナルティとしてルートを評価する.

ドローンのルートの作成方法

step:1 bから出発するドローンのルートは, σ_{goal}においてbから出発するドローンのルートをそのまま使用する.

step:2 b以外の停留点から出発するドローンのルートについては, $\sigma_{cuerrent}$のドローンのルートのルートとする. この時, そのドローンのルートにstep1でbに割り当てられた顧客点が入っていた場合, その顧客点をスキップするようなルートとする.

5.3.2.2 b-out 操作

現在の解を$\sigma_{cuerrent}$とする. 現在の解からbを削除する. このとき, 新しく生成する解の配達車のルート, ドローンのルートは以下のように決定する.

配達車のルートの作成方法

$\sigma_{cuerrent}$における配達車のルートで, bをスキップするようなルートを作成し, 配達車のルートとする.

ドローンのルートの作成方法

step:1 $\sigma_{cuerrent}$でb以外の停留点から出発するドローンのルートをそのまま使用する.

step:2 step1で作成したルートに対し, $\sigma_{cuerrent}$でbから配送されていた顧客点をそれぞれ最もコストが小さくなるように挿入していく.

5.3.3 パスリリンキングのアルゴリズム

本研究ではパスリリンキングによる探索を以下の手順で行なった.

step:1 reference-set の中からランダムに解を2つ選び, 小節5.3.2の手法を用いて, 一方の解をもう一方の解に近づけていく.

step:2 一つずつ近づく過程で生成された解それぞれに対しinsert-swap近傍を用いた局所探索法による改善を行う.

step:3 step2生成された解の中で, 「reference-set のいずれの解とも使用している停留点が2つ以上異なる」かつ「reference-set の中で最も評価値の悪い解よりも評価値が良い」場合, reference-set の中で最も評価値の悪い解と交換する. この際, 上記の条件を満たしていなくても, reference-set の中で最も評価値の良い解よりも評価値が良かった場合も上記のように交換を行う.

step1〜step3を制限時間内で繰り返し, reference-set の中で最も良い評価価の解を出力する.

5.3.4 パスリリンキングの高速化

パスリリンキングを行う過程で生まれる全ての解に対してinsert-swap近傍を用いた局所探索法による改善を行うと, 計算量が大きくなってしまう. そこで, insert-swap近傍を用いた局所探索法を行う解と行わない解を以下の基準で定めパスリリンキングの高速化を行なった. 以下の二つの基準のどちらかに当てはまる解にのみ, insert-swap近傍を用いた局所探索法による改善を行う.

insert-swap近傍を用いた局所探索法による改善を行う基準

- σ_{start} を σ_{goal} に近づける過程で生成された解のうち, 一つ前に生成した解よりも評価値が良い, かつ, 一つ後に生成された解よりも評価値が良い解であること.
- σ_{start} に最も近い解, または, σ_{goal} に最も近い解であること.

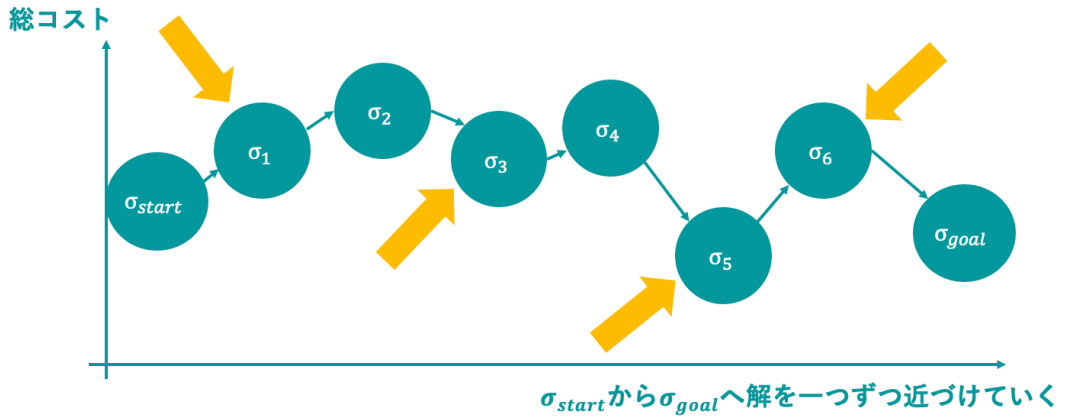

図5.3.4: パスリリンキングの高速化 insert-swap近傍を用いた局所探索法を行う基準

図5.3.4の黄色い矢印で示された解がinsert-swap近傍を用いた局所探索法を行う解である. こ

のように, insert-swap 近傍を用いた局所探索法を行う基準を定めることにより, 全ての解に対し, insert-swap 近傍を用いた局所探索法を行う必要がなくなり, より精度の高い解を高速に求解することが期待できる.

