Vol.2015-AL-151 No.4
2015/1/13

被覆制約付き配送計画問題に対する効率的局所探索法

高田 陽介[1,a)]　橋本 英樹[1]　柳浦 睦憲[1]

概要：被覆制約付き配送計画問題 (m-CTP) は, 集合被覆問題 (SCP) と配送計画問題 (VRP) を組み合わせた問題である. 訪問することのできる点 (訪問点) 集合 V と被覆されなければならない点 (被覆点) 集合 W が与えられ, 各 $v \in V$ は W の一部を被覆する. 問題の目的は m 個の V 上の巡回路の距離の総和を最小化することである. ただし各被覆点は巡回路のいずれかの頂点により被覆されなければならない. 既存の手法では, 訪問する点集合を定めたのちに VRP に対する局所探索法を利用し, 巡回路のみを改善するものが多かった. 本研究では訪問する点集合と巡回路を同時に探索する局所探索法として, パスの再構築法と 1-del/1-ins という操作を組み込んだものを提案し, これらの効率的実現法を示す. 提案手法では被覆制約を破った解への移動も許可し, 被覆制約の違反度合いをペナルティとしたペナルティつきコストで解を評価する. さらに探索状況に応じてペナルティ重みを適応的に変動させることで, より高い性能を実現した.

1. はじめに

被覆制約付き配送計画問題（m-CTP）は，集合被覆問題（SCP）と配送計画問題（VRP）を組み合わせた問題であり，以下のように表される．まず, 訪問することのできる n_v 個の点 (訪問点) 集合 $V = \{v_0, v_1, \ldots, v_{n_v-1}\}$ と被覆されなければならない n_w 個の点 (被覆点) 集合 $W = \{w_1, w_2, \ldots, w_{n_w}\}$ からなる無向グラフ $G = (V \cup W, E_1 \cup E_2)$ を定義する．ここで E_1 と E_2 は辺集合 $E_1 = V \times V$，$E_2 \subseteq V \times W$ であり，E_1 の各辺 (v_i, v_j) には距離 c_{ij} が定義されている．本研究では c_{ij} が三角不等式を満たす場合を考える．各訪問点 v_i は $(v_i, w_j) \in E_2$ を満たすすべての w_j を被覆する．m-CTP の目的は，デポと呼ばれる訪問点 v_0 を出発して V 内のいくつかの点を訪問し，再びデポに戻ってくるような高々 m 個の巡回路の距離の合計を最小化することである．ただし，求めた巡回路に含まれる点の全体が W の点すべてを被覆しなければならない．

現実社会への応用として，例えば，郵便ポストの設置位置を決める問題がある [4]．各住宅からある程度近い位置に 1 つはポストがなければならず，郵便局は郵便物回収のための移動距離を最小にしたい．この問題は，ポストの設置位置候補の集合を V，住宅の集合を W，ポスト v_i とポスト v_j との距離を c_{ij}，位置の近い住宅とポストの間に辺 e ($\in E_2$) が存在すると考えることで，m-CTP として扱うことができる．その他の例としては発展途上国におけるヘルスケアチームの訪問経路を求める問題がある [3]．ヘルスケアチームは限られた数の村しか訪れることができないが，すべての村の患者が徒歩で診察を受けに来られなければならない．このとき，ヘルスケアチームはなるべく多くの患者を診察するために移動時間を最小化したい．この問題は，すべての患者が徒歩で診療を受けられるという制約の下で移動時間を最小化する問題として扱うことができる．

m-CTP に対しては，まず SCP を解いて訪問する点を定めたのち，VRP を解いて巡回路を決めるという解法が考えられる．しかし一般的な m-CTP に対しては，訪問点集合と巡回路を同時に考慮した解法が有効であることが村上により示されている [5]．また，訪問点集合と巡回路を同時に考慮したような巡回路の構築法である H-1-CTP[1] が提案されているが，局所探索法タイプの解法についてはあまり研究されていない．そこで本研究では，訪問点集合と巡回路を同時に探索する 1-del/1-ins とパスの再構築という操作を含む二種類の局所探索法を提案し，それらを用いた解法を提案する．さらにこれらの操作については改善解を逃すことがないという保証の下で探索範囲を狭めることで，解の精度を落とすことなく計算時間を短縮する方法を提案する．

計算実験により Hà ら [2] による手法と比較したところ同等の性能であることが確認できた．また大規模な問題例を扱った村上 [6] による手法とも比較し，計算時間と精度の両方において優位な結果を得ることを確認した．

2. 問題

使用できる車両数の上限を m とする．上述のように，

[1] 名古屋大学大学院情報科学研究科
Nagoya University
[a)] takada.yousuke@d.mbox.nagoya-u.ac.jp

各訪問点 v は $(v,w) \in E_2$ を満たすすべての w を被覆する．m-CTP はすべての被覆点 $w \in W$ を被覆するような高々 m 個の巡回路（訪問する点 $v \in V$ の順番）の組を決める問題である．点集合 $T(\subseteq V)$ の点はいずれかの車両に必ず訪問されなければならない．各訪問点間の距離 c_{ij} は三角不等式を満たす．つまり，任意の v_i, v_j, v_k に対して $c_{ij} + c_{jk} > c_{ik}$ が常に成り立つ場合を考える．各訪問点 v_i には要求量 a_i が設定されており，各巡回路の要求量の総和が各車両の容量 p を超えてはならない．また各車両には移動に伴うコストの上限 q も与えられる．以上の制約を満たした上で巡回路のコストの総和を最小化する問題である．

