発見的解法：収集/配送経路問題への適用

沼田　一道

順序や組合せを決定する最適化問題においては，厳密な最適解を見いだすのが困難なことが多い．このような場合に活躍するのが発見的解法である．できるだけ良い解を現実的な時間内に求める発見的解法は，個々の問題に柔軟に対応できるので，極めて実用的である．本稿では，訪問点の組分けと各組の巡回を同時に最適化する収集/配送経路問題（VRP）に即して発見的解法を紹介し，ポストから郵便物を収集する際の問題を扱ってみる．

キーワード：発見的解法，配送経路問題，巡回セールスマン問題，局所探索

1. はじめに

「詰込む」，「選ぶ」，「巡る」，「分割する」，「覆う」等の作業を含む仕事を「うまく」行おうとするとき，我々は様々なタイプの組合せ最適化問題に出会う．ただし，それらは何らかの列挙操作なしに最適解を求めるのが困難と思われる性質の悪い問題であることが多い．

厳密な最適解を求めるのが困難な場合でも，「解けません」では済まないので，できるだけ良い解を現実的な時間内に求める必要が生ずる．この要求に応えるのが「発見的解法」である．陰的列挙を土台とした汎用ソルバーにより，かなりのサイズまで厳密に解ける問題もあるが，例えば「巡回」を含む問題などは定式化（問題の数式表現）そのものが組合せ的サイズになるので，発見的解法に頼らざるを得ない．

本稿では，巡回を含む組合せ最適化問題として収集/配送経路問題（VRP：Vehicle Routing Problem）を取り上げ，それに即して発見的解法を紹介し，ポストから郵便物を収集するVRPの実例を解いてみる．

2. 発見的解法

発見的解法（ヒューリスティックス）とは，解構成要素の評価値あるいは解自体の評価値（目的関数値）を用いて良い解を得る可能性が高いと思われる方向に解を構成していく，あるいは解の探索を進めていく方法である．問題に応じて評価基準を設定しコンピュータを用いて広く組織的に探索を行うので，経験や勘に基づく直感的な方法よりは格段に良い解を与える．探索の基本的枠組みは単純なので高速に実行できるし，様々なモデルの変更にも柔軟に対処できる．また，計算時間をある程度犠牲にし，メタ戦略[5]を適用したり，緩和解情報を利用したりして得られる解は（結果的に）最適であることも少なくなく，「準最適解」と呼ばれたりする．

発見的解法は大きく二つ分類される．

- 構築法：何もないところから評価基準に従い解を構成していく．
- 改善法：何らかの手段で得た初期解を逐次改善していく．

単純な構築法は，評価値の高い順に要素を解に取り込んでいくので「貪欲解法」と呼ばれることもある．改善法は，現在の解を少しだけ変更して得られる解全体の集合（近傍）を探索し，その中に現在解より良い解があれば，その解を次の現在解として探索を繰り返すので，局所探索法とも呼ばれる．コンピュータの性能が十分でない時代には構築法が使われたが，高速なCPUを手軽に利用できる現在では局所探索法が主流である．

良く知られた巡回セールスマン問題（TSP：Traveling Salesman Problem）を例として構築法と改善法の違いを見ておこう．TSPは「巡る」作業に関するものでVRPの部分問題として現れる．