6 計算実験

この章では二段階配送計画問題に対して, 提案手法である 3 つのメタ戦略で求解し, 比較をする. 計算実験には 2.3 GHz Intel Core i5, 16 GB 2133 MHz LPDDR3 を搭載した計算機を使用した. 問題例は全てランダムに生成した. 実験に用いるプログラムは python を用いて実装した. α, β はそれぞれ 1000 とした.

表1：作成した問題例一覧

問題例	顧客数	停留点数	配達車両数	各配達車に積載するドローンの数	配達車の容量制約	ドローンの容量制約
20_10_1	20	10	1	3	25	3
20_10_2	20	10	1	3	25	3
20_10_3	20	10	1	3	25	3
20_10_4	20	10	1	3	25	3
20_10_5	20	10	1	3	25	3
40_20_1	40	20	2	3	25	3
40_20_2	40	20	2	3	25	3
40_20_2	40	20	2	3	25	3
40_20_3	40	20	2	3	25	3
40_20_3	40	20	2	3	25	3
80_40_1	80	40	4	3	25	3
80_40_2	80	40	4	3	25	3
80_40_3	80	40	4	3	25	3
80_40_4	80	40	4	3	25	3
80_40_5	80	40	4	3	25	3
160_80_1	160	80	8	3	25	3
160_80_2	160	80	8	3	25	3
160_80_3	160	80	8	3	25	3
160_80_4	160	80	8	3	25	3
160_80_5	160	80	8	3	25	3
200_100_1	200	100	10	3	25	3
200_100_2	200	100	10	3	25	3
200_100_3	200	100	10	3	25	3
200_100_4	200	100	10	3	25	3
200_100_5	200	100	10	3	25	3

表1に作成した問題例と，その設定の一覧を示す．

表 2 : ランダム多スタート法

	ランダム多スタート insert-swap			ランダム多スタート 1del-1ins		
問題例	値	時間 [s]	反復回数	値	時間 [s]	反復回数
20_10_1	6598.69	30	121	6576.93	30	32
20_10_2	5491.67	30	130	5536.05	30	35
20_10_3	5947.87	30	126	5870.32	30	30
20_10_4	4730.36	30	114	4787.94	30	35
20_10_5	5793.99	30	128	5915.29	30	36
40_20_1	9854.53	100	186	10002.87	100	28
40_20_2	10624.38	100	175	10629.74	100	24
40_20_3	9037.79	100	193	9169.31	100	27
40_20_4	9427.34	100	171	9410.79	100	27
40_20_5	9581.68	100	186	9572.73	100	25
80_40_1	15691.32	200	88	15692.27	200	10
80_40_2	15603.26	200	102	15763.49	200	11
80_40_3	15497.39	200	78	15729.82	200	11
80_40_4	14855.24	200	111	14878.18	200	9
80_40_5	16706.70	200	97	16820.19	200	11
160_80_1	23934.76	2000	139	25502.38	2000	6
160_80_2	24072.25	2000	124	28872.51	2000	6
160_80_3	22923.53	2000	119	24081.52	2000	6
160_80_4	23566.88	2000	108	24899.65	2000	6
160_80_5	25596.43	2000	149	25951.40	2000	6
200_100_1	28559.97	3000	124	30462.95	3000	4
200_100_2	27953.86	3000	113	29625.92	3000	4
200_100_3	28665.05	3000	118	30032.66	3000	4
200_100_4	27542.27	3000	117	27978.55	3000	4
200_100_5	28577.78	3000	118	29983.22	3000	4

表 2 はランダム多スタート法の実験結果である. 制限時間を設定し, 実行した. 表 2 中の「反復回数」は制限時間内に反復した回数を示す. ランダム多スタート法は解空間をランダムに探索するため, 解一つ一つに対する近傍探索に重点を置くランダム多スタート-1del-1ins よりも, 解の生成個数が多いランダム多スタート-insert-swap の方が有効であることを確認した.

