# 観光スポットの組合せを考慮した時刻依存観光経路問題

# Time-Dependent Tourist Routing Problem Considering Combinations of Sightseeing Spots

**情報工学専攻　松井 楓馬**
Fuma MATSUI

## 概要

現実における観光経路は移動時間が時刻に依存するとともに観光スポットの組合せや訪問順序によって得られる満足感が異なる場合が考えられる．そのような観光経路問題に対し，本研究では時刻依存オリエンテーリング問題を拡張し，観光スポット間の時刻依存の移動速度及び, 観光経路内の順序関係を考慮し, より高い利益を得ることができる観光経路設計をモデル化した．この新たなモデルに対して, 反復局所探索法と Ant Colony System の 2 種類による発見的解法を提案する．

**キーワード:** 観光経路，時刻依存経路問題，反復局所探索法，蟻コロニー最適化，メタ戦略．

## 1　はじめに

本研究に関連する経路問題として，オリエンテーリング問題 (Orienteering Problem:OP) が挙げられる．OP は NP 困難 [1] であるため，厳密な最適解を求めるアルゴリズムは時間がかかり，ほとんどの研究はアニーリング法やタブー探索法など発見的解法に集中している．この OP の拡張の一つとして時刻依存オリエンテーリング問題 (TD-OP) が挙げられる．OP と同様 TD-OP も NP 困難 [2] である．多くの既存研究では時間的整合性（FIFO 準拠）を仮定するが，現実の交通渋滞をリアルに模倣するにはこれだけでは不十分である．また，観光経路問題に対する研究は様々あるが，複数の観光スポットの訪問順序や組合せによる効用の変化などは，著者が知る限りまだ見つかっていない．本論文は TD-OP を拡張し，複数日におけるスポット間の訪問順序関係を用いた，新たな観光モデルを考える．

## 2　時刻依存移動速度モデル

本研究で使用する速度モデルにおいて，スポット間の車両の速度，つまり移動時間は，走行する時間帯と辺カテゴリーに依存する．これらは事前に決められているものであり，移動中に振り分けられるわけではない．この速度モデルには，朝と夕方の混雑のピークを反映した 2 つの不等間隔の時間帯と，通常の交通時間帯が組み込まれている．これに加えて，以下の 5 つの辺カテゴリーも組み込まれている．

- 常に混雑している：終日を通して交通量が多い繁華街を表す.
- 朝にピーク：生活圏から都心に通じる道路を表し，一般的に朝の交通量が多い.
- 2 回のピーク：高速道路に近い道路を表し，朝，夕方の 2 回ピークが訪れる.
- 夕方にピーク：都心から生活圏に通じる道路を表し，一般的に夕方の交通量が多い.
- めったに混まない：あまり交通量のない道路を表し，通常よりも早く移動できる.

速度 $v_{c,k}$ は，時間帯 $k$ $(1 \leq k \leq K)$ と辺カテゴリー $c$ $(1 \leq c \leq C)$ の組み合わせごとに定義される．$K$=4,$C$=5 で使用される速度表を表 1 に示す．表示された速度行列は，先入先出の法則にしたがう．つまり，先に出発した車両は目的地に早く到着することを保証する．さらに，日中の経路のみを構築したいので，旅行者は午前 8 時以降に出発し，午後 9 時までに宿もしくは終了地点に戻ってくると仮定する.

**表 1**　使用速度表

| | 朝のピーク | 通常 | 夕方のピーク | 通常 |
|---|---|---|---|---|
| 時間帯 | 7am-9am | 9am-5pm | 5pm-7pm | 7pm-9pm |
| アークカテゴリー ($c$) | | | | |
| 1. いつも忙しい | 0.5 | 0.81 | 0.5 | 0.81 |
| 2. 朝のピーク | 0.5 | 0.7 | 1 | 1.5 |
| 3. 朝, 夕 2 つのピーク | 0.5 | 1.5 | 0.5 | 1.5 |
| 4. 夕のピーク | 1 | 1.5 | 0.5 | 0.7 |
| 5. めったに混まない | 1.5 | 1.5 | 1.5 | 1.5 |

