分散グラフアルゴリズム

研究ノート

泉　　泰介*

Distributed Graph Algorithm

Key Words：Distributed System, Graph Theory, Algorithm

はじめに

現代の情報処理において情報蓄積、計算が通信網の上において分散的に処理されることはもはや日常的であり、来るべきIoT（Internet of Things）時代に向かってその傾向はさらに加速を続けている。また、それに伴い、規模・複雑さの両面において、分散システムの効率的かつ安全な協調動作（分散計算）の実現は、大きな困難性を伴う課題として認識されている。なお、ここでいう「計算」とは、必ずしもアルゴリズム的な「問題解決」のみを指すだけではなくシステム自体の「制御」を含む広義の計算を指す。実世界における分散システムは極めて大規模化しており、実証的、実験的なアプローチのみから新たな技術を検証・検討していくことは容易ではない。数理的、理論的な側面から分散計算の問題構造を明らかにし、その困難性の本質を理解する試みは分散計算理論と呼ばれ、アルゴリズム理論の一部として古く70年代より研究され続けている。

分散計算の理論において、通信ネットワークは通常グラフ構造を用いてモデル化される（図1）。このとき、ネットワークのトポロジカルな構造に立脚した何らかの組み合わせ最適化問題、例えば最短経路問題、彩色問題、最小カット問題といった基本問題等をネットワーク自身が自律的に計算したいという要求は自然な問題設定であり、また多くの応用が存

* Taisuke IZUMI
1978年8月生まれ
現在、大阪大学　大学院情報科学研究科
コンピュータサイエンス専攻　准教授
博士（情報科学）
専門／分散システム，アルゴリズム理論
TEL：06-6879-4116
FAX：06-6879-4119
E-mail：t-izumi@ist.osaka-u.ac.jp

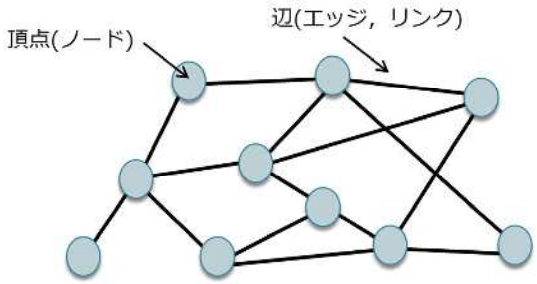

図1：グラフの例

在する。例えば、最短経路問題はネットワークにおける効率的なラウティング手法、彩色問題は無線ネットワークの帯域割り当て、最小カット問題はネットワークにおける脆弱部分の発見等に密接に関連している。このような、システムの動作系そのものを問題の入力としたネットワーク上で動作するグラフアルゴリズムは特に分散グラフアルゴリズムと呼ばれる。

分散グラフアルゴリズム設計の難しさ

分散グラフアルゴリズムを設計する最も単純な方法は、ネットワークの全トポロジ情報をある単一ノード（リーダー）が収集し、逐次アルゴリズム[1]を用いて解を計算することである。理屈の上では、この方法は効率的な逐次アルゴリズムを持つようなあらゆる問題に対して適用可能な万能解法ではあるが、大規模なネットワークにおいて全トポロジの情報量は巨大であり、それを単一ノードがすべて収集するというアプローチは現実的でないことは容易に想像できる。よって、分散グラフアルゴリズムの設計においては、各々の計算主体が入力データ（ネットワークのトポロジ情報）の全容を知ることなく自律的に

[1] 分散アルゴリズムと区別するために，従来の単一計算機上で動作するアルゴリズムはこのように呼ばれる。

解を出力しなければならないという点が難しさの本質となる。

多くの通信を伴うシステムにおいては、実行時間における支配的な要素は計算機内部の計算よりも通信に伴う遅延であるため、通常、分散グラフアルゴリズムの性能はメッセージ交換に要する（同期）ラウンド数で評価される。ここでは、1同期ラウンドにおいて、各ノードは自身とリンクにより接続されている隣接ノードへとメッセージを送受信して、その結果をもとに内部計算を実行するものとする。通信におけるリンクの帯域幅には上限があり、一般には既知の上限値 B(bits/ ラウンド) がモデルのパラメタとして与えられている。アルゴリズム全体の実行時間は多くの場合、帯域幅 B、ネットワークのノード数 n、リンク数 m、またネットワークの直径（最も離れた2ノード間のホップ数）D の関数として評価される。また、ネットワークが大規模であるという仮定から、B の値は、通常は n よりも大幅に小さい値、典型的には $B = \log_2 n$ が設定されることが多い。

