Res. Rep. Fac. Eng. Mie Univ., Vol. 25, pp. 81−96 (2000)

論　文

巡回セールスマン問題の近似アルゴリズムについて

坂上知英, 吉澤慎*, 太田義勝, 大山口通夫

Tomohide SAKAGAMI, Makoto YOSHIZAWA, Yoshikatsu OHTA and Michio OYAMAGUCHI

(情報工学科 Department of Information Engineering)

(Received September 18, 2000)

Abstract

The Traveling Salesman Problem (TSP) is the task of finding a route through a given set of cities with shortest possible length. Many practical applications(VLSI design, etc.) can be modeled as a TSP. But, TSP is NP-hard, so the efficient approximation algorithms have been studied so far. In this paper, we show new approximation algorithms for TSP and the experimental results for these algorithms.

Keywords: Traveling Salesman Problem, Approximation Algorithm

はじめに

巡回セールスマン問題 (Traveling Salesman Problem:以下 TSP と略す) とは、複数個の都市を一度づつ巡回するとき、巡回の総コストが最小であるものを見つける問題である [1,2]。TSP が効率的に解けると、経路計画やスケジューリングなど、さまざまな分野での作業の効率化や費用の削減に貢献するため、世界中の科学者に研究されてきた。

しかし、TSP は多項式時間では最適解を得ることができないだろうと予想されている問題であり、効率的に最適解を求める手法は存在しないだろうと考えられている。そこで、最適でなくとも、ある程度の精度を持った近似解を短時間に求めることが必要かつ重要とされ、様々な TSP に対する近似アルゴリズムが研究されてきた。TSP 近似アルゴリズムは初期解をまず求めて（構築法），それを改善していく（改善法）という２つのフェーズから構成される．

本稿では、従来の TSP に対する近似アルゴリズムで用いられている構築法，改善法を改良した新しい方法をいくつか提案する．また，従来の TSP に対する近似アルゴリズムと本研究の方法との比較を実験的に行なった結果を示す。

次に本論文の構成を示す。まず第１章では、TSP について簡単に述べる。第２章では、従来の TSP に対する近似アルゴリズムについて述べる。第３章では、本研究の TSP 近似アルゴリズムである構築法と改善

*現在，三重電子計算センター

法について述べる。第4章では、従来の近似アルゴリズムと考案した近似アルゴリズムの、計算時間、精度の比較結果を示す。

1 巡回セールスマン問題

本節では巡回セールスマン問題 (Traveling Salesman Problem:TSP) について述べる。

1.1 巡回セールスマン問題とは

巡回セールスマン問題とは次のような問題である。複数個の都市があり、各都市間に非負のコストが割り当てられているとき、すべての都市を一度づつ訪問する巡回路を決定することを考える。このとき、巡回路の総コストが最小である巡回路を決定する問題を巡回セールスマン問題といい、その巡回路を最適巡回路という。

TSP は都市間のコストの割り当て方によって様々な種類に分類される。例えば、都市 i から都市 j へのコストと都市 j から都市 i へのコストが同じである問題を対称巡回セールスマン問題といい、異なる問題を非対称巡回セールスマン問題という。その他にも、都市を平面上に配置し、2次元上の距離をコストに割り当てたものや、都市を球面上に配置し、球面上の距離をコストに割り当てたものなどがある。

本研究では、都市を平面上に配置し、都市間の距離をコストに割り当てる2次元上の TSP のみを考えた。

1.2 巡回セールスマン問題の現状

n 都市の TSP の最適解を求めることを考える。n 都市を巡回する可能な巡回路の総数は、対称巡回セールスマン問題の場合 $(n-1)!/2$ 個、非対称の場合 $(n-1)!$個となり、総当たりで最適巡回路を決定しようとすると、n=30 程度でも現実的な時間で解くことはできない。また、総当たりでなく、効率的に最適巡回路を決定する方法も考案されていない。

そこで、最適でなくとも、ある程度の精度を持った近似解を短時間に求めるということが必要とされ、様々な TSP に対する近似解法が研究されている。

2 従来の TSP 近似アルゴリズム

TSP 近似アルゴリズムを求めるのに初期解をまず求めて（構築法），それを改善していく（改善法）という二つのフェーズからなるアプローチがよくとられている．本研究でも，このアプローチにより近似アルゴリズムの検討を行った．本節ではまず従来の TSP 近似アルゴリズムで用いられている構築法と改善法について述べる。

