数理解析研究所講究録
906巻 1995年 24-31

巡回セールスマン問題 (Traveling Salesman Problem) の貪欲アルゴリズムについて

横山 光一
Mitsukazu YOKOYAMA
(mitsu@cs.titech.ac.jp)
東京工業大学 工学部 情報工学科 4年 渡辺研究室
〒152 東京都目黒区大岡山 2-12-1

1 序論

巡回セールスマン (最適化) 問題 (Traveling Salesman (Optimization) Problem) は，代表的な NP-困難問題として知られている [GJ83]．また，近似解を求める問題や，都市間のコストを三角不等式を満たすものに制限したり，2値に制限したりした場合においての最適化問題も，NP-困難であることが知られている [GJ83, PY93]．

本論文において，巡回セールスマン問題 とは以下のような問題 (以下は，TSP と略す) である．

> **入力** : コストつき完全無向グラフ G
> **問題** : G 上でコストの和が最小となるハミルトン閉路を求めよ．

また，最小のハミルトンパスを求める問題 とは以下のような問題 (以下は，TSP† と略す) である．

> **入力** : コストつき完全無向グラフ G
> **問題** : G 上でコストの和が最小となるハミルトンパスを求めよ．

ここでの グラフ $G=(C,E)$ は，n 個の 都市 からなる集合 C とそれぞれの都市間の コスト つき 辺 の集合 E からなる完全無向グラフで，具体的には，図1.1のようなグラフである．すなわち，$C=\{c_1,c_2,\ldots,c_n\}$ であり，辺 c_ic_j のコストを表す $d(c_i,c_j)$ が E の要素である．また，ハミルトン閉路 ， ハミルトンパス とはそれぞれ，与えられたすべての頂点 (この場合は都市) をちょうど1度ずつ通るような閉路，パス (閉路ではない) のことであり，以下では，単に 閉路 ， パス と呼ぶこともある．

すなわち，TSP とは，「あるセールスマンが，ある都市から出発して，すべての都市をちょうど1度ずつ訪れて，再び出発した都市に戻ってくるときに，最小のコストで済ますにはどういったルートをとればよいか？」を考える問題である．

本論文では，TSP(辺のコストが2値) や TSP† に対して，ある種の貪欲アルゴリズム (Greedy Algorithm) は，最悪解でない解を見つけることができることを示した．

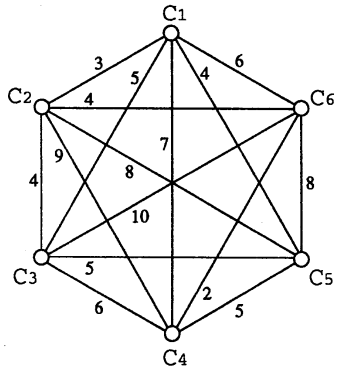


図1.1 TSP や TSP† に入力されるグラフの例

2 準備

本論文において用いられる記号や語句は，一般に用いられているものがほとんどだが，主なものと，まぎらわしいものについてここで定義しておく．

TSP や TSP† において，グラフのハミルトン閉路やハミルトンパスの中で，コストの和が最小のものを 最適解 と呼び，最大のものを 最悪解 と呼ぶ．最悪解である とは，「最悪解であって，最適解ではない」ことを意味し，最悪解でない とは，「最適解であるか，最悪解ではない」ことを意味する．すなわち，すべての閉路 (パス) のコストの和が等しいグラフを コスト不変 のグラフと定義すると，コスト不変のグラフにおいては，いずれの言葉も意味を持たない．

閉路やパスは $[c_{k_1}, c_{k_2}, \ldots, c_{k_i}]$ などと表し，c_{k_1}，c_{k_2}，… の順に 訪れる 閉路 (パス) と同じ意味である．また，都市番号 とは，入力された都市 c_i の添字 i のことで，訪れる順番のことではない．