表 3 : 反復局所探索法

反復局所探索法			
問題例	値	時間 [s]	反復回数（kick の回数）
20_10_1	6576.93	30	49
20_10_2	5491.67	30	43
20_10_3	5870.32	30	46
20_10_4	4787.94	30	53
20_10_5	5793.99	30	43
40_20_1	9900.70	100	27
40_20_2	10920.07	100	27
40_20_2	9149.88	100	33
40_20_3	9375.15	100	26
40_20_3	9549.34	100	32
80_40_1	15755.28	200	9
80_40_2	16090.52	200	10
80_40_3	15488.09	200	8
80_40_4	14864.06	200	8
80_40_5	17409.28	200	7
160_80_1	24059.03	2000	5
160_80_2	24313.35	2000	5
160_80_3	23196.14	2000	5
160_80_4	23407.70	2000	5
160_80_5	25574.85	2000	5
200_100_1	28990.64	3000	4
200_100_2	28047.68	3000	3
200_100_3	28855.64	3000	3
200_100_4	27481.39	3000	3
200_100_5	28702.52	3000	3

表 3 は反復局所探索法の実験結果である. 制限時間を設定し, 実行した. 表 2 中の「反復回数」は制限時間内に反復した回数 (kick 操作をした回数) を示す. 反復局所探索法では顧客点数 20〜40 程度の小規模な問題例に対しては, 有効であることが確認できた. 一方で, 顧客点数 80〜200 の中, 大規模の問題例に対しては 1del-1ins 近傍を用いた局所探索法の実行時間が長く, 現実的な時間で広い解空間を探索することができなかったため, パスリリンキングよりも良い結果を出すことはなかった. より良い結果を求解するためには, 1del-1ins 近傍の局所探索法の高速化を検討する必要があると考えられる.

表 4 : パスリリンキング

	パスリリンキング				パスリリンキング 高速化			
問題例	値	時間 [s]	パスリリンキングの回数	insert-swap の回数	値	時間 [s]	パスリリンキングの回数	insert-swap の回数
20_10_1	6598.69	30	443	2407	6588.71	30	661	1912
20_10_2	5491.67	30	564	2694	5491.67	30	756	1848
20_10_3	5913.05	30	780	2707	5913.05	30	954	2517
20_10_4	4701.14	30	520	2334	4656.88	30	754	1646
20_10_5	5793.99	30	401	29590	5793.99	30	552	2973
40_20_1	9818.15	100	299	1494	9730.55	100	829	2169
40_20_2	10435.38	100	135	1444	10425.72	100	619	2075
40_20_2	8976.49	100	295	1526	8976.49	100	725	2602
40_20_3	9314.49	100	284	1475	9314.49	100	928	2172
40_20_3	9509.05	100	244	1354	9550.64	100	610	2076
80_40_1	15573.06	200	60	1232	15510.95	200	238	1474
80_40_2	15442.37	200	64	1620	15283.87	200	491	1488
80_40_3	15253.01	200	109	1208	14994.37	200	466	1423
80_40_4	15013.99	200	68	1225	14674.94	200	237	1646
80_40_5	16845.22	200	76	1348	16654.68	200	243	1510
160_80_1	24416.97	2000	17	3967	23654.29	2000	394	3212
160_80_2	24509.38	2000	21	5492	23246.72	2000	542	2970
160_80_3	23327.18	2000	20	1529	22506.37	2000	493	3049
160_80_4	23732.91	2000	24	7093	22751.35	2000	493	3049
160_80_5	25639.25	2000	21	5285	24253.28	2000	541	2806
200_100_1	28974.04	3000	17	3509	27913.10	3000	419	2796
200_100_2	29060.15	3000	22	8930	27291.26	3000	315	3135
200_100_3	29381.56	3000	24	6790	27807.38	3000	410	2856
200_100_4	28133.32	3000	25	6201	26981.31	3000	482	2750
200_100_5	29433.13	3000	25	7032	28132.71	3000	370	3241

表 4 はパスリリンキングの実験結果である. 制限時間を設定し, 実行した. 表 2 中の「insert-swap の回数」は制限時間内に 4.2 節の insert-swap 近傍を用いた局所探索法を行なった回数を示す. パスリリンキングは全てのサイズの問題例に対し, 有効な解法であることが確認できた. パスリリンキングの高速化を行なったことにより, より短い実行時間で精度の高い解を求解することが可能であることが確認できた.