TSP：n個の訪問すべき都市と各都市間の移動コストが与えられる．ある（任意の）都市から出発した

ぬまた　かずみち
東京理科大学 工学部経営工学科
〒162-8601 新宿区神楽坂1-3

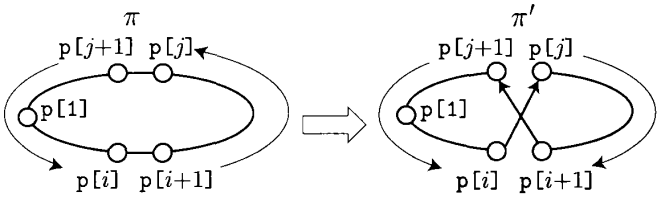

図1　2-opt 近傍

セールスマンが，すべての都市を1度ずつ訪問して出発点に戻るとき，移動コストの総和が最小となる巡回路を求めよ．

構築法の例として Nearest Neighbor 法を，逐次改善法の例として 2-opt 法を挙げる．

Nearest Neighbor 法　k 番目に訪問する都市（点）を p[k] に格納し，π=(p[1], p[2], ⋯, p[n]) で巡回路を表す．

step 0 ：適当な都市を選び p[1] に格納する；
　　その都市を既訪問とする；$k \leftarrow 1$

step 1 ：p[k] からの移動コストが最も小さい未訪問都市を選んで p[$k+1$] に格納；
　　その都市を既訪問とする；$k \leftarrow k+1$

step 2 ：$k<n$ ならば step 1 へ，
　　そうでなければ π を近似解として終了．

最初はコストの小さい枝が選ばれるが，最後のほうになると自由度がなくなり，コストの大きい枝を選ぶ可能性が大きくなるので，あまり良い解は得られない．

2-opt 法　現在解 π=(p[1], p[2], ⋯, p[n]) の近傍を

$$U(\pi)=\{(\mathrm{p}[1], \mathrm{p}[2], \cdots, \mathrm{p}[i], \mathrm{p}[j], \mathrm{p}[j-1], \cdots, \mathrm{p}[i+1], \mathrm{p}[j+1], \mathrm{p}[j+2], \cdots, \mathrm{p}[n]) \mid 1 \le i \le n-2,\ i+2 \le j \le n \text{ or } n-1\}$$

で定義して局所探索法を行う．

$U(\pi)$ は図1のように π の2つの枝を交換してできる巡回路全体の集合であり，2-opt 近傍とも呼ばれる．TSP に対する逐次改善法については4節で詳しく述べる．

3.　収集/配送経路問題

VRP は「分担して」して「巡る」問題である（図2）．

VRP：配送拠点（デポ）にある品物を，積載量制限のある複数台の搬送車（ビークル）で分担して，各需要点の要求量を満たすよう配送する．ビークルはデポを出発し担当する需要点に要求量分ずつ順次配送してデポに戻る．各ビークルの移動コストの総和を最小にする分担と巡回を求めよ．

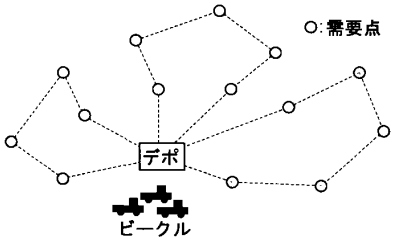

図2　VRP の構成要素

「配送」の代わりに「収集」を考えても同じ問題である．物品の収集/配送を伴わず，「分担して訪問する」だけの問題は，m 人巡回セールスマン問題と呼ばれる．

ビークルの集合を $K=\{1, 2, \cdots, m\}$，需要点の集合を $I=\{1, 2, \cdots, n\}$，デポを番号0で表す．デポを含む点間の移動コスト c_{ij}，需要点 i の要求量 q_i，ビークル k の積載量限度 Q_k が与えられているものとする．

ここでは VRP を，多少変則的であるが，各ビークルの担当需要点集合を「変数」とし，「総移動コスト最小化する需要点割当て（組分け）問題」として定式化する．また，関数 $f(S)(S \subset I')$ で点集合 S を巡回する最小の移動コストを表す．

$$\text{minimize} \quad \sum_{k=1}^{m} f(\{0\} \cup S_k) \tag{1}$$

$$\text{sub. to} \quad S_k \in 2^I \quad k \in K, \quad \bigcup_{k=1}^{m} S_k = I \tag{2}$$

$$S_k \cap S_l = \phi \quad k, l (k \neq l) \in K \tag{3}$$

$$\sum_{i \in S_k} q_i \le Q_k \quad k \in K \tag{4}$$

S_k は第 k ビークルが担当する需要点集合，$f(\{0\} \cup S_k)$ はデポと S_k を巡回する最小の移動コストである．発見的解法の多くは，この定式化に基づく．