分散グラフアルゴリズムの実例：最小全域木問題

分散グラフアルゴリズム設計の雰囲気を伝えるために、ここでは最小全域木問題に対するアルゴリズムの設計例を紹介したい。最小全域木問題は、リンクに何らかの重み（コスト）が与えられている状況において、全ノードを連結にするような辺の部分集合でコスト和が最小のものを求める問題と定義される[2]。分散システムの文脈においては、最小全域木問題はあるノードからのメッセージを全ノードへと放送する経路でコスト最小のものを求める問題とみなすことができるが、グラフ理論における基礎的な問題の一つとして、それ以外にも数多くの応用がある。

最小全域木問題を計算する最も良く知られた逐次アルゴリズムは、以下に示すクラスカル法と呼ばれる手法である。

1. まず、すべてのリンクを重みの昇順に整列する。
2. 整列された順序に従いリンクを1本ずつ取り出し、最小全域木の辺として解に加えていく。ただし。次に取り出したリンクが既に解に加えた辺集合と閉路 (ループ) を成すときは、その辺を加えず、スキップする。

この手続きは任意のグラフ、任意の重みの割り当てに対して最小全域木を構成することが証明されている。クラスカル法は逐次アルゴリズムとしては極めて動作がシンプルであり、また実装も容易である。しかしながら、すべての辺を整列して1本ずつ処理するという手続きは本質的に直列的であり、分散計算における並行性、並列性を十分に生かすことができない。例えばリーダーノードが収集通信操作により最小コストの未処理リンクを順次探して、それを解に加えるかどうか判定を下す、という形であれば、クラスカル法を分散アルゴリズムとして実現することは可能である、しかしながらこの実現法を用いた場合の計算ラウンド数は、愚直に実装すると $O(mD)$ ラウンドが必要であり、少し工夫したとしても $O(n)$ ラウンドが限界である[3]。いずれの場合においてもネットワークの規模に対して線形的に計算時間が増加することになり、やはり大規模なネットワークに対して適用することは困難である。そこで、以下に示すようなより並列処理に適したアルゴリズムを検討してみよう。

1. アルゴリズムは、最小全域木の断片の集合を管理しながら、複数の断片をリンクの追加により結合して、最終的に一つの結合された木（最小全域木）を出力する。初期状態において、すべてのノードは自分のみからなる木の断片であるとする。
2. 各々の木の断片は、自身と他の断片を繋ぐリンク（外向辺と呼ぶ）のうち、重みの最も小さいものを選び、それを全域木のリンクとして加える。加えたのち、リンクの両端点が属する断片は一つの断片に結合される。

このアルゴリズムは、逐次アルゴリズムの文脈ではボルフカ法と呼ばれる。アルゴリズムの対比のため、クラスカル法、ボルフカ法の動作例をそれぞれ図2に示す。クラスカル法と同様に、このアルゴリズムも任意のグラフ、任意の重みに対して正しい最小全

[2] この定義は木であることを要求していないが，コスト和最小のものは必然的に木になる。

[3] ここで O(・) はランダウの記号である。あまり馴染みのない読者におかれては，「$O(f(n))$」を「$f(n)$ に比例する」と読み替えても差し支えない。

