連結制約と被覆制約を持つ施設配置問題に対する発見的解法

M2012MM049 山之内亮介
指導教員：佐々木美裕

1　はじめに

本研究では，連結制約と被覆制約を持つ施設配置問題 (CCFLP) をネットワーク上で考える．この問題では，配置される p 個の施設によってすべての需要点が被覆されること (被覆制約) に加え，任意の需要点間 (OD ペア) が施設を経由して連結となること (連結制約) が要求される．目的は，OD ペア間の距離の総和を最小化することである．施設を電気自動車 (EV) の急速充電器と考えると，対象地域内のすべての OD ペア間を EV で往来可能となるように p 個の急速充電器を配置する問題となる [7]．他の応用例として，センサネットワーク構築の最適化などが挙げられる [1, 2, 5, 6]．

同様の問題に対して，山之内，保田 [7] は最適化ソフトウェアを用いて厳密な最適解を求めた．しかし，大規模な問題の解を求めることはできていない．そこで，本研究では，この配置問題に対する発見的解法を提案し，大規模な問題の解を求めることを目的とする．

2　モデルの説明

V_C，V_D を $V_C \cap V_D = \emptyset$ を満たす点集合，2 つの枝集合 E_C^{all} と E_D^{all} を $E_C^{all} = \{(u,v)|u \in V_C, v \in V_C\}$，$E_D^{all} = \{(u,v)|u \in V_C, v \in V_D\}$ とし，$E_C \subseteq E_C^{all}$，$E_D \subseteq E_D^{all}$ と定義する．$V = V_C \cup V_D$ と $E = E_C \cup E_D$ に対し，無向グラフ $G = (V, E)$ を考える．$W \subseteq V_C$ に対し，$W \cup V_D$ による G の誘導部分グラフを $G_W = (W \cup V_D, E_W)$ とする．任意の無向グラフ $G = (V, E)$ と G の各枝の長さ $l : E \to \mathbb{R}$ で定義されるネットワーク $\mathcal{N} = (V, E)$ 上において，$d_G(u, v)(u \in V, v \in V)$ を $\mathcal{N} = (V, E)$ 上の最短経路の長さとする．ただし，u から v へ到達不可能な場合は，$d_G(u, v) = \infty$ と仮定する．また，すべての OD ペアの集合 Π^{all} を $\Pi^{all} = \{(u,v)|u \in V_D, v \in V_D\}$ とする．OD ペアの集合 $\Pi \subseteq \Pi^{all}$ に対して，$\Pi \subseteq \Pi^{all} = \{(u,v)|u \in V_D, v \in V_D\}$ とすると，CCFLP は次のように定式化できる．

[CCFLP]

$$\text{minimize} \qquad \sum_{(u,v)\in\Pi} d_{G_W}(u,v) \tag{1}$$

$$\text{subject to} \qquad W \subseteq V_C,\ |W| = p. \tag{2}$$

G_W が非連結の場合，目的関数値が ∞ となるので，G_W が連結とならなければならない．よって，CCFLP は，W による $G_C = (V_C, E_C)$ の誘導部分グラフが連結であること (連結制約) と，すべての $v \in V_D$ について $\{(u,v)|(u,v) \in E_W, u \in W\} \neq \emptyset$ とすること (被覆制約) の制約を満たす $W \subseteq V_C$ の中で目的関数 (1) を最小化するものを見つける問題となる．ここで，V_C を急速充電器の配置候補点集合，1 回の充電で EV が走行できる限界距離を L として，

$$E_C = \{e \in E_C^{all} \mid l(e) \leq L\}, \tag{3}$$

$$E_D = \{e \in E_D^{all} \mid l(e) \leq L/2\}, \tag{4}$$

$$\Pi = \{(u,v) \in \Pi^{all} \mid d_{G_L}(u,v) \leq L/2\} \tag{5}$$

とすれば，CCFLP は EV 専用急速充電器の配置問題 [7] と等価となる．ただし，G_L は，需要点集合 V_D と枝集合 $\{(u,v)|u \in V_D, v \in V_D\}$ で定義される無向グラフである．

