4N-01

時空間データ分析のためのSpatialHadoopの拡張

瀧本 祥章[†]　杉浦 健人[‡]　佐々木 勇和[§]　石川 佳治[‡¶]
[†] 名古屋大学工学部電気電子・情報工学科　[‡] 名古屋大学大学院情報科学研究科
[§] 名古屋大学未来社会創造機構　[¶] 国立情報学研究所

1 はじめに

Hadoopをはじめとする MapReduce フレームワークは、バッチ処理や機械学習といった様々な分野においてビッグデータ分析に対する非常に有用なフレームワークとして知られている。その一方、スマートフォンなどのモバイルデバイスの普及によって生成される空間データはここ最近で爆発的に増加した。これらのことから空間データに関しても Hadoop のような MapReduce フレームワークによって並列分散して効率よく処理することが求められている。しかし、Hadoop は空間上の性質を考慮していないので、その処理の効率性が制限されるという問題がある。一方で、空間データの処理では道路ネットワークなどに関してグラフ処理が必要になることがある。反復処理が前提となるグラフ処理に関しては、一回ごとの処理で HDFS に書き出す必要がある Hadoop は効率的とは言えない。

そこで、本稿では空間データを SpatialHadoop[1] で扱い、グラフ処理を GraphX[2] で行うことにより効率的に処理を行うことを提案し、適用事例として最人気ルート (MPR, most popular route) の探索 [3] を行う。論文 [3] では MPR の発見手法が提案されているが、発見自体が興味の中心であり、処理の効率化は十分に議論されていなかった。特に、今日の並列分散処理技術の活用については未検討であり、多くの可能性が存在する。

2 利用するシステム技術

2.1 SpatialHadoop について

SpatialHadoop は空間データをネイティブサポートする初の MapReduce フレームワークである [1]。Hadoop 上での空間インデックス (グリッドファイル, R 木, R+木) を使用することで、データを複数のファイルに分割し、各ファイルの MBR(minimal bounding rectangle) を保持する。これにより、従来の Hadoop より高速な空間データ処理の実現をしている。

2.2 GraphX について

GraphX はグラフ処理を並列分散環境で行うための Spark API で、大容量のグラフ構造データを扱うことができるフレームワークである。グラフ処理中の反復処理において、繰り返しごとにファイルシステムに書き出す必要がなく、インメモリで処理を行うので効率的に実行することが可能である。

3 MPR 探索の概要

論文 [3] ではまず、軌跡データベースから移動ネットワークを構築する。この移動ネットワークはノードが道路上の交差点または軌跡の終点を表し、エッジが両端のノード間を、他のノードを経由せずに移動できることを表す。その後、構築した移動ネットワーク上で目的地に対する各ノードの人気度を求め、MPR を発見する。本稿では、移動ネットワークの構築を SpatialHadoop、ノードの人気度の計算を GraphX にて、並列分散処理で実現する。これらの行程についてそれぞれ簡単な説明と SpatialHadoop または GraphX を適用する方法を 4, 5 章で述べる。

4 移動ネットワークの構築

4.1 概要

論文 [3] では、道路ネットワークに関する地図情報は利用せず、移動軌跡データから直接的に移動情報を抽出する。その第一歩として、道路上の交差点の検出を行う。その方法として、coherence (coh) という値を定義し、DBSCAN[4] に類似したクラスタリングを行っている。2 つの軌跡上の点 p, q が与えられたとき、coherence は式 (1) で定義される。

$$coh(p,q) = \exp\left[-\left(\frac{dist(p,q)}{\delta}\right)^{\alpha}\right] \cdot |\sin\theta|^{\beta} \quad (1)$$

なお、式 (1) 中の $dist(p,q)$ は p,q 間のユークリッド距離を、θ は p と q の移動方向の角度の差 ($0 \le \theta \le \pi$) を表す。また、α, β は調整用のパラメータで、δ はスケーリング係数である。DBSCAN ではユークリッド距離が閾値以上の点をまとめているのに対して、この方法では coherence が閾値 τ 以上であるような点をクラスタリングし、大きさが閾値 φ 以上であるものをノードとしている。

4.2 SpatialHadoop の導入

coherence については $coh(p,q) \ge \tau$ より $dist(p,q) \le \delta \cdot \sqrt[\alpha]{-\ln\tau}$ が成り立ち、空間的な性質を利用することができる。実際、このクラスタリングは SpatialHadoop によって 3 ステップの MapReduce で実行することができ、その処理手順は以下のようになる。

1. 部分クラスタリング
   mapper で各グリッド内の点をそれぞれ抽出す

Extension of SpatialHadoop for Spatio-temporal Data Analysis
Yoshiaki Takimoto[†], Kento Sugiura[‡], Yuya Sasaki[§], Yoshiharu Ishikawa[‡¶]
[†] Department of Electrical and Electronic Engineering and Information Engineering, School of Engineering, Nagoya University
[‡] Graduate School of Information Science, Nagoya University
[§] Institute of Innovation for Future Society, Nagoya University
[¶] National Institute of Informatics