2.1 定式化

m-CTP を整数計画問題として定式化する．ここで各被覆点 $w_l \in W$ に対し，w_l を被覆する訪問点の集合を S_l と定義する．つまり，$S_l = \{v_i \mid (v_i, w_l) \in E_2\}$ である．また，変数 y_{hk} を，訪問点 v_h が巡回路 k に含まれれば 1，それ以外では 0 の値をとる 0-1 変数とする．変数 x_{ijk} を，巡回路 k に辺 (v_i, v_j) が含まれれば 1，それ以外では 0 の値をとる 0-1 変数とする．これらを用いて，m-CTP は以下のように定式化できる．

$$\min \quad \sum_{k=1}^{m} \sum_{i=0}^{n_v-1} \sum_{j=i+1}^{n_v} c_{ij} x_{ijk} \tag{1}$$

$$\text{s.t.} \quad \sum_{k=1}^{m} \sum_{v_h \in S_l} y_{hk} \geq 1 \quad (w_l \in W), \tag{2}$$

$$\sum_{k=1}^{m} y_{hk} \leq 1 \quad (v_h \in V \setminus \{v_0\}), \tag{3}$$

$$\sum_{i=0}^{h-1} x_{ihk} + \sum_{j=h+1}^{n_v} x_{hjk} = 2y_{hk} \quad (v_h \in V \setminus \{v_0\},\ 1 \leq k \leq m), \tag{4}$$

$$\sum_{k=1}^{m} \sum_{\substack{v_i \in S, v_j \in V \setminus S \\ \text{or} \\ v_j \in S, v_i \in V \setminus S}} x_{ijk} \geq 2 \sum_{k=1}^{m} y_{hk} \quad (S \subset V \setminus \{v_0\},\ v_h \in S), \tag{5}$$

$$\sum_{k=1}^{m} y_{hk} = 1 \quad (v_h \in T \setminus \{v_0\}), \tag{6}$$

$$\sum_{h=1}^{n_v} a_h y_{hk} \leq p \quad (1 \leq k \leq m), \tag{7}$$

$$\sum_{i=0}^{n_v-1} \sum_{j=i+1}^{n_v} c_{ij} x_{ijk} \leq q \quad (1 \leq k \leq m), \tag{8}$$

$$y_{hk} \in \{0,1\} \quad (v_h \in V,\ 1 \leq k \leq m), \tag{9}$$

$$x_{ijk} \in \{0,1\} \quad (1 \leq i < j \leq n_v - 1,\ 1 \leq k \leq m). \tag{10}$$

式 (2) は被覆制約を表している．式 (3) は，訪問点 v_h は高々 1 本の巡回路にしか含めることはできないという意味である．式 (4) は次数制約を意味し，v_h が巡回路に含まれる場合は v_h に繋がる辺のうち 2 本が巡回路に含まれ，v_h が巡回路に含まれない場合は v_h に繋がる辺は巡回路に含まれないということである．式 (5) は部分閉路除去制約を意味する．まずデポ v_0 を含まない任意の部分集合 $S \subset V \setminus \{v_0\}$ を考える．S 内の点のうち少なくとも 1 つが巡回路に含まれる場合，S 内の点と $V \setminus S$ 内の点を結ぶ辺のうち 2 本以上が必ず巡回路に含まれる．つまり，すべての閉路はデポを含まなければならないということである．式 (6) は T の要素は必ず訪問しなければならないということを表している．式 (7) は容量制約，式 (8) は距離制約を意味し，式 (9) と式 (10) はそれぞれ y_{hk} と x_{ijk} が 0-1 変数であることを表している．

3. 提案手法

本節では m-CTP に対して局所探索法に基づく解法を提案する．

探索中により広い解空間を探索するため被覆制約を破った解への移動も許可し，評価関数として以下の評価関数 f を用いる．σ を現在の解，$c(\sigma)$ を現在の解のコスト，$\tilde{W}(\sigma)$ を被覆されていない被覆点集合， b_i を各被覆点 w_i に割り当てられたペナルティ重みとし，評価関数 f を

$$f(\sigma) = c(\sigma) + \sum_{w_i \in \tilde{W}(\sigma)} b_i \tag{11}$$

と定義する．これは全ルート長の総和に被覆制約の違反度合いをペナルティとして足したものである．

提案手法の概要を以下に示す．まずランダムに選んだ訪問点 1 点のみからなる巡回路を m 個用意する．その後，未訪問点をひとつずつ巡回路に追加する操作を，容量制約または距離制約により点を挿入できなくなるか被覆制約を満たすまで繰り返す．初期解を生成したら，タブーリストを用いた 1-del/1-ins により訪問点集合と巡回路を同時に改善する．1-del/1-ins では探索中に実行可能解へ移動するたびに V-OPT を適用して解を改善する．1-del/1-ins によって見つかった最小コストの実行可能解に対し，パスの再構築法を適用しさらなる改善を試みる．これらの一連の操作を行ったのち，現在の解に小さな変形を施すことによって得られた解から再び探索を行うということを繰り返し，探索の中で得られた最小コストの実行可能解を出力する．以下に提案手法全体の枠組みを示す．