TSP や TSP† に対する 計算時間 は，入力の長さではなく都市数 n を基準として議論する．

3 TSP† に対する貪欲アルゴリズムについて

3.1 アルゴリズム *Greedy*1 の説明

TSP† に対し，次のような貪欲アルゴリズム *Greedy*1 を考える．

入力 ： コストつき完全無向グラフ G，出発都市 c_i
出力 ： ハミルトンパス — $Greedy1(G, c_i)$

より正確には 図 3.1 のようなアルゴリズムである．

次に，計算時間を考える．都市間のコストの大小の比較の回数を計算時間の単位とする．アルゴリズム *Greedy*1 の各ステップにおける比較の回数は，

Algorithm *Greedy*1(**input** G, c_i); % $G=(C,E)$, c_i は出発都市
begin
  $Ans[1] \leftarrow c_i$; % Ans は解を入れる配列で，$Ans[j]$ には j 番目に訪れる都市が入る．
  $C' \leftarrow C - \{c_i\}$; % C' はまだ訪れていない都市の集合
  for $j \leftarrow 2$ **to** n **do**
  begin
    $Ans[j] \leftarrow d(Ans[j-1], c)$ が最小になる都市 $c \in C'$; % <注>
    $C' \leftarrow C' - \{Ans[j]\}$;
  end-for;
  output Ans;
end.
<注> そのような都市 c が複数個存在する場合は，都市番号が小さいものを優先する．

図3.1 アルゴリズム *Greedy*1

Step	内容	比較の回数
$Ans[2] \leftarrow c_{k_2}$	$n-1$ 個の数字の中から最小の数字を選ぶ．	$n-2$
$Ans[3] \leftarrow c_{k_3}$	$n-2$ 個の数字の中から最小の数字を選ぶ．	$n-3$
⋮	⋮	⋮
$Ans[n-1] \leftarrow c_{k_{n-1}}$	2個の数字の中から小さい数字を選ぶ．	1
$Ans[n] \leftarrow c_{k_n}$	1個残っている数字を選ぶ．	0

<注> c_{k_j} は j 番目に訪れる都市のことである．

のようになるので，計算時間 $f(n)$ は，比較の回数を合計して，

$$f(n) = \sum_{k=2}^{n}(n-k) = \frac{1}{2}n^2 - \frac{3}{2}n + 1 = \mathcal{O}(n^2)$$

となる．

3.2 アルゴリズム *Greedy*1 の解析

以下は，この先用いられる語句や記号の定義である．

1. 「アルゴリズム *Greedy*1 にグラフ G と出発都市 c_i を入力して動かす」ことを，アルゴリズム *Greedy*1 を G において c_i から適用させる (G が重要でないときは，単にアルゴリズム *Greedy*1 を c_i から適用させる) と呼び，得られたパスを *Greedy*1(G, c_i) と表す．

2. アルゴリズム *Greedy*1 を適用させたときに出発した都市を s と表し，以下，訪れた都市を順に，s_1 , s_2 , ⋯ , s_{n-1} と表す．便宜上，$s = s_0$ として考える．
すなわち，*Greedy*1$(G, c_1) = [s, s_1, s_2, \ldots, s_{n-1}]$ である．

3. パスを表記する際に現れる，$\underline{s_i \nearrow s_j}$ は $s_i, s_{i+1}, \ldots, s_j$ を表し，$\underline{s_i \searrow s_j}$ は $s_i, s_{i-1}, \ldots, s_j$ を表し，$s_i \nearrow s_i$ や $s_i \searrow s_i$ や s_i, s_i は s_i 自体を表す．
例えば，$Greedy1(G, c_1) = [s \nearrow s_{n-1}]$ である．

ここでは，以下の定理を証明する．

定理 3.1

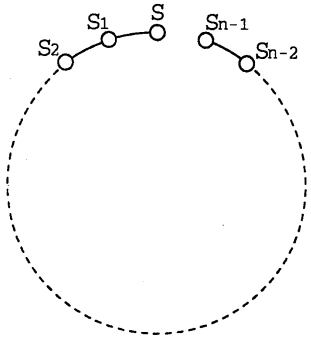
TSP† において，アルゴリズム *Greedy*1 を 2 つの都市から適用させれば，最悪解でないパスが得られる．

< 証明 >

「$Greedy1(G, c_1)$ は最悪解である．」と仮定 (仮定 A とする) すると，$Greedy1(G, c_1)$ のグラフは後に示す **補題 3.2** のように完全に決定される (図 3.2 参照)．よって，アルゴリズム *Greedy*1 を $s_{n-1}(=c_n)$ から適用させると，$Greedy1(G, c_1)$ より良いパス $[s_{n-1}, s \nearrow s_{n-2}]$ が得られる．□ **< 定理 3.1 の証明 終わり >**

