巡回セールスマン問題への招待III

久保　幹雄

...tabu search. Simulated Annealing and the genetic algorithm, schemes for controlled randomization for integer programs, may have significant potential.

CONDOR

(Operations Research: The Next Decade, Operations Research, 1988)

Annealing is a potentially valuable tool but in no ways panacea.

David Johnson

(Graph Coloring by Simulated Annealing, Operations Research, 1991)

9. ローカルサーチの周辺

前回作った2-opt法は、高速ではあるが、得られた解は十分に良いとは言えない。一方、最近の計算機の高速化、廉価化に伴い、長い間ジョブを実行させておいても、それほどクレームがつかなくなった。そのため、さらに時間をかけても、より良い解が得たいという要求が生まれてくるのは、自然な欲求であろう。以下では、そのような方法を考える。

最も簡単に実現できる方法は、初期解を変えて2-opt法を何回も(時間の許す限り)繰り返す方法である。この簡単な方法でさえ、あえて名前はあげないが、最近マスコミを騒がしている幾つかの新(?)解法よりも良い解を算出する。(どうしても名前を知りたい人は、巡回セールスマン問題への招待I, 文献[2]参照。)

良い解を探すためのもう1つの自然な方法は、近傍を大きくとることである。これは、暗闇の中の登山者に、より強力な懐中電灯を持たせることに相当する。1度にk本の枝を入れ替えることによる近傍を用いたローカルサーチをk-opt法と呼ぶ。前回作成した2-opt法のような計算上の工夫をした3-opt法は高速でかつ比較的良い解を算出する。計算時間は、都市数$n=1$万程度までは、ほとんど線形オーダーを少し上回る速さで増加する。実を言うと、任意のnに対して「ほぼ線形オーダー」というのは嘘である。John Bentleyは、profilerを用いた詳細な数値実験によってk-optには`Flip ()`を行うときに発生する$\Omega(n^{1.74})$の項があることを突き止めた。しかし、この項の係数は非常に小さいので、nが10万を超えないと効いてこない。超大規模問題に対しては、巡回路を表すデータ構造に、さらに工夫を凝らす必要が出てくるが、通常の範囲の問題なら前回用いたデータ構造で十分高速である。さて、さらなる改良のために、単に$k \geq 4$の

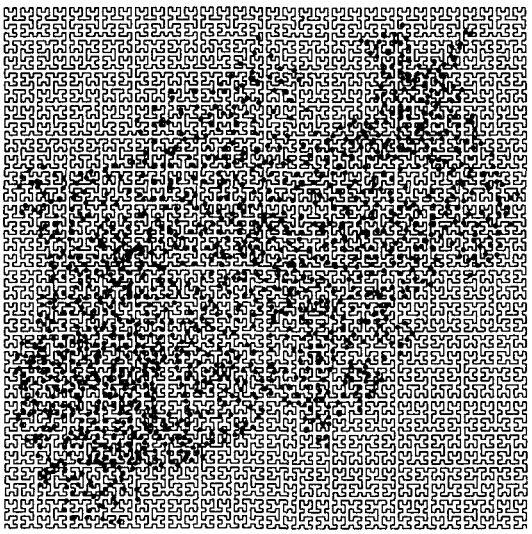

図7: 巡回セールスマン問題ギャラリー5: 1379都市問題とHilbert Curve. 左下から始まり右下に至る空間充填曲線の順に都市を辿ると, 空間充填曲線法による近似解が得られる.

