情報処理学会論文誌　データベース　Vol.9　No.2　34–45 (June 2016)

観光ルート推薦のための効率的な制約条件

新妻 弘崇[1,a)]　新井 晃平[2,b)]　太田 学[1,c)]

受付日 2015年12月20日, 採録日 2016年2月26日

概要：観光ルート推薦の問題は，なんらかの目的関数を最大化する最適化問題として定式化することができる．たとえば旅行者の興味や満足度をスコアとして表した関数の値を最大化する観光ルートを求める問題として定式化することができる．この最適なルートを求める問題は Travelling Salesman Problem（TSP）と類似した整数計画問題として記述することができる．しかし TSP と同様の定式化をすると不適切な解が頻繁に計算される問題が生じる．たとえば一筆書きにならない途切れたルートが頻繁に計算される問題が生じる．この問題を解決する制約条件を本研究では提案する．提案する制約条件には計算時間が大幅に増大する問題がある．そこで計算時間を短縮する制約条件をさらに追加する手法を提案する．提案手法の有効性を示すために 2 種類の目的関数を使った実験を行った．1 つ目は訪問する観光スポットの人気度の総和，2 つ目は旅行者をマルコフ遷移モデルで表した場合の対数尤度を目的関数として設定し実験を行った．どちらの目的関数を使用した実験でも，提案手法によって計算される観光ルートは，Greedy な解法で計算したルートと比べておおよそ 2 倍の目的関数の値を与えることを確認した．

キーワード：観光ルート推薦，Twitter

Efficient Constraints for Tour Recommendation

Hirotaka Niitsuma[1,a)]　Kohei Arai[2,b)]　Manabu Ohta[1,c)]

Received: December 20, 2015, Accepted: February 26, 2016

Abstract: Tour recommendation can be formulated as finding an optimized tour that maximizes a given objective function. For example, tour recommendation can be formulated as maximizing the score that represents the degree of a user's interest as an objective function. The optimization problem can be solved as the integer programming problem using the similar formulation as used in Travelling Salesman Problem (TSP). However, the integer programming problem gives invalid solutions frequently. For example, the optimized tour frequently includes interrupted routes. This research proposes additional constraints for the integer programming problem that can resolve this problem. However, the integer programming problem including the proposed additional constraint requires huge computational time to solve. This research also proposes other additional constraints that can resolve this computational time problem. The proposed formulations are evaluated using two objective functions. The first objective function is the score which represents the popularity of each spot. The second objective function is the likelihood of a Markov chain model of tourists. We show the formulation using these additional constraints can compute better solutions that have almost a twice better score than a greedy method using these two objective functions.

Keywords: travel route recommendation, Twitter

1. はじめに

観光に関する情報をブログ，Twitter，Flickr などから抽出する様々な研究 [1], [2], [3], [4], [5] が行われている．この抽出された情報は，旅行者の好みに合ったお土産などの情報を推薦するために使うことができる [1], [2]．本研究では抽出された情報を使って，旅行者の好みに合った観光

[1] 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University, Okayama 700–8530, Japan
[2] 株式会社両備システムソリューションズ
Ryobi System Solutions, Okayama 702–8006, Japan
[a)] niitsuma@suri.cs.okayama-u.ac.jp
[b)] pv6x88nw@s.okayama-u.ac.jp
[c)] ohta@de.cs.okayama-u.ac.jp

ルートを推薦する手法を提案する．

観光ルートを推薦する手法としては，観光スポット間のリンクに注目する方法とノードに注目する方法の 2 つがある．リンクに注目する方法とは，次に訪問する観光スポットへのリンク，つまり旅行者がスポットを訪れる順序に注目した方法である．たとえば，多くの旅行者が実際に訪れた順序を正解データと見なし，出発地点と目的地点を与えれば，多くの旅行者が満足するであろうルートを推薦する手法 [3], [6] である．ノードに注目する方法とは，それぞれの観光スポットつまりノードごとの良さを満足度などのスコアとして数値化し，そのスコアの合計が最大となるような観光ルートを求める手法である [4], [5]．どちらのルート探索問題も Travelling Salesman Problem（TSP）[7] と類似した組合せ最適化問題となる [4]．具体的には，与えられた時間や費用などのコストの範囲内で，満足度などの目的関数を最大化する観光ルートを探索する問題として定式化できる．本研究ではこの問題を，観光ルート推薦と呼ぶこととする．このような目的関数を最大化する最適化問題として観光ルート推薦を扱っている研究としては，文献 [3], [4], [5], [8], [9], [10] などがある．この最適化問題は，TSP に使われる記法と同じ記法で記述することができ，また解法も TSP に使われる方法を利用できる．

倉島ら [3] は，旅行者をマルコフ遷移モデルでモデル化し，遷移確率が最大となるルートを探索する手法を提案した．具体的には，旅行者が観光に使える時間の範囲内で，尤度という目的関数を最大化する問題として観光ルート推薦を定式化した．しかし倉島らの研究では，目的関数が最大となる旅行ルートの探索に遺伝的アルゴリズムなどの局所最適解を回避できるアルゴリズムを使っていないため，必ずしも最適なルートが計算されるわけではない．

Lee ら [10] は，目的関数として電気自動車の充電待ち時間を最小化する観光ルート推薦手法を提案した．この問題の目的関数はバッテリーの状態によって変化する階段関数を含む非線形関数となる．そのため Lee らの定式化は線形な最適化手法を利用するのが困難な問題となっている．この研究では，非線形な問題を扱える最適化手法である遺伝的アルゴリズムを使って最適なルートを求めている．

Lim [4] は他の旅行者がたくさん訪れる場所ほど人気が高いと考え，旅行者の満足度をこの人気の高さを使って表現した．この満足度を表す関数は，線形関数となるように定式化された．そして，この線形関数である満足度が最大となるルートを探索する問題を整数計画問題 [7] として定式化した．

TSP や，観光ルート推薦の問題などの最適なルートを求める問題を整数計画問題として解く場合には，巡回路除去制約 [7], [11] と呼ばれる制約条件を常に満たさなければならない．具体的にはループが存在したり，すべてがつながっておらず一筆書きになっていなかったりする不適切なルートが生成されるのを避ける必要がある．この条件は問題の大きさを N とすると，おおよそ 2^N のオーダーの数の制約条件式となる．2^N のオーダーの問題は扱うのは困難であるため，これを N^2 のオーダーに近似する Miller-Tucker-Zemlin formulation（MTZ）[11] という制約条件式が TSP では使われている．Lim [4] は，この近似を観光ルート推薦の問題にそのまま適用した．しかし，観光ルート推薦の問題にこの近似をそのまま適用すると，頻繁に不適切なルートが生成される．この現象については 4 章の実験で示す．

2. 関連研究

ここでは旅行者のモデル化方法に注目して関連研究について述べる．なお，本研究で観光スポットまたはスポットと呼ぶ対象は，他の研究で Point of Interest（POI）[4] と呼ばれる対象やランドマーク [3] と呼ばれる対象と同じ意味のものである．