補題 3.2

仮定 A のもとでは，グラフ G は，図 3.2 のように完全に決定される (証明は省略する)．



1. $d(s_i, s_{i+1}) \leq d(s_{i+1}, s_{i+2})$ $(0 \leq i \leq n-3)$．
2. $d(s_i, s_{i+1}) = d(s_i, s_j)$ $(0 \leq i \leq n-2,\ i+2 \leq j \leq n-1)$．
3. $d(s, s_1) < d(s_{n-2}, s_{n-1})$．
4. $s_i = c_{i+1}$ $(1 \leq i \leq n-1)$．

図 3.2 補題 3.2

ところで，アルゴリズム *Greedy*1 を 1 つの都市から適用させるときの計算時間は，$\mathcal{O}(n^2)$ だったので，c_1，c_n の 2 つの都市から適用させるときの計算時間も $\mathcal{O}(n^2)$ である．すなわち，TSP† を解くにあたって，アルゴリズム *Greedy*1 は，$\mathcal{O}(n^2)$ の計算時間で，最悪解でないパスを見つけることができる．

ところで，九州大学の岩間一雄教授の御指摘により，以下のことも分かった．

定理 3.3

TSP† において，アルゴリズム *Greedy*1 を 1 つの都市から適用させれば，最悪解でないパスが得られる．

< 証明 >

コストの異なる辺が出ている都市 (c_d とする) から *Greedy*1 を適用させて得られるパス $Greedy1(G, c_d)$ について考える．

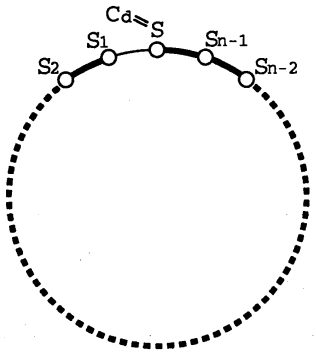


図3.3

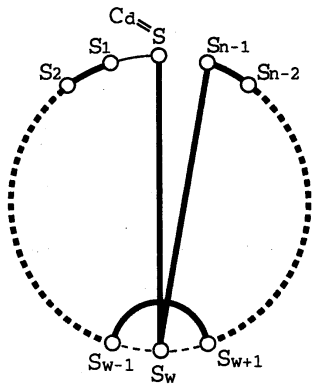


図3.4

(1) $d(s,s_1) < d(s,s_{n-1})$ のとき (図 3.3 参照)

パス $[s_1 \nearrow s_{n-1}, s]$ が $Greedy1(G,c_d)$ より悪い解になり，明らかに $Greedy1(G,c_d)$ は最悪解でない．

(2) $d(s,s_1) = d(s,s_{n-1})$ のとき (図 3.4 参照)

仮定より，s を通って $[s,s_1]$ よりコストの大きい辺が必ず存在する．その辺を $[s,s_w]$ とすると，パス $[s_1 \nearrow s_{w-1}, s_{w+1} \nearrow s_{n-1}, s_w, s]$ が $Greedy(G,c_d)$ より悪い解になり，$Greedy1(G,c_d)$ は最悪解でない．

(1), **(2)** より，$Greedy1(G,c_d)$ は最悪解でない．□ **< 定理 3.3 の証明 終わり >**

併せて，次のようなことも分かった．

事実 3.4

TSP† において，コスト不変のグラフは，すべての辺のコストが等しいグラフしかない．

< 証明 >

グラフ上にコストの異なる辺が存在すると仮定すると，必ずある都市 (c_d とする) からコストの異なる辺が出ていることになる．それらの辺を $[c_d,c_e]$, $[c_d,c_f]$ とすると，

$$\begin{cases} [c_d, c_e, \ldots, c_f] \\ [c_e, \ldots, c_f, c_d] \end{cases} \quad (\ldots\text{部は同じパスである})$$

の 2 つのパスのコストは明らかに異なる．よって，コスト不変のグラフは，すべての辺のコストが等しいグラフしかない．□ **< 事実 3.4 の証明 終わり >**

この **事実 3.4** から，次のことが明らかに成り立つ

系 3.5

TSP† において，最悪解でないパスを得るためには，コストの異なる辺が出ている都市で，その 2 つの都市を選び，コストの小さい辺を作る都市からコストの大きい辺を作る都市へ向けてランダムにパスを作れば，そのパスは最悪解にならない．

