B-015

GPU による回遊中心性及び利便中心性の並列計算

Parallelized Calculation of Migration Centrality and Convenience Centrality Using GPU

大石 真生 † 渡邉 貴之 ‡
Mao Oishi Takayuki Watanabe

1. はじめに

近年，様々な分野において大規模ネットワーク・複雑ネットワークの分析に注目が集まっている．ネットワーク内の各ノードの重要性を指標化する中心性は, 様々なネットワークの分析に用いられるが，ネットワークの規模が増大すると計算には多大なコストを要するという問題がある．文献 [1] において，著者らは集合媒介中心性の計算に対して GPU による並列化を行い, 計算の高速化が可能な事を示した．

本研究では，観光行動分析への応用を目的として，集合回遊中心性及び集合利便中心性の計算における GPU による計算の高速化を検討する．これらの中心性は，観光リソース（Wi-Fi スポットやデジタルサイネージなど）の適切配置を実現する指標となる新たな中心性である [2]．これらの中心性はノード数（観光スポット数）の 3 乗オーダーの計算量が必要であり，ノード数の増加とともに計算コストが著しく増加する．そのため，GPU を用いた並列計算による計算の高速化を検討する．

2. 実距離を考慮したネットワークに対する中心性指標

媒介中心性や近接中心性などの既存の中心性指標では，各リンクの移動コスト (距離) はすべて等しいと仮定される．そのため，観光スポットや交差点などをノードとしたネットワークのように, ノード間の実距離が異なるような現実問題への応用には限界が想定される．そこで, 重要観光スポット抽出などの現実問題への応用を考慮し，各リンクに距離を導入した新たな中心性概念として回遊中心性及び利便中心性が提案されている [2].

2.1. 回遊中心性

観光スポット集合を $S = \{s, t, v, ...\}$ とする．文献 [2] では, ある観光スポット s からスポット v を経由して，スポット t へ行く際の寄り道距離を以下のように定義している．

$$D(s,t;v) = d(s,v) + d(v,t) \tag{1}$$

ここで, 観光スポット間の距離 $d(s,v)$ 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ている．そして, 任意の s から t への途中において，v の回遊中心性は以下のように定義される．

$$mBWC(v) = \frac{\sum_{s\in S\setminus\{v\}} \sum_{t\in S\setminus\{v,s\}} \frac{d(s,t)}{D(s,t;v)}}{(|S|-1)(|S|-2)} \tag{2}$$

すなわち, 回遊中心性は途中で立ち寄るのが容易である度合いを表す指標と考えられる．一方, 集合回遊中心性では, スポット集合 S の部分集合として R を考える．また, スポットペア s から t への移動で，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を次式で定義している．

$$D(s,t;R) = \min_{r\in R}\{d(s,r) + d(r,t)\} \tag{3}$$

結果として, 任意の s から t への途中において，R の集合回遊中心性は以下のように定義される．

$$smBWC(R) = \frac{\sum_{s\in S\setminus R} \sum_{t\in S\setminus R\cup\{s\}} \frac{d(s,t)}{D(s,t;R)}}{(|S|-K)(|S|-K-1)} \tag{4}$$

2.2. 利便中心性

利便中心性とは, 任意のスポットから辿り着くのが容易である度合いを表す指標である．まず, 任意のスポット v の利便中心性は以下のように定義される．

$$mCLC(v) = \frac{|S|-1}{\sum_{s\in S\setminus\{v\}} d(s,v)} \tag{5}$$

また, スポット集合 S の部分集合として R を考える．各スポット s から部分集合 R 内のスポットへの距離 r を用いて，R の集合利便中心性は以下のように定義される．

$$smCLC(R) = \frac{|S|-K}{\sum_{s\in S\setminus R} \min_{r\in R} d(s,r)} \tag{6}$$

3. 貪欲解法による集合中心性の計算アルゴリズム

文献 [2] では, 第 2 章で紹介した 2 種類の集合中心性の目的関数のサブモジュラ性 [3] に着目し, 貪欲解法により集合 R を求めている．文献 [2] で示されている計算手順をアルゴリズム 1 に示す．