前述の倉島らの研究 [3] では，Flickr から収集した旅行者のジオタグ情報を，観光スポット間の遷移を表すマルコフモデルとして表現することで旅行者をモデル化している．旅行者ごとの好みの違いはマルコフモデルの隠れ変数として扱われ，この隠れ変数と旅行履歴の関係は，文章中の単語の出現確率を表すモデルである Probabilistic Latent Semantic Analysis [12] を使ってモデル化されている．具体的には，単語が観光スポット，文章が旅行者の観光スポットの訪問履歴に対応すると考えたモデルが使われている．そして，このモデルのパラメータを文章中の単語の出現確率を計算する場合と同様に Expectation Maximization アルゴリズム [12] で推定する．

位置情報付きツイートを使って倉島ら [3] と同様の処理を行う研究もある．中嶋ら [2] は位置情報付きツイートと Foursquare や Instagram のサービス，および旅行者のツイートに頻繁に現れる特徴語を用いてツイートを収集し，それぞれのユーザのタイムラインから観光ルートを抽出した．ツイートから旅行者の情報を抽出する場合には，位置情報だけでなく関連するツイートの会話などのテキスト情報も収集することができる．このテキスト情報を使うと倉島らのモデル [3] では名前のついていない隠れ変数として扱われていた旅行者ごとの好みの違いをより明確に表現できるようになる．具体的には収集したツイートを，手がかり語や品詞の特徴から「食事」,「景観」,「行動」の 3 つのカテゴリに分類した．このカテゴリ分類を使うことで，名前のついていない隠れ変数ではなく「食事」,「景観」,「行動」という明確に名前と意味をともなう分類を使った旅行者のモデルを作ることができる．

しかし中嶋らの手法 [2] では，Twitter ユーザのタイムラインからのみ観光スポットを収集しているため，得られる観光スポットの数が少ないという問題があった．そこで

我々 [5] は Yahoo! 知恵袋の「地域，旅行，お出かけ」カテゴリに出現する出現頻度の高い地名を観光スポットとする手法を提案した．ここで地名とは観光したい目的地の周辺の施設一覧を Google Places API を用いて取得した施設名である．たとえば京都駅周辺の施設名一覧を Google Places API から取得すると，「清水寺」，「伏見稲荷大社」などの周辺施設名一覧が得られる．この施設名一覧には公衆トイレなどの観光スポットの名称としては一見不適切な施設名が含まれることがあるため，この研究では Yahoo! 知恵袋の「地域，旅行，お出かけ」カテゴリに出現する出現頻度の少ないを施設名を除外した．この方法により，適当な地名のみを得ることができる．この地名を手がかり語とすることで，観光に関連のあるツイートを収集することができる．本研究では，この手法で収集した観光ルートデータを実験に用いる．

旅行者ごとの好みの違いを，その旅行者の観光ルートの中の代表点で表現する手法も提案されている．前述の Lee らの研究 [10] では，旅行者の観光ルートの中で代表的な観光スポットを数個与え，これを出発点として遺伝的アルゴリズムを使って観光ルートの突然変異や進化をさせる手法が提案されている．この方法は，観光ルートそのものを遺伝的アルゴリズムの遺伝子コードとして扱うことができるため，特に遺伝的アルゴリズムに適した特徴表現方法となっている．

3. 巡回路除去制約の近似

観光ルート推薦の問題，すなわち与えられたコストの範囲内で，旅行者の満足度を最大化する観光ルートを探索する問題は TSP と同様に次のように定式化できる．

$$\underset{x_{11},x_{12},...,x_{NN}}{\text{maximize}} \quad f(x_{11},x_{12},...,x_{NN}) \tag{1}$$

$$\text{subject to} \quad \sum_{j=2}^{N} x_{1j} = 1 \tag{2}$$

$$x_{i1} = 0, \qquad i = 1,...,N \tag{3}$$

$$\sum_{j=2}^{N} x_{ij} \leq 1, \quad i = 2,...,N-1 \tag{4}$$

$$\sum_{i=1}^{N-1} x_{ij} \leq 1, \quad j = 2,...,N-1 \tag{5}$$

$$\sum_{i=1}^{N-1} x_{iN} = 1 \tag{6}$$

$$x_{Ni} = 0, \qquad i = 1,...,N \tag{7}$$

$$x_{ii} = 0, \qquad i = 1,...,N \tag{8}$$

$$g(x_{11},x_{12},...,x_{NN}) \leq g_{max} \tag{9}$$

$$x_{ij} \in \{0,1\}$$

ここで x_{ij} は 0，1 の 2 値をとる変数であり，スポット i からスポット j への移動があるときに 1 となり，それ以外の場合は 0 となる．つまり i から j への移動を表す変数である．観光ルートの最初の出発スポットは $i=1$ 番目のスポット，最終到着スポットは $i=N$ 番目のスポットに固定する．このとき，出発スポット $i=1$ からは，必ず $j=2,3...,N$ までのスポットのいずれかに移動があるため $x_{12},x_{13},...,x_{1N}$ のどれか 1 つが必ず 1 となる．この条件を記述したのが制約条件式 (2) である．同様の制約条件は，最終到着スポットである N 番目のスポットについても成り立つ．最終到着スポット N には必ず $i=1,2,...,N-1$ のスポットのいずれかからの移動があるため $x_{1N},x_{2N},...,x_{N-1,N}$ のどれか 1 つが必ず 1 となる．この条件を記述したのが制約条件式 (6) である．$i=1,N$ 番目以外のスポット $i=2,3,...,N-1$ については必ずしも通る必要がないため，同様の制約条件式の値は 1 か 0 になる．これは不等式制約条件として記述できる．この不等式制約条件式を記述したのが制約条件式 (4) と (5) である．f は最大化したい目的関数，すなわち旅行者の満足度である．g は制約を受けるコストを表す．たとえば，観光に使える時間や費用である．以下では g をコスト関数と呼ぶこととする．g_{max} はコストの上限である．

この最適化問題はコストの制約条件式 (9) をとりのぞき，不等式制約条件式 (4) と (5) を等式制約条件式に変更すると TSP と等価となる．この等式制約条件式への変更により，すべてのスポットを 1 回は訪問するルートが計算されるようになる．一方で，式 (4) と (5) を不等式制約条件式とすると，すべてのスポットを必ずしも訪問しなくても良いようになる．ただし，この不等式制約条件式への変更は，すべてがつながっておらず一筆書きになっていない不適切なルートが生成されてしまう問題を生じる．具体的には i へ向かうルートがある場合，つまり $\sum_{j=1}^{N-1} x_{ji} = 1$ のときに，逆に i から出発するルートがない場合，つまり $\sum_{j=2}^{N} x_{ij} = 0$ となってしまう場合を生じてしまう．この問題の解決には後で導入する制約条件式 (15) を制約条件に追加する必要がある．