2.1 構築法

構築法とは与えられた都市、都市間のコストから何らかの方法で巡回路を作る方法である。次に本研究で扱った構築法を示す。

2.1.1 Greedy 法

Greedy 法とは次のような構築法である。

- 各都市間の辺をコストの昇順にソートする
- 以下を、辺のコストが小さい順に調べ、すべての都市を訪問する巡回路ができるまで繰り返す
  - 辺を加えた構築解が次の 2 つの条件を満たすならば辺を構築解に加える
    1. 各都市の次数が 2 を越えない
    2. すべての都市を訪問しない巡回路を作らない

この方法は都市数 n のとき、$O(n^2 \log n)$ で実現でき、$O(\log n)$ の精度保証を持つ。性質としては、コストの小さい辺から構築解に加えていくため、部分的には最適巡回路と同じ辺を多く含む傾向があり、実験的には改善法にもかかりやすいことが知られている。しかし、構築の最終段階ではコストの小さい辺を選べず、極端にコストの大きい辺を選ぶ傾向があり、解の精度自体はあまり良くないことが知られている。

2.1.2 Nearest Neighbor 法

Nearest Neighbor 法 (NN) とは次のような構築法である

- 適当な都市を選び、出発点とする
- すべての都市を訪問するまで以下を繰り返す
  - まだ未訪問の都市で、現在いる都市から最も近い都市を選ぶ
  - 現在いる都市と選んだ都市を接続し、選んだ都市に移動する
- 現在いる都市と出発点を接続する

この方法は都市数 n のとき、$O(n \log n)$ で実現でき、$O(\log n)$ の精度保証を持つが、本研究では $O(n^2)$ の実現にとどめた。性質としては、コストの小さい辺から解に加えていくので、Greedy 法と同じく、部分的に最適巡回路と同じ辺を多く含む傾向があり、改善法にかかりやすい。また、この方法も構築の最終段階で極端にコストの大きい辺を選ぶ傾向があり、解の精度は良くない。

2.1.3 Nearest Insertion 法

Nearest Insertion 法 (NI) とは次のような構築法である。

- n 個の都市の集合を V とし、都市 i,j 間のコストを $C_{i,j}$ とするとき、V から適当な都市を一つ選び、その 1 都市のみの退化した部分巡回路 (セルフループ) を作る
- 部分巡回路上の都市の集合を X、部分巡回路上にない都市の集合を $V - X$ とするとき、$X = V$ となるまで以下を繰り返す

- $\min_{k \in V-X} \min_{j \in X} C_{k,j}$ を達成するを kを求める
- 部分巡回路上の連続した都市 i, jについて、
  $\min C_{k,i} + C_{k,j} - C_{i,j}$ を達成する i, j間に kを挿入し部分巡回路に加える

この方法は都市数 n のとき、$O(n^2)$ で実現でき、精度保証 2、すなわち、最適解の 2 倍以下の解を算出する。精度保証の点で、Greedy 法、Nearest Neighbor 法より優れたアルゴリズムであるが、部分的には最適巡回路からほど遠い巡回路を構築する傾向があり、改善法にかかりにいくことが知られている。

2.1.4 Farthest Insertion 法

Farthest Insertion 法 (FI) とは次のような構築法である。

- n 個の都市の集合を Vとし、都市 i,j 間のコストを $C_{i,j}$とするとき、Vから適当な都市を一つ選び、その 1 都市のみの退化した部分巡回路 (セルフループ) を作る
- 部分巡回路上の都市の集合を X、部分巡回路上にない都市の集合を $V - X$とするとき、$X = V$となるまで以下を繰り返す
  - $\max_{k \in V-X} \min_{j \in X} C_{k,j}$ を達成するを kを求める
  - 部分巡回路上の連続した都市 i, jについて、
    $\min C_{k,i} + C_{k,j} - C_{i,j}$ を達成する i, j間に kを挿入し部分巡回路に加える