アルゴリズム 1 貪欲解法による集合 R の求解

```
k ← 0, R ← ∅
while k < K do
    r̂ ← arg max_{r∈S\R} F(R ∪ {r})
    R ← R ∪ {r̂}
    k ← k + 1
end while
return R
```

ただし，$F(R)$ は 2 種類の中心性の目的関数を統一的に表記したものである．アルゴリズム 1 の時間計算量は，抽出スポット数 K と全スポット数 $|N|$ から $O(K|N^3|)$ となることがわかる．

4.GPU による並列計算

本章では, 第 3 章で紹介した集合 R の貪欲解法を対象に,GPU を用いた並列化の詳細について述べる．ただし, 紙面の制約上, 集合回遊中心性の並列化についてのみ示す．

本研究では NVIDIA 社の GPU で採用されている Kepler アーキテクチャをターゲットに CUDA 環境を用いて並列化を行った．具体的には, アルゴリズム 1 の逐次処理を含む基本的なプログラムは CPU 側で処理し, 目的関数の増分を計算する処理 (アルゴリズム 1 の 3 行目) について GPU 側での並列化を行った．目的関数の増分計算では,R に対して任意のスポット r を加えた式 (4) を繰り返し計算する必要がある．特に, 式 (4) の 2 重和の演算部分は容易に並列化可能である．CPU 側の疑似コードをアルゴリズム 2 に示す．ここで,N は

†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

全スポット数,K は抽出スポット数,L は全スポット間のリンク数であり $L = N(N-1)/2$ となる．アルゴリズム 2 では，並列化した処理は kernel 関数呼び出しによって GPU 側で実行される．また，スポットペア間におけるスポット回遊度の計

アルゴリズム 2 集合 R を求めるための CPU 側の処理

$R[i], i = 0..K-1$ ▷ 抽出スポット配列
$V[i], i = 0..N-1$ ▷ 回遊中心性の増分配列
$F[i], i = 0..N-1$ ▷ 抽出スポットフラグ配列
$D[i], i = 0..N*N-1$ ▷ 距離配列
$E[i], i = 0..L-1$ ▷ 累積回遊度配列
$a_1[i], i = 0..L-1$ ▷ スポットペア始点リスト
$a_2[i], i = 0..L-1$ ▷ スポットペア終点リスト
t ▷ thread 数
b_N ▷ ノード処理用 block 数
b_L ▷ リンク処理用 block 数

1: **for** $k \in 0..K-1$ **do**
2: V_reset_kernel$[b_N, t](V)$
3: **for** $r \in 0..N-1$ **do**
4: **if** $F[r] = 1$ **then** continue
5: **end if**
6: cal_kernel$[b_L, t](a_1, a_2, R, D, E, V, r, k, N)$
7: **end for**
8: $element_index \leftarrow$max_element(V)
9: $r \leftarrow element\_index$
10: add_kernel$[b_L, t](a_1, a_2, R, D, E, r, k, N)$
11: $R[k] \leftarrow r, F[r] \leftarrow 1$
12: $x \leftarrow 0.0$
13: **for** $i \in 0..L-1$ **do**
14: $x \leftarrow x + E[i]$
15: **end for**
16: **end for**

算は，全ての組み合わせで計算を行うため，逐次処理の場合スポット数が多くなればなるほど，計算に時間を要する．そのため，スポット回遊度の計算を GPU 側で行う cal_kernel 関数を用意した．cal_kernel 関数の詳細をアルゴリズム 3 に示す．並列処理の中で排他制御が必要なものについては,atomic 演算を使用している．さらに，アルゴリズム 2 の 9 行目では，配列内の最大値を GPU で求める CUDA のライブラリ関数を利用した．そして,cal_kernel 関数で求めたそれぞれの回遊度を配列 E に追加していく処理もまた，計算に時間を要する．そのため，回遊度を追加する処理を GPU 側で行う add_kernel 関数を用意した．add_kernel 関数の詳細をアルゴリズム 4 に示す．