3.1 Miller-Tucker-Zemlin formulation

前述の観光ルート推薦の定式化には巡回路除去制約 [7], [11] が常に満たされるわけではないという問題がある．具体的にはループを含むルートが生成されてしまう問題がある．この問題を避けるには次の制約条件を追加する必要がある [7].

$$\sum_{i,j \in S} x_{ij} \leq |\mathcal{S}| - 1, \qquad \mathcal{S} \subseteq \{2,3,...,N\} \tag{10}$$

ここで $\mathcal{S}$ は 1 番目以外のスポットの集合の部分集合である．ループが存在しないならば，スポットの任意の部分集合 $\mathcal{S}$ 内のリンクの数は，その部分集合の大きさ $|\mathcal{S}|-1$ 以下に必ずなる．この条件を記述したのが上記の制約条件で

ある．この制約条件はすべての部分集合 $\mathcal{S}$ について考慮する必要があり，合計でおおよそ $O(2^N)$ 個の制約条件を生じる．このため，現実的には扱うのが困難な制約条件である．そこで以下の近似がよく使われる．

$$q_1 = 1$$
$$q_i - q_j + 1 \leq (N-1)(1-x_{ij}), \quad i,j \in \{2,3,...,N\} \tag{11}$$

この制約条件は Miller-Tucker-Zemlin formulation (MTZ) [11] と呼ばれる．ここで q_i はスポット i を何番目に訪問するかの順番を表す変数である．たとえば $i=3$ のスポットを 5 番目に訪問する場合は $q_3 = 5$ となる．この q_i は Sequential Formulation [13] と呼ばれる．この制約条件の意味は，x_{ij} が 0 と 1 になる場合に次式のように場合分けすると理解できる．

$$q_i - q_j \leq \begin{cases} N-2 & x_{ij}=0 \\ -1 & x_{ij}=1 \end{cases} \tag{12}$$

$x_{ij}=1$ となる場合には j は i の後に訪問するので $q_j \geq q_i+1$ となる．それ以外の場合は q_i と q_j は最大値 N から最小値 2 の間の任意の値をとってよい．つまり $q_i - q_j$ は $2-N$ から $N-2$ の間の値となる．この制約条件の上限側のみを記述しているのが式 (12) の $x_{ij}=0$ の場合である．

MTZ を使った近似は TSP に対しては良い近似を与える．しかし観光ルート推薦の問題では MTZ のみでは不適切なルートを除外するのには不十分である．ここで**不適切なルート**とは次のいずれかの条件が成立するルートのことである．

- ループの含まれるルート
- 一筆書きにならない途切れたルート．具体的には $x_{ij}=1$ のときに $x_{jk}=1$ となるような k が存在しない場合．

MTZ のみを追加の制約条件として前述の最適化問題の解を計算すると不適切なルートが頻繁に計算され，解を計算することができず深刻な問題となる．この現象については 4 章の実験で示す．

本研究では，このように TSP の単純な拡張だけでは不適切なルートが計算されてしまう問題を解決する制約条件を提案する．

3.2　提案する制約条件

MTZ はすべてのスポットを 1 回は訪問するという TSP の条件が暗黙の前提となっている．しかし観光ルート推薦の問題では，候補スポットをすべて訪問する必要はない．候補スポットが N あり，このうち $M < N$ スポットしか訪問しない場合を考えよう．この場合，Sequential Formulation は訪問するスポットについては $q_i \leq M$ となり定まるが，訪問しないスポットは何番目になってもよい．実際に MTZ のみで観光ルート推薦の問題を解くと q_i は重複を含む様々な値をとる．たとえば 4 番目に訪問するスポットが複数出現する $q_5 = q_2 = 4$ となるような場合が現れる．そこで訪問しないスポットは $q_i > M$ となるようにする制約条件を導入する．こうすることで q_i は重複した値をとりにくくなる．また重複が少なくなった結果，解を探索するべき空間に無意味な解が含まれる可能性も減る．たとえば $q_5 = q_2 = 4$ は $M=3$ の場合に生じる可能性があるが，$M=3$ の場合には q_5 はどんな値なっても良い．しかし $q_5 = 2,3,...$ とすべての可能な解を探索する必要がないよう，適当な範囲内に固定されるようにするのである．

まず初めに以下の制約条件を導入する．

$$\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij} = q_N - 1 \tag{13}$$

すべての x_{ij} を合計すると，(訪問するスポットの数 -1) だけ 1 があるため，(訪問するスポットの数 -1) と同じ数になる．スポット N は最後に訪問するスポットであるため，q_N は訪問するスポットの数 M と同じ値になる．これを等式制約条件として導入したのがこの制約条件である．

次に q_i の和に注目する．数列 $(q_2, q_3, ..., q_N)$ は理想的には数列 $(2,3,...,N)$ に何回か適当な置換を行った数列と同じとなる．つまりその和 $q_2+q_3+...+q_N = 2+3+...+N$ を計算すると

$$\sum_{i=2}^{N} q_i = \frac{N(N+1)}{2} - 1 \tag{14}$$

となる．この制約条件を導入することで前述の，同じ順番のスポットが複数生じる状態，たとえば $q_5 = q_2 = 4$ のような状態が生じにくくなる．なお訪問するスポット数 M は最適化問題の解が計算されるまで分からないため，問題を解く前に決めることはできない．そのため $q_2, q_3, ..., q_M$ の和を問題を解く前に計算することはできない．解が計算される前であっても計算でき，なおかつ $M=N$ の場合であっても成り立つ安全側な $q_2, q_3, ..., q_N$ の和が計算されている点に注意する．

実際に MTZ のみで観光ルート推薦問題を解くと，一筆書きにならない解が頻繁に現れる．具体的には i へ向かうルートがあっても，逆に i から出発するルートがない場合が頻繁に現れる．この問題は次の制約条件を導入することで解決することができる．

$$\sum_{j=1}^{N} x_{ji} = \sum_{j=2}^{N} x_{ij}, \quad i = 2,...,N-1 \tag{15}$$

ここで $\sum_{j=2}^{N} x_{ij}$ は i から出発するリンクの数であり，i がルートに含まれる場合には 1 となり，そうでない場合は 0 となる．$\sum_{j=1}^{N} x_{ji}$ は i へ到着するリンクの数でありこれも

同じ値となる．この 2 つが等しくなる等式制約条件を導入するのが式 (15) である．

次に式 (13) から $q_N = M$ となる点に注目し，スポット i がルートに含まれるかどうか，つまり $q_i < q_N = M$ となるかを判定する制約条件を導入する．スポット i がルートに含まれるかどうか判定するには，i へ向かうリンクの集合 $\{x_{ji}|j = 1, 2, ..., N\}$ と i から出発するリンクの集合 $\{x_{ij}|j = 1, 2, ..., N\}$ の和集合である次の集合 $\ominus_i$ に注目するとよい．

$$\ominus_i = \{x_{ij}|j = 1, 2, ..., N\} \cup \{x_{ji}|j = 1, 2, ..., N\} \quad (16)$$