## 3　スポット組合せ順序モデル

実際の観光経路において，訪問する観光スポット間には順序や組合せなどによる相乗効果があると考えられ

る．本研究で提案する観光モデルは，観光スポットの組合せによる効果をスポット $i \to j$ の順序で訪問したときの評価値 $h_{i,j}$ で表す．

- $i$ から $j$ への訪問順序に $-$ の関係．：$h_{i,j} < 0$
- $i$ から $j$ への訪問順序に $+$ の関係．：$h_{i,j} > 0$
- $i$，$j$ の組合せに $+$ の関係．：$h_{i,j} > 0 \cap h_{j,i} > 0$
- $i$，$j$ の多くとも一方しか訪問しない．：$h_{i,j} < 0 \cap h_{j,i} < 0$

また，観光経路を考える際，飲食店や宿などいくつかのカテゴリーに属する観光スポットは，訪問する個数が厳密に定められている．この制約を本研究で提案する順序関係値で表すことができる．例として，あるカテゴリーに属するスポットが $m$ ヵ所存在し，そのうちちょうど $k$ ヵ所訪れなければならないとする．このカテゴリーに属するスポット $a$ 利益を本来の利益より $\alpha$ 大きく設定し，$a, b$ 相互に $-\beta$ の順序関係値があるとする．ちょうど $k$ ヵ所のスポットを訪問するときの評価値が最大となるためには，前後の $k-1$，$k+1$ ヵ所のスポットを訪問するときの評価値よりも高くなるようにすればよい．つまり，以下の 2 式を同時に満たす．

$$k\alpha - \frac{(k-1)k}{2}\beta > (k-1)\alpha - \frac{(k-1)(k-2)}{2}\beta$$
$$k\alpha - \frac{(k-1)k}{2}\beta > (k+1)\alpha - \frac{k(k+1)}{2}\beta$$

上記の 2 式を共に満たす $\alpha$，$\beta$ の関係式は以下のように表すことができる．

$$(k-1)\beta < \alpha < k\beta$$

この結果から，あるカテゴリーに属するスポット数 $m$ は直接関係ないことがわかる．本研究ではこのように制約条件に対する順序関係値を与えることにより，制約条件を直接扱う代わりに，目的関数値を最大化することで自動的に制約条件を満たす解を発見するモデルを扱う．また，本研究モデルの頂点が属するカテゴリーとして，宿カテゴリー，飲食店カテゴリー，その他のカテゴリーの 3 種類を用いる．

## 4　メタ戦略

メタ戦略とは現実的な計算時間で最適解を求めることが困難な組合せ最適化問題に対して，ヒューリスティックな手法に基づいて短時間で高精度の近似解を求める近似解法の枠組みである．本研究では以下のメタ戦略アルゴリズムを構築した．

### 4.1　局所探索法

局所探索法とは, 現在の解の近傍を調べ, 改善解があれば移動するアルゴリズムである. 本研究では挿入近傍, 置換近傍，交換近傍をの 3 種類を作成した．

#### 4.1.1　挿入近傍

挿入近傍とは，現在の解に未訪問の頂点を挿入することで得られる近傍である．未訪問の頂点の挿入を試みるたびに，解全体の時間のかかる評価を行わないようにするため，現在の各頂点について，解が実行不可能になる前にその頂点への訪問を延期できる最大時間 (max_shift) を記録しておく．これにより，効率的な実行可能性の判定と解の更新が可能になる．頂点 $y$（未訪問頂点）を $x$ と $z$ の間に挿入しようとするとき，時刻依存経路問題における余分移動時間 $\Delta tt$ は以下の式と等しくなる．

$$\Delta tt = (tt_{x,y} + st_y) + tt_{y,z} - tt_{x,z}$$

頂点 $y$ は，余分移動時間が頂点 $z$ の max_shift 以下の場合にのみ，現在の解に挿入できる．

$$\Delta tt \leq \text{max\_shift}_z$$

その後，$y$ 以降の頂点の移動時間を再計算し，以下の式に基づいて max_shift 変数を再計算する．

$$\begin{aligned}\text{max\_shift}_i = {} & \text{previous\_max\_shift}_i \\ & + (\text{previous\_}ta_i - \text{new\_}ta_i)\end{aligned} \tag{1}$$