Algorithm *Greedy2*(**input** G, c_i); % $G=(C,E)$, c_i は出発都市
begin
  $Ans[1] \leftarrow c_i$; % Ans は解を入れる配列で，$Ans[j]$ には j 番目に訪れる都市が入る．
  $C' \leftarrow C - \{c_i\}$; % C' はまだ訪れていない都市の集合
  $Ans[2] \leftarrow$ $d(c_i, c)$ が最小になる都市 $c \in C'$; % <注>
  $C' \leftarrow C' - \{Ans[2]\}$;
  $Ans[n] \leftarrow$ $d(c_i, c)$ が最小になる都市 $c \in C'$; % <注>
  $C' \leftarrow C' - \{Ans[n]\}$;
  for $j \leftarrow 3$ **to** $n-1$ **do**
  begin
    $Ans[j] \leftarrow$ $d(Ans[j-1], c)$ が最小になる都市 $c \in C'$; % <注>
    $C' \leftarrow C' - \{Ans[j]\}$;
    $j \leftarrow j+1$;
  end-for;
  $Ans[n+1] \leftarrow c_i$;
  output Ans;
end.
<注> そのような都市 c が複数個存在する場合は，都市番号が小さいものを優先する．

図 4.1 アルゴリズム *Greedy2*

4 TSP に対する貪欲アルゴリズムについて

4.1 アルゴリズム *Greedy2* の説明

TSP に対し，次のような貪欲アルゴリズム *Greedy2* を考える．

入力 : コストつき完全無向グラフ G，出発都市 c_i
出力 : ハミルトン閉路 — *Greedy2*(G, c_i)

より正確には 図 4.1 のようなアルゴリズムである．

また，計算時間はアルゴリズム *Greedy1* と同様に求められ，$\mathcal{O}(n^2)$ である．

4.2 アルゴリズム *Greedy2* の解析

以下は，この先用いられる語句や記号の定義である．

1. 「アルゴリズム *Greedy2* にグラフ G と出発都市 c_i を入力して動かす」ことを，アルゴリズム *Greedy2* を G において c_i から適用させる (G が重要でないときは，単に アルゴリズム *Greedy2* を c_i から適用させる) と呼び，得られた閉路を *Greedy2*(G, c_i) と表す．

2. アルゴリズム *Greedy2* を適用させたときに出発した都市を $\underline{s}$ と表し，以下，訪れた都市を順に，$\underline{s_1}$，$\underline{s_2}$，$\cdots$，$\underline{s_{n-1}}$ と表す．便宜上，$\underline{s = s_0}$ として考える．
すなわち，$Greedy2(G, c_1) = [s, s_1, s_2, \ldots, s_{n-1}, s]$ である．

3. パスや閉路を表記する際に現れる，$\underline{s_i \nearrow s_j}$ は $s_i, s_{i+1}, \ldots, s_j$ を表し，$\underline{s_i \searrow s_j}$ は $s_i, s_{i-1}, \ldots, s_j$ を表し，$s_i \nearrow s_i$ や $s_i \searrow s_i$ や s_i, s_i は s_i 自体を表す．
例えば，$Greedy2(G, c_1) = [s \nearrow s_{n-1}, s]$ である．

4. パスや閉路を表記する際に現れる，$\underline{c_i \uparrow c_j}$ は $c_i, c_{i+1}, \ldots, c_j$ を表し，$\underline{c_i \downarrow c_j}$ は $c_i, c_{i-1}, \ldots, c_j$ を表し，$c_i \uparrow c_i$ や $c_i \downarrow c_i$ や c_i, c_i は c_i 自体を表す．

5. $d(s_{i-1}, s_i) \neq d(s_i, s_{i+1})$ を満たす s_i $(1 \leq i \leq n-2)$ や $d(s, s_1) \neq d(s, s_{n-1})$ を満たす s を $\underline{\text{切替点}}$ と呼ぶ (s_{n-1} は切替点と呼ばない)．

6. $Greedy2(G, c_i)$ 上の切替点を k_j が大きい (すなわち，s の添字の数字が大きい) 順に，$\underline{s_{k_1}}$，$\underline{s_{k_2}}$，$\cdots$ とする．