4.　VRP の解法：割当てと巡回

実際的な VRP を解くには，発見的解法が使われている．古くから知られた方法として，割当てと巡回路形成を同時に行うセービング法や幾何的な基準で S_k を決定してから $f(\cdot)$ を計算するスウィープ法などがある[3][6]．これらは（半）構築法であり高速ではあるが精度的には改善の余地もある．最近では，CPU の飛躍的な高速化を背景として，$f(\cdot)$ の計算とそれによる S_k の更新を繰返す逐次改善/局所探索法が提案されている．後者を念頭に $f(\cdot)$ の計算法と S_k の更新法を紹介する．

4.1　$f(\cdot)$ の計算

$f(\{0\} \cup S_k)$ の計算は，TSP の求解に他ならない．S_k を変化させながら何回も計算するので，高速でそ

れなりに良い解を求める逐次改善法—2-opt法，3-opt法，Lin-Kernighan法など—を用いるのが普通である．以下のようにプログラムされた関数fは，対象点数nと，各点の識別番号/初期訪問順(p[1], p[2], …, p[n])を受け取り，2-opt法で逐次改善して局所最適巡回路を求め，その総移動費用を値として返す．

```
function f(n, p)
{repeat   change:=false;
  for i:=1 to n-2 do{
    if i=1 then ne: =n-1 else ne:=n;
    for j:=i+2 to ne do{
      i1:=i+1;
      if j=n then j1:=1 else j1:=j+1;
      delta:=c[p[i], p[j]]+c[p[i1], p[j1]]
                -c[p[i], p[i1]]-c[p[j], p[j1]];
      if delta<0 then{
        for k:=i+1 to i+⌊(j-i)/2⌋ do{
          tmp:=p[k];
          p[k]:=p[i+j+1-k];
          p[i+j+1-k]:=tmp;
        }; change:=true;
      }}}
  until (not change);
  w:=c[p[n], p[1]];
  for i:=1 to n-1 do w:=w+c[p[i], p[i+1]];
  f:=w
}/* end of function f*/
```

8行目では，巡回路π'とπ（図1）の移動コストの差(*delta*)を計算している．*delta*が負であれば改善である．2-opt法では，π'が改善解であれば即時に現在解をそれで更新し($\pi \leftarrow \pi'$)，探索を継続する．この操作を現在解πの近傍$U(\pi)$内に改善解が存在しなくなるまで繰り返す．改善できなくなったときの現在解(p[1], …, p[n])は局所最適である．

3-opt法は3本の枝交換によって2-opt法と同様の操作を行う（図3）．2-opt法と比べ所要時間は増すが精度は高くなる．具体的なアルゴリズムは文献[1]に記載されている．

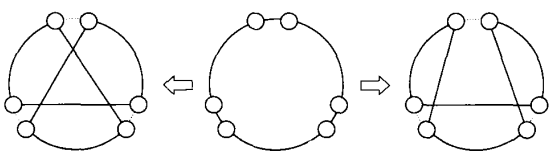

図3　3-opt法の枝交換

交換する枝をさらに増やせば精度は高くなるが，所要時間も急速に大きくなるので実用には耐えない．所要時間を抑えつつ精度を上げる方法としてLin-Kernighan法（以下LK法）がある[2][4]．ここで，2-opt法の基本操作を「最初に除く枝d，次に加える枝a，次に除く枝b，最後に加える枝x」と分解し，最後に加えたxを後続操作の最初に除く枝dとして「2-opt基本操作」の連鎖を考える．x_kまでの連鎖で$k+1$本の枝交換を実現できる．

