4H-01

確率付き集合回遊中心性に基づく重要観光スポットの並列計算

Parallelized Calculation of Major Sightseeing Spots Based on Set Detour Centrality with Probability

大石 真生 † 湯瀬 裕昭 ‡ 斉藤 和巳 ‡ 渡邉 貴之 ‡
Mao Oishi Hiroaki Yuze Kazumi Saito Takayuki Watanabe

1. はじめに

近年, 大規模ネットワーク分析の高速化を目的とした,GPU(Graphics Processing Unit) による並列計算に期待が集まっている [1]-[2]. 集合回遊中心性は文献 [4] において提案された新たな中心性指標であり, 途中で立ち寄るのが容易である度合いを表す指標である. この中心性はノード数の 3 乗オーダーの計算量が必要となるため, ノード数の増加とともに計算コストが著しく増加する. 我々は, すでに文献 [5] において, 集合回遊中心性に基づくスポット選定の計算が GPU によって並列化可能なことを示した. 一方, 集合回遊中心性に対して, 距離や人気度に伴うスポット選択確率を加味した中心性として確率付き集合回遊中心性が提案されている [3]. 距離や人気度をどの程度重視するかをパラメータとして指定した際に, 即座に観光スポットの推薦ができるようなシステムの実現が望ましい. そのため本研究では, 確率付き集合回遊中心性に基づく重要観光スポット選定の GPU による計算の高速化を検討する.

2. 集合回遊中心性と確率付き集合回遊中心性

2.1. 集合回遊中心性

観光行動分析に有効な中心性として提案されている集合回遊中心性は, あるノードから他のノードへ移動する際に途中で立ち寄るのが容易である度合いを表す指標である [4]. 地図上の観光スポット（以下, 単にスポットと呼ぶ）をネットワークのノードとみなし, 各リンクにスポット間の実距離を導入する. まず, スポット集合を $S = \{s, t, v, ...\}$ とし, スポット s とスポット t の最短距離を $d(s,t)$ とする. そして, スポット集合 S の部分集合として R を考える. スポットペア s から t への移動において, 部分集合 R 内のスポット r を寄り道スポットとして経由するときの最短距離は次式で定義される.

$$D(s,t;R) = \min_{r \in R}\{d(s,r) + d(r,t)\} \tag{1}$$

ここで, スポット間の距離は交差点情報を用いた最良優先探索により計算した標準的な測地距離を用いる. 結果として, 集合 R に対する集合回遊中心性は以下のように定義される:

$$SDC(R) = \sum_{s \in S \setminus R} \sum_{t \in S \setminus R \cup \{s\}} \frac{d(s,t)}{D(s,t;R)} \tag{2}$$

$SDC(R)$ を最大にするスポット集合を求めることで, 観光客の回遊性を向上させるスポット抽出ができる.

2.2. 確率付き集合回遊中心性

確率付き集合回遊中心性は, 距離や人気度に基づくスポットを選択する確率を加味して計算を行う. 文献 [3] では, 回遊者の行動をモデル化するために,Levy Flight モデルに伴う距離に依存したパラメータを導入している.Levy Flight モデルとは移動距離を d とした時に, その発生確率 $P(d)$ がベキ則 $P(d) \propto d^{-\lambda}$ に従う行動モデルである. このモデルを用いたスポット遷移確率 $p_1(t \mid s;\theta_1)$ は, 次式で定義される.

†静岡県立大学経営情報学部 School of Management and Informatics, University of Shizuoka
‡静岡県立大学 ICT イノベーション研究センター ICT Innovation Research Center, University of Shizuoka

Algorithm 1 貪欲法による立ち寄り容易スポット集合の計算

```
k ← 1, R_0 ← ∅
while k ≤ K do
    r̂_k ← arg max_{r∈S\R_{k-1}} PSDC(R_{k-1} ∪ {r})
    R_k ← R_{k-1} ∪ {r̂_k}
    k ← k + 1
end while
return R_K
```

$$p_1(t \mid s;\theta_1) = \frac{d(s,t)^{-\theta_1}}{\sum_{v \in S} d(s,v)^{-\theta_1}} \tag{3}$$

ここで,θ_1 は距離に対するパラメータであり,Levy Flight のベキ係数 λ に対応する. また, あるスポットの人気度を加味するために, スポット t の人気度を $f(t)$ とし, 人気度に対するパラメータを θ_2 とすると, スポット t の選択確率 $p_2(t;\theta_2)$ は次式で定義される.

$$p_2(t;\theta_2) = \frac{f(t)^{\theta_2}}{\sum_{v \in S} f(v)^{\theta_2}} \tag{4}$$