具体的には

$$\psi_i = \sum_{x \in \ominus_i} x = \begin{cases} 2 & \text{スポット } i \text{ がルートに含まれる} \\ 0 & \text{スポット } i \text{ がルートに含まれない} \end{cases} \quad (17)$$

となる．このスポット i がルートに含まれる場合と含まれない場合について明確に場合分けする制約条件を以下のように導入することができる．

$$q_i \geq \begin{cases} q_N + 1 & \psi_i = 0 \\ q_N + 2 - N & \psi_i = 2 \end{cases} \quad (18)$$

$$q_i \leq \begin{cases} q_N + N - 2 & \psi_i = 0 \\ q_N - 1 & \psi_i = 2 \end{cases} \quad (19)$$

$$(i = 2, 3, ..., N - 1)$$

式 (18) はスポット i がルートに含まれない場合には $q_i \geq q_N + 1$ となり，それ以外の場合には $2 \leq q_i \leq N$ の間のどの値をとってもよいと仮定して $q_i - q_N$ がとりうる最小の値 $2 - N$ と比較して $q_i - q_N \geq 2 - N$ が満たされるという制約条件を表している．

式 (19) はスポット i がルート含まれる場合には $q_i \leq q_N - 1$ となり，それ以外の場合には $2 \leq q_i \leq N$ の間のどの値をとってもよいと仮定して $q_i - q_N$ がとりうる最大の値 $N - 2$ と比較して $q_i - q_N \leq N - 2$ が満たされるというという制約条件を表している．

この制約条件は次のように変形すると線形の整数計画問題で扱うことができる形になる．

$$q_i \geq q_N + \frac{1 - N}{2}\psi_i + 1 \quad (20)$$

$$q_i \leq q_N + \frac{1 - N}{2}\psi_i + N - 2 \quad (21)$$

4. 制約条件の比較実験

次に 3.2 節で導入した制約条件式を，乱数を使って生成した人工的な問題を使って比較する．以下では目的関数 f とコスト関数 g が次の線形な関数である場合を考える．

$$f(x_{11}, x_{12}, ..., x_{NN}) = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ij}x_{ij} \quad (22)$$

$$g(x_{11}, x_{12}, ..., x_{NN}) = \sum_{i=1}^{N}\sum_{j=1}^{N} g_{ij}x_{ij} < g_{max} \quad (23)$$

このとき f_{ij}, g_{ij} を 0 から 1 の範囲の乱数に設定して問題を 100 個生成する．この 100 個の問題を使って比較評価を行う．なお g_{max} は 1 から N の乱数とし，問題の大きさは $N = 7$ とした．このランダムな問題 100 個を使って制約条件の違いによる影響を調べたのが**表 1** である．実験は 5 つの制約条件 (15)，(13)，(14)，(20)，(21) の組合せ $2^5 = 32$ 通りすべてについて行った．この 32 通りの制約条件の組合せについて，ランダムな問題 100 個を整数計画ソルバを使って解く．整数計画のソルバとしては CPLEX 12.6.2.0 [*1] を使った．計算機環境は Intel Core i7-2600K 3.40 GHz の CPU を使った Gentoo Linux のシステムを使用した．表 1 では 100 個の問題を解くのにかかった合計計算時間と，その計算で使用した制約条件を各制約条件の列に • のマークをつけて示している．また 100 個の問題の中で不適切な解が計算された問題の数 K も K/100 として表に示した．32 通りの制約条件の組合せの中で，100 個の問題すべてについて探索した中で最も良い目的関数値となる解を計算できた制約条件の組合せは 16 通りあった．この探索した中で最も良い目的関数値となる解を本研究では**最良解**と呼ぶこととする．100 個の問題すべてについて

表 1　ランダムな問題 100 個を使った制約条件の評価
Table 1　Evaluation of the proposed constraints on randomly generated 100 problems.

制約条件式 (15)	(13)	(14)	(20)	(21)	不適切な解の数	計算時間 (秒)
•	•		•	•	0/100	1.97571
•			•	•	0/100	1.84688
•					0/100	1.84566
•			•		0/100	1.83840
•	•			•	0/100	1.79710
•		•		•	0/100	1.74248
•				•	0/100	1.66593
•	•		•		0/100	1.65752
•	•	•	•	•	0/100	1.65204
•	•				0/100	1.63231
•	•	•			0/100	1.63012
•		•	•		0/100	1.60313
•		•	•	•	0/100	1.53527
•		•			0/100	1.52602
•	•	•	•		0/100	1.51717
•	•	•		•	0/100	1.45281
MTZ のみ					10/100	1.43398

[*1] http://www-01.ibm.com/software/commerce/optimization/cplex-cp-optimizer/

最良解を計算した制約条件の組合せ 16 通りと，制約条件に MTZ のみを使った場合のみを表には示した．100 個の問題のうち 1 つでも最良解が計算できなかった制約条件の組合せ 15 通りについては省略した．ただし MTZ のみを使用した場合も最良解を計算できなかったが，省略せず表に示している．

表の一番下が，どの追加の制約条件も使わなかった場合，つまり MTZ のみを使った場合である．MTZ のみの場合は，不適切な解が 100 個の問題中 10 個の問題について計算されている．この結果から MTZ のみの場合は 10 回中 1 回は不適切な解を計算してしまうため，意味のある解を計算できないことが分かる．ただし MTZ のみの場合が最も計算は高速である．Lim の研究 [4] では巡回路除去制約の問題を解決する制約条件として MTZ のみが使われていた．Lim の手法は，適用する問題によっては解が計算できない可能性があることを，この実験結果は示している．

表の結果では最良解が計算できる場合すべてに制約条件 (15) が含まれている．このことから良い解を計算するには制約条件 (15) が重要となることが分かる．しかし制約条件 (15) だけで十分なわけではなく，他の制約条件と組み合わせることで計算時間が 2 割程度，短縮されることがある．

計算時間が短縮される理由については，次のように考察できる．整数計画問題は分岐限定法を使って解かれることが多い [7], [14]．本研究で利用した整数計画ソルバーである CPLEX も分岐限定法を使った解法を採用している[*2]．分岐限定法では探索する枝を減らせれば計算時間も短縮できる．本研究では，このような探索する枝を減らせる，つまり解を探索する空間が小さくなるような制約条件を提案している．たとえば，制約条件式 (14) がないならば $(q_1, q_2, q_3, ...) = (1, 1, 1, ...)$ といった無意味な解についても良い解であるかどうかを確認する計算が必要となる．制約条件式 (14) があれば，このような無意味な解候補が最初から除外され計算時間が短縮される．このように整数計画問題に依存した特定の制約条件を追加することで分岐限定法の探索する解を減らす方法は，分岐カット [14] と呼ばれるテクニックの 1 つである．本研究で提案する計算時間を短縮するための制約条件は，観光ルート推薦の問題に特化した分岐カットであると見なせる．文献 [14] では，分岐カットによる計算時間の減少について理論的評価をするのは困難であり実験的評価が重要であると述べられている．提案手法についても理論的に計算時間が減少することが保証されているわけではないといえる．また複数の分岐カットを組み合わせる問題についても理論的評価をするのは困難であるといえる．そこで本研究では実験的に計算時間の減少について示している．

[*2] http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.6.2/ilog.odms.cplex.help/refcppcplex/html/branch.html

表 2　TSPLIB のベンチマーク問題を使った評価
Table 2　Evaluation of the proposed constraints on TSPLIB problems.