3　解法

本研究では，2 段階に分けて問題を解く．第 1 段階では，貪欲算法とけちけち法に基づく発見的解法を用いて，連結制約と被覆制約を満たす解を生成する．第 2 段階では，第 1 段階で求めた解を初期解として近傍探索，アニーリング法を行い，改善解を算出する．

3.1　初期解の生成

本節では，初期解の生成方法として，貪欲算法とけちけち法に基づく発見的解法を提案する．これらの解法において，W に追加および削除する $u \in V_C$ を選択する順番を決定するために，各点の「重要度」を最短経路数え上げ問題 [4] を用いて設定する．最短経路数え上げ問題は，最短経路として使用される枝の回数を数え上げるものであるが，本研究では点の使用回数を求めるものとする．最短経路数え上げ問題によって求めた使用回数が多い点ほど，OD ペア間の距離の総和が小さくできると考えられるので，これを点の「重要度」として定める．$\mathcal{N} = (V, E)$ 上において，対象とする OD ペア (u, v) の数は (5) より，$|\Pi|$ であるので，最短経路は全部で $|\Pi|$ 本存在する．このとき，$u \in V_C \subseteq V$ がこれら $|\Pi|$ 本の最短経路を構成する点として使われた回数を $w(u)$ とし，これを施設配置の選択基準として使用する．最短経路の算出にはダイクストラ法を用いる．

3.1.1　貪欲算法

この解法では，すべての $u \in V_C$ の中で，重要度 $w(u)$ の高い順に W に追加し，連結制約，被覆制約および $|W| \geq p$ を満たすまで W を更新する．

3.1.2　けちけち法

この解法では，はじめに $W := V_C$ とし，重要度 $w(u)$ の低い順に，W から $u \in W$ を削除し，$|W| = p$ を満たすまで W を更新する．ただし，$u \in W$ を W から削除することにより誘導部分グラフ G_W が非連結になる場合は削除しない．

3.1.3　貪欲算法の改良

この解法では，施設を配置する $u \in V_c$ を選ぶ基準として，重要度 $w(u)$ に加えて，ネットワーク上で $u \in V_C$ に隣接している $v \in V_D$ の数 $c(u)$ を用いる．この $c(u)$ を，本研究では $u \in V_C$ の被覆数と呼ぶ．

貪欲算法では評価値が最良である解を 1 つだけ保存するのが一般的であるが，ここでは，複数の解を保存し，よ

り良い解を求める．解を生成する際に，$u \in V_C$ の被覆数 $c(u)$，重要度 $w(u)$ を評価基準とする貪欲算法を用いる．

[改良型貪欲算法]

入力: $\mathcal{N} = (V, E)$; **出力:** $W_{ik} \subseteq V_C$

ステップ 0: $i := 1$ とする．また，すべての $v \in V_D$ に対して $d(v) := |\{(u, v)|(u, v) \in E_D, u \in V_C\}|$ を求め，$F := \{u \in V_C|(u, v') \in E_D, d(v') = 1\}$ とする．

ステップ 1: $W_{ik} := F(k = 1, \cdots, \sigma)$，$C'_i := V_C \backslash F$，$D'_i := V_D \backslash \{v \in V_D|(u', v) \in E_D, u' \in F\}$ とする．また，すべての $u_i \in C'_i$ に対して $c(u_i) := |\{(u_i, v)|(u_i, v) \in E_D, v \in D'_i\}|$ を求める．

ステップ 2: すべての $u_i \in C'_i$ を $c(u_i)$ の大きい順に並べ替え，大きい順に σ 個だけ $u_i^1, \cdots, u_i^\sigma$ とする．ただし，$c(u_i)$ の値が同じ $u_i \in C'_i$ が複数ある場合，$w(u_i)$ の値が大きいものを優先する．