くぼ　みきお　東京商船大学　流通情報工学
〒135 東京都江東区越中島 2-1-6
e-mail: kubo@ship2.ipc.tosho-u.ac.jp

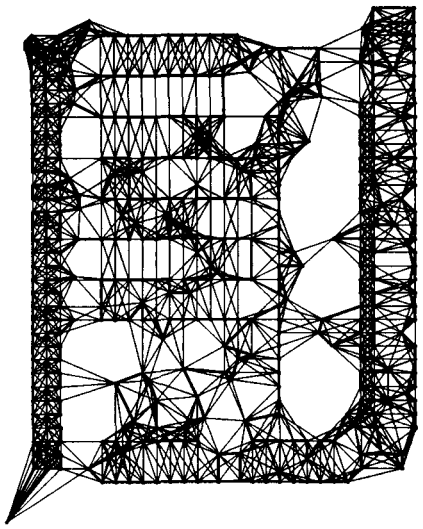

図 8: 巡回セールスマン問題ギャラリー 6: 442 都市問題の 10 個の近傍 (細線) と最適巡回路 (太線).

k-opt 法を適用するのでは、あまり芸がない。実際、単純な 4-opt 法では計算時間の増加に見合うほど良い解は得られない。Lin と Kernighan (§ 4 参照) は、kを可変にして、改良度が正のうちは、どんどんkを増やしていくという、賢い探索法を開発した。もちろん、計算時間に余裕を持っている場合は、初期解を変えて 3-opt 法や Lin と Kernighan の解法を繰り返し行うことによって、さらに良い解を得ることができる。

以上が、計算時間に余裕がある場合の古典的アプローチであるが、最近では、モダン・ヒューリスティックスまたはメタ戦略と呼ばれる新しいパラダイムが幾つか提案されている。この範疇に含まれる解法としては、(Artificial) Neural Net 法、Genetic Algorithm、Evolution Algorithm、Tabu Search、Simulated Annealing 法などがある。

私は、これらのパラダイムに対して、2 つの危惧を持っている。1 つは、流行ものとしての上滑りな研究と過大評価が氾濫することである。名前が sexy なことからマスコミの一時的な注目を浴び、すぐに消え去ってしまった理論がたくさんある。また、過度の期待は、その反動で全く見向きもされない理論になってしまう危険性をはらんでいる。もう 1 つの危惧は、これらの新しいパラダイムを全く無視することである。しばしば、新しい理論の創世期には、不完全で質の悪い論文が多く出回りがちであるが、それらの論文に対する評価だけで、むざむざ便利な道具 (になる可能性のあるもの) を全く眼中から外してしまうことは避けねばならない。我々が成すべきことは、実務家がこれらのパラダイムをいつでも道具箱から取り出して使えるように、どのパラダイムがどういう問題に対して有効か (また、有効でないか) を詳細に検討し、OR の 1 つの道具として洗練させ、準備しておくことである。

以下では、モダン・ヒューリスティックスに対しての正しい認識を与えるために Tabu Search と Simulated Annealing 法を紹介し、どのようにコード化するのか実演する。アルゴリズムを実現するには、前回示した 2-opt 法と同じようにデータ構造や細部の設計が必要になる。ここで、Tabu Search と Simulated Annealing 法を選んだ理由は、解法が明確に定義でき、また 2-opt 法と同様に比較的簡単に実現できるためである。

10. Tabu Search を作ろう!

前回、ローカルサーチの説明に用いた闇夜の登山者の例を使って説明する。ローカルサーチにおける登山者の目的は、なるべく高い地点で眠ることであった。そのため、回りを懐中電灯で照らしながら高い地点へ移動していく戦略を用いた。今度は、登山者のゲームのルールを少し変えて、「なるべく高い地点を通過すること」とする。登山者は、夜が空けるまで歩き回り、途中通過した最も高い地点の標高を得点として得る。このルールの変更は登山者から見ると理不尽であるが、計算機を用いて最良解を探索するときには自然である (計算機は、途中での最良解を保存することができる!)。

このようにルールを変更すると、回りに標高の高い地点がなくなったときでも、登山者は休まずに歩き続けるであろう。妥当な戦略として、なるべく下り坂の勾配が小さい方向へ進むことが考えられる。賢い登山者ならば、懐中電灯で照らせる範囲の風景を記憶しておき、前に通過した地点には、なるべく戻らないようにするであろう。このように、記憶を頼りに、常に標高の最も高い方向を目指すのが、Fred Glover によって提案された Tabu (Taboo) Search と呼ばれる解法である [1, 2]。Tabu (または、Taboo とも綴る) とは、「禁断」を意味し、前に通過した地点に戻ることを避けるための記憶を「禁断リスト (Tabu List)」と呼ばれるリストに保持しておくことから、このような怪しい名前