ベンチマーク名	MTZ のみ	(13) (14)
gr24	36 秒	22 秒
bays29	6 秒	4 秒
bayg29	4 秒	4 秒

実験的な評価の結果，次のことがいえる．提案する制約条件の中には同時に使うとかえって計算時間が増加してしまうか，または最良解を計算できない場合が存在する．特に制約条件 (20) と (21) は同時に使うと計算時間が増加してしまうかまたは，最良解を計算できない傾向がある．最良解を計算できる組合せで最も計算時間が短いのは (15), (13), (14), (21) の制約条件の組合せと (15), (13), (14), (20) の組合せである．

この計算時間が短縮される効果について確認するために TSP についても同様の実験を行った．TSP の場合は，すべてのスポットを必ず通るため，制約条件 (15), (20), (21) には意味がない．しかし制約条件 (13), (14) には TSP においても探索空間を減らす効果がある．この制約条件を使って計算時間が短縮できることを示したのが**表 2** である．実験には TSP のベンチマーク集である TSPLIB [15] の大きさが 30 以下のベンチマークを 3 つを使用した．3 つのベンチマークのうち 2 つで計算時間がおおよそ 2/3 となっているのが分かる．1 つのベンチマークでは変化がなかったが，これは提案する制約条件では探索空間を減らす効果が得られなかったためである．

5. 実データを使った実験

ここでは実データを用いた実験について説明する．具体的には 2 章で説明した文献 [5] の研究の方法で Twitter から収集した観光ルートデータを用いた実験を行う．まず初めに旅行者のモデルについて説明し，次にそのモデルを使って目的関数 f とコスト関数 g を定め，最後に最適化計算の出力を示す．

5.1 旅行者のモデル

文献 [5] の方法で Twitter から収集した旅行者の旅行履歴を集計することで，倉島ら [3] の確率モデルと同様のモデルを作ることができる．具体的には次の確率を旅行者の旅行履歴から計算する．

$$P(j|i,u) = \sum_c P(j|c,i)P(c|u) \tag{24}$$

ここで $P(j|i,u)$ は旅行者 u が，スポット i から j に移動する確率である．この確率が最大になる旅行ルート $i_1, i_2, ..., i_M$, を求めることで，旅行者 u が最も満足する観光ルートを求めることができる．$P(c|u)$ は旅行者 u が，カ

テゴリ $c \in \{$ 食事, 景観, 行動 $\}$ のどれかに興味をもつ確率，$P(j|c,i)$ はカテゴリ c に興味をもった旅行者が，スポット i から j に移動する確率である．$P(c|u)$ は旅行者 u の旅行履歴からそれぞれのカテゴリに属する履歴の比率を求めれば容易に計算することができる．$P(j|c,i)$ は以下で計算する．

$$P(j|c,i) = P(j|i)\frac{P(c|j,i)}{P(c|i)} \tag{25}$$

ここで $P(j|i)$ はすべての旅行者についてのスポット i から j に移動する確率であり，旅行履歴から比率として計算することができる．$P(c|i)$ は直前にスポット i にいた旅行者が次のスポットとしてカテゴリ c の対象に興味を持つ確率である．本研究では直前のスポットは次のスポットでの興味に影響しないと仮定し，次の近似を導入する．

$$P(c|i) \approx P(c) \tag{26}$$

$P(c)$ はすべての旅行履歴における，それぞれのカテゴリの比率である．$P(c|j,i)$ は i から j に移動した旅行者がカテゴリ c に興味を持つ確率である．この確率についても，直前のスポットは次のスポットでの行動に影響しないと仮定し，次の近似を導入する．

$$P(c|j,i) \approx P(c|j) \tag{27}$$

$P(c|j)$ はスポット j がカテゴリ c に所属する度合いを表す．これも旅行履歴の比率から計算する．以上で式 (24) に示した確率 $P(j|i,u)$ が計算でき，$P(j|i,u)$ が最大となる旅行ルート $i_1, i_2, ..., i_N$, を 3 章で述べた整数計画問題を使って求めることができる．本研究では倉島ら [3] の方法と同様に対数尤度が最大となるルートを求める問題と考え目的関数 f を次のように設定し，3 章の最適化画問題を解く．

$$f_{kura} = \sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij} \log \frac{P(j|i,u)}{\beta} \tag{28}$$

β は倉島ら [3] の研究で unigram rescaling と呼ばれているパラメータである．本研究では式 (28) の対数尤度が負の値をとらないように調節した．

同様に旅行者の旅行履歴を集計することで，Lim [4] の研究で使われている満足度も次のように計算することができる．

$$f_{Lim} = \sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij}(P(i) + \sum_{c} P(i|c)P(c|u)) \tag{29}$$

ここで $P(i)$ はスポット i の人気度を表し，すべての旅行履歴の中でスポット i が出現する割合を集計することで計算する．$P(c|u)$ は旅行者 u がカテゴリ c に興味を持つ度合いである．

f_{kura} と f_{Lim} はそれぞれ観光スポット間のリンクに注目した目的関数と，ノードに注目した目的関数である．具体的には，f_{kura} は他の旅行者が多く通ったリンクを通るルートが選択されるほど大きな値となる目的関数である．f_{Lim} は他の旅行者が多く通ったノードつまり観光スポットが選択されるほど大きな値となる目的関数である．この 2 つの目的関数について評価実験を行う．

コスト関数 g は合計旅行時間の上限の制約を与えるものとする．具体的には次式のようにする．

$$g = \sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij}(d_{ij} + v_i) \tag{30}$$

ここで

$$d_{ij} = \text{Googlemap API で移動手段に自動車を使う設定にした場合の } i \text{ から } j \text{ への移動時間} \tag{31}$$

$$v_i = \text{スポット } i \text{ の平均滞在時間} \tag{32}$$

である．スポット i の平均滞在時間 v_i は，旅行履歴からすべての旅行者がスポット i にどれだけの時間滞在したかを集計することで計算する．ただし本研究の実験では平均滞在時間 v_i は安全のために実際の滞在時間よりも長めの時間になる計算をしている．たとえば旅行者が観光スポット A に到着したときにツイートをしてから A に 80 分滞在し，次にスポット B に 60 分滞在したがツイートを 1 回もしなかった場合を考える．このような履歴に存在しないスポット B に訪問したことを推測するのは困難であるため，スポット A に 80+60=140 分滞在したと見なす計算を本実験では行っている．その結果として滞在時間は実際の滞在時間よりも長めの滞在時間が計算される．