m-CTP に対する反復局所探索法（ILS）

Step 1 構築法により解を生成する．

Step 2 各被覆点 w_i に対しペナルティ重み b_i を設定する．

Step 3 探索開始時の解を $\sigma_{\text{before}} := \sigma$ として記憶しておく．

Step 4 1-del/1-ins を用いた局所探索を行う．その際実行

可能解へ移動するたびにその解に対して V-OPT を適用し，得られた解から 1-del/1-ins の探索を再開する．

Step 5 Step 4 で見つかった最小コストの実行可能解に対してパスの再構築法による局所探索を行う．

Step 6 $f(\sigma_{\mathrm{before}}) < f(\sigma)$ であるなら $\sigma := \sigma_{\mathrm{before}}$ とする．$\sigma_{\mathrm{before}} := \sigma$ とする．

Step 7 現在の解に小さな変化を加える．

Step 8 終了条件を満たしていれば探索中に見つかった実行可能解の中でコストが最小の解を出力する．そうでなければ Step 4 へ戻る．

各ステップの詳細を以下で紹介する．

3.1 V-OPT

本節では 1-del/1-ins の探索中に用いる V-OPT を紹介する．V-OPT とは VRP に対する局所探索であり，VRP に対する標準的な近傍である 2-opt 近傍，2-opt*近傍，Swap 近傍，Relocate 近傍を用いる．2-opt 近傍とは 1 つの巡回路上の 2 つの辺を付け替えて得られる解集合のことである．2-opt*近傍は 2-opt 近傍を拡張したもので，異なる 2 つの巡回路に含まれる辺同士を付け替えた解集合である．Swap 近傍では訪問済みの 2 つの点を入れ替える操作を行い，Relocate 近傍ではある巡回路上の部分パスを別の巡回路へ挿入する．これらの近傍操作をそれぞれ改善がなくなるまで行うことで，解に含まれる点集合を変えない範囲で解の改善を試みる．

3.2 1-del/1-ins

ここでは 1-del/1-ins による局所探索法を紹介する．1-del/1-ins は現在のルートから訪問点を一つ除去する 1-del，現在のルートに訪問点を一つ追加する 1-ins，これら二つを同時に行う 1-del-ins の 3 つの近傍操作からなる．この探索中ではタブーリストを用いる．タブーリストには挿入あるいは削除が起こった訪問点を記憶する．点をタブーリストに追加するたびにその点のタブー期間 t を $t_{\min} \leq t \leq t_{\max}$ の範囲でランダムに設定し，リストに追加されてから解の移動が t 回起こるまではその点の挿入あるいは削除を禁止する．探索が終了したら，探索中で得られた最小コストの実行可能解 σ_{best} を出力する．以下に全体の流れを示す．

1-del/1-ins による局所探索

Step 1 1-del を改善がなくなるまで適用する．

Step 2 1-ins を改善がなくなるまで適用する．

Step 3 Step 1 と Step 2 のいずれかで f の値に改善があった場合は Step 1 へ戻る．

Step 4 1-del-ins を適用する．

Step 5 Step 1 から Step 4 までのいずれかのステップで f の値に改善があった場合は Step 1 へ戻る．

Step 6 ペナルティ重みの値を更新する．

Step 7 Step 1 から Step 6 までを規定回数繰り返したら σ_{best} を出力して終了する．そうでない場合は Step 1 へ戻る．

以下で各操作の詳細を説明する．

3.2.1 1-del 操作

1-del は巡回路から点をひとつ削除する．アルゴリズムの流れを以下に示す．

Step 1 現在の解 σ が訪問する点全ての集合を V_1，未探索の点集合を $\tilde{V} := V_1$ とする．

Step 2 $\tilde{V} = \emptyset$ ならば σ を現在の解として出力し終了する．そうでなければ任意の点 $v \in \tilde{V}$ を選び，$\tilde{V} := \tilde{V} \setminus \{v\}$ とする．

Step 3 v を σ から除去して得られる解 σ' が $f(\sigma') < f(\sigma)$ を満たすなら $\sigma := \sigma'$ とする．そうでなければ Step 2 へ戻る．

Step 4 σ が実行可能解であるならば，σ に V-OPT を適用して得られた解を $\hat{\sigma}$ とし，$\sigma := \hat{\sigma}$ とする．$c(\sigma) < c(\sigma_{\mathrm{best}})$ を満たすならば $\sigma_{\mathrm{best}} := \sigma$ とする．Step 2 へ戻る．