#### 4.1.2　置換近傍

置換近傍は，現在の解の頂点を未訪問頂点と置き換えることで得られる近傍である．ここでは，$y$ を置換候補頂点，$z$ を置換される現在の解の頂点，$x$ と $w$ をそれぞれ $z$ の前頂点と次頂点と定義する．以下の不等式が成り立つとき，置換が可能である．

$$\Delta tt = ((tt_{x,y} + st_y) + tt_{y,w}) - ((tt_{x,z} + st_z) + tt_{z,w})$$
$$\Delta tt \leq \text{max\_shift}_w$$

置換可能な頂点に対して解の差分計算が行われる．解が改善するならば，置換が実行される．その後，(1) 式と同様に移動時間 max_shift 変数の再計算を行う．

#### 4.1.3　交換近傍

本問題は移動経路の順序によっても解の総利益に影響が出るため，現在の解の 2 つの頂点を交換しようとする交換近傍が行われる．構成した経路の頂点を交換することで解が改善される可能性がある．交換する頂点を $y$, $z$ とし，$y$ の前頂点を $k$，次頂点を $l$ とし，$z$ の次頂点を $n$ とする．$y, z$ の 2 点がともに同じカテゴリーに属す

ならば，日をまたいで交換可能であり，一方が宿カテゴリー以外であれば異なるカテゴリーであっても同日内の他頂点と交換可能である．もし 2 頂点が交換可能であれば，一時的に 2 点を交換し，解の差分計算が行われる．解の評価値が改善されたならばそれが実行解かどうかを試す．

$$\begin{aligned}\Delta score \quad &= \left(h_{y,z} + \sum_{i=l}^{n}(h_{y,i} + h_{i,z})\right) \\ &\quad - \left(h_{z,y} + \sum_{i=l}^{n}(h_{z,i} + h_{i,y})\right) > 0\end{aligned}$$

選択された 2 頂点が日をまたぐ交換であれば，交換する 2 頂点は 2 回の置換近傍で行うことができる．日をまたがない 2 頂点の交換の場合は $y$ の到着時刻の再計算が必要になるが，以下の式を満たせば交換可能である．

$$\text{new\_}ta_y < \text{max\_shift}_n + \text{previous\_}ta_z$$

### 4.2 反復局所探索法

反復局所探索法 (Iterated Local Serach，ILS) とは，過去の探索で得られた解にランダムな変形 (本研究では Kick と呼ぶ) を加えたものを初期解として局所探索法を繰り返し適用する手法である. 局所探索法とは近傍解集合から良好な近傍解への移動を繰り返し，改善解がなくなった場合，局所解に至ったとして探索を終了するアルゴリズムである．詳細は Algorithm1 に示す．

#### 4.2.1 初期解構築

初期解は主に貪欲法で作成される．具体的には，未訪問頂点と終了点との移動時間の制限を考慮しながら，訪問することでスコアが上昇する見込みのある頂点を挿入できなくなるまで，解の終端に頂点を追加する．

### 4.3 ACS(Ant Colony System)

本節では，TD-OP における解法 [3] にも適用された ACS について述べる．ACS は蟻コロニー最適化 (ACO) に代表される構成的メタ戦略であり，複数の解を独立に構成し，フェロモンと呼ばれる記憶構造を用いて移動した辺をマークし，他への誘引を可能にする．詳細は Algorithm2 に示す．

**Algorithm 1** Iterated Local Search

$k \leftarrow 1$
$NoImprovement \leftarrow 0$
$Max\_k \leftarrow 2 * L$
$Max\_Itr \leftarrow 10000$
Initialized Solution $sol$
$sol_{best} \leftarrow sol$
**while** $NoImprovement < Max\_Itr$ **do**
  **while** Until the solution is no longer improved **do**
    $sol \leftarrow$Local Search($sol$)
  **end while**
  **if** $F(sol) \geq F(sol_{best})$ **then**
    $sol_{best} \leftarrow sol$
    $k \leftarrow 1$
    $NoImprovement \leftarrow 0$
  **else**
    $NoImprovement \leftarrow NoImprovement + 1$
  **end if**
  $Kick(sol, k)$
  $k \leftarrow k + 1$
  **if** $k = Max\_k$ **then**
    $k \leftarrow 1$
  **end if**
**end while**
**return** $x_{best}$