$d_1, a_1, b_1, x_1=d_2, a_2, b_2, x_2=d_3, a_3, b_3, x_3=d_4 \cdots$

LK法はこの連鎖系列を可変深度探索する．すなわち，

- 連鎖系列の中で選択の余地がある$d_1, a_1, a_2, \cdots$について，系列長（探索深度）を「良い解の可能性」を指標として制御しながら，バックトラック探索する．
- k回目に加える枝a_kはd_kの両端点から出る枝の中で，短いほうから*width*本目までを候補とする．*width*はk（レベル）によって異なる．
- あるレベルから戻るとき，改善系列（x_kで止めたとき改善量が正のもの）がすでに見つかっていれば，当該系列の操作を施したものを次の対象巡回路として第1レベルから探索を再開する．
- 第1レベルから戻るとき，LK法の探索は終了するが，得られている巡回路にさらに2-opt法を施して最終的な巡回路とする．これは，試みる枝を*width*本に限定しているので2-optによる改善機会を見逃している可能性があるからである．

上記は文献[4]で提案されたmodified Lin-Kernighan (mLK)法の概略である．LK法（mLK法）は高速で精度が高い．

4.2　$\{\boldsymbol{S}_k\}_{k=1, 2, \ldots, m}$の計算

可能な組分け全体の集合$\mathbf{S}=\{(2), (3), (4)$を満たす$\boldsymbol{S}=(S_1, S_2, \cdots, S_m)\}$を探索して，目的関数(1)をできるだけ小さくする$\boldsymbol{S}^*$を見いだす．様々な探索方法が考えられるが，最も基本的なのはS_iの1つの要素をS_jに移す「1移動近傍」を用いた局所探索である．すなわち，適当な（ランダムあるいは構築法で作成した）初期組分け$\boldsymbol{S}_0$から出発し，近傍内の目的関数を減少させる組分けに$\boldsymbol{S}$（現在解）を更新していく．このとき，グループ間$(S_{i1}, S_{i2}, \cdots, S_{ic})$での循環的要素移動が目的関数の改善をもたらす場合もある．このような移動はc回の1移動操作($S_{i1} \rightarrow S_{i2}$; $S_{i2} \rightarrow S_{i3}$; …; $S_{ic-1} \rightarrow S_{ic}$; $S_{ic} \rightarrow S_{i1}$)で代替できるが，途中で目

的関数値が（いったん）増加すると循環は実現されない．そこで，改悪であっても（一定期間）解の更新を続け，何回か更新した後での改善を期待する．この方法は「タブー探索」と呼ばれている．「タブー」という名称は，「最近移動させた要素の逆方向の移動を禁ずる」仕組みに由来する[1]．タブー探索はVRPに対して効果的であるが，パラメータの設定等に若干の調整が必要である．次節の「例題」においては，メタ戦略（タブー探索のような探索能力強化の仕組み）を用いない近傍探索だけの単純解法を紹介する．

5. 例題—ポストからの郵便物収集—

本節では，郵便局（集配局）がその管理下のポストから郵便物の収集を行うとき，「何人の作業員で，どのように分担し，どのように巡回するか」という問題をVRPの例題として取り上げる．対象とするのは，都下S区のC郵便局と96個のポストで，その位置関係は図4に示す通りである[2]．

5.1 問題設定

この問題を通常のVRPのように「総移動距離を最小化する」という目標で扱うと各職員の稼働時間に長短の差が生ずる．そこで，「各職員の稼動時間をできるだけ均等に保ちながら短くする」ことを考える．前提を含めて問題を整理するとつぎのようになる．

- 職員は郵便局を出発し，分担して全ポスト（$V=\{1,2,\cdots,n\}$）から郵便物を集め，郵便局に戻る．
- 各点間の移動費用をその所用時間とする（職員の移動速度はすべて同一で一定）．
- 郵便物の容量・重量は考えない（1人ですべてを運ぶことも可能）．
- 1つのポストから郵便物を回収するのに要する時間（d）はすべて等しい．
- 1日毎の各職員の稼動時間を均等化する．
- 職員数 m をパラメータとし，m を変えて解く．