3.2.2 1-ins 操作

1-ins は巡回路へ点をひとつ挿入する．アルゴリズムの流れを以下に示す．

Step 1 現在の解 σ が訪問しない訪問点全ての集合を V_2，未探索の点集合を $\tilde{V} := V_2$ とする．

Step 2 $\tilde{V} = \emptyset$ ならば σ を現在の解として出力し終了する．そうでなければ任意の点 $v_i \in \tilde{V}$ を選び，$\tilde{V} := \tilde{V} \setminus \{v_i\}$ とする．

Step 3 最小の $c_{ij} (i \neq j)$ を満たす v_j を選ぶ．ただし v_j の候補としては v_i をその前後のうち距離の増加の少ない方に挿入しても容量制約も距離制約も破らないものに限る．

Step 4 v_i を v_j の前後のうち距離の増加の少ない方に挿入したときの解 σ' が $f(\sigma') < f(\sigma)$ を満たすならば，$\sigma := \sigma'$ とする．そうでなければ Step 2 へ戻る．

Step 5 σ が実行可能解であるならば，σ に V-OPT を適用して得られた解を $\hat{\sigma}$ とし，$\sigma := \hat{\sigma}$ とする．$c(\sigma) < c(\sigma_{\mathrm{best}})$ を満たすならば $\sigma_{\mathrm{best}} := \sigma$ とする．Step 2 へ戻る．

3.2.3 1-del-ins 操作

まず現在のルートから訪問点をひとつ除去する．その後，未訪問の点の中から，現在のルートに挿入したときに評価関数が改善するものを選択し，ルートに挿入する．詳しい手順を以下に示す．

Step 1 現在の解 σ が訪問する点全ての集合を V_1，σ が訪問しない訪問点全ての集合を V_2 とし，未探索の訪問済み点集合を $\tilde{V}_1 := V_1$，未探索の未訪問点集合を $\tilde{V}_2 := V_2$ とする．

Step 2 $\tilde{V}_1 = \emptyset$ ならば σ を現在の解として出力し終了する．そうでなければ任意の点 $v_i \in \tilde{V}_1$ を選び，$\tilde{V}_1 := \tilde{V}_1 \setminus \{v_i\}$，$\tilde{V}_2 := V_2$ とする．

Step 3 v_i を σ から除去したときの解を σ' とおく．

Step 4 $\tilde{V}_2 = \emptyset$ ならば Step 2 へ戻る．そうでなければ任意の点 $v_j \in \tilde{V}_2$ を選び，$\tilde{V}_2 := \tilde{V}_2 \setminus \{v_j\}$ とする．

Step 5 V_1 の中で最小の $c_{jk}(j \neq k)$ を満たす v_k を選ぶ．ただし v_k の候補としては v_j をその前後のうち距離の増加の少ない方に挿入しても容量制約も距離制約も破らないものに限る．

Step 6 v_j を v_k の前後のうち距離の増加の少ない方に挿入したときの解を σ'' とおく．もし $f(\sigma'') < f(\sigma)$ ならば，$\sigma := \sigma''$ とする．そうでなければ Step 4 へ戻る．

Step 7 σ が実行可能解であるならば σ に V-OPT を適用して得られた解を $\hat{\sigma}$ とし，$\sigma := \hat{\sigma}$ とする．$c(\sigma) < c(\sigma_{\mathrm{best}})$ を満たすならば $\sigma_{\mathrm{best}} := \sigma$ とする．Step 2 へ戻る．

3.2.4 ペナルティ重みの更新

1-del/1-ins による局所探索法ではペナルティ重みを探索状況に応じて自動的に変化させ，より広い範囲へと探索が進むようにした．ペナルティ重みの更新の具体的な手順を以下に示す．ここで α_{inc} と α_{dec} は $\alpha_{\mathrm{inc}} > 1$ と $0 < \alpha_{\mathrm{dec}} < 1$ を満たすパラメータである．

(i) 前回の更新以降の探索で一度でも実行不可能解が得られた場合は，探索中に一度でも未被覆になった点の重みを α_{inc} 倍する．

(ii) 前回の更新以降の探索で一度も実行不可能解が得られなかった場合は，すべての点の重みを α_{dec} 倍する．

3.2.5 1-del-ins の高速化

1-del-ins において探索する点の候補をすべて考えると，削除する点が $O(|V|)$ 個，削除する点それぞれに対し挿入する点が $O(|V|)$ 個あるので，$O(|V|^2)$ となる．しかし 1-del と 1-ins どちらを適用しても改善がない状態であるとすると，改善解を逃さずに探索の候補を減らすことができる．ここで，1-del-ins において削除する点を v_{del}，挿入する点を v_{ins} とおき，v_{del} の前後の辺の長さを e_1, e_2，v_{del} を削除することで追加される辺の長さを e_3，v_{ins} を挿入することで追加される辺の長さを e_4, e_5，削除される辺の長さを e_6 とおく（**図 1**，**図 2**）．また v_{del} を削除することで被覆されなくなる点のペナルティ重みの総和を a_{del}，v_{ins} を挿入することで新たに被覆される点のペナルティ重みの総和を a_{ins} とおく．解に含まれるどの点を削除しても改善がなく，解に含まれないどの点を挿入しても改善がない場合，削除と挿入を同時に行って改善が起こるような点の組は，そのうちの v_{ins} による 1-ins 操作が実行可能である場合には，