この方法は都市数 n のとき、$O(n^2)$ で実現でき、精度保証は与えられていない。性質としては Nearest Insertion と同じような性質を持つことが知られている。Nearest Insertion 法と違い、部分巡回路から遠い都市を挿入していくのは、構築の早い段階で巡回路の全体の形を作ることがねらいである。精度保証は与えられていないが、実験的には Nearest Insertion よりも良い解を算出することが知られている。

2.2 改善法

改善法とは何らかの方法で得られた巡回路をもとに、さらにコストの小さい巡回路を探す方法である。次に本研究で扱った改善法を示す。

2.2.1 2-opt 法

2-opt 法とは次のような改善法である。

- 巡回路を入力させる
- 以下を改善ができなくなるまで繰り返す
  - 適当な都市 a を選び、巡回路で a の次の都市を b とする
  - 都市 a の近傍 N 都市から a,b 以外の都市 c と、巡回路で c の次の都市 d を選ぶ

– 都市 a から b への辺のコストを $C_{a,b}$ と書くとすると

$$C_{a,b} + C_{c,d} > C_{a,c} + C_{b,d}$$

であれば辺 ab, cd を ac, bd に入れ換えて改善する

2-opt 法は、図 4.1 のように 2 辺を入れ換えて改善する方法であり、辺が交差している巡回路の改善に威力を発揮する改善法である。この方法は理論的には、最悪の場合、指数オーダーの改良回数を必要とし、解の精度をいくらでも悪くできる問題を作ることができるが、実際にはそのような問題は稀であり、実験的には優れた改善法である。

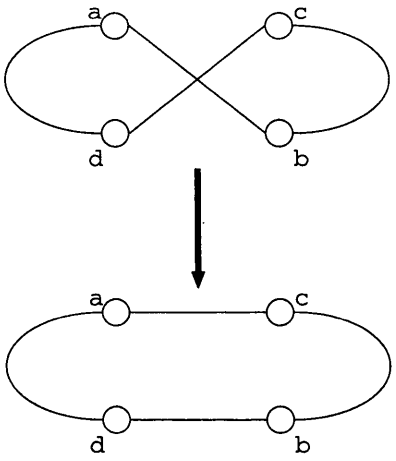


図 3.1: 2-opt 法の改善例

3 本研究の TSP 近似アルゴリズム

本節では本研究で提案する TSP 近似アルゴリズムの構築法と改善法について述べる。本研究では構築法として Greedy 法を改良した改良 Greedy 法 [4]，NN 法を改良した DNN 法 [5] の二つの方法を提案する．また，改善法として 2-opt 法を改善した 2-path-opt 法 [4] と Combination-opt 法 [6] の二つの方法を提案する．

3.1 本研究の構築法

3.1.1 改良 Greedy 法

前節で取り上げた Greedy 法は構築の最終段階で極端にコストの大きい辺を選んでしまう。これは、目先の利益だけを考えて構築しているためだと考えられる。そこで、Greedy 法の辺を選ぶ基準を変更し、後々のことも考慮しつつ、貪欲に構築していく方法を考えた。以下に改良 Greedy 法のアルゴリズムを示す。

- 各都市 i について
  - 都市 i から i 以外の都市への辺のコストの平均 μ_i を求める

- 都市 i から都市 j への辺のコスト $C_{i,j}$ を $C_{i,j}-(\mu_i+\mu_j)$ に書き換え、昇順にソートする
- 以下を、辺のコストが小さい順に調べ、すべての都市を訪問する巡回路ができるまで繰り返す
  - 辺を加えた構築解が次の 2 つの条件を満たすならば辺を構築解に加える
    1. 各都市の次数が 2 を越えない
    2. すべての都市を訪問しない巡回路を作らない

この方法は都市数 n のとき、$O(n^2 \log n)$ で実現できる。精度保証を与えることはできていない。

Greedy 法と異なる点は、辺のコストを書き換えてから Greedy に進めていく点である。この方法は、コストが小さい辺で、かつ、その辺の端点につながる辺の平均コストが大きいものから解に加えていく。このように構築していくことで良い解が算出されるという保証はないが、直観的説明としては、もし、そのような辺が解に加えられないとすると、いずれはその辺の端点につながる辺が選ばれることになる。すなわち、平均コストが大きい辺の中から辺が選ばれることになり、結果的に解全体のコストが上がることが考えられる。つまり、この方法は、そのような損失を未然に防いでいると考えることができる。

3.1.2 Division Nearest Neighbor 法 (DNN 法)

従来の各構築法の特徴として、FI 法, NI 法は改善法にかかりにくく、NN 法は改善法にかかりやすいとなっている。NN 法は最も近い未訪問都市を結んでいく。そのため構築の初期段階ではコスト増分は少ないが、後半の段階では残った都市を結んでいくことになり、非常に長いパスを追加することもある。そこで、本研究では分割 Nearest Neighbor 法 (Division Nearest Neighbor;DNN) を提案する。以下に DNN 法のアルゴリズムを示す.