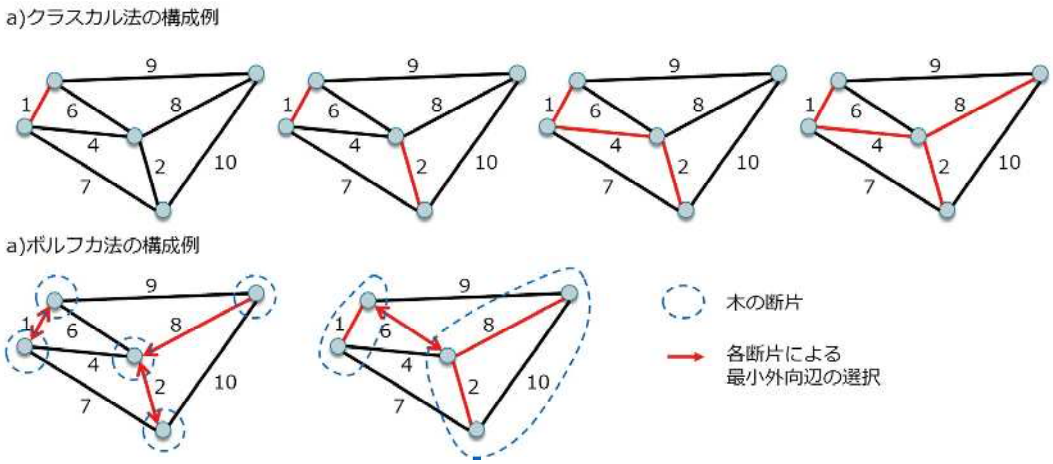

図2：クラスカル法とボルフカ法の動作例

域木を出力することが保証されている。さらに、各断片が最小重みの外向辺を選択する手続きは断片ごとに独立並行に実行可能なので、分散アルゴリズムとして実現することを考えると、こちらの手法はより適切なように見える。実際、全断片がそれぞれ最小外向辺を選んで断片を結合すると、断片の個数は少なくとも半減するため、全体としては（同時並行な）結合を高々 $\log_2 n$ 回繰り返すだけで最小全域木は構成できる。これは n に比して大幅に小さい回数である。しかしながら、実はボルフカ法も、単純な実装ではそれほど計算時間が良くならないことが分かる。問題点は、断片内部における最小外向辺の発見にある。最小外向辺の発見は断片内部ですでに張られている木を用いて収集通信操作により実現されるが、断片のサイズが大きいとき、この収集操作が元のグラフの直径 D よりも大きな時間を必要とする場合がある。実際、ある特定のグラフ及び辺重みに対して、ボルフカ法の自然な分散実装が $(n\log_2 n)/2$ ラウンドを必要とするような問題例が存在することが知られている。

結局のところ、上に挙げたアプローチはいずれも n に比例するラウンド数を要することになり、残念ながらそのままでは大規模ネットワークに対して適用可能なものではないという結論に至る。では、どのようにするとより高速なアルゴリズムを構成することができるのだろうか？実は、上に挙げた2つのアルゴリズムを組み合わせることがその解となる。なぜそのようなことが可能となるか、クラスカル法とボルフカ法を見比べながら検討してみよう。まず、ボルフカ法と同様に、クラスカル法もある意味で木の断片を結合していく手法とみなすことができる。ただし、複数の断片を並列に結合することはなく、リーダーが選んだ辺により断片が1つずつ逐次的に減少していく。すなわち、断片の個数を n 個から十分小さい個数に減らすまでの時間が、クラスカル法の実行における支配的な部分である。その一方、ボルフカ法の実行における支配的な部分は、各断片における最小外向辺の発見であるが、その手続きの実行に要する時間が大きくなるのは、断片の大きさが十分大きくなった後、すなわち、断片の個数がある程度少なくなった後である。よって、断片の個数が多い間はボルフカ法で高速に断片の数を減らし、ある程度数が少なくなったらクラスカル法へとスイッチすることで全体の実行時間を抑えることができる。実際にはさらにいくつかの補助的なアイデアを必要とするが、おおよそ上述のような手法を用いることで、$O(\sqrt{n}\log_2 n + D)$ ラウンドまで最小全域木の構成時間を向上させることが可能である。このラウンド数は、ほぼ理論的な限界に一致する、すなわち、この計算時間より大幅に構成時間の短縮を達成することが不可能であることも証明されている。

おわりに

本稿では、分散グラフアルゴリズムと呼ばれる研究分野について、その背景と問題意識を概説するとともに、基礎問題の一つである最小全域木問題を例題として、アルゴリズム設計の難しさ、面白さを紹介した。本稿により、当該分野へと興味を持つ方が

少しでも増えるようであれば、著者としては幸いである。

最後に、興味を持たれた方へと向けて、さらなる学びのための参考文献を挙げて本稿を締めさせていただく。グラフアルゴリズムに限らない分散アルゴリズム全般について日本語で読める文献のうち、現時点でも入手可能な唯一の教科書として[1]を挙げる。分散グラフアルゴリズムにフォーカスを絞った和書は残念ながら存在しないが、洋書であれば[2]は同分野のバイブルと呼べる一冊である。分散アルゴリズムに限らない、グラフアルゴリズム全体を学ぶための教科書は和書、洋書問わず数多く出版されているが、優れた一冊として[3]を挙げておきたい。

参考文献

[1] 増澤，山下「適応的分散アルゴリズム」共立出版，2010年。
[2] D. Peleg「Distributed Computing : Locality-Sensitive Approach」SIAM, 2000年。
[3] Kleinberg, Tardos（浅野他訳）「アルゴリズムデザイン」共立出版，2008年。