P_1 と P_2 を組み合わせることにより, $p(t \mid s;\theta_1,\theta_2)$ は以下で定義される.

$$p(t \mid s;\theta_1,\theta_2) = \frac{p_1(t \mid s,\theta_1)p_2(t,\theta_2)}{\sum_{v \in S} p_1(v \mid s,\theta_1)p_2(v,\theta_2)} \tag{5}$$

このとき, 確率付き集合回遊中心性は以下で定義される.

$$PSDC(R) = \sum_{s \in S \setminus R} \sum_{t \in S \setminus R} p_2(s;\theta_2)p(t \mid s;\theta_1,\theta_2)\frac{d(s,t)}{D(s,t;R)} \tag{6}$$

２種類のパラメータを調整することで, 観光客の好みに合った観光スポットの推薦ができるシステムへの応用も期待できる.

3. 近似解法アルゴリズムと計算の並列化

上述した中心性の目的関数 $PSDC$ を最大にするスポット集合 R を求める近似的手法として, 貪欲法と局所探索法を用いる [4]. 貪欲法によるスポット集合の計算手順を Algorithm 1 に示す. この手順では, 集合 R にスポットを 1 つ追加する際に目的関数の値の増分が最も多くなるスポットを選定する処理を K 回繰り返している. また, 適当な k 個のスポットからなる集合 R の初期解 R_k が与えられた際に, 局所探索法を用いて解を改善することが可能である.

本研究では, 貪欲法でスポットを新たに 1 つ選定するたびに,Algorithm 1 の **while** ループ内で局所探索法を繰り返し行う計算手順において, 並列化による計算の高速化を検討する.

そして, 本研究では並列処理に対応したプログラムを, NVIDIA 社の GPU で採用されている Kepler アーキテクチャと CUDA(Compute Unified Device Architecture) 開発環境を用いて実装する.

近似解法アルゴリズムに対して,GPU による並列化の適用を考える. 並列化の方針として,1 thread に対して 1 スポットの処理を割り当てて並列化する Spot-based と，1 thread に対して 1 スポットペアの処理を割り当てて並列化する Pair-based が考えられる.

Algorithm 2 並列版貪欲法の CPU 側の処理

$V[i], i = 1..|S|$ ▷ 集合回遊中心性の増分配列
$R[i], i = 1..K$ ▷ 抽出スポット配列
$F[i], i = 1..|S|$ ▷ 抽出済みスポットフラグ配列
$D[i][j], i = 1..|S|, j = 1..|S|$ ▷ 距離行列
$E[i][j], i = 1..|S|, j = 1..|S|$ ▷ 回遊度行列
$P[i][j], i = 1..|S|, j = 1..|S|$ ▷ スポット選択確率行列
$a_1[i], i = 1..|S|(|S|-1)/2$ ▷ スポットペア始点リスト
$a_2[i], i = 1..|S|(|S|-1)/2$ ▷ スポットペア終点リスト
t ▷ 1block あたりの thread 数
b_s ▷ Spot-based における block 数
b_p ▷ Pair-based における block 数

1: $k \leftarrow 1$
2: **while** $k \leq K$ **do**
3: *RESET_V_KERNEL$[b_s, t](V)$
4: **for** $h \leftarrow 1$ to $|S|$ **do**
5: **if** $F[h] = 1$ **then**
6: **continue**
7: **end if**
8: *CAL_KERNEL$[b_p, t](a_1, a_2, D, E, P, V, h)$
9: **end for**
10: $h \leftarrow$*MAX_ELEMENT(V)
11: *ADD_KERNEL$[b_p, t](a_1, a_2, D, E, P, h)$
12: $R[k] \leftarrow h$
13: $F[h] \leftarrow 1$
14: $k \leftarrow k + 1$
15: **end while**
16: **return** R

Algorithm 3 スポットペア間の回遊度を計算する kernel 関数の詳細

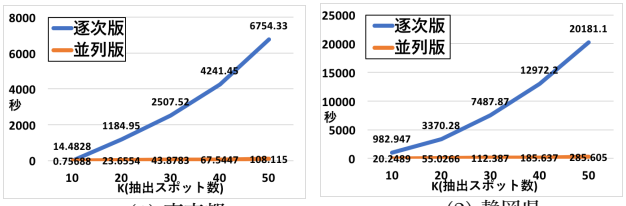
1: **function** cal_kernel(a_1, a_2, D, E, V, h)
2: **for each** thread i **in parallel do**
3: $i \leftarrow a_1[i], j \leftarrow a_2[i]$
4: $v \leftarrow P[i][j] * (D[i][j]/(D[i][h] + D[h][j]))$
5: **if** $v < E[i][j]$ **then**
6: $atomicAdd(V[h], v - E[i][j])$
7: **end if**
8: **end for**
9: **end function**