ステップ 3: $W_{ik} := W_{ik} \cup \{u_i^k\}$，$C_{ik} := C'_i \backslash \{u_i^k\}$，$D_{ik} := D'_i \backslash \{v \in D'_i|(u_i^k, v) \in E_D\}(k = 1, \cdots, \sigma)$ とする．

ステップ 4: $k := 1$ とする．

ステップ 5: $(\mathcal{N}, W_{ik}, C_{ik}, D_{ik})$ を入力として，**[被覆制約を満たす点集合の探索]** へ．

ステップ 6: (G_C, W_{ik}) を入力として，**[連結制約を満たす点集合の探索]** へ．
また，$k = \sigma$ ならば，ステップ 7 へ進む．そうでなければ，$k := k + 1$ とし，ステップ 5 へ戻る．

ステップ 7: $W_{ik}(k = 1, \cdots, \sigma)$ の目的関数値をそれぞれ求める．目的関数値の 1 番の小さい W_{ik} を保持し，$F := F \cup \{u_i^k\}$ とする．$i = p$ ならば W_{ik} を出力して終了．そうでなければ，$i := i + 1$ とし，ステップ 1 へ戻る．

以下に被覆制約を満たす点集合の探索と，連結制約を満たす点集合の探索のアルゴリズムを記述する．

[被覆制約を満たす点集合の探索]

入力: $\mathcal{N} = (V, E), W_0 \subseteq V_C, C, D$; **出力:** $W \supseteq W_0$

ステップ 0: $W := W_0$ とする．

ステップ 1: すべての $u \in C$ に対して $c(u) := |\{(u, v)|(u, v) \in E_D, v \in D\}|$ を求め，$S := \{u \in C \mid c(u) = \max_{u \in C} c(u)\}$ とする．

ステップ 2: $u \in S$ のうちで $w(u)$ が最大であるものを u^* とし，$W := W \cup \{u^*\}, C := C \backslash \{u^*\}, D := D \backslash \{v \in D|(u^*, v) \in E_D\}$ とする．$D = \emptyset$ であれば，W を出力して終了する．$D \neq \emptyset$ であれば，ステップ 1 へ戻る．

[連結制約を満たす点集合の探索]

入力: $G_C = (V_C, E_C), W_0 \subseteq V_C$; **出力:** $W \supseteq W_0$

ステップ 0: $W := W_0$ とし，W による G_C の誘導部分グラフを G_W とする．

ステップ 1: G_W が非連結であればステップ 2 へ進む．G_W が連結で $|W| \geq p$ であれば，W を出力して終了する．G_W が連結で $|W| < p$ であれば，ステップ 6 へ進む．

ステップ 2: G_W の連結成分を $H_W^j(V_j, E_j)(j = 1, \cdots, m)$ とする．連結成分 $H_W^j(j = 1, \cdots, m)$ を 1 点 s_j に縮約して得られるグラフを $\hat{G}(\hat{V}_C, \hat{E}_C)$ とし，$\hat{G}$ と $\hat{G}$ の各枝の長さ $l : e \mapsto 1(e \in \hat{E}_C)$，点の重み $m : s_j \mapsto |V_j|(j = 1, \cdots, m)$ で定義されるネットワークを $\hat{\mathcal{N}} = (\hat{V}_C, \hat{E}_C)$ とする．また，$\hat{S} = \{s_j|j = 1, \cdots, m\}$ とする．

ステップ 3: $\hat{\mathcal{N}} = (\hat{V}_C, \hat{E}_C)$ 上の $\hat{S}$ の中で $d_G(u, v)(u, v \in \hat{S}, u \neq v)$ の長さが最短となる s_j 間のみを数え上げ，s_j の重みを考慮した点の重要度 $\hat{w}(u)(u \in V_C \backslash W)$ を求める．

ステップ 4: $u^* = \underset{u \in V_C \backslash W}{\operatorname{argmax}} \hat{w}(u), W := W \bigcup \{u^*\}$ とする．