5.2 Greedy な解法

文献 [5] の研究では Greedy な解法を使って観光ルート推薦を行う方法を提案した．Algorithm 1 にそのアルゴリズムを示す．ただし本研究では文献 [5] の研究とは旅行者の特徴を表現する方法が異なるため，そのための修正を加えたアルゴリズムとなっている．このアルゴリズムの詳細は次のようなものである．まず 1 行目で候補となるエッジの集合 EdgeCandiates を与える．特に断りがない場合はすべてのスポット間をつなぐすべてのエッジ $\{(i,j)|i,j = 1,2,...,N, i \neq j\}$ が EdgeCandiates に与えられているとする．次に初期ルートとして出発スポットと最終スポットを直接つなぐルート $(i_{start}, i_{goal}) = (1, N)$ を暫定解として route に代入する．以下では $\mathcal{T}$ を暫定解の集合とする．route は暫定解の集合 $\mathcal{T}$ の中で最も良い解を表すとする．3 行目では，暫定解の集合 $\mathcal{T}$ として初期ルートが 1 つだけ含まれる集合を設定する．5 行目では，暫定解の集合 $\mathcal{T}$ の中で最も良い解を route に代入する．6 行目では，その時点での最良解である route= $(i_1, i_2, ...i_j, i_{j+1}, ...)$ に EdgeCandiates のエッジ (i_j, i_k) または (i_k, i_{j+1}) を使って挿入

可能なノード i_k を挿入したルート $(i_1, i_2, ...i_j, i_k, i_{j+1}, ...)$ を生成する．そのような route にノードを 1 つ挿入したルートは複数できることがある点に注意する．たとえば $(i_3, i_k) \in$ EdgeCandiates であり，$(i_{k'}, i_2) \in$ EdgeCandiates であるならば，$\mathcal{T} = \{(i_1, i_2, i_3, i_k, i_4, ...), (i_1, i_{k'}, i_2, ...), ...\}$ といった集合が 6 行目の処理では生成される．なおこのとき $(i_k, i_4) \in$ EdgeCandiates，$(i_1, i_{k'}) \in$ EdgeCandiates である必要はない．7 行目では $\mathcal{T}$ の中でループのあるルートを除外している．8 行目では $\mathcal{T}$ の中でコスト制約を満たさないルートを除外している．9 行目は route にノードを 1 つ挿入したルートの集合の中で妥当な解が存在しなかった場合に処理を終了することを示している．そうでない場合は 5 行目にもどり route にノードを挿入する処理を繰り返す．こうしてこれ以上ノードが挿入できなくなるまで route を改良していくことで良いルートを求める．

Algorithm 1 Greedy solution

1: EdgeCandiates $= \{e_1 = (i_1, j_1), e_2 = (i_2, j_2), ...\}$
2: route $= (i_{start}, i_{goal})$
3: $\mathcal{T} = \{\text{route}\}$
4: **repeat**
5: route $= \arg\max_{r \in \mathcal{T}} f(r)$
6: $\mathcal{T} = \{r | r = (i_1, i_2, ...i_j, i_k, i_{j+1}, ...) \wedge$ route $= (i_1, i_2, ...i_j, i_{j+1}, ...) \wedge$ $(\ (i_k, i_{j+1}) \in$ EdgeCandiates $\vee$ $(i_j, i_k) \in$ EdgeCandiates $)\ \}$
7: $\mathcal{T} = \{r | \neg\text{loop}(r), r \in \mathcal{T}\}$
8: $\mathcal{T} = \{r | g(r) \leq g_{max}, r \in \mathcal{T}\}$
9: **until** $\mathcal{T} = \emptyset$
10: **return** route

このアルゴリズムは整数計画問題として問題を解いた場合に計算される解が不適切な解であるとき，つまり $x_{ij} = 1$ となる j が存在するが $x_{jk} = 1$ となる k が存在しない解が計算されたときに，この不適切な解を不適切でない解に修正するのにも使うことができる．5.4 節の実験では不適切な解はこの方法で解を修正して，他の不適切な解が計算されない方法との比較を行う．具体的には，候補エッジの集合を EdgeCandiates=$\{(i,j)|x_{ij} = 1\}$ と設定することで，不適切でない解に修正された解を求めることができる．

5.3　実験データ

実験データとして [5] の方法で収集した京都の「清水寺」，「伏見稲荷大社」の 2 つの観光スポットの周辺観光ルートデータを用いる．この方法を使うと京都には約 150 の観光スポットが抽出される．次にこの 150 カ所の観光スポットを通る旅行者のツイートを収集する．具体的には 2014 年 12 月 21 日から 2014 年 12 月 23 日までの間に投稿されたツイートのうち，この 150 カ所の観光スポットを通る旅行者のツイートを約 700 ツイート収集した．

しかしこの 150 カ所の観光スポットには数回しか旅行者が訪問しないスポットが多く含まれていたため，本研究では訪問回数が上位 4 割の約 60 カ所の観光スポットのみを利用する．この上位 4 割のスポットのみを訪問する長さ 4 以上の観光ルートは 19 ルートあり，この 19 ルートを使って評価実験を行う．**図 1** にその 19 ルートの 1 つを示す．旅行ルートの可視化には Google Maps API[*3] を使用した．ただし移動時間には式 (31) の移動に自動車を使ったと仮定して Google Maps API から取得した移動時間を使用した．

		f_{Lim}	滞在時間（分）	次への移動時間 (分)	合計時間 (分)
A	天龍寺	0.26	82.55	2.86	
B	祇王寺	0.27	40.72	14.30	
C	等持院	0.19	119.85	4.00	
D	龍安寺	0.27	148.64	29.00	
E	茶寮都路里		138.61		
	合計	0.99	530.38	50.16	580.54

図 1　旅行者が車で移動したと仮定した場合に通るルート
Fig. 1　Travel route assuming the sampled user drove a car.

5.4　評価実験

目的関数として式 (28) の $f = f_{kura}$ を使う場合と式 (29) の $f = f_{Lim}$ を使う場合についてそれぞれの目的関数の値を最大にする旅行ルートを求める実験を行う．具体的には次の手順で実験を行う．

(i) 旅行者のツイートのタイムラインから観光スポットの訪問順番 $(i_1, i_2, i_3, ..., i_M)$ を抽出する．
(ii) 訪問したスポットから旅行者のカテゴリごとの好みの度合い $P(c|u) = \sum_{k=1}^{M} P(c|i_k)P(i_k)$ を計算する．
(iii) $P(c|u)$ から目的関数 $f(x_{11}, x_{12}, ..., x_{NN})$ を計算する．
(iv) 自動車で移動したと仮定して旅行にかかった時間 $g_{max} = \sum_{k=1}^{M} v_{i_k} + \sum_{k=1}^{M-1} d_{i_k i_{k+1}}$ を計算する．ここで v_{i_k} は式 (32)，$d_{i_k i_{k+1}}$ は式 (31) から計算する
(v) 出発スポットを i_1，最終到着スポットを i_M として $g(x_{11}, x_{12}, ..., x_{NN}) \leq g_{max}$ の範囲内で目的関数 $f(x_{11}, x_{12}, ..., x_{NN})$ が最大となるルートを，設定した最適化手法で計算する．

最適化手法としては 3 章の整数計画問題として最適ルートを求める手法と Algorithm 1 に示す Greedy な解法を使う．また不適切な解が求まった場合には Algorithm 1 を使って

[*3] https://developers.google.com/maps/

表 3　京都の実データ 19 ルートを使った評価実験
Table 3　Evaluation of the proposed constraints using 19 real tours in Kyoto.