がつけられた。

ちなみに、Gloverとは独立にPierre Hansenによってほぼ同じ解法が提案されている[3]。Hansenは彼の解法をSteepest Ascent Mildest Descent法と呼んでいる。個人的には、この名前の方がTabu Searchよりもアルゴリズムの内容を良く表していると思うが、ここではTabu Searchと呼ぶことにする。なお、LinとKernighanを含めたローカルサーチの古典的な研究の多くも、Tabu Searchと同様のアイディア(1度探索した解は探索から外すこと)を使っていることは注目に値する。

ローカルサーチの場合と同じように、Tabu Searchの疑似コードを示そう。Tabu Searchでは、近傍から禁断リストΞを除いた中から、最も標高$c(x)$の高い地点へ移動する。移動を行うための関数$move(x)$を以下のように定義する。

$$move(x) = \begin{cases} x' & \text{if } c(x') \geq c(y) \text{ for all } y \in N(x) \setminus \Xi \\ \emptyset & \text{if } N(x) \setminus \Xi = \emptyset \end{cases}$$

この関数を用いることによってTabu Searchの概要は以下のように書くことができる。なお、TL (Tabu Lengthを表す)は禁断リストΞの長さである。

```
procedure tabu search
 1 t := 0
 2 x_0 := some initial feasible solution
 3 Ξ := ∅
 4 TL := a positive integer
 5 while stopping-criterion ≠ yes do
 6     x_{t+1} := move(x_t)
 7     Ξ := Ξ ∪ {x_t} \ {x_{t-TL}}
 8     t := t + 1
 9 return x
```

さて、禁断リストには何を入れたら良いのであろうか? 最も単純な考えは、解そのものを保持しておくことであるが、大体の場合は、この方法ではうまくいかない。筆者のささやかな経験から、以下の指針を提示しておく。「問題の構造を決める小さい集合をとりなさい!」

この指針にしたがえば、巡回セールスマン問題に対しては、点集合または枝集合が候補であるが、ここでは点集合を用いた以下のような簡単な方法を採用する。2-opt法において枝ab, cdを除いてac, bdを加えたとき(すなわち関数`Flip(a,b,c,d)`を呼んだとき)、点aは、しばらくの間は改良を行う枝の始点としては使わない。

この操作をプログラム化するための最も簡単な方法は、点集合に対応する1次元配列`LS[]`†を用いることである。まず、`LS[]`には0を入れておき、`Flip(a,b,c,d)`を呼んだとき、`LS[a]`に正の数`TL`を入れる。毎イテレーションごとに、正の`LS`を持つ点は、その値を1ずつ減らしていき、`LS[a]`=0になったら、点aを再び使うことができるものとする(9行)。Tabu Searchは、2-opt法と同様に任意の巡回路`tour[]`およびその長さ`length`を入力として、以下のように書ける(データ構造については§8参照)。

```
 1 void Tabu_Search()
 2 { int i, j, a, b, c, d, tmp;
 3   int iter, delta, astar, bstar, cstar, dstar;
 4   unsigned int LS[n];
 5   for(iter=0; iter<ITER_MAX; iter++){
 6     delta=−9999;
 7     for(i=0; i<n; i++){
 8       a = tour[i];
 9       if(LS[a]){ LS[a]−−; continue; }
10       b = Next(a);
11       for (j = head[a]; j < head[a+1]; j++){
12         c    = adja[j];
13         if(LS[c] | b==c) continue;
14         d    = Next(c);
15         if(b==d | a==d) continue;
16         tmp=Dis(a,b)+Dis(c,d)−Dis(a,c)−Dis(b,d);
17         if (tmp > delta){
18             delta=tmp; astar=a; bstar=b;
19             cstar=c; dstar=d;
20         }
21       }
22     }
23     length −= delta;
24     Flip (astar, bstar, cstar, dstar);
25     LS[astar] = TL;
26   }
27 }
```

