

はじめに

巡回路除去制約の
近似

MTZ

提案する制約条件

制約条件の比較
実験

実データを使った
実験

まとめ

観光ルート推薦のための効率的な制約条件

中島 健希

富山県立大学
u120031@st.pu-toyama.ac.jp

December 22, 2023

背景

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.

目的

観光ルート推薦の問題は TSP と類似した問題となる. しかし, TSP と異なり全てのスポットを通る必要がないため, TSP に使われる MTZ をそのまま使うと不適切なルートが計算されてしまう. そこでこの問題を解決することを目的とした制約条件を提案する.

概要

TSP や、観光ルート推薦の問題を整数計画問題として解くときには、巡回路除去制約を常に満たさなければならない。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。

$$\begin{array}{ll} \text{maximize} & f(x_{11}, x_{12}, \dots, x_{NN}) \end{array} \quad (1)$$

$$\begin{array}{ll} \text{subject to} & \sum_{j=2}^N x_{1j} = 1 \end{array} \quad (2)$$

$$x_{i1} = 0, \quad i = 1, \dots, N \quad (3)$$

$$\sum_{j=2}^N x_{ij} \leq 1, \quad i = 2, \dots, N-1 \quad (4)$$

$$\sum_{i=1}^{N-1} x_{ij} \leq 1, \quad j = 2, \dots, N-1 \quad (5)$$

$$\sum_{i=1}^{N-1} x_{iN} = 1 \quad (6)$$

$$x_{Ni} = 0, \quad i = 1, \dots, N \quad (7)$$

$$x_{ii} = 0, \quad i = 1, \dots, N \quad (8)$$

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

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

図 1: 観光ルート探索問題の定式化

概要

前述の観光ルート推薦の定式化には巡回路除去制約が常に満たされるわけではない. 具体的にはループを含むルートが生成されしまう問題がある. 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. この制約条件はすべての部分集合について考慮する必要があるため, 扱うのが困難. そのため, 以下の近似が使われる.

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, \quad S \subseteq \{2, 3, \dots, N\} \quad (10)$$

図 2: 追加した制約条件

$$\begin{aligned} q_1 &= 1 \\ q_i - q_j + 1 &\leq (N - 1)(1 - x_{ij}), \quad i, j \in \{2, 3, \dots, N\} \end{aligned} \quad (11)$$

図 3: MTZ

概要

候補スポットが N であり, このうち M スポットしか訪問しない場合を考える. この場合, 訪問するスポットは q_i は M 以下となり定まるが, 訪問しないスポットは何番目になっても良い. そこで訪問しないスポットは q_i が M より大きくなるようにする制約条件を導入する.

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

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

$$\sum_{j=1}^N x_{ji} = \sum_{j=2}^N x_{ij}, \quad i = 2, \dots, N-1 \quad (15)$$

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

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

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

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

図 4: 追加した制約条件

はじめに

巡回路除去制約の
近似

MTZ

提案する制約条件

制約条件の比較
実験

実データを使った
実験

まとめ

概要

次に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. 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_{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.

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

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

図 5: 追加した制約条件

制約条件の比較実験

7/15

結果

100 個の問題の全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み合わせは 16 通りあった. 実験的な評価として, 制約条件 (20) と (21) は同時に使うと時間が増加するか, 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み合わせで最も計算時間が短いのは (15),(13),(14),(21) の組み合わせ.

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

Table 1: ランダムな問題 100 個を使った制約条件の評価

はじめに

巡回路除去制約の
近似

MTZ

提案する制約条件

制約条件の比較
実験

実データを使った
実験

まとめ

結果

計算時間短縮効果の確認のために,TSP にも同様の実験を行った. 制約条件 (13),(14) には TSP において探索空間を減らす効果がある.3 つのうち 2 つで計算時間が 2/3 となっている.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.

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

Table 2: TSPLIB のベンチマーク問題を使った評価

旅行者のモデル

確率モデルを旅行者の旅行履歴から計算する. 式 (24). 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 を仮定する. 同様に旅行者の旅行履歴から, 満足度も計算することができる. この2つの目的関数について評価実験をする. 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.

$$P(j|i, u) = \sum_c P(j|c, i) P(c|u) \quad (24)$$

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

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

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

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

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

Greedy な解法を使って観光ルート推薦を行う方法を示す. このアルゴリズムは不適切な解を不適切ではない解に修正するのに使える.