目的関数	制約条件式 (15)	(13)	(14)	(20)	(21)	不適切な解の数	計算時間（秒）	目的関数 f の値
f_{kura}	•	•	•		•	0/19	10,835.08	31.391
	•		•	•	•	0/19	10,851.26	31.497
	•	•	•	•	•	0/19	10,848.95	31.497
	•		•		•	0/19	10,835.27	31.497
	•	•			•	0/19	10,830.79	31.497
	•	•		•		0/19	10,829.71	31.497
	•			•		0/19	10,829.50	31.497
	•	•	•	•		0/19	10,826.68	31.497
	•				•	0/19	10,826.23	31.497
	•	•	•			0/19	10,813.20	31.497
	•	•				0/19	10,810.89	31.497
	•		•			0/19	10,809.26	31.497
	•					0/19	10,809.21	31.497
	MTZ のみ					18/19	6.31	7.013
	Greedy					0/19	0.00	17.894
f_{Lim}	•	•	•	•	•	0/19	10,923.37	2.095
	•		•	•	•	0/19	20,604.68	2.120
	•				•	0/19	16,948.91	2.120
	•	•	•		•	0/19	10,906.92	2.120
	•		•		•	0/19	19,191.23	2.131
	•					0/19	17,146.39	2.131
	•	•			•	0/19	10,910.32	2.131
	•	•	•	•		0/19	10,879.48	2.131
	•	•				0/19	10,837.05	2.131
	•		•			0/19	17,800.25	2.131
	•			•		0/19	17,332.88	2.131
	•	•		•		0/19	10,879.43	2.131
	•	•	•			0/19	10,846.75	2.131
	MTZ のみ					18/19	23.25	0.733
	Greedy					0/19	0.00	1.070

不適切でない解に修正する．

実験結果を**表 3** に示す．実験結果は表 1 と同様の方法で記述した．ただし表 1 と異なり目的関数 f の値も追記してある．目的関数の値が最良解よりも大幅に劣っている制約条件の組合せは省略してある．Algorithm 1 のみを使った場合は Greedy と表した．

制約条件として MTZ のみを使った整数計画問題は非常に高速に計算をすることができるが，ほとんどの解は不適切な解となっている．Algorithm 1 を使って不適切でない解に修正しても目的関数の値，すなわち満足度スコアが他の制約条件を追加した場合の半分程度の値にしかならない．Algorithm 1 のみを使った場合も同様である．

制約条件 (15) を追加すると最も良い解が計算できることが分かる．さらに他の制約条件を追加した場合の結果は，Lim [4] の提案する満足度スコア f_{Lim} を目的関数とした場合と，マルコフ遷移のモデルすなわち目的関数として f_{Kura} を使った場合で異なる．

f_{Lim} を使った場合は制約条件 (15) を追加すると最も良い解が計算できるが，制約条件 (15) のみの場合は 5 時間近い計算時間がかかっている．しかし他の制約条件を追加することで 5 時間の計算時間が 3 時間程度まで短くなる場合がある．また計算時間が短くなっても計算される目的関数の値はほぼ同じである．

f_{Kura} を使った場合は，制約条件 (15) 以外の制約条件を追加しても，計算時間も目的関数の値もほぼすべて同じ値である．目的関数 f_{Kura} はマルコフ遷移モデルを使って構成されているため，スポットの訪問順番の情報が目的関数の中に埋め込まれている．そのため f_{Lim} の場合よりも訪問するスポットの順番の探索は簡単な問題となる．このため探索空間を減らす制約条件を追加しても計算時間が減らないと考えられる．

目的関数を f_{Lim} とした場合と表 1 のランダムな問題についての結果を合わせて考察すると，計算時間を短くできる制約条件の組合せは次の条件を満たす制約条件であると

		f_{Lim}	滞在時間（分）	次への移動時間 (分)	合計時間 (分)
A	天龍寺	0.26	82.55	23.47	
B	渉成園	0.27	65.95	23.53	
C	野宮神社	0.27	147.19	25.29	
D	茶寮都路里		138.61		
	合計	0.80	434.30	72.29	506.59

図 2　MTZ のみ
Fig. 2　MTZ only.

いえる．

- 制約条件 (13) を必ず使う．
- 制約条件 (14)，(20)，(21) のうち 1 つ以上を必ず使う．
- ただし制約条件 (20) と (21) を同時に使わない．

しかし目的関数に f_{Kura} を使った場合には，制約条件 (14) と (21) を同時に使うと目的関数の値が最良解よりも悪くなる場合が生じた．これらの結果を総合すると良い解を計算でき，なおかつ計算時間を短くできる制約条件の組合せは，以下の 3 つの組合せであると結論できる．

- 制約条件 (15)，(13)，(14) の組合せ
- 制約条件 (15)，(13)，(20) の組合せ
- 制約条件 (15)，(13)，(21) の組合せ

目的関数を f_{Kura} とした場合は f_{Lim} とした場合よりも全体的に計算時間が小さくなる傾向がみられる．これは事前に頻繁に通るリンクの情報を設定した方が問題が簡単であることを示している．

次に計算された旅行ルートの例について考察する．**図 2**，**図 3**，**図 4** にそれぞれの最適化手法で計算されたルートの例を示す．旅行ルートの可視化には図 1 と同様に Google Maps API を使用した．これらのルートは図 1 の旅行者が通ったルートをベースとして目的関数に f_{Lim} を使った場合に，それぞれの最適化手法で計算したルートである．すなわち図 1 と同じ出発スポット，最終到着スポット，全旅行時間を設定して，最適なルートを計算した結果が図 2 から図 4 である．またスポットごとの目的関数 f_{Lim} の値も示した．図 2 は制約条件に MTZ のみを使って整数計画ソルバを使って求めたルートである．図 3 は制約条件 (15) を追加して整数計画ソルバを使って求めたルートである．なお，表 3 で示した同じ最良解が計算されるが計算時間を短くするための制約条件がさらに追加されている場合については，計算されるルートも同じであるため省略する．図 4

		f_{Lim}	滞在時間（分）	次への移動時間 (分)	合計時間 (分)
A	天龍寺	0.26	82.55	2.86	
B	祇王寺	0.27	40.72	28.11	
C	長楽館	0.17	23.42	26.20	
D	実光院	0.26	26.31	42.99	
E	渉成園	0.27	65.95	2.30	
F	豊国神社	0.30	90.39	3.70	
G	茶寮都路里		138.61		
	合計	1.52	467.95	106.16	574.11

図 3　提案手法
Fig. 3　Proposed method.

		f_{Lim}	滞在時間（分）	次への移動時間 (分)	合計時間 (分)
A	天龍寺	0.26	82.55	10.99	
B	仁和寺	0.29	169.77	11.92	
C	祇王寺	0.27	40.72	28.00	
D	豊国神社	0.30	90.39	3.70	
E	茶寮都路里		138.61		
	合計	1.11	522.04	54.62	576.66

図 4　Greedy
Fig. 4　Greedy.