アルゴリズム 3 GPU 側で実行される kernel 関数の詳細 1

1: **function** cal_kernel$(a_1, a_2, R, D, E, V, r, k, N)$
2: **for each** thread idx **in parallel do**
3: $i \leftarrow a_1[idx], j \leftarrow a_2[idx]$
4: $s \leftarrow r, min \leftarrow D[i*N+r] + D[r*N+j]$
5: **for** $h \in 0..N-1$ **do**
6: $g \leftarrow R[h], d \leftarrow D[i*N+g] + D[g*N+j];$
7: **if** $d < min$ **then** $min \leftarrow d, s \leftarrow g$
8: **end if**
9: **end for**
10: $v \leftarrow D[i*N+j]/(D[i*N+s] + D[s*N+j])$
11: $atomicAdd(V[r], v - E[idx]);$
12: **end for**
13: **end function**

5. 評価実験と考察

本研究で実装した GPU による集合回遊中心性の集合 R を求める並列計算処理の性能を観光スポットネットワークを用いて評価する．実行環境は，表 1 の通りである．

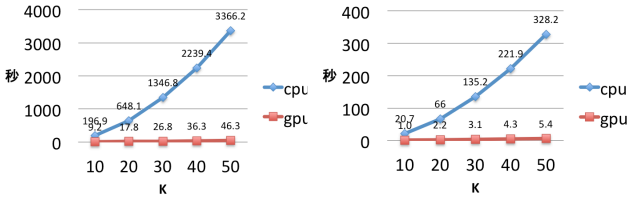
アルゴリズム 4 GPU 側で実行される kernel 関数の詳細 2

1: **function** add_kernel$(a_1, a_2, R, D, E, r, k, N)$
2: **for each** thread idx **in parallel do**
3: $i \leftarrow a_1[idx], j \leftarrow a_2[idx]$
4: $s \leftarrow r, min \leftarrow D[i*N+r] + D[r*N+j]$
5: **for** $h \in 0..N-1$ **do**
6: $g \leftarrow R[h], d \leftarrow D[i*N+g] + D[g*N+j];$
7: **if** $d < min$ **then** $min \leftarrow d, s \leftarrow g$
8: **end if**
9: **end for**
10: $E[idx] \leftarrow D[i*N+j]/(D[i*N+s]+D[s*N+j])$
11: **end for**
12: **end function**

表 1: 実行環境

CPU	Intel Core-i7-3930K（3.20GHz）
RAN	64GB
OS	Cent OS 5.8（64bit）
GPU	NVIDIA GTX-TITAN（SMX 数 15 基） 5GB RAM
Compiler	NVCC5.0

使用した観光スポットネットワークは,Tokyo ネットワーク (スポット数 :894, リンク数 :399171) と Kyoto ネットワーク (スポット数 :428, リンク数 :91378) の 2 つである．図 1



(1)Tokyo ネットワーク (2)Kyoto ネットワーク
図 1: K を増加させた際の計算時間の比較

に示すように,CPU 版と GPU 版の実行速度を比較した結果は,Tokyo ネットワーク,Kyoto ネットワークのどちらのネットワークも抽出スポット数が増加するにつれて GPU 版の実行速度の高速化率が向上していることが分かる．全体としての効用を上げるために局所改善を用いた場合，より最適な解が求められると考えられる．今後は、局所改善が施された貪欲解法アルゴリズムで GPU による高速化を実装したい．

謝辞

本研究は、総務省 SCOPE(No.142306004) の助成を受けた．また、静岡県立大学経営情報学部の斉藤和巳教授と鈴木優伽氏には集合回遊中心性の基本プログラムとサンプルデータを提供していただいた．ここに深謝する．

参考文献

[1] 赤池由樹， 大久保誠也， 武藤伸明， 斉藤和巳， 渡邉貴之， “GPU による集合媒介中心性に基づく看板配置問題の並列計算，” 第 12 回情報学ワークショップ， Nov. 2014.

[2] 伏見 卓恭， 斉藤 和巳， 武藤伸明， 池田哲夫， 風間 一洋， “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， Jul. 2015.

[3] G. L. Nemhauser， L. A. Wolsey， and M. L.Fisher, “An analysis of approximations for maximizing submodular set function，” Mathematical Programming, Vol.14, pp. 265–294， 1978.