る。また、複数のグリッドにまたがるクラスタの発見用に、各グリッドの端から $\delta \cdot \sqrt[q]{-\ln \tau}$ 以内の距離にある (図 1 の灰色部分) 点を抽出する。reducer によってそれぞれクラスタリングを行い、クラスタ ID を付与する。

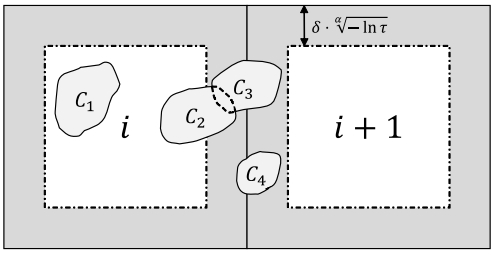

図 1: クラスタの例

2. 結合するクラスタの発見
   mapper でグリッドごとに作成したクラスタ (図 1 では C_1, C_2) とグリッドの外周部分から作成したクラスタ (図 1 では C_3, C_4) を比較して、同一の点を含むクラスタの ID(図 1 では C_2 と C_3) を検出する。reducer ではその検出結果をまとめる。

3. クラスタの結合
   mapper によって手順 2 で検出したクラスタ ID を統一することでクラスタを結合し、reducer で閾値 φ と大きさを比較する。

5 人気関数の計算方法

5.1 概要

論文 [3] では経路の人気度を経由するノードの積としている。各ノードの人気度は、以下のようにもとまる。まず、目的地が d であるときのノード n_i からノード n_j への遷移確率 $\Pr_d$ を 式 (2) で定義する。

$$\Pr_d(n_i \to n_j) = \frac{\sum_{traj \in (n_i, n_j)} func(traj, d)}{\sum_{traj \in all\ outgoing\ edges} func(traj, d)} \quad (2)$$

さらに、ノードの人気度を t 回以内の遷移で d に到着する確率 (t はグラフの直径) と定義する。ただし、$func$ は軌跡と点の最短 (ユークリッド, ネットワーク) 距離 $dist_s$ を用いて $\exp(-dist_s(traj, d))$ を表す。

5.2 GraphX の導入

ノードの人気度の計算はグラフ上での反復処理として記述できる。よって、GraphX を用いて以下のように求めることができ、その設定は次のようになる。

事前準備
すべてのノードで空のリストを所持
すべてのエッジの遷移確率を計算

初期メッセージ
エッジ方向に ID と遷移確率を送信

受信時の処理
受信した ID と確率を保存
同時に同じ ID を受信した場合は確率を加算

送信メッセージ
エッジ方向に受信した ID と
それぞれの値と遷移確率の積を送信

また、実際の処理はの流れは以下のようになる。

1. 目的地を d として事前準備を行い、初期メッセージを送る (図 2)。

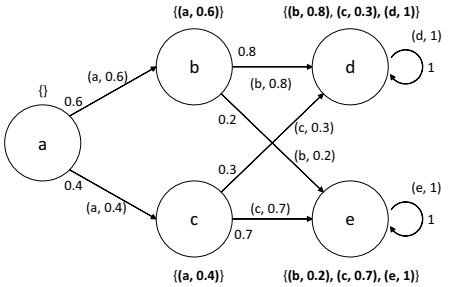

図 2: 初期メッセージの送信

2. メッセージを受信したノードは送信メッセージを送る (図 3)。

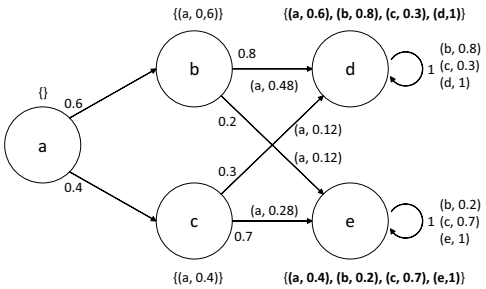

図 3: メッセージの送信

3. 2 を t 回繰り返し行う。

処理後に、目的地が所持するリストが各ノードの人気度となる (リストにない頂点は 0)。よって、図 3 の場合の人気度は $(\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}) = (0.6, 0.8, 0.3, 1, 0)$ となる。

6 まとめ

本稿では、大規模な空間データやグラフの処理に対して、それぞれ SpatialHadoop, GraphX を用いることによって並列分散処理して実行することを提案し、その適用事例として MPR の探索手法を開発した。

謝辞
本研究の一部は科研費 (25280039) , 地球環境情報統融合プログラム (DIAS) , 文科省「ビッグデータ利活用のための研究開発」による。

参考文献

[1] A. Eldawy and M. F. Mokbel. A Demonstration of SpatailHadoop: An Efficient MapReduce Framework for Spatial Data. In *VLDB*, pp. 1230–1233, 2013.

[2] GraphX | Apache Spark. `https://spark.apache.org/graphx/`.

[3] Z. Chen, H. T. Shen, and X Zhou. Discovering Popular Routes from Trajectories. In *ICDE*, pp. 900–911, 2011.

[4] J. Han, M. Kamber, and J. Peo. *DATA MINING Concepts and Techniques*, chapter 10.4.1. Morgan Kaufmann, third edition, 2011.