表5: 各手法の結果まとめ

問題例	ランダム多スタート insert-swap	ランダム多スタート 1del-1ins	反復局所探索法	パスリリンキング	パスリリンキング 高速化
20_10_1	6598.69	**6576.93**	**6576.93**	6598.69	6588.71
20_10_2	**5491.67**	5536.05	**5491.67**	**5491.67**	**5491.67**
20_10_3	5947.87	**5870.32**	**5870.32**	5913.05	5913.05
20_10_4	4730.36	4787.94	4787.94	4701.14	**4656.88**
20_10_5	**5793.99**	5915.29	**5793.99**	**5793.99**	**5793.99**
40_20_1	9854.53	10002.87	9900.70	9818.15	**9730.55**
40_20_2	10624.38	10529.74	10920.07	10435.38	**10318.54**
40_20_3	9037.79	9169.31	9149.88	8976.49	**8948.34**
40_20_4	9427.34	9410.79	9375.15	9314.49	**9300.32**
40_20_5	9581.68	9572.73	9549.34	9509.05	**9495.44**
80_40_1	15691.32	15692.27	15755.28	15573.06	**15510.95**
80_40_2	15603.26	15763.49	16090.52	15442.37	**15283.87**
80_40_3	15497.39	15729.82	15488.09	15253.01	**14994.37**
80_40_4	14855.24	14878.18	14864.06	15013.99	**14674.94**
80_40_5	16706.70	16820.19	17409.28	16845.22	**16654.68**
160_80_1	23934.76	25502.38	24059.03	24416.97	**23654.29**
160_80_2	24072.25	2887.51	24313.35	24509.38	**23246.72**
160_80_3	22923.53	24081.52	23196.14	23327.18	**22506.37**
160_80_4	23566.88	24899.65	23407.70	23732.91	**22751.35**
160_80_5	25596.43	25951.40	25574.85	25639.25	**24253.28**
200_100_1	28559.97	30462.95	28990.64	28974.04	**27913.10**
200_100_2	27953.86	29625.92	28047.68	29060.15	**27291.26**
200_100_3	28665.05	30032.66	28855.64	29381.56	**27807.38**
200_100_4	27542.27	27978.55	27481.39	28133.32	**26981.31**
200_100_5	28577.78	29983.22	28702.52	29433.13	**28132.71**

表5は各手法の実験結果まとめを示す. 5つの手法のうち最も目的関数値が小さい結果をボールド体で表記している.

7　まとめ

本研究では二段階配送計画問題を提案し, 提案手法としてランダム多スタート法, 反復局所探索法, パスリリンキングの 3 つのメタ戦略による求解を行った. 計算実験により, 小規模な問題例に対しては反復局所探索法による求解が優れており, 中規模, 大規模な問題に対しては, パスリリンキングによる求解が優れていることがわかった. パスリリンキングの高速化を行なったことで, 短時間で精度の高い解の求解が可能であることを確認した.

謝辞

本研究の遂行にあたり, 熱心な指導と助言をいただきました橋本英樹准教授に深く感謝の意を表します. 日々の研究室生活においては橋本研究室の岡本優太くんには大変お世話になりました.

参考文献

[1] ヤマトグループ csr 報告書 2013. https://www.yamato-hd.co.jp/csr/report/pdf/2013.pdf, 2013.

[2] Gurobi. https://www.gurobi.com/, 2019.

[3] 令和 2 年度宅配便等取扱実績関係資料. https://www.mlit.go.jp/report/press/content/001418260.pdf, 2020.

[4] Emile Aarts, Emile HL Aarts, and Jan Karel Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 2003.

[5] Yong Sik Chang and Hyun Jung Lee. Optimal delivery routing with wider drone-delivery areas along a shorter truck-route. *Expert Systems with Applications*, Vol. 104, pp. 307–317, 2018.

[6] Fred W Glover and Gary A Kochenberger. *Handbook of metaheuristics*. Springer Science & Business Media, 2006.

[7] Paolo Toth and Daniele Vigo. *The vehicle routing problem*. SIAM, 2002.

[8] 今堀慎治, 柳浦睦憲. 概説メタ戦略. オペレーションズリサーチ: 経営の科学, Vol. 58, No. 12, pp. 695–702, 2013.

[9] 瀧本修斗, 高田陽介, 岩田麻希, 高橋輝, 今井純志, 早川敬一郎, 胡艶楠, 佐々木美裕, 小野廣隆, 柳浦睦憲. ハブ・スポーク配送計画問題に対する反復局所探索法, 日本オペレーションズ・リサーチ学会 2019 年秋季研究発表会アブストラクト集, 2-a-5, 2019.

[10] 劉楊成漢, 鈴木勉. トラックとドローンの併用による配達効率化分析, 日本オペレーションズ・リサーチ学会 2019 年秋季研究発表会アブストラクト集, 2-b-5, 2019.