は Algorithm 1 の Geedy な解法で求めたルートである．

最適化が十分でない図 2 と図 4 のルートは移動効率が悪いのがわかる．たとえば図 4 では A，C，B，D，E の順

番で訪問した方が移動時間が少なくなるはずであるが，計算されたルートは余分な移動時間が必要なルートとなっている．

提案手法を使って計算した図 3 のルートは訪問するスポットの数が Greedy な解法や MTZ のみを使った場合と比べておよそ 1.5 倍である．訪問するスポットは，京都駅付近の渋滞しやすいスポットを避けて車で移動しやすい駅から離れたスポットが多く選択されている．移動しやすいスポットを選択することで訪問するスポットの数を 1.5 倍にしたとも考察できる．この結果として目的関数の値も 1.5 倍近い値となっている．提案手法は非常に効率的なルートを計算するのがわかる．

6. おわりに

観光ルート推薦の問題は Travelling Salesman Problem（TSP）と類似した問題となる．しかし TSP と異なりすべてのスポットを通る必要がないため，TSP に使われる Miller-Tucker-Zemlin formulation（MTZ）をそのまま使うことはできない．そこでこの問題を解決できる制約条件 (15) を本研究では提案した．この制約条件によって，適切でなおかつより良い解を求めることができるが計算時間が増加する問題が生じた．この計算時間の問題の解決方法として，制約条件をさらに追加することで解の探索範囲を狭め，計算時間を少なくする方法を提案した．実験の結果，制約条件 (13)，(14) の組合せ，制約条件 (13)，(20) の組合せ，または制約条件 (13)，(21) の組合せの 3 通りの組合せに計算時間を減らす効果があることが確認できた．

今後の課題として，滞在時間を実際より長く推定してしまう問題がある．またツイートから旅行者の交通手段を推定し，実際の交通手段による移動時間に基づく推薦手法の評価を検討している．

参考文献

[1] 難波英嗣：観光情報の自動編纂, 知能と情報, Vol.26, No.1, pp.9–15 (2014).

[2] 中嶋勇人，新妻弘崇，太田　学：位置情報付きツイートを利用した観光ルート推薦，情報処理学会研究報告，データベース・システム研究会報告，Vol.2013-DBS-158, No.28, pp.1–6 (2013).

[3] 倉島　健，岩田具治，入江　豪，藤村　考：写真共有サイトにおけるジオタグ情報を利用したトラベルルート推薦，電子情報通信学会技術研究報告，Vol.109, No.450, pp.55–60 (2010).

[4] Lim, K.H.: Recommending Tours and Places-of-Interest Based on User Interests from Geo-tagged Photos, *Proc. 2015 ACM SIGMOD on PhD Symposium*, SIGMOD '15 PhD Symposium, New York, NY, USA, pp.33–38, ACM (2015).

[5] 新井晃平，新妻弘崇，太田　学：Twitter を利用した観光ルート推薦の一手法，第 7 回データ工学と情報マネジメントに関するフォーラム G7-6，pp.1–8 (2015).

[6] 松本絢香，中村健二，小柳　茂：制限時間内の観光ルート推薦システム，情報処理学会第 77 回全国大会講演論文集，Vol.2015, No.1, pp.589–591 (2015).

[7] 藤江哲也：整数計画法による定式化入門，オペレーションズ・リサーチ：経営の科学，Vol.57, No.4, pp.190–197 (2012).

[8] 野中さつき：東京ディズニーシーにおける最適巡回路，南山大学情報理工学部数理情報学部卒業論文 (2013).

[9] Mathias, M., Moussa, A., Zhou, F., Torres-Moreno, J., Poli, M., Josselin, D., El-Bèze, M., Linhares, A.C. and Rigat, F.: Optimisation Using Natural Language Processing: Personalized Tour Recommendation for Museums, *CoRR*, Vol.abs/1501.01252 (2015).

[10] Lee, J., Kim, S.-W. and Park, G.-L.: A Tour Recommendation Service for Electric Vehicles Based on a Hybrid Orienteering Model, *Proc. 28th Annual ACM Symposium on Applied Computing*, pp.1652–1654, ACM (2013).

[11] Miller, C.E., Tucker, A.W. and Zemlin, R.A.: Integer Programming Formulation of Traveling Salesman Problems, *J. ACM*, Vol.7, No.4, pp.326–329 (1960).

[12] Hofmann, T.: Probabilistic Latent Semantic Analysis, *Proc. 15th Conference on Uncertainty in Artificial Intelligence*, pp.289–296 (1999).

[13] Orman, A.J. and Williams, P.: A Survey of Different Integer Programming Formulations of the Travelling Salesman Problem, *Advances in computational management science*, Vol.9, No.9, pp.93–108 (2006).

[14] 藤江哲也：整数計画問題に対する分枝カット法とカットの理論（OR 研究の最前線），オペレーションズ・リサーチ：経営の科学，Vol.48, No.12, pp.935–940 (2003).

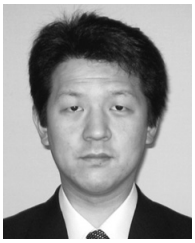
[15] Reinelt, G.: TSPLIB—A Traveling Salesman Problem Library, *ORSA Journal on Computing*, Vol.3, No.4, pp.376–384 (1991).

新妻 弘崇　（正会員）

1993 年大阪大学工学部応用物理学科卒業．1995 年同大学大学院工学研究科応用物理学専攻博士前期課程修了．1999 年奈良先端科学技術大学院大学情報科学研究科情報システム学専攻博士後期課程修了．博士（工学）．2007 年岡山大学大学院自然科学研究科助教．機械学習ならびに Web 情報検索の研究に従事．電子情報通信学会，IEEE，ACM 各会員．

新井 晃平

2015 年岡山大学工学部情報系学科卒業．同年株式会社両備システムソリューションズ入社．大学在学中に Twitter のツイートを利用した観光ルート推薦に関する研究に従事．



太田 学　（正会員）

1994 年東京大学工学部電気工学科卒業．1999 年同大学大学院工学系研究科電気工学専攻博士課程修了．博士（工学）．東京都立大学大学院工学研究科助手，岡山大学大学院自然科学研究科助教授，准教授を経て，2013 年より岡山大学大学院自然科学研究科教授．Web 情報検索ならびに電子図書館の研究に従事．電子情報通信学会，日本データベース学会，IEEE 各会員．

（担当編集委員　土方 嘉徳）