1. 最も距離の離れた 2 都市 (S1), (S2) を選択する
2. (S1), (S2) を通る直線で都市群を 2 つのグループ [G1], [G2] に分ける。ただし (S1), (S2) はどちらにも属さない
3. (S1) を開始都市として、[G1] に対して NN 法を行う
4. グループ [G1] で最後に訪問した都市と (S2) を結ぶ
5. (S2) を開始都市として、[G2] に対して NN 法を行う
6. グループ [G2] で最後に訪問した都市と (S1) を結ぶ

DNN 法の特徴は、仮想的に往路・復路に分けていることである。仮想的というのは、実際にははっきりとした往復路ができるわけではないが、少なくとも境界線をまたぐ辺は現れないので、往路・復路と区別して考えることができる。

分割後、都市の数が半数ずつになるが理想であるが、この分割方法では保証はされていない。往路・復路それぞれの始点・終点は全都市の中で最も外側にあるほうが効率がいいため、この条件を優先させた。

都市数の条件を満たすには、曲線で分割することになり、計算が難しくなるため本研究では直線による分割で実現する。

また、Nearest Neighbor 法を 2 回に分けて適用するため、Nearest Neighbor 法の特徴である" 構築後半段階の長いパス" が純粋な Nearest Neighbor よりも多い。このため、構築解の精度は Nearest Neighbor 法による構築に比べ、悪くなることが予想される。しかし DNN 法は" 改善可能性" の高い初期解を生成することを目的とするので、構築解の精度は問わないことにする。

本研究で実現した Nearest Neighbor 法は $O(N^2)$ であり、Division Nearest Neighbor 法は各グループの都市数が等しければ $2 \times O((\frac{N}{2})^2)$ となるので、DNN 法の時間計算量は $O(N^2)$ である。

3.2 本研究の改善法

3.2.1 2-path-opt 法

2-opt 法では 2 辺を入れ換えて改善するが、もっとたくさんの辺を入れ換えることで改善できないかと考えた。そこで、2-opt 法を拡張して、巡回路から 2 つのパスを選び、そのパスを用いて改善する 2-path-opt 法を考案した。以下にその方法を示す。

- 巡回路を入力させる
- 以下を改善ができなくなるまで繰り返す
  - 巡回路から長さ L のパスを選び、パスの端点を a,b とする
  - a の近傍 N 都市からパス ab と重複しない長さ L のパス cd を選ぶ
  - $C_{a,b} + C_{c,d} > C_{a,c} + C_{b,d}$ であるなら以下の操作を行なう
    - 2 本のパス上の都市を、a から c、b から d へのパスの組に分けるとき、そのすべての分け方に対して、
      - 最短パスを求め、2 本のパスの和が最小であれば記録しておく
    - 求めた 2 本のパスのコストの和が元の 2 本のパスのコストの和より小さければ入れ換えて改善する
- 巡回路を出力する

2-path-opt 法は、図 5.1 にのように、巡回路上の 2 本のパスで最適化を行ない、改善する。この方法で最も時間のかかる部分は 2 本のパス上の都市を 2 組にわけ、それぞれの組の最短パスを求める部分である。2 本のパス上の都市を a から c、b から d のパスの組に分ける分け方の総数は $2^{2(L-2)}$通りであり、それぞれに対して最短パスを求める。本研究では、最短パスを求める部分は動的計画法で実現した。用いた動的計画法では、長さ L のパスの最短パスを求めるのに $L^2 2^L$ に比例した時間がかかる。よって、一回の改善に $2^{2(L-2)} \times L^2 2^L$ に比例した時間がかかり、L に関して指数関数的であるが、L がある程度小さければ十分短時間で計算可能である。実験的には L=4 で 2-opt 法の 2 倍程度の計算時間がかかることを後で示す。本研究ではこの方法に計算量、精度保証を与えることができなかった。