ステップ 5: $\hat{G}(\hat{V}_C, \hat{E}_C)$ の連結成分の数が m より小さい場合，ステップ 1 へ戻る．そうでなければ，ステップ 4 へ戻る．

ステップ 6: $\hat{C} := V_C \backslash W$ とする．$\hat{C}$ のうちで $w(u)$ が最大であるものを u^* とし，$W := W \bigcup \{u^*\}$ とする．$|W| = p$ ならば W を出力して終了する．そうでなければステップ 6 へ戻る．

3.2 近傍探索

本研究では，近傍 $N(W)$ を，$v \in V_D$ に対して，$\{(u_1, v) \in E_D|u_1 \notin W\}$ と $\{(u_2, v) \in E_D|u_2 \in W\}$ を満たす u_1，u_2 を $W \cup \{u_1\} \backslash \{u_2\}$ とすることより得られる解とする．式で表すと以下のようになる．

$$N(W) = \{W'|W' = W \cup \{u_1\} \backslash \{u_2\}, u_1 \notin W, u_2 \in W, (u_1, v) \in E_D, (u_2, v) \in E_D\}.$$

解の評価関数 $\tilde{f}$ は OD ペアの移動距離の総和である．探索空間は実行可能領域内とする．確率 $\rho(0 \leq \rho < 1)$ で現在の解よりも悪くなる遷移を許す．探索の終了条件は，あらかじめ定めた探索回数に達したときとする．

3.3 アニーリング法

アニーリング法 (SA 法) とは，近傍探索を実行する過程に確率的な振る舞いを加え局所最適解に陥らないようにした手法である．ある確率で現在の解よりも悪くなるような遷移も許し，その確率を温度と呼ばれるパラメータ t で制御する．評価値が悪くならない解への遷移確率を 1 とし，改悪解への遷移確率を改悪量を Δ として $e^{-\Delta/t}$ とする．SA 法では初期温度，温度の更新方法，ループ (一定温度での反復) およびアルゴリズムの終了条件を決定する必要がある．これらの決定方法を以下に述べる．

本研究では，温度の更新方法について，Kirkpatrick ら [3] に従う．初期の採択割合は 1 に非常に近くなることが望ましいので，初期温度を高めに設定する必要がある．ここでは初期温度 $t_{\max}$ を，探索の過程で初めて見つかった改悪解を W'_0，そのときの暫定解を W_0 としたとき，$t_{\max} = -(\tilde{f}(W'_0) - \tilde{f}(W_0))/\ln(0.9)$ と定める．

温度の更新方法は，$t := \alpha \cdot t(0 < \alpha < 1)$ とする．また，暫定解が見つかった時点の温度 t_{found} を記憶し，その後，

探索を $|V_D|^2$ 回反復して暫定解が更新されなかった場合，温度を $t := t_{\text{found}}$ と一旦高くする．

ループの終了条件は，$|V_D|$ 回反復したときとする．探索の終了条件はループの中で解の受理頻度が p_{freeze} 以下となる反復が r_{freeze} 回続いたときとする．p_{freeze} と $r_{\text{freeze}}(0 < p_{\text{freeze}} < 1 \leq r_{\text{freeze}})$ はパラメータとする．

[SA 法]

入力: $\mathcal{N} = (V, E), W \subseteq V_C$**; 出力:** $W^* \supseteq W$

ステップ 0: $W^* := W$ とする．

ステップ 1: 以下のステップ a, b, c, d および e を，ループの終了条件が満たされるまで反復する．

a: $N(W)$ 内の解をランダムに 1 つ選び W' とする．

b: $W' \cup V_D$ による誘導部分グラフ $G_{W'}$ が非連結ならばステップ a へ戻る．$G_{W'}$ が連結ならばステップ c へ進む．