もちろん、アルゴリズムの途中での最良解を記憶しておき、更新の度に巡回路を保存する必要があるが、ここでは省略している。上で作成したTabu Searchは、2つのパラメータ`ITER_MAX`および`TL`を含んでいるが、これらに適正値を与える仕事が残っている。

20世紀における最も尊敬すべき数学の教師であるGeorge Pólyaは、定理の証明において何の動機もなし

†`LS`は, Life Span (寿命)を意味する. そのため, このTabu Searchの変種をLSM (Life Span Methodまたは Local Searchもどき)と命名した. 「Local Searchもどき」は, Tabu Serchと同じ発想が、古典的なLocal Searchの文献にすでに含まれていたことを示す.

図 9: 巡回セールスマン問題ギャラリー 7: 4461 都市問題の近似解: この解は Nearest Neighbor 法 + 2-opt 法によって求められた.

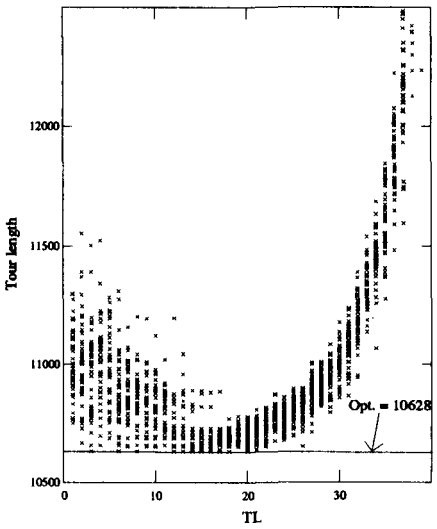

図 10: パラメータ TL による巡回路長の変化

に導入された道具を Deus ex machina (天から降ってきた機械神) と呼び、それを使うことを避けるように説いている。ヒューリスティック解法の開発においても Deus ex machina を禁じるべきであると考えられる。モダン・ヒューリスティック解法の開発においては、色々な工夫 (と同時にパラメータが必要になる) を次々と追加する傾向があるが、なぜその工夫を導入する必要があるのか、また、パラメータ設定はどのような根拠でしたのかを、明確な動機付けを伴って述べる必要がある。

上で作った Tabu Search を例にして、どのようにして Deus ex machina を天に戻すかを説明する。まずは、簡単な方から片づけよう。ITER_MAX は、イテレーション数の上限であり、どの程度の計算時間で終了させたいのかに依存して決めれば良い。ここでは、アルゴリズムを $O(n^2)$ 時間で終了させるために、ITER_MAX を $O(n)$ (例えば $100n$) と設定しておく (上の Tabu Search の 1 イテレーションは、点に接続する枝の本数の上限を定数に抑えている (§ 8 参照) ので、$O(n)$ の時間を要することに注意)。

問題になるのは、Tabu Search の最重要パラメータ TL の決定である。何の実験的裏付けもなしで「7 が幸運の数であるので TL に 7 を用いる」などという暴挙は間違ってもしてはならない。パラメータ適正化は、通常実験的解析によって行なわれる。ここでは、アメリカ合衆国 48 都市問題 (巡回セールスマン問題への招待 I: 図 1 参照) を対象として、TL の適正値を見つける。初期解をランダムな順列としたときの実験の結果を図 10 にあげる。実験結果からパラメータ TL の適正値は 20 付近であることが分かる。もちろん、他の巡回セールスマン問題の例題に対して同様の実験を行うことによって、より一般的かつ深い知見が得られるが、ここでは省略する。

11. Annealing を作ろう!

Tabu Search よりも多少長い歴史を持ち、かつポピュラーなモダン・ヒューリスティックとして Simulated Annealing 法 [6, 9] がある。この解法は S. Kirkpatrick, C. Gelatt, and M. Vecchi と V. Černý が独立に提案したものであると言われているが、その土台は、1953 年に N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller [8] によって築かれていた。Simulated Annealing 法の別名を紹介すると Monte Carlo Annealing, Probabilistic Hill Climbing, Statistical Cooling, Stochastic Relaxation, Metropolis Algorithm などがある。個人的には、確率的丘登り (Probabilistic Hill Climbing) が素直で良いと思うが、ここでは、ポピュラーな Simulated Annealing 法の名前を使うことにする。