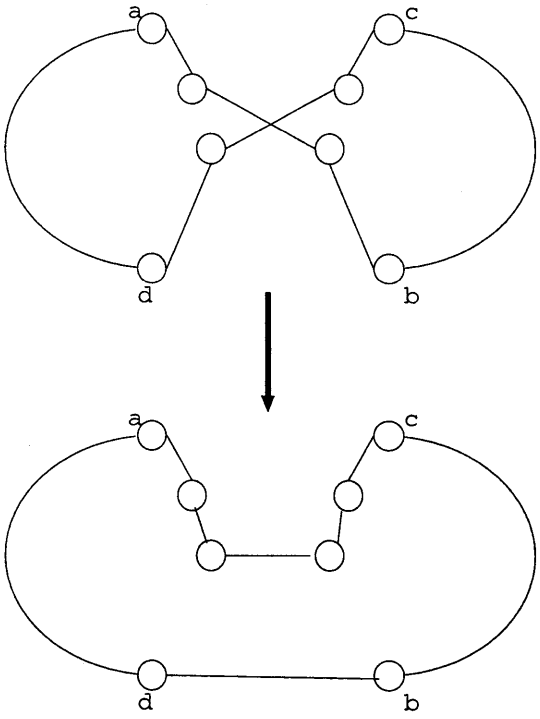


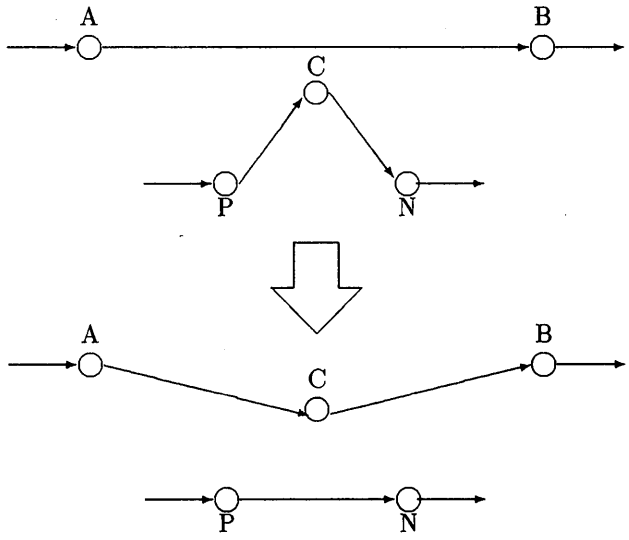
図 4.1: 2-path-opt 法の改善例

3.2.2 1.5-opt 法

本研究では、2-opt でカバーできない局所最適化に対処する改善法を用意した。

任意の点を、別の辺中に移動させるもので、ある隣接した都市 (A), (B) と、単一の都市 (C) を選択する。都市 (C) の巡回路上での前の都市を (P)、次の都市を (N) とする。もし、都市間の距離 $|AC|+|CB|+|PN|$ が $|AB|+|PC|+|CN|$ よりも小さいならば、(P) と (N) を繋ぎ、(C) を (A)−(B) 間に挿入する。

$|AB|$ が非常に長い辺の場合に特に有効で、Nearest Neighbor 法の後半の段階で発生する長いパスを崩していくことができる。



1.5-opt 法

近傍都市と近傍都市密度

2-opt では改善可能な 2 辺は交わっていることが条件である。つまり、この 2 辺を構成する 4 都市は比較的近い位置にある。したがって、基準都市 (A) の近傍から都市 (C) を選択し、改善可能かどうかを調べる方法を取ることによって、全体から検索するよりも計算量を減少させることができる。明らかに改善不可能な組み合わせはチェックしなくなっている。

近傍都市

近傍都市の個数は、全都市数の log をとって決定する。都市の数に比例させると計算量を抑えるという点で不利になるからである。また、都市数が少ない問題に対してはなるべく多く確保したいが、都市数が多い問題ではある程度の個数を確保すれば十分だからである。

全都市数を Citys とすると、近傍都市数 Neighbor は以下のように決定する。

$$Neighbor = 10 * \log_2 Citys$$

(ただし $Neighbor > Citys$ の場合は $Neighbor = Citys/2$)