c: $\Delta = \tilde{f}(W') - \tilde{f}(W)$ とする．

d: $\Delta \leq 0$ ならば確率 1 で，$\Delta > 0$ ならば確率 $\mathrm{e}^{-\Delta/t}$ で $W := W'$ とし，ステップ e へ進む．そうでなければ，ステップ a へ戻る．

e: $\tilde{f}(W) \leq \tilde{f}(W^*)$ ならば，$W^* := W$ とし，ステップ a へ戻る．

ステップ 2: 探索の終了条件を満たされれば，W^* を出力して終了する．そうでなければ，温度 t を更新した後ステップ 1 へ戻る．

4 計算実験

4.1 準備

本節では，300km 四方の領域内に需要点と候補点をランダムに配置した 2 つの例題を用いて，3 節で述べた解法の評価を行う．各例題の需要点数，候補点数，OD ペア数および閾値 L は表 1 の通りである．

表 1 実験に使用したデータ

例題	$\lvert V_D\rvert$	$\lvert V_C\rvert$	$\lvert\Pi\rvert$	枝の距離の閾値 L
1	30	50	403	100km
2	50	80	1159	

提案解法は，Microsoft[1] Visual C++[1]2010 Express で実装した．厳密解を求めるための厳密解法として，山之内，保田 [7] のモデルを用いる．最適化計算には，最適化ソフトウェア IBM[2] ILOG[1] CPLEX[2] Optimization Studio 12.5 を使用する．使用した計算機に搭載されている CPU は Intel[3] Core[3] i7-960 Processor, メモリは 24GB である．

4.2 初期解の生成

3.1 節で述べた解法により得られた値と厳密な最適値を比較する．各解法で，例題 1 に対し p の値を 1〜40，例題 2 に対し p の値を 1〜66 と変えて実行した．厳密解法と各提案解法の実行可能解が得られる最小の p の値，最大の実行時間を表 2 に示す．

表 2 より，計算時間は，山之内，保田 [7] モデルでは p の値によって変化し，18 時間程度要するものも存在するが，提案解法では全て 8 秒以内である．貪欲算法では，p の値が小さい場合，実行可能解を求めることができないが，その他の解法では，p の値が小さい場合でも，実行可能解を求めることができた．

表 2 初期解生成法の実行結果 (例題 1，2)

例題	実行可能解が得られる最小の p			実行時間 (秒)		
	厳密解法	貪欲算法	けちけち法	厳密解法	貪欲算法	けちけち法
1	11	20	12	3145.8	0.1	0.1
2	15	42	17	66632.8	0.1	0.1

例題	実行可能解が得られる最小の p			実行時間 (秒)		
	改良型貪欲算法			改良型貪欲算法		
	$(\sigma = 1)$	$(\sigma = 3)$	$(\sigma = 5)$	$(\sigma = 1)$	$(\sigma = 3)$	$(\sigma = 5)$
1	12	11	11	0.1	0.8	1.4
2	16	15	15	0.1	4.1	7.8

提案解法 (けちけち法，改良型貪欲算法) で得られた値と，厳密な最適値との相対誤差を，図 1，2 に示す．貪欲算法はで得られた値はすべてけちけち法で得られたものと同じとなった．