Algorithm 1 Greedy solution

```

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

```

図 7: Algorithm 1

実験データ

実験データとして京都の「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. 約 150 ヶ所の観光スポットが抽出される. この 150 ヶ所の観光スポットをとる旅行者のツイートを約 700 ツイート収集した. しかし数回しか旅行者が訪問しないスポットが多くあったため, 訪問回数が上位 4 割のスポットのみ利用する.



地図データ ©2015 Google, ZENRIN

	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 8: 旅行者が車で移動したと仮定した場合に通るルート

評価実験

目的関数として式 (28) を使う場合と式 (29) を使う場合で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。

実験内容

- (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 $(i_1, i_2, i_3, \dots, i_M)$ 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 $f(x_{11}, x_{12}, \dots, x_{NN})$ 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 $g(x_{11}, x_{12}, \dots, x_{NN}) \leq g_{max}$ の範囲内で目的関数 $f(x_{11}, x_{12}, \dots, x_{NN})$ が最大となるルートを、設定した最適化手法で計算する。

実験結果

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み合わせは、以下の3つであると結論できる。

- ・ 制約条件 (15),(13),(14) の組み合わせ
- ・ 制約条件 (15),(13),(20) の組み合わせ
- ・ 制約条件 (15),(13),(21) の組み合わせ

目的関数	制約条件式					不適切な解の数	計算時間 (秒)	目的関数の値
	(15)	(13)	(14)	(20)	(21)			f の値
f_{area}	*	*	*	*	*	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	7.613
	Greedy					0/19	0.00	17.894
f_{time}	*	*	*	*	*	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

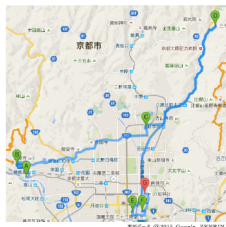
Table 3: 京都の実データ 19 ルートを使った評価実験

実データを使った実験

14/15

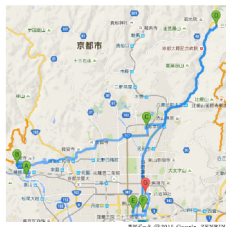
旅行ルートの考察

提案手法を使って計算した場合, 訪問するスポットの数が他に比べて 1.5 倍. これは渋滞しやすいスポットを避け, 車で移動しやすいスポットが多く選択されているためである. 移動しやすいスポットを選ぶことで数を 1.5 倍にしたと考察できる. 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.



	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 龍王寺	0.27	40.72	28.11	
C 長家館	0.17	23.42	26.20	
D 実光院	0.26	26.31	42.99	
E 涉成園	0.27	65.95	2.30	
F 豊国神社	0.30	90.39	3.70	
G 赤倉郡路里		138.61		
合計	1.52	467.95	106.16	574.11

図 10: MTZ のみ



	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 龍王寺	0.27	40.72	28.11	
C 長家館	0.17	23.42	26.20	
D 実光院	0.26	26.31	42.99	
E 涉成園	0.27	65.95	2.30	
F 豊国神社	0.30	90.39	3.70	
G 赤倉郡路里		138.61		
合計	1.52	467.95	106.16	574.11

図 11: 提案手法



	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 龍王寺	0.27	40.72	28.00	
D 豊国神社	0.30	90.39	3.70	
E 赤倉郡路里		138.61		
合計	1.11	522.04	54.62	576.66

図 12: Greedy

はじめに

巡回路除去制約の
近似

MTZ

提案する制約条件

制約条件の比較
実験

実データを使った
実験

まとめ

まとめ

TSP に使われる MTZ をそのまま使うことができない. そこでこの問題を解決できる制約条件 (15) を提案した. これにより, 適切でより良い解を求めれたが計算時間が増加した. 制約条件をさらに追加することで計算時間を減らす方法を提案した. そして実験を行った. 結果として制約条件 (13),(14) の組み合わせ, 制約条件 (13),(20), 制約条件 (13),(21) の3通りに計算時間を減らす効果があった.

今後の課題

滞在時間を実際より長く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.