近傍都市密度

改善法を適用する場合、巡回路が改善可能な間、基準となる辺を選出してから改善対象の辺や点を選び、改善可能性を計算する。この基準辺の選択順序が変わると改善法の適用順序が変わり、求まる近似解は異なったものになる可能性がある。このばらつきは、改善法のアルゴリズムには関係ない。

そこで本研究では、問題に固有な順序で改善法を適用するために、近傍都市密度の高い都市から基準辺を選択する方法をとった。

すべての都市に対して、Neighbor 個の近傍都市への距離の合計を計算し、この合計が小さいものほど周辺の都市への距離が近い、つまり近傍都市の密度が高いと判定する。近傍都市密度順に改善法を適用することによって、問題に固有な順序で改善法を適用することができ、ばらつきのない近似解を得ることができる。

Combination-opt 法

本研究での改善法の適用は、2-opt, 1.5-opt 両方の特性を活かすため、次のような順序で適用する。これを Combination-opt 法 (COMB) と呼ぶ。

1. 基準都市 (A) を選択
2. (A) の次の都市 (B) と辺 (AB) を作る
3. 都市 (A) の近傍から都市 (C) を選択
4. 辺 (AB), 都市 (C) で 1.5-opt の改善可能性を計算
5. 改善可能な場合
   (a) 1.5-opt を実行
   (b) 1. へ戻る
6. 改善不可能な場合
   (a) (C) の次の都市 (D) と辺 (CD) を作る
   (b) 辺 (AB), 辺 (CD) で 2-opt の改善可能性を計算
   (c) 改善可能な場合、2-opt を実行
   (d) 1. へ戻る

基準都市 (A) がすべての都市を一巡したとき、その間に一度も改善が実行されていなければ、その巡回路は 1.5-opt, 2-opt による改善がこれ以上できない状態にあるので、改善法は終了する。

本研究で実現した 2-opt, 1.5-opt は、N個の都市が $10 \times \log_2 N$個の近傍都市に対して改善を試みるため、時間計算量は $O(N \log N)$ である。

改善法の応用

本研究では入力データは都市の座標 (x, y) の羅列であり、この座標の定義順に連続した番号 (都市番号) を都市の識別に用いている。前節の手順 1. で基準都市を選択する順番は、都市番号順に行うものと、近傍都市密度順に行う 2 通りで行った。

都市番号順に行う Combination-opt を COMB1、近傍都市密度順に行うものを COMB2 と表記する。

4 実験結果

本節では，構築法と改善法に対して，Greedy 改良法＋ 2-path-opt 法と DNN 法＋ Combination-opt 法の組み合わせに対して従来の方法と実験的にその性能の比較を行った結果を示す.

4.1 Greedy 改良法＋ 2-path-opt 法

4.1.1 比較方法

従来の方法と、考案した方法を実験的に比較した。比較には TSP の問題例の標準的なデータベースである TSPLIB[3] の 2 次元ユークリッド上の問題 7 1 個を使用した。UNIX の time コマンドを用いて計算時間を計測し、

$$\frac{\text{近似解} - \text{最適解}}{\text{最適解}} \times 100$$

で、誤差の計算を行ない、各方法について平均を計算した。また、分散も計算した。

構築法の比較では、構築解の比較と、近傍数 N=10 の 2-opt 法を用いて改善法のかかりやすさの比較を行なった。

改善法の比較では、入力として、従来の方法と考案した改良 Greedy 法の出力結果を用い、2-path-opt 法のパス長 L=4 に固定し、近傍数 N を変化させて比較した。

4.1.2 比較結果

改良 Greedy 法は、Greedy 法とそれほどかわらぬ計算時間で、精度があがった (表 5.1 参照)。改善法にかかりやすいかどうは、改良 Greedy 法の解の精度がもともと良いため、よくわからなかった (表 5.2 参照)。

2-path-opt 法は 2-opt 法とくらべ、解の精度はあがったが，計算時間が 2 倍近く時間がかかった (表 5.3 参照)。

構築法	平均計算時間 (秒)	平均誤差	誤差の分散
NN	0.34	23.85	35.85
NI	1.05	21.58	16.91
FI	1.08	11.18	27.84
Greedy	53.87	17.74	20.97
Greedy 改	61.08	8.77	10.42