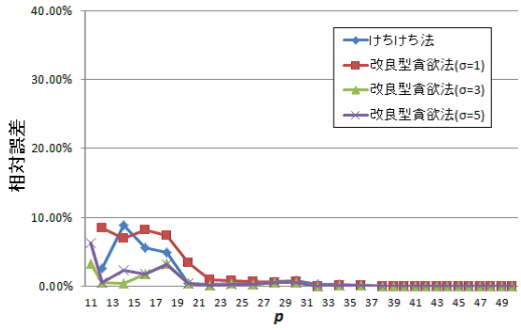

図 1 例題 1 の厳密な最適値との相対誤差

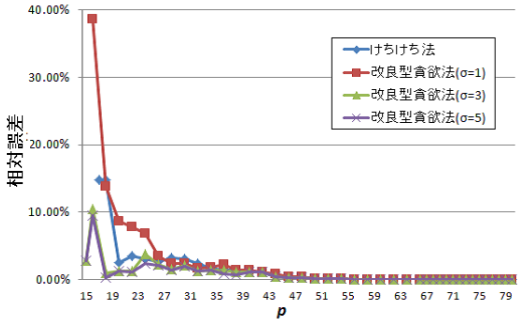

図 2 例題 2 の厳密な最適値との相対誤差

図 1，2 より，p の値が，例題 1 ではおよそ 20，例題 4 ではおよそ 45 より大きくなると，どの解法でも相対誤差が 1% よりも小さくなった．$\sigma = 5$ としたときの改良型貪欲算法は，各例題で精度の良い解を求めることができるが，例題 1 の $p = 11$ の場合のように $\sigma = 3$ としたときよりも精度の悪い解を求める場合も存在する．よって，単純に σ の値を大きく設定するだけでは，精度の良い解を求めることができるわけではないことがわかる．

4.3 近傍探索

4.2 節で求めた解を初期解とし，近傍探索法と SA 法を用いて改善解を求める．近傍探索の終了条件となる探索回数は，SA 法が終了条件を満たしたときの探索回数と同数とする．また，確率 ρ を 0 と 0.1 と変えて，各例題，各初期解で実験する．

SA 法のパラメータは $\alpha = 0.9$，$p_{\text{freeze}} = 0.05$，$r_{\text{freeze}} = 20$ とする．今回は，例題 1 の $p = 12, 14$ と例題 2 の $p = 16, 18$ のときの解のみを実験する．また，解法の精度を正確に評価するため，近傍 $N(W)$ 内の解をランダムに選択する際に使用する計算機上のパラメータである乱数の種 (seed) を変更し，各例題に対し 3 度の計算実験を行う．

各初期解から近傍探索，SA 法を用いて求めた解の相対誤差と実行時間を表 3，4 に示す．記述している値は 3 度の計算実験で得られた値の平均である．また，けちけち法では，例題 2 の $p = 16$ とした場合に解を求めることができなかったので，N/A とする．

表 3，4 より，どの解法を用いても，各例題で大幅に解は改善され，厳密解を求めたものもあった．また，どの解法でも実行時間に差はあまりない．近傍探索の ρ を 0 と 0.1 とした場合を比較すると，0.1 の場合の方が精度の良い解を求める可能性が高いことがわかる．例題 1 では，SA 法を用いたときに比較的精度の良い解を求めているが，例題 2 では $\rho = 0.1$ としたときの近傍探索の方が精度の良い解を求めている．

表 3　例題 1 の近傍探索 ($\rho = 0, 0.1$)，SA 法の実行結果

近傍探索法 ($\rho = 0$)								
p	相対誤差 (%)				実行時間 (秒)			
	けちけち法	改良型貪欲算法			けちけち法	改良型貪欲算法		
		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$
12	0.62	0.31	0.00	0.00	1.8	2.4	2.0	2.1
14	3.66	0.34	0.43	0.43	2.7	3.2	3.1	3.0
近傍探索法 ($\rho = 0.1$)								
p	相対誤差 (%)				実行時間 (秒)			
	けちけち法	改良型貪欲算法			けちけち法	改良型貪欲算法		
		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$
12	0.00	0.00	0.00	0.00	1.8	2.4	2.1	2.1
14	0.68	1.06	0.32	0.46	2.6	2.7	2.5	2.5
SA 法								
p	相対誤差 (%)				実行時間 (秒)			
	けちけち法	改良型貪欲算法			けちけち法	改良型貪欲算法		
		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$
12	0.19	0.00	0.00	0.00	3.8	5.1	4.5	4.5
14	0.00	0.00	0.18	0.06	4.7	5.8	4.9	5.0