case 1. v_{del} を削除することで被覆されなくなる点を v_{ins} が被覆する場合

case 2. 巡回路において v_{del} があった場所に v_{ins} を挿入する場合

の 2 通りのみであると言える．一方，1-ins において，v_{ins} を挿入したら評価関数には改善が起こるものの容量制約や距離制約を破ってしまうために挿入できない場合，上記以外の組でも改善が起こりうる．よって

case 3. v_{ins} を挿入するとペナルティ付きコストは改善するが容量制約または距離制約を破る場合

も考える必要がある．

case 1 においては任意の v_{del} に対応する v_{ins} の候補として，v_{del} を削除したときに被覆されなくなる点を被覆するような未訪問の点を列挙すればよい．

case 2 では各 v_{del} に対する v_{ins} の候補はすべての未訪問の点となりうるが，その中で改善が起こる v_{ins} の候補を絞ることができる．case 2 において改善が起こる必要十分条件は

$$e_4 + e_5 - a_{\mathrm{ins}} < e_1 + e_2 - a_{\mathrm{del}} \tag{12}$$

である．この条件が成り立つすべての v_{ins} を列挙すればよいことがわかる．ここで，

$$e_4 < e_1 \tag{13}$$

が成り立つ場合とそうでない場合を考える．式 (13) が成

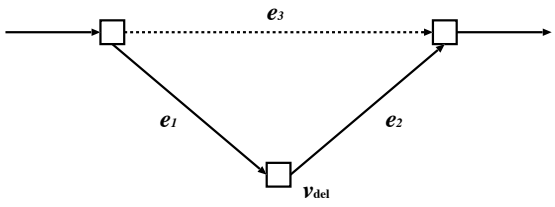

図 1 v_{del} の削除に伴う各辺の長さ

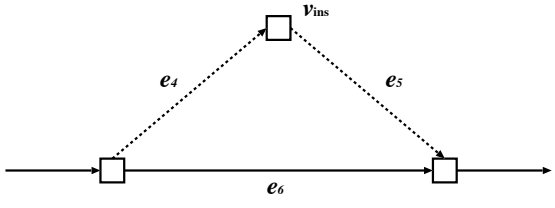

図 2 v_{ins} の挿入に伴う各辺の長さ

り立たないという条件を用いて式 (12) を変形すると

$$\begin{aligned} e_4 + e_5 - a_{\mathrm{ins}} &< e_1 + e_2 - a_{\mathrm{del}} \\ e_5 &< e_1 + e_2 - e_4 + a_{\mathrm{ins}} - a_{\mathrm{del}} \\ &= e_2 + (e_1 - e_4) + a_{\mathrm{ins}} - a_{\mathrm{del}} \\ &\leq e_2 + a_{\mathrm{ins}} - a_{\mathrm{del}} \end{aligned} \tag{14}$$

となり，さらにすべての未訪問の点の中で，挿入したときのペナルティ減少量の最大値を $a_{\max}$ とすると式 (14) より

$$\begin{aligned} e_5 &< e_2 + a_{\mathrm{ins}} - a_{\mathrm{del}} \\ &< e_2 + \max_{v_{\mathrm{ins}}} a_{\mathrm{ins}} - a_{\mathrm{del}} \\ &= e_2 + a_{\max} - a_{\mathrm{del}} \end{aligned} \tag{15}$$

となる．つまり，case 2 においては任意の v_{del} に対して式 (13)，式 (15) のいずれかを満たすような v_{ins} を探索すればよいと言える．ここで式 (13)，式 (15) は，v_{del} を決定すると右辺が定数となり，左辺は v_{ins} により決まる変数となる．さらに左辺の e_4 と e_5 はそれぞれ v_{del} の前後の点から v_{ins} までの距離に相当する．つまり v_{ins} の候補として，v_{del} の 1 つ前の点からの距離が e_1 未満である点と v_{del} の 1 つ後の点からの距離が $e_2 + a_{\max} - a_{\mathrm{del}}$ 未満である点のみを探索すれば，改善を逃すことなく探索範囲を狭めることができると言える．この探索を実装するため，各訪問点 v_i に対し c_{ij} の値の昇順に v_j $(1 \leq j \leq n_v, i \neq j)$ をソートしたリストを記憶しておく．このリストは近傍リストと呼ばれる．v_{del} の前後の点の近傍リストを参照し，リストの先頭からそれぞれ式 (13)，式 (15) が成り立たなくなるまで辿ることで上記の探索を実行することができる．1-ins 操作の Step 3 や 1-del-ins 操作の Step 5 においても，この近傍リストを用いることで高速化を図っている．

case 3 では，1-ins では挿入できないが v_{del} を削除することで挿入することができるようになる v_{ins} を効率的に探索することを考える．具体的には，任意の v_{del} を削除したときにできる距離や要求量の各制約までの余裕を考え，その余裕に収まるような点の中で最も評価関数値が改善するような点を高速に探す．そこで，問題例の制約が