表 5.1: 構築法の比較

構築法	平均計算時間 (秒)	平均誤差	誤差の分散
NN	13.76	8.01	18.39
NI	13.75	12.26	11.02
FI	13.76	9.58	21.68
Greedy	13.81	7.38	11.51
Greedy 改	13.76	5.53	6.12

表 5.2: 2-opt(N=10) を適用したときの比較

改善法	近傍数	平均計算時間 (秒)	平均誤差	誤差の分散
2-opt	10	13.77	8.56	18.89
	20	12.59	8.32	18.20
	30	12.49	8.28	17.96
	40	12.76	8.26	17.89
2-path-opt (L=4)	10	21.35	8.31	15.37
	20	23.85	7.71	14.29
	30	25.73	7.51	14.03
	40	26.73	7.32	13.66

表 5.3: 改善法の比較

4.2 DNN 法+ Combination-opt 法

データは TSPLIB より、2 次元平面上の対称巡回セールスマン問題 (att48, att532, pr1002, pr2392, rl5915, brd14051, pla33810, pla85900) を使用した。いずれの問題も、問題名後尾の数字が都市の数を表している。

構築解 (初期解)・改善解 (近似解) の精度は、

$$\frac{\text{初期解 (}or\text{近似解) コスト} - \text{最適解コスト}}{\text{最適解コスト}} \times 100 \qquad (\%)$$

とし、最適解が求まっておらず、[下界, 上界] が与えられている問題 (rl5915, brd14051, pla33810, pla85900) に対しては

$$\frac{\text{初期解 (}or\text{近似解) コスト} - \text{上界}}{\text{上界}} \times 100 \qquad (\%)$$

で表す。

また、上界の値は、1999 年 12 月時点で TSPLIB で公表されている値を使用した。

4.2.1 初期解および近似解の精度

実験において生成した初期解コスト・近似解コストの精度を示す。

問題・改善法 \ 構築法		FI	NI	NN	**DNN**
att48	初期解	11.10%	7.62%	11.69%	**24.67%**
	2-opt	9.65%	6.93%	2.40%	**3.10%**
	COMB1	2.60%	2.77%	3.75%	**2.41%**
	COMB2	2.20%	3.45%	2.13%	**2.41%**
att532	初期解	20.19%	19.82%	23.76%	**42.93%**
	2-opt	9.15%	11.18%	12.03%	**3.30%**
	COMB1	7.65%	7.47%	7.41%	**5.75%**
	COMB2	8.38%	7.80%	8.45%	**6.50%**
pr1002	初期解	25.38%	22.16%	24.00%	**27.95%**
	2-opt	12.15%	11.01%	16.29%	**5.37%**
	COMB1	10.43%	8.29%	10.75%	**6.64%**
	COMB2	8.15%	8.49%	11.88%	**5.86%**
pr2392	初期解	28.25%	29.48%	27.11%	**29.71%**
	2-opt	12.24%	14.97%	13.85%	**6.14%**
	COMB1	9.61%	11.56%	10.20%	**6.71%**
	COMB2	10.12%	11.25%	9.72%	**5.39%**

–表は次ページへ続く–

問題・改善法 \ 構築法		FI	NI	NN	DNN
rl5915	初期解	31.94%	32.04%	29.81%	**27.64%**
	2-opt	10.58%	9.66%	9.67%	**6.62%**
	COMB1	8.26%	7.55%	8.71%	**7.04%**
	COMB2	8.91%	6.66%	8.67%	**5.80%**
brd14051	初期解	29.10%	27.63%	22.34%	**26.91%**
	2-opt	11.51%	14.49%	12.47%	**5.67%**
	COMB1	8.52%	8.48%	7.58%	**5.51%**
	COMB2	8.17%	8.57%	8.75%	**5.42%**
pla33810	初期解	22.50%	21.34%	20.55%	**68.37%**
	2-opt	8.98%	9.03%	8.32%	**9.97%**
	COMB1	5.54%	5.76%	6.39%	**8.57%**
	COMB2	6.13%	6.02%	5.67%	**8.34%**
pla85900	初期解	23.67%	24.52%	23.32%	**16.12%**
	2-opt	8.94%	9.56%	8.75%	**9.98%**
	COMB1	6.09%	6.15%	5.51%	**3.72%**
	COMB2	5.99%	6.35%	5.54%	**3.78%**