表 4　例題 2 の近傍探索 ($\rho = 0, 0.1$)，SA 法の実行結果

近傍探索法 ($\rho = 0$)								
p	相対誤差 (%)				実行時間 (秒)			
	けちけち法	改良型貪欲算法			けちけち法	改良型貪欲算法		
		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$
16	N/A	6.90	6.90	3.63	N/A	7.5	7.9	5.6
18	7.52	1.25	0.26	0.26	11.1	9.1	9.9	10.6
近傍探索法 ($\rho = 0.1$)								
p	相対誤差 (%)				実行時間 (秒)			
	けちけち法	改良型貪欲算法			けちけち法	改良型貪欲算法		
		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$
16	N/A	7.73	4.60	3.63	N/A	19.1	21.2	14.4
18	1.43	0.56	0.13	0.00	10.0	8.6	9.5	10.2
SA 法								
p	相対誤差 (%)				実行時間 (秒)			
	けちけち法	改良型貪欲算法			けちけち法	改良型貪欲算法		
		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$
16	N/A	4.60	6.90	3.63	N/A	7.4	12.1	7.8
18	0.37	1.26	0.31	0.18	10.5	8.9	15.9	17.5

5　おわりに

本研究では，CCFLP に対する発見的解法を複数提案した．初期解の生成において，p の値が小さい場合，貪欲算法では実行可能解を求めることができなかったが，けちけち法ではできた．また，実行時間も非常に短く，実用的な解法である．改良型貪欲算法では，短時間で厳密解に近い解を求めることが可能だが，パラメータ σ の設定が難しい．近傍探索，SA 法を用いることで，高速に高精度の解を求めることが可能であり，厳密な最適解を求めることもできた．ただし，改良型貪欲算法と同様に，適切なパラメータの設定が難しく，また，問題の規模が大きいと解の評価値を計算する際に膨大な時間を費やしてしまう．

今後は，近傍探索と SA 法のパラメータの値や終了条件を変化させ，繰り返し実験を行うことで，CCFLP に対する最適な戦略を見つけ出したい．また，解の評価値の計算時間の短縮を考えていきたい．現在は探索解の評価を行う度に，すべての OD ペア間の距離を算出しているため，OD ペアの数によって大きく時間が変化してしまう．よって，すべての OD ペア間の距離を算出するのではなく，暫定解と探索解の変更された要素によって，経由する施設が変化する OD ペア間のみの距離を算出することで計算時間の短縮ができると考える．

参考文献

[1] T. Furuta, M. Sasaki, F. Ishizaki, A. Suzuki, and H. Miyazawa (2009) : A new clustering model of wireless sensor network using location theory, *Journal of the Operations Research Society of Japan*, Vol. 52, pp. 366-376.

[2] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan (2002) : An application-specific protocol architecture for wireless microsensor networks, *IEEE Transactionson Wireless Communications* 1, pp. 660-670.

[3] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi (1983) : Optimisation by simulated annealing, *Science*, 220, pp. 671-680.

[4] 大山達雄 (1998) :「最短経路数え上げ問題とその応用」，日本 OR 学会春季研究発表会アブストラクト集 1998，pp. 166-167.

[5] 佐々木美裕 (2007) :「センサネットワーク構築と最適化問題」，第 19 回 RAMP シンポジウム論文集，pp. 31-44.

[6] M. Sasaki, T. Furuta, F. Ishizaki, and A. Suzuki (2012): A mathematical programming approach to the multi-round topology construction problem in wirless sensor networks, *Journal of the Operations Research Society of Japan*, Vol. 55, pp. 199-208.

[7] 山之内亮介，保田将弘 (2012):「電気自動車専用急速充電器の最適配置問題」，南山大学数理情報学部 2011 年度卒業論文.

[1] Microsoft，Visual C++は米国 Microsoft Corporation の米国およびその他の国における登録商標です.

[2] IBM，ILOG, CPLEX は IBM Corporation の登録商標です.

[3] Intel, Core は Intel Corporation および子会社の米国およびその他の国における登録商標です.