**Algorithm 2** Ant Colony System -input parameters: $\alpha, \beta, \rho,$ max_ants, $\tau_{init} N_{n_i}^{max}$

$s_{ib} \leftarrow 0,\ s_{gb} \leftarrow 0,\ N_{ni} \leftarrow 0,$ iteration $\leftarrow 0$
**while** iteration $< N_c$ **do**
  Initialize $\tau(\tau_{\text{init}})$
  **for** $i \leftarrow 1$ to max_ants **do**
    $sol_i \leftarrow$Construct solution $(\tau, \alpha, \beta, \rho)$
    local pheromone update($sol_i, \tau$)
    Caliculate max_shift($sol_i$)
    Local solution($sol_i$)
  **end for**
  $sol_{ib} \leftarrow \max(F(sol_1), F(sol_2), \ldots, F(sol_{\text{max\_ants}}))$
  **if** $sol_{ib} > sol_{gb}$ **then**
    $sol_{gb} \leftarrow sol_{ib}$
    $N_{ni} \leftarrow 0$
  **else**
    $N_{ni} \leftarrow N_{ni} + 1$
  **end if**
  Global pheromone update($\tau, sol_{ib}, N_{ni}, N_{ni}^{max}$)
  iteration$\leftarrow$ iteration +1
**end while**

## 5 数値実験

### 5.1 実験結果

今回は作成した 2 種類の提案手法にて，実験を行った．既存の TD-OP データセット [4] のうち 100 頂点を超えるものに任意の頂点に宿を割り当て，確率 $p$ に従い順序関係値を割り当てたものを適用した．各頂点の利益は 1〜30 の乱数の中から選択され，宿カテゴリーに属する頂点以外の滞在時間は 30 分, 1 時間のどちらかをランダムに割り当てる．出発地点，終了地点の利益は 0 に等しく，観光日数 $L$，1 日の最大観光時間 $t_{max}$ の値，順序関係値の付与確率 $p$ の値を変更したもので行った．また，宿や飲食店カテゴリーに属する頂点の利益やカテゴリー間の順序関係値にパラメータ (3 章の $\alpha, \beta$) が加算

されるが，これらは最終的に再計算されており，表に記載のスコアはパラメータの値を除いたものである．

**表 2**　ランダムに作成された初期解による結果 ($L = 2$)

| | $score$ | 頂点利益 | 順序関係値の和 | 平均実行時間 |
|---|---|---|---|---|
| データセット | $max/avg$ | $max/avg$ | $max/avg$ | $avg_time(sec)$ |
| p4_9 | 718/685.6 | 497/467.8 | 258/217.8 | 30.2173 |
| p4_10 | 793/752.7 | 503/487.3 | 297/265.4 | 37.9703 |
| p4_11 | 850/819.5 | 542/514.1 | 336/305.4 | 42.1003 |
| p4_12 | 941/886.3 | 619/549.9 | 367/336.4 | 50.7581 |
| p4_13 | 1009/967 | 638/586.2 | 418/380.8 | 57.1227 |

**表 3**　貪欲法によって作成された初期解による結果 ($L = 2$)

| | $score$ | 頂点利益 | 順序関係値の和 | 平均実行時間 |
|---|---|---|---|---|
| データセット | $max/avg$ | $max/avg$ | $max/avg$ | $avg_time(sec)$ |
| p4_9 | 706/678.3 | 488/455.7 | 247/222.6 | 34.5858 |
| p4_10 | 782/737.6 | 513/481.6 | 300/256 | 43.5055 |
| p4_11 | 841/807.6 | 565/530.8 | 293/276.8 | 46.2635 |
| p4_12 | 927/894.6 | 584/558 | 374/336.6 | 52.8927 |
| p4_13 | 907/885.6 | 598/548.1 | 361/337.5 | 43.7036 |