(i) 個数制約（各点の要求量がすべて等しいときの容量制約）のみである場合
(ii) 個数制約ではなく容量制約または距離制約がある場合

の 2 つに場合を分けて考える．

(i) の場合においては，各 v_{del} に対して改善量の最も大きな v_{ins} を探し出す操作を $O(1)$ で行う方法を提案する．ここで，1-ins 操作において評価関数値は改善するが制約を破ってしまうために挿入しなかった点の集合を I とおく．1-ins 操作を行う際に，I の中で改善量が大きな方から挿入場所が互いに異なる 3 点を巡回路ごとに記憶しておく．巡回路 k における上記 3 つの点の集合を I_k とする．I_k の構築は 1-ins による探索に付随して行うことができる．その後，1-del-ins において各 v_{del} に対し v_{del} が含まれる巡回路を k とし，挿入場所が v_{del} の前後ではない点の中で改善量が最大となる点を I_k から選び，その点を v_{ins} とする．制約は個数制約のみであるので，v_{del} を削除することで v_{ins} は必ず挿入することができる．また上記の I_k の構築法で，各 v_{del} に対して最も改善量が大きなものが必ず I_k に含まれると言える．各 v_{del} に対して最大で 3 つの点を調べるだけであるので，case 3 を満たすような v_{ins} を定数時間で探すことができる．

(ii) の場合，完全二分木を用いた方法により，$O(\log |I|)$ 時間で v_{ins} を探索できる（詳細は省略する）．

現実的には case 1 と case 2 には含まれないような $|I|$ の要素数は非常に少ないことが期待されるため，各 v_{del} に対して $|I_k|$ の要素を改善量の降順にソートし，その先頭から順番に探索を行っても現実的には高速に動作すると考えられる．一方，(ii) に対する高速化手法の実現には複雑なデータ構造の実装が必要となる．

3.3 パスの再構築法

本節では m-CTP に対する局所探索法の近傍として用いる部分パスの再構築法について説明する．巡回路から部分パスを除去することで被覆制約付き最短経路問題を作成し，それを解くことで得られるパスを元の解のものと置き換え，解を改善する．以下にこの操作の概要を示す．

パスの再構築法

Step 1 現在の解 σ に含まれる長さ β（β はパラメータ）以下の部分パス全ての集合を U とし，未探索の部分パス集合を $\tilde{U} := U$ とする．

Step 2 探索によって $\tilde{U}$ に含まれるパスが解から無くなった場合はそのパスを $\tilde{U}$ から削除する．また探索によって $\tilde{U}$ に含まれないパスが解に追加された場合はそのパスを $\tilde{U}$ へ追加する．$\tilde{U} = \emptyset$ ならば現在の解を出力し終了する．そうでないならば任意の部分パス $u \in \tilde{U}$ を選び，$\tilde{U} := \tilde{U} \backslash \{u\}$ とする．

Step 3 σ から u を除去し，部分問題を作成する．

Step 4 部分問題を解いて得られたパスを u' とする．

Step 5 σ の部分パス u を u' に置き換えた解 σ' が $f(\sigma') < f(\sigma)$ を満たすならば，$\sigma := \sigma'$ とする．

Step 6 Step 2 に戻る．

Step 3 における部分問題の作成方法を 3.3.1 節で，Step 3 における部分問題の解法を 3.3.2 節で説明する．

3.3.1 部分問題の作成法

部分問題の作成法の概要は以下のようである．現在の解

のあるルートに着目し，そのルートから 2 つの点 v_s, v_t を選ぶ．v_s から v_t へのパスに含まれる訪問点 (v_s, v_t 以外) をルートから除去したと仮定し，そのときに被覆されなくなる被覆点の集合を $W' \subseteq W$ とおく．また W' の点をひとつでも被覆するような訪問点の集合を $V' \subseteq V$ とおく．パスを除去されたルートに含まれる訪問点の要求量の総和を p から引いたものを p'，総距離を q から引いたものを q' とおく．p' を要求量の上限，q' を経路長の上限とした上で，v_s を出発し W' の点をすべて被覆するように V' 上の点を通って v_t へと到達するような最短の経路を求める問題 (被覆制約付き最短経路問題) を部分問題として出力する．

3.3.2 部分問題の解法

本節では 3.3.1 節で作成した部分問題である，被覆制約付き最短経路問題の解法として動的計画法（DP）を使用したものを提案する．$\hat{V}$ を訪問済みの点集合 ($\hat{V} \subseteq V'$)，$g(\hat{V}, v_i)$ を v_s を出発し $\hat{V}$ 内の点を通って $v_i \in \hat{V}$ に到達するような経路の中で最短のものの長さ，c_{ij} を v_i から v_j への距離とする．提案する DP の漸化式は

$$g(\{v_i\}, v_i) = c_{si} \quad (v_i \in V') \tag{16}$$

$$g(\hat{V}, v_i) = \begin{cases} \min\limits_{v_j \in \hat{V} \setminus \{v_i\}} g(\hat{V} \setminus \{v_i\}, v_j) + c_{ij} \\ \quad (\text{if } \sum\limits_{v_k \in \hat{V}} a_k \leq p' \text{ or } g(\hat{V} \setminus \{v_i\}, v_j) \leq q') \\ \infty \quad (\text{otherwise}) \end{cases} \tag{17}$$