列ラベルはそれぞれ Farthest Insertion 法、Nearest Insertion 法、Nearest Neighbor 法、そして本研究で考案した Division Nearest Neighbor 法を用いて初期解を生成したことを示す。行ラベルは各問題に対して、構築した初期解の精度、既存の 2-opt 法で改善した近似解の精度、本研究で考案した Combination-opt 法で改善した近似解の精度となっている。

問題によって差はあるものの、従来の構築法よりも本研究で考案した Division Nearest Neighbor 法で初期解を構築した場合のほうが、よい精度を持った近似解を得ることができている。また、従来の改善法 2-opt 法のみを用いた場合よりも、2-opt 法に 1.5-opt 法を交えた Combination-opt 法で改善した場合のほうが、よい精度を持った近似解を得ている。

4.2.2 計算時間

問題	計算時間	近似解精度
att48	1秒	2.41%
att532	2秒	6.50%
pr1002	5秒	5.86%
pr2392	15秒	5.39%
rl5915	100秒	5.80%
brd14051	500秒	5.42%
pla33810	1時間	8.34%
pla85900	5時間	3.78%

計測には Pentium II 450MHz の計算機を使用し、プログラムの開始と終了時の時間の差を取った。プロセッサ時間ではなく、またデータファイルの読み込みなどの前処理も含んでいる。近似解精度は構築法 DNN、改善法 COMB2 の時のものを参考程度に載せた。

構築法・改善法の組み合わせの違いによる差は、±10%以内で、目立った差はなかった。また、従来の 2-opt 法のみによる改善と、Combination-opt 法による改善も、同程度の計算時間であった。

4.2.3 評価

構築法による初期解の精度では、DNN 法は他の構築法に比べて精度の悪い解を構築するが、その後改善法を適用することによって他の構築解よりも精度のよい近似解を求めることができた。

また、構築法 DNN を用いて近傍都市密度の高い順に改善を行った方法が、比較的他の近似アルゴリズムよりも精度のよい近似解を得ることができた。

本研究では、2 次元平面上の対称巡回セールスマン問題のみを考えたため、都市のデータは座標の組で与えられている。そのため、メモリ使用量のオーダーは $O(N)$ で十分であるが、構築や改善の段階で莫大な回数の距離の計算を行う。各都市間のコストを与えれば、コストの計算を毎回行う必要はないが、メモリ使用量は $O(N^2)$ となる。これは非対称巡回セールスマン問題の場合でも同じである。都市間のコストを記憶するに十分なメモリを持った計算機を用いるか、コスト表をメモリに記憶できる程度の都市数の問題を解く際には、更なる高速化は望めると考える。

おわりに

本研究では、TSP に対する近似アルゴリズムの研究として、従来の近似アルゴリズムを改良した新しい構築法と改善法を考案し、従来の近似アルゴリズムとの比較を行なった。比較の結果から、考案した近似アルゴリズムは、実験的には従来のものよりも精度が良いことがわかった。

今後の課題としては、考案した解法の高速化と理論的検証, ならびに, 遺伝的アルゴリズム [6], ニューロコンピューティングなどを用いた近似アルゴリズムの検討などが挙げられる。

参考文献

1. 山本芳嗣、久保幹雄、「巡回セールスマン問題への招待」、朝倉書店, 1997

2. TSPLIB: http://www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95/

3. Gerhard Reinelt: "The Traveling Salesman – Computational Solutions for TSP Applications", Lecture Notes in Computer Science 840, Springer-Verlag, 1994

4. 坂上知英、「TSP 近似アルゴリズムの研究」, 平成 11 年度 三重大学工学部 情報工学科 卒業論文, 1999

5. 吉澤慎、「巡回セールスマン問題の近似アルゴリズムに関する研究」, 平成 11 年度 三重大学工学部 情報工学科 卒業論文, 1999

6. 河田俊郎、「並列処理に関する研究・遺伝的アルゴリズムを用いた巡回セールスマン問題の並列化」, 平成 10 年度 三重大学工学部 情報工学科 卒業論文, 1998