**表 4**　順序関係値による変化 (ACS)($L = 2$)

| | $score$ | 頂点利益 | 順序関係値の和 | 平均実行時間 |
|---|---|---|---|---|
| データセット | $max/avg$ | $max/avg$ | $max/avg$ | $avg_time(sec)$ |
| p4_20% | 969/944.6 | 627/569.1 | 418/375.5 | 35.7013 |
| p4_30% | 1006/980.3 | 595/560.2 | 467/420.1 | 34.3633 |
| p4_40% | 1179/1152 | 559/520.1 | 676/631.9 | 37.148 |
| p4_50% | 1270/1245.4 | 600/558.3 | 758/687.1 | 43.8141 |

**表 5**　順序関係値による変化 (ILS)($L = 2$)

| | $score$ | 頂点利益 | 順序関係値の和 | 平均実行時間 |
|---|---|---|---|---|
| データセット | $max/avg$ | $max/avg$ | $max/avg$ | $avg_time(sec)$ |
| p4_20% | 864/834.8 | 562/528.6 | 343/306.2 | 45.0012 |
| p4_30% | 880/836.9 | 546/494.3 | 380/342.6 | 51.5656 |
| p4_40% | 1021/988 | 506/475.5 | 546/512.5 | 46.0604 |
| p4_50% | 1113/1069.4 | 547/490 | 668/579.4 | 43.8141 |

## 6　考察

メタ戦略において大事な要素となるはずの初期解だが，今回の実装した結果にあまり関係ないことが，表 2，表 3 から読み取れる．また，表 4，表 5 より，どちらの手法においても順序関係値の和の合計は増加傾向にあることから，たとえ順序関係値に負の効果があったとしても経路全体の総利益における影響は大きなものだと考えられる．計算時間に着目すると，どちらも 30 秒を超えてしまうことが多く，今回提案した max_shift による評価法も挿入，置換可能場所は高速に評価できるが，解の差分計算で時間がかかってしまうのが原因だと考えられる．

次に，ACS と ILS だが，今回の実験結果では ACS の方が全体の面での性能がいいことが読み取れる．これは時刻依存経路問題のようなグラフが動的に変化する場合，多様な解をより多く見つけることのできる蟻コロニーの特徴が起因しているようにも思える．今回実装した ILS は，Kick として繰り返し解から頂点を削除する方法を採用した．例えば，局所解から 1 頂点のみを削除し，挿入近傍によってまた同じ経路を再構築してしまうことがある．これもまた局所解に陥りやすい原因になったのだろうと考える．

## 7　まとめ

本研究では，既存の TD-OP を拡張し，2 頂点間における訪問順序関係を用いた新たなモデルを作成した．このモデルを実際の観光経路作成に適用することで，実際の観光スポットの訪問順序や組合せを考慮した，複数日の観光経路を作成することができると考える．また，既存手法に加え新たに近傍を作成し，2 種類のアルゴリズムを比較することで，性能や発見的手法による効果をより詳しく比較することができた．

## 8　今後の課題

本モデルは異なる 2 地点間の訪問順序における関係値を利用したが，3 頂点以上の訪問順序における複雑な設定もパラメータの値を適宜調整することで表現することができると考える．アルゴリズムの面に関しても今回は ACS と反復局所探索法を実装したが，タブー探索法などほかにも様々な発見的解法が挙げられる．今後はそれらについても検討する価値があるかもしれない．

## 参考文献


[1] Golden, B., Levy, L., and Vohra, R. The orienteering problem. *Naval Research Logistics*, 34, 307 – 318 (1987).

[2] Fomin, F., and Lingas, A. Approximation algorithms for time-dependent orienteering. *Information Processing Letters*, 83, 57 – 62 (2002).

[3] Verbeeck, C., Sörensen, K., Aghezzaf, E. H., and Vansteenwegen, P., A fast solution method for the time-dependent orienteering problem, *European Journal of Operational Research*, 236(2), 419 – 432 (2014).

[4] The Orienteering Problem: Test Instances -The Time-Dependent Orienteering Problem (online), available from https://www.mech.kuleuven.be/en/cib/op (参照 2024-02-01)