と表される．$g(\hat{V}, v_i)$ に相当するラベルを $l_{\hat{V},v_i}$ とおく．また $l^t_{\hat{V}}$ を，v_s を出発して $\hat{V}$ 内の点をすべて通り v_t へ到達するような経路の中で最短のものの長さとする．今回は $\hat{V} = \emptyset$ から $\hat{V}$ のサイズを徐々に拡大しながら $l_{\hat{V},v_i}$ を計算していき，最終的に最小の $l^t_{\hat{V}}$ を求めるプログラムを実装した．ただし，$\hat{V} \cup \{v_i\}$ の点がすべての被覆点を被覆した時点でそれ以上 $\hat{V}$ を拡大することなく，$l^t_{\hat{V}\cup\{v_i\}} = l_{\hat{V},v_i} + c_{it}$ を計算する．これは距離 c_{ij} が三角不等式を満たしており，巡回路に点を追加してもコストは減少しないためである．以下に具体的なアルゴリズムを示す．

DP を用いた部分問題の解法

Step 1 $g(\hat{V}, v_i)$ に対応する $2^{|V'|} \times (|V'|)$ の大きさの表を作成する．各セルの値 $l_{\hat{V},v_i}$ ($\hat{V} \subseteq V', v_i \in \hat{V}$) の初期値を ∞ とする．初期値からの更新が起こったセルを表す $(\hat{V}, v_i)$ の組を記憶するキューを用意する．$l_{\{v_i\},v_i} := c_{si}$ ($v_i \in V'$) と表を更新し，キューに各 $(\{v_i\}, v_i)$ を入れる．

Step 2 キューが空である場合は Step 8 へ進む．そうでない場合はキューから $(\hat{V}, v_i)$ を取り出し，それぞれを現在の訪問済み点集合と現在の点とする．

Step 3 $\hat{V}$ 内の点をすべて訪れて被覆制約を満たす場合は $l^t_{\hat{V}\cup\{v_i\}} := l_{\hat{V}\cup\{v_i\},v_i} + c_{it}$ と表の値を更新し，Step 2 へ戻る．

Step 4 未訪問の訪問点の中で未探索の点の集合を $\tilde{V}$ とし，$\tilde{V} := V' \setminus \hat{V}$ とする．

Step 5 $\tilde{V} = \emptyset$ の場合は Step 2 へ戻る．そうでなければ訪問点 $v_j (\in \tilde{V})$ を選び，$\tilde{V} := \tilde{V} \setminus \{v_j\}$ とする．

Step 6 v_j が以下の条件を満たさない場合は Step 5 へ戻る．

(i) v_j を訪れることで新たに被覆点が被覆される
(ii) $l_{\hat{V},v_i} + c_{ij}$ が q' を超えない
(iii) $\sum_{v_k \in \hat{V} \cup v_j} a_k$ が p' を超えない
(iv) $l_{\hat{V},v_i} + c_{ij}$ が $l_{\hat{V}\cup v_i,v_j}$ より小さい

Step 7 $l_{\hat{V}\cup v_i,v_j} = \infty$ ならば $(\hat{V} \cup v_i, v_j)$ をキューへ追加する．$l_{\hat{V}\cup v_i,v_j} := l_{\hat{V},v_i} + c_{ij}$ として Step 5 へ戻る．

Step 8 得られた $l^t_{\hat{V}}$ の中で最小のものを部分問題の最適解として出力する．

各状態に対し Step 4 で $\tilde{V}$ のサイズが $O(|V'|)$，Step 6i で v_j に対し新たに被覆する点が存在するかどうかの判定に $O(|W'|)$ の計算時間を要する．また $\hat{V}$ は全部で $2^{|V'|+2}$ 通りある．よって全体の計算時間は $O(|V'| \cdot |W'| \cdot 2^{|V'|})$ となる．

4. 実験結果

この節では Hà らによる結果 [2] と村上による結果 [6] を比較対象として計算実験を行った結果を示す．プログラムは C++を用いて実装し，計算実験には 1.7GHz Intel Core i5 プロセッサ，4GB RAM を搭載する計算機を用いた．今回は 3.2.5 節で提案した 1-del-ins の高速化手法を用いず，実験に使用したプログラム内の 1-del-ins では各 v_{del} に対してすべての未訪問点を v_{ins} の候補として探索している．

Hà らの用いた問題例は X-$|T|$-$|V|$-$|W|$-p と名付けられる．ここで X は問題例を作成するのに用いた TSPLIB の問題例の名前を表している．また p は 1 つの巡回路が訪問できる頂点数を表している．距離制約は設定されていない．実験結果を表 1 に示す．表に記載されている時間はアルゴリズムが終了するまでにかかった CPU 時間で，単位は秒である．括弧内の時間は最良解が得られた時間を表している．最適解の値と一致している部分は太字で，Hà らの手法による値よりも良い場合は斜体で表示している．表 1 より，最適解の求められているすべての問題例に対して提案手法により最適解を求めることができていることがわかる．またすべての問題例において Hà らの結果と等しいか，あるいはより低いコストの解を出力できている．さらに，Hà らの手法では計算時間が大幅に増加してしまっている問題例に対しても，提案手法では短時間で精度の良い解を