ここでは，以下の定理を証明する．

定理 4.1

TSP において，都市間のコストを 0 か 1 に制限したとき，アルゴリズム *Greedy2* をすべての都市から適用させれば，最悪解でない閉路が得られる．

< 証明の方針 >

次のような仮定を設け，切替点が 3 個以上，2 個，1 個，なしのそれぞれについて，矛盾を示すことにより **定理 3.1** と同様に証明できる．

仮定

1. $d(c_i, c_j) \in \{0, 1\}$ とする．
2. $Greedy2(G, c_1)$ は最悪解である (これより，$n \geq 4$ が成り立つ)．
3. アルゴリズム *Greedy2* をどの都市から適用させても，得られる閉路は最悪解である．
すなわち，$\forall i\ Greedy2(G, c_i) = Greedy2(G, c_1)$ $(2 \leq i \leq n)$ である．

アルゴリズム *Greedy2* を 1 つの都市から適用させるときの計算時間は，$\mathcal{O}(n^2)$ だったので，n 都市すべてから適用させる場合には $\mathcal{O}(n^3)$ になる．しかしながら，実際には n 都市すべてから適用させる必要はなく，c_1 と，c_1 から適用させて得られた閉路 ($Greedy2(G, c_1)$) 上の，s_{n-1}，s_{n-2}，s_{k_1}，s_{k_2} の計 5 つの都市から適用させれば十分であることも分かった．それらの都市は記憶しておけばよいので，計算時間は，$\mathcal{O}(n^2)$ に下げることができる．すなわち，TSP を解くにあたって，都市間のコストを 0 か 1 に制限した場合，アルゴリズム *Greedy2* は，$\mathcal{O}(n^2)$ の計算時間で，最悪解でない閉路を見つけることができる．

また，**定理 4.1** では都市間のコストを 0 か 1 に制限していたが，一般には，単に 2 値に制限するだけでも同じことが成り立つ．すなわち，次の定理が成り立つ．

定理 4.2

TSP において，都市間のコストを 2 値に制限したとき，アルゴリズム *Greedy2* を 5 つの都市から適用させれば，最悪解でない閉路が得られる．

< 証明 >

0,1 をそれぞれ，2 値のコストの小さい方，大きい方に置き換えれば同様に証明できる．□ **< 定理 4.2 の証明 終わり >**

5 結論

本論文では，TSP(都市間のコストが 2 値) や TSP† に対して，ある種の貪欲アルゴリズムは最悪解でない解を見つけることができることを示した．アルゴリズム *Greedy*1 も *Greedy*2 も「最適解や最適解の近似解を見つける」のではなく,「最悪解を出さない」ことだけが保証されたアルゴリズムである．しかし，2 つともかなり **単純** な **貪欲** アルゴリズムであるにもかかわらず,「最悪解を出さない」という結果はある意味でおもしろい.「最悪解を出さない」ことしか示してないが，実際には，最適解からの離れ具合でもある程度いい成績を残せるかもしれない．例えば，グラフをランダムに与えたときの，最適解からの離れ具合を統計にとってみるだけでもおもしろいと思う．

また，実際の問題において「最悪解を出さない」ことが嬉しい局面は，

- そんなにいい解は必要ないが，最悪解だけは避けたい (例えば，ロシアンルーレットなど) 場合
- 1 つだけ最適解があり残りは最悪解で，その差が大きい場合

くらいしかないが，そういった場合に「最悪解を出さない」ことは意味を持ってくる．そういった場合の問題について考えてみるのもおもしろいと思う．

今後の課題としては，

- TSP の都市間のコストを 2 値に限らない場合についても同様のことを示す．
- 入力グラフに制限を加えた問題での近似精度を考える．

などが挙げられる．

謝辞

最後に，本研究を進めるにあたりあたたかい御指導を賜わりました渡辺治助教授に心からの感謝を申し上げます．加えて，本研究に対して熱心な議論と多くの貴重な助言を頂きました築地立家さんをはじめとする渡辺研究室の皆様にもお礼を申し上げます．

参考文献

[GJ83] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company(1983), 113-117, 147.

[PY93] C.H. Papadimitriou and M. Yannkakis, *The Traveling Salesman Problem with Distances One and Two*, Mathematics of Operations Research(1993), Vol.18, No.1.