図4 C郵便局と96個のポスト

稼働時間の均等化を「最も時間のかかる職員の稼働時間最小化」と考え，min-max型の集合分割問題として以下のように定式化する．

$$\text{minimize} \quad \max_{1\le k\le m}\{f(\{0\}\cup S_k)+d\cdot|S_k|\} \tag{5}$$

$$\text{sub. to} \quad S_i\subset V,\ S_i\cap S_j=\phi,\ \bigcup S_i=V \tag{6}$$

ポスト間の移動費用を一律に d（郵便局-ポスト間は $d/2$）だけ加算しておけば，目的関数の第2項は不要である．

5.2 順列空間を探索する解法

ポスト番号 $\{1,2,\cdots,n(=96)\}$ の順列 σ に m 分割 $\boldsymbol{S}(\sigma)\in\mathbf{S}_m$ を対応させ，それが与える目的関数(5)の値を σ の評価値として，$\varphi(\sigma)$ と書く．ポストの m 分割空間 $\mathbf{S}_m$ を直接探索する代わりに，できるだけ小さな評価値 $\varphi(\sigma)$ を与える σ を順列全体の集合 $\mathbf{P}_n$ の中で探し，間接的に良い m 分割を求める[3]．$\varphi(\sigma)$ は，例えば，以下のA，Bのような手順で近似的に計算する．

方法A　σ に $m-1$ 個の「区切り」を入れ，各部分を S_k とみなす．

$$\sigma(1)\cdots|\sigma(t_2)\cdots\cdots|\sigma(t_k)\cdots\cdots|\sigma(t_m)\cdots\sigma(n)$$
$$t_1=1 \quad t_2 \qquad t_k \qquad t_m$$

各グループの先頭番号 $\boldsymbol{t}=(t_1(=1),t_2,\cdots,t_m)$ と順列 σ で決まる m 分割を $(S_1(\sigma,\boldsymbol{t}),S_2(\sigma,\boldsymbol{t}),\cdots,S_m(\sigma,\boldsymbol{t}))$ とする．そして，σ を固定し $\boldsymbol{t}$ を変化させたときの目的関数の最小値を $\varphi_A(\sigma)$ とする．すなわち，

$$\varphi_A(\sigma)=\min_{\boldsymbol{t}}\ \max_{1\le k\le m}\{f(\{0\}\cup S_k(\sigma,0))\}.$$

方法B　最初に分割の各組を空集合とし，$\sigma(1),\sigma(2),\cdots,\sigma(n)$ の順に1点ずつ，m 個の組のいずれかに，それらのうちで最大となるものの費用を最小に抑えるように配分していく．すべてを配分したときに得られる m 分割が $\boldsymbol{S}(\sigma)$ であり，それが与える目的関数値を $\varphi_B(\sigma)$ とする．

方法AまたはBにより，問題(5)-(6)は $\varphi_{A/B}(\sigma)$ の最小化問題に帰着する．この最小化は，σ の任意の2要素の交換から成る互換近傍系 $W(\sigma)$ を用いた $\mathbf{P}_n$ の局所探索で行う．

[1] こうしないと，解を改悪更新した後，次の更新で元の解へ戻り（これは改善），探索が堂々巡りをしてしまう可能性が高い．

[2] 平成19年のデータ[7]．

[3] 必ずしも能率的な方法ではないが，単純さと汎用性を考慮して採用する．

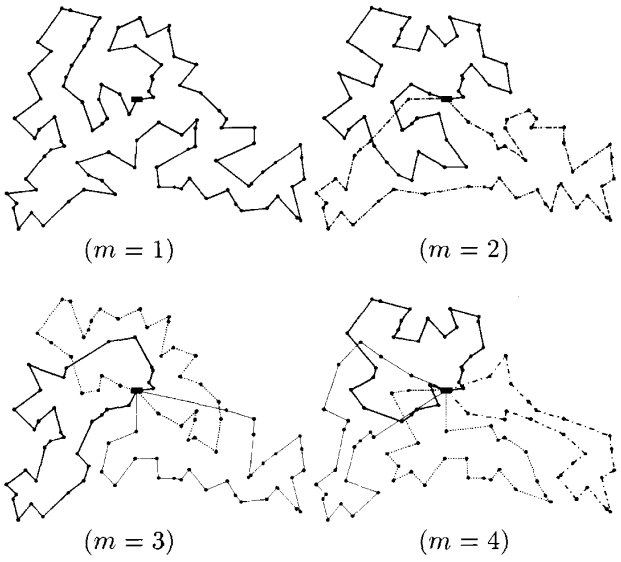

図5　m 人の巡回路

step 0：$\sigma \leftarrow \sigma_0$；(ランダムな初期順列)