Algorithm 1 の 3 行目および 4 行目の処理に対して,GPU による並列処理を実行する kernel 関数を組み込んだアルゴリズムを, 新たに Algorithm 3 として示す.Algorithm 3 の 3 行目から 10 行目が,Algorithm 1 の 3 行目に該当し,Algorithm 3 の 11 行目が,Algorithm 1 の 4 行目に該当する.Algorithm 3 において*が付された関数は,GPU によって並列処理される kernel 関数である.Algorithm 3 の 3 行目の確率付き集合回遊中心性の増分配列 V を初期化する処理は,Spot-based の並列処理となる. 一方, スポットペア間の回遊度を計算する 8 行目の処理や, 選択したスポットを集合 R に追加し回遊度行列 E を更新する 11 行目の処理は Pair-based の並列処理となる.Pair-based の kernel 関数の詳細を Algorithm 3 に示す. また, 局所探索法や, その他の kernel 関数の詳細は紙面の都合上割愛する.

4. 評価実験と考察

本章では, 確率付き集合回遊中心性の目的関数を最大にするスポット集合 R の近似解を求める計算を, 第 3 章で示したアルゴリズムを用いて実行し評価する. その際,CPU のみで計算する逐次版,CPU と GPU により計算する並列版の両者の計算時間を比較することで並列化による計算の高速化を評価する. 本評価実験は,CPU:Intel Core-i7-3930K,RAM:64GB,OS:Cent OS 5.8（64bit）の実行環境にて行った. また,GPU は NVIDIA 社の GTX-TITAN 5GB RAM を使用し,Compiler は NVCC5.0 を使用した. 使用したデータセットの対象地域は, 東京都スポットネットワークと, 静岡県スポットネットワークの 2 つである. スポット数はそれぞれ 894 と 1266, リンク数はそれぞれ 399171 と 800745 である. 今回は, スポット間の距離行列 D やスポットの選択確率行列 P はあらかじめ計算済みとした.

評価実験の結果を図 1 に示す. 逐次版と並列版の計算時間を比較した結果は, どちらのネットワークにおいても抽出スポット数が増加するにつれて並列版の逐次版に対する高速化率が増加していることが分かる.



(1) 東京都 (2) 静岡県

図 1: K を増加させた際の計算時間の比較

次に, 実装した並列版のスポット選定処理を組み込んだ観光スポット推薦システムのプロトタイプを構築した. 構築したシステムのフロントエンドは Web アプリケーションとして実装し, サーバサイドの PHP プログラムを介して並列版の C++プログラムを実行する形式とした. 図 2 に示す通り, システムの画面上で抽出スポット数と θ_1, θ_2 をユーザが任意に設定できる方式とした.

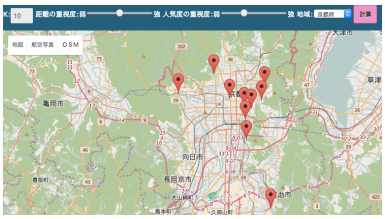


図 2: 観光スポット推薦システムのプロトタイプ

5. まとめ

本研究では, 確率付き集合回遊中心性に基づく重要観光スポット選定の GPU による計算の高速化を実現し, 観光スポット推薦システムのプロトタイプを構築した.

謝辞

本研究は, 総務省 SCOPE(No.142306004) の助成を受けた. ここに深謝する.

参考文献

[1] Z.Shi, and B.Zhang, “Fast network centrality analysis using GPUs,” BMC Bioinformatics, vol.12, May 2011.
[2] P.R.Pande and D.A.Bader, “Computing Betweenness Centrality for Small World Networks on a GPU,” in Proc. of the 15th Annual High Performance Embedded Computing Workshop (HPEC), Sept. 2011.
[3] 鈴木, 伏見, 斉藤, 風間,“回遊行動モデルに基づく重要観光スポット抽出法”, 情報処理学会第 77 回全国大会, 2015 年 3 月.
[4] 伏見, 斉藤, 武藤, 池田, 風間,“実距離を考慮した中心性指標の提案と重要観光スポット抽出への応用, ” 人工知能学会論文誌, Vol.30, No.4, 2015 年 7 月.
[5] 大石, 湯瀬, 斉藤, 渡邉,“集合回遊中心性に基づく立ち寄り容易スポットの並列計算”, 第 13 回情報学ワークショップ, 2015 年 12 月.