再び登山者の例で説明しよう。登山者がSimulated Annealing法を使うには、寒暖計と乱数表を常備している必要がある。まず、懐中電灯で回りを適当に照らし、照らした場所の標高と現在地点の標高の差Δを測定する。照らした地点の標高が現在地点よりも低くなければ(すなわち$\Delta \geq 0$ならば)、ローカルサーチの場合と同様に、その場所に移動する。もし、$\Delta < 0$のときは、寒暖計によって現在の気温Tを測定する。また、乱数表により$[0,1]$の一様乱数を得て、その値が$e^{\Delta/T}$より小さかったら、やはり移動する。大きかったら、その場にとどまり、再び適当な場所を懐中電灯で照らす。山の気温は明け方が最も冷えて、夜が空けるまでにはTが徐々に下がっていき0度になるとすると、登山者はローカルサーチの場合と同様に小高い丘の上につくことになる。

Simulated Annealing法における移動を制御するための関数は以下のように定義される。

$$P(x) = \begin{cases} x' \in N(x) & \text{if } c(x') \geq c(x) \\ x' \in N(x) & \text{if } c(x') < c(x), \text{ w.p. } e^{\frac{c(x')-c(x)}{T}} \\ x & \text{otherwise} \end{cases}$$

上の定義を与えられると、「なぜ確率$e^{\frac{c(x')-c(x)}{T}}$で悪い方向への移動を許可するのか? 他の関数ではいけないのか?」といった自然な疑問が生じてくる。この定義はDeus ex machinaであり、私には手品のように見える! 実は、指数関数を用いるのは絶対条件ではなく、理論的な収束を示すときに扱い易いから使っているに過ぎず、ある性質を満たす関数なら、何でも漸近的に最適解に収束することが示されている。実際には、指数関数のような計算機にとっては高価な関数を使わずに、その近似で十分である。乱暴なようだが、指数関数の1次近似$1 + \frac{c(x')-c(x)}{T}$でも、得られる解の良さにはほとんど影響がないことが実験によって確認されている。しかし、安全のためには分布関数の近似法(例えばRectangle-Wedge-Tail法: [7, § 3.4]参照)を使う方法が推奨できる。

Simulated Annealing法において最も重要なことは、温度の設定である。登山者は自分では温度の制御ができないが、計算機上では温度を自由に設定できる。理論的に優れた性質を持つ冷却方法の1つとして対数冷却(logarithmic cooling)と呼ばれる方法がある。この方法では、kイテレーションでの温度は、ある定数Cに対して$C/\log k$と設定される。Simulated Annealing法の初期の売り文句は、十分に長い対数冷却を行えば最適解を確率1で得ることができるということであった。しかし、最適解を確率1で得るために必要な理論的な計算時間は、(なんと!) $O(n^{n^{2n-1}})$であり、これは、全列挙による解法$O((n-1)!)$や動的計画法による解法$O(n^2 2^n)$よりはるかに大きな数である。このような理論的結果は、実際にこの解法の適用を考えている実務家にとっては、何の役にもたたない。

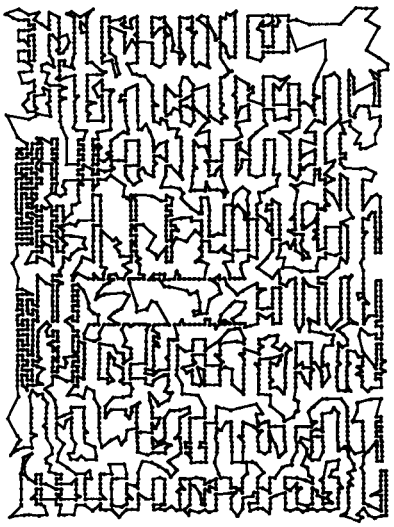

図11: 巡回セールスマン問題ギャラリー8: Tabu Searchで得られた3038都市問題の近似解.