表 1　Hà らの結果との比較

		Hà らの手法		提案手法	
問題例	最適値	目的関数	時間	目的関数	時間
A1-1-50-50-4	10,271	**10,271**	0.80	**10,271**	1.08 (0.02)
A1-1-50-50-5	9220	**9220**	0.78	**9220**	0.96 (0.01)
A1-1-50-50-6	9130	**9130**	0.81	**9130**	0.91 (0.02)
A1-1-50-50-8	9130	**9130**	0.81	**9130**	1.97 (0.02)
A1-10-50-50-4		17,973	3.26	*17,953*	0.93 (0.01)
A1-10-50-50-5	15,440	**15,440**	3.81	**15,440**	0.87 (0.03)
A1-10-50-50-6	14,064	**14,064**	3.85	**14,064**	0.88 (0.02)
A1-10-50-50-8		13,369	3.92	13,369	1.39 (0.04)
A2-1-100-100-4	11,885	**11,885**	2.89	**11,885**	1.39 (0.07)
A2-1-100-100-5	10,234	**10,234**	2.82	**10,234**	1.75 (0.10)
A2-1-100-100-6		10,020	2.92	10,020	2.50 (0.26)
A2-1-100-100-8		9093	2.92	9093	2.01 (0.18)
A2-20-100-100-4		26,594	43.91	*26,455*	1.83 (0.04)
A2-20-100-100-5		23,419	37.29	*23,255*	1.87 (0.15)
A2-20-100-100-6		20,966	39.50	20,966	2.34 (0.07)
A2-20-100-100-8		18,418	42.42	*18,415*	2.54 (0.13)

表 2　村上の結果との比較
($|V| = 400, |T| = 5, p = 75, q = 150, m = 20$)

	村上の手法		提案手法	
$\lvert W\rvert$	目的関数	時間	目的関数	時間
400	1318	50.70	1188	56.16
800	1360	60.00	1132	42.16
1200	1452	65.08	1204	42.22
1600	1477	69.07	1227	42.00
2000	1523	129.23	1247	24.39

表 3　村上の結果との比較
($|V| = 500, |T| = 5, p = 100, q = 150, m = 20$)

	村上の手法		提案手法	
$\lvert W\rvert$	目的関数	時間	目的関数	時間
500	1262	29.46	1187	50.00
1000	1344	50.23	1052	30.87
1500	1392	45.67	1155	44.75
2000	1452	567.1	1100	50.90
2500	1465	138.07	1185	55.49

出力できていることがわかる．

村上の用いた問題例はランダムに生成されたものであり，訪問点数を $|V| = 400, 500$，被覆点数を $|W| = |V|, 2|V|, \ldots, 5|V|$ としている．実験結果を表 2，表 3 に示す．ILS の終了条件は 60 秒を超えるまでとし，それまでに得られた最良解の値とその解が得られた時間を表に記載している．時間の単位は秒である．すべての問題例に対して村上による結果よりコストの低い解を出力できていることがわかる．

5. まとめ

本研究では被覆制約付き配送計画問題に対して 1-del/1-ins とパスの再構築法という操作を含む局所探索法を提案した．1-del/1-ins は解から点を削除する 1-del，解に点を挿入する 1-ins，点の削除と挿入を同時に行う 1-del-ins という 3 つの近傍操作からなる．探索中ではタブーリストを用いることにより，同じ解の行き来を防ぎ探索が広く進むようにした．パスの再構築法は解に含まれる部分パスを再構築し，解を改善する操作である．再構築の際に生成する部分問題を厳密に解く動的計画法を提案した．計算実験により既存の研究と比較し，精度や計算時間の点でより良い結果を得ることができた．

参考文献

[1] Gendreau, M., Laporte, G. and Frédéric, S.: The covering tour problem, *Operations Research,* 45 (1997) 568–576.
[2] Hà, M. H., Bostel, N., Langevin, A. and Rousseau, L.-M.: An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices, *European Journal of Operational Research,* 226 (2013) 211–220.
[3] Hodgson, M. J., Laporte, G. and Semet, F.: A Covering Tour Model for Planning Mobile Health Care Facilities in Suhum District, Ghana, *Journal of Regional Science,* 38 (1998) 621–638.
[4] Labbé, M. and Laporte, G.: Maximizing user convenience and postal service efficiency in post box location, *Belgian Journal of Operations Research,* 26 (1986) 21–35.
[5] 村上啓介：被覆制約付き巡回路問題に対する発見的解法，スケジューリング・シンポジウム 2012 講演論文集, (2012) 35–40.
[6] 村上啓介：列生成法と局所探索法を用いた被覆制約付き配送計画問題に対する解法，スケジューリング・シンポジウム 2014 講演論文集, (2014) 117–122.