　$UB \leftarrow +\infty$

step 1：$\tau^* \leftarrow \arg \min_{\tau \in W(\sigma)} \varphi(\tau)$；

step 2：**if** $\varphi(\tau^*) < UB$ **then** {$\sigma^* \leftarrow \tau^*$；

　$UB \leftarrow \varphi(\tau^*)$；$\sigma \leftarrow \tau^*$；**goto** step 1}

step 3：UB (値)，$\boldsymbol{S}(\sigma)$ (解)を出力，終了．

UB は暫定値/上界値であり，それまでに探索した実行可能解が与える最小の目的関数値である．

5.3　計算結果

以上の方法で，問題(5)-(6)を $m=1, 2, \cdots$ と変化させながら解いた結果を図5に示す．f の計算には手軽さと精度を考慮して3-opt法を使用している．$\varphi(\tau)$ の計算はA，Bの両方法で行ってみた．3以上の m についてはBの方が良い結果を与える．図の $m=2$ はA，$m=3, 4$ はBを用いた結果である．計算時間は，Core 2 Duo 2.0 GHz のノートPCで15分 ($m=2$)，90分 ($m=3$)，30分 ($m=4$) 程度である[4]．$m=1$ はTSPであり一瞬で解ける．

m が4程度までは自然な形で地域的に分割されていくが，5以上になると地域的な重なりが大きくなる．

6.　まとめ

本稿では，収集/配送経路問題 (VRP) の基本モデルを紹介し，それが「分担 (組分け・分割)」と「巡回」の2つの問題を含むことを示した．

VRPは「巡回」の問題を含むので汎用のソルバーでは対処できず，自分で解法を用意しなくてはならないことが多い．そのようなときに使える道具として，「巡回」に対する2,3-opt法，mLK法，「分担」に対する局所探索法を紹介した．これらは手軽な発見的解法であるが，上手に使えば，実用に耐える結果を導くこともできる．

これらの手法の適用例として，「ポストからの郵便物収集」取り上げ，「問題設定，解法準備，求解」の過程を説明した．求解においては，問題に即した若干の工夫と簡単なプログラミングが必要とされる．

大規模な問題を扱うときは，既存のライブラリ (専用ソルバー) を利用することになるかもしれないが，手ごろな問題で一連の作業に触れておくことは良い経験となる．本稿が，「収集/配送経路問題」に初めて取り組む際の出発点として，少しでも役に立てば幸いである．

参考文献

[1] S. Lin: Computer solutions of the traveling-salesman problem, *BSTJ*, Vol. 44, 2245-2269 (1965).

[2] S. Lin and B. W. Kernighan: An effective heuristic algorithm for the traveling-salesman problem, *Operations Research*, Vol. 21, 498-516 (1973).

[3] E. L. Lawler et al.: *The Traveling Salesman Problem* (chap. 12), JOHN WILEY & SONS (1985).

[4] K.-T. Mak and A. J. Morton: A modified Lin-Kernighan traveling-salesman heuristic, *Operations Research Letters*, Vol. 13, 127-132 (1993).

[5] 柳浦，茨木：組合せ最適化—メタ戦略を中心として—，朝倉書店 (2001).

[6] 久保：ロジスティクス工学，朝倉書店 (2001).

[7] 鈴木：郵便物収集を行う人員数と作業時間に関する研究，平成19年度東京理科大学工学部経営工学科卒業論文 (2008).

[4] WindowsXP, Delphi 6で作成，実行．