Simulated Annealing法の本当の性能は、上のような理論的な結果ではなく、実験的解析によるものでなければならない。AT & TのDavid Johnsonらのグループは、グラフ分割問題、グラフ彩色問題、数分割問題を例としてSimulated Annealing法の実験的解析を行った[4, 5](巡回セールスマン問題に対しては、現在執筆中だそうである)。彼らの結果は決して成功例ばかりではなく、中にはSimulated Annealing法が、従来の方法に全く歯が立たなかった問題もあると報告している。それにも関わらず、このプロジェクトは実験的解析の模範として、非常に高い評価を受けている。実験結果を要約したのが冒頭のJohnsonの言葉であるが、同じことがSimulated Annealing法以外のモダン・ヒューリスティック解法に対しても言える。すなわち、ローカルサーチの変形であるこれらの解法は、比較的簡単にコード化できるので便利ではあるが、決して万能薬ではない。

Johnsonらのグループの実験でも推奨されている

実用的な温度の制御法として幾何学的冷却(geometric cooling)と呼ばれる方法がある。この方法においては、同一温度で均衡に達するまで移動を繰り返し、その後で温度を定数(例えば0.95)倍することによって温度を下げる。

この冷却方法を採用したSimulated Annealing法の概要は、前に定義した関数$P(x)$を用いて以下のように書くことができる。

```
procedure simulated annealing
 1 x := some initial solution
 2 T := T_START
 3 while T ≥ T_END do
 4     while equilibrium-criterion ≠ yes do
 5         x := P(x)
 6     T := T × T_FACTOR
 7 return x
```

ここで導入したパラメータは、Simulated Annealing法を作るための最低限のパラメータ(初期温度T_START、最終温度T_END、温度を下げる比率T_FACTOR)である(パラメータの数は、むやみに増やさないように注意する!)。

効率の良いアルゴリズムを実現するためには、近傍に対しても多少の工夫がいる。Simulated Anneaing法の基本は、近傍の中からランダムに要素を選択することである。この部分の実現方法として、まず最初に思いつくのは、ランダムに2つの点a,cを発生させてから、aの次の点b、cの次の点dを求め、枝ab,cdを除き、枝ac,bdを加えたときの巡回路長を評価する方法である。この方法は、枝acが非常に長く、改良の見込みがない場合でも正直にチェックを行うので、明らかに非効率的である。

2-opt法で用いたのと同じように、不必要に長い枝は考慮から外すことが効率的なSimulated Annealing法を実現させるための鍵である。ここでは、点をランダムに発生させるのでなく、改良の始点aおよびaに隣接する点を順番に探索していきながら、確率的に移動を行う方法を採用する。これは、登山者の例では、懐中電灯を適当な方向に向けるのではなく、決められた方向から時計回りに照らしていくことに相当する。このとき、点の添字はエンドレスに0から$n-1$を繰り返すことになる。そのための関数を記述しておこう。

```
1 int Increment( int i )
2 {   if(i==n-1) return   0;
3       else    return ++i;
4 }
```

前に示したSimulated Annealing法の疑似コードでは、同一温度での均衡の判定条件を正確に記述していなかった。ここでは、近傍全体を何回か網羅したら温度を下げる簡便法をとることにしよう。そのためのパラメータとして、SIZEを導入する。改良の始点になる点を変えていき、近傍をSIZE回巡回したら温度を変えることにする。

上で述べた要素を入れることによって、Simulated Annealing法のプログラムは以下のように書くことができる(14行目の`random()`は$[0,1)$の一様乱数を与える関数である)。

```
 1 void Simulated_Annealing()
 2 { int i=0, j, a, b, c, d, tmp;
 3   float T;
 4   for(T=T_START; T≥T_END; T*=T_FACTOR){
 5     for (count=0; count++; count < SIZE * n){
 6       a = tour[i];
 7       b = Next(a);
 8       for (j = head[a]; j < head[a+1]; j++){
 9         c = adja[j];
10         if(b==c) continue;
11         d = Next(c);
12         if(b==d | a==d) continue;
13         tmp = Dis(a,b)+Dis(c,d)-Dis(a,c)-Dis(b,d);
14         if ( tmp ≥ 0 | random() < 1+ tmp/T ){
15                 length -= tmp;
16                 Flip (a, b, c, d);
17         }
18       }
19       i = Increment(i);
20     }
21 }
22 }
```

残念ながら、ここではSimulated Annealing法のパラメータの最適化やTabu SearchとSimulated Annealing法の性能評価について書くスペースはない。比較を行うためには種々の工夫の導入によるアルゴリズムのチューンアップが必要であり、当然LinとKernighanの解法や3-opt法との比較も必要である。これらの課題については熱心な読者に託すことにしよう。

12. おわりに

まだまだ書き足らないことがたくさんある。多面体論やそれを用いた切除平面法による最適化手法、PとNPの狭間をしらべるWell Solved Special Case、より複雑な現実問題(例えば、Vehicle Routing Problem)への拡張、並列計算と分枝限定法による最適化手法など、枚挙に暇がない。残念ながら、今回の連載では、これらの話題を含めることはできなかった。より詳しい情報に関しては、私が書いたサーベイまたは巡回セールスマン問題のみを扱った大著(巡回セールスマン問題への招待I, 文献[3,4])を参照されたい。

なお、本連載で用いたプログラム(をきちんと動くように宣言、入力を付加したもの)は、私に電子メールを送っていただければ、研究用としてなら喜んで提供するつもりである(フロッピーディスクによる配布はご勘弁願いたい)。また、筆者らの研究グループでは、グラフ分割問題、グラフ彩色問題、2次割当問題、ジョブショップスケジューリング問題、最大安定集合問題(最大クリーク問題)、N-Queen問題に対しても、効率的なプログラムおよびBenchmark問題を持っている。どの分野に興味を持っていて、どのような研究を行うかを、電子メールで知らせていただければ、巡回セールスマン問題と同様に喜んで提供するつもりである。

我々の目的は単に目新しい方法を開発することではなく、真に有効な方法を探索することである。そのためには、歴史(§4参照)が教えてくれるように、古くからある解法を見直し、さらに新しいアイディアを入れていくことが重要である。また、開発された解法に対して、正確かつ公正な評価を行うことは、組合せ最適化問題を実務における応用に使うときの重要な指針となり、日本国内におけるORのさらなる普及のためにも重要である。この連載を機に、一人でも多くの読者がこの分野に興味を持ってくれて、一緒に研究や討論を行う仲間になってくれることを切に願っている。

謝辞

最新の情報を教えて下さったBellcoreのWilliam Cook博士、たくさんのコメントを頂いた(筑波大学の福田公明先生をはじめとする)茗荷谷クラブの皆さん、実験を手伝って下さった早稲田大学の森戸研究室の皆さん(そして、もちろん森戸晋教授!)、このような我侭な連載を許して下さった東京工業大学の森雅夫教授をはじめとする編集委員の皆さんに心から感謝の意を表します。

参考文献

[1] F. Glover. Tabu search I. *ORSA Journal on Computing*, 1:190–206, 1989.

[2] F. Glover. Tabu search II. *ORSA Journal on Computing*, 2:4–32, 1989.

[3] P. Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. In *The Congress on Numerical Methods in combinatorial Optimization*, Capri, March 1986.

[4] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, part I, graph partitioning. *Operations Research*, 37:865–892, 1989.

[5] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, part II, graph coloring and number partitioning. *Operations Research*, 39:378–406, 1991.

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[7] D. E. Knuth. *The Art of Computer Programming, volume 2: Seminumerical Algorithms*. Addison-Wesley, 1969; second edition, 1981.

[8] N. Metropolis, A. Rosenblut, M. Rosenblut, A. Teller, and E. Teller. Equation of state caluculation by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[9] V. Černý. Themodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimmization and Applications*, 45:41–51, 1985.