4H-02

極大独立集合問題における並列性と解の精度

根本望† 野村直也† 藤井昭宏† 田中輝雄†

工学院大学†

1. はじめに

グラフ理論におけるグラフとは,頂点集合 V,頂点同士結ぶ辺集合 E で構成される.

グラフ理論の用語として,他の頂点と辺で結ばれていることを「隣接」,頂点から出ている辺の数を「次数」と呼ぶ. 頂点の部分集合$v \subset V$で,v内の全ての頂点が互いに隣接していないような集合を独立集合と呼び,集合に含まれる頂点を独立点と呼ぶ.

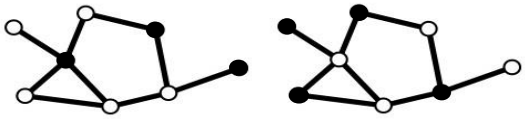
独立集合で,これ以上頂点を追加すると集合内の頂点同士で隣接関係が生じてしまうような集合を極大独立集合という. 極大独立集合を求める時,並列性と解の精度の両立が課題となっている.

本研究では,解の精度を保ちながら,並列性のある手法を提案し,有効性を評価する.

2. 極大独立集合問題

2.1 極大独立集合

極大独立集合の図解を図 1 に示す. 黒く塗りつぶされている点が独立点である. 独立点の個数を解の精度と定義する. (a)よりも(b)の方が解の精度が高い.



(a) (b)

図 1 極大独立集合

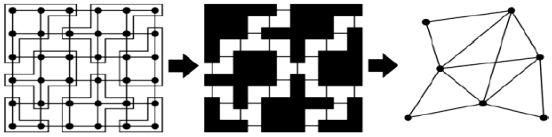
2.2 SA-AMG 法向け極大独立集合問題

大規模な連立一次方程式に対する有効なソルバの一つに SA-AMG 法がある. 図 2 は SA-AMG 法のアルゴリズムの一部で, 与えられた行列をグラフに変換し, グラフ上の隣接している頂点同士でグループ分けを行い(a), 1 つのグループを 1 つの頂点とする(b)ことで, よりサイズの小さい問題を作る(c)というフェーズである[1][2]. このフェーズにおいて, グラフの極大独立集合を求めた結果を適用することが有効とされている[2][3].

Parallelism and Accuracy of Solution in Maximal Independent Set Problem

Nozomu Nemoto†, Naoya Nomura†, Akihiro Fujii†, Teruo Tanaka†

‡Kogakuin University



(a) (b) (c)

図 2 SA-AMG 法

2.3 従来手法と問題点

```
1.  UNPROC = ∅
2.  INDEPEND = ∅
3.  forall i ∈ V{
4.     i.value = i.degree + random(0~1)}
5.  UNPROC ← V
6.  DO WHILE UNPROC ≠ ∅
7.     forall i ∈ UNPROC{
8.        INDEPEND ← i
9.        forall j ∈ i.neighbor {
10.          if i.value < j.value
11.             INDEPEND = INDEPEND\i
12.       end if}
13.       if i ∈ INDEPEND
14.          UNPROC = UNPROC \ i ∪ i.neighbor
15.       end if}
```

図 3 並列性の高い従来手法

従来の並列性重視の解法を図 3 に示す[3].

まず,各頂点に独立点に選ばれる優先度(value)を表した数値を与える(3,4 行目). 数値は各頂点の次数(degree)に 0 から 1 の乱数を加えた値となる. 次に,各頂点は自身との隣接頂点集合(neighbor)と数値比較を行い(8,9 行目)、その結果自身の値が最大ならば,その頂点自身及び隣接する全て頂点の消去を行う(12,13 行目). 6~14 行目までの処理を,独立点であるか消去されたかのどちらでもない頂点の集合(UNPROC)が空になるまで繰り返す.

従来手法は,各点を独立点として選択できるかどうかは並列に処理することができるため,サイズを大きくしても高速である. ただし,独立点の選ばれ方は不規則なので,解の精度はサイズを大きくするのに比例して悪くなる.

3. 提案手法

独立点が不規則に選ばれることを防ぐため,最初

```
1.  /*図3の1~5行目と同じ処理*/
2.  ARRAY = ∅
3.  INDEPEND ← v1 ∈ UNPROC
4.  forall i ∈ UNPROC ∩ i ∈ INDEPEND{
5.      forall j ∈ i.neighbor ∩ j ∈ UNPROC{
6.          forall k ∈ j.neighbor ∩ k ∈ UNPROC{
7.              k.value = k.value + 1
8.              if k ∉ ARRAY
9.                  ARRAY ← k
10.             end if}
11.     UNPROC = UNPROC \ i ∪ i.neighbor
12.   end if}
13. if UNPROC = ∅ exit(0)
14. else
15. forall i ∈ UNPROC ∩ i ∈ ARRAY{
16. /*図3の8~12行目と同じ処理*/}
17. go to row 3
18. end if
```

図4　提案手法全体像

に独立点を1つ選び,その頂点を中心として周囲から独立点を決めていくことを提案する

最初に独立点を1つだけ選ぶ(2行目). 独立点と隣接する頂点を消去し(5,6行目),消去された頂点と隣接する頂点が未処理である場合,頂点の優先度に1加える(7,8行目). 優先度を更新した頂点を集合ARRAYに加える(9,10行目). ARRAY内に格納された頂点を,従来と同様に自身の隣接頂点と数値比較し,独立点の判定を行う(15,16行目). そして3行目に戻り,消去,値の更新,配列への追加の一連の処理を同様に行い,UNPROCが空となった時処理を終了する.

4. 数値実験

4.1 実験環境

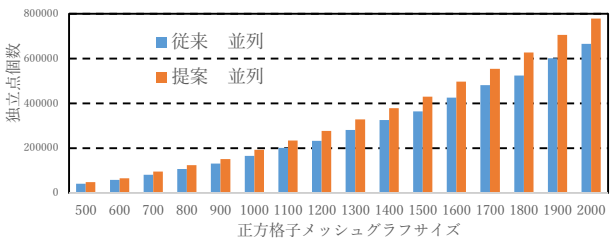
CPU は Intel(R)Xeon(R) CPU E5-2623 v3 @3.00GHz, コンパイラは gcc オプションは-O3 と-fopenmp を用いた.

4.2 実験内容

従来手法と提案手法をそれぞれ実行し, 解の精度と実行時間を計測した. スレッド数は4, 対象としたグラフは正方格子メッシュグラフで, サイズは500~2000の100刻みとした.

4.3 結果

従来手法と提案手法をそれぞれスレッド並列で実行した結果の解の精度の比較を図5に示す.



また, 提案手法の逐次版と並列版の実行時間を比較することで, 並列性を評価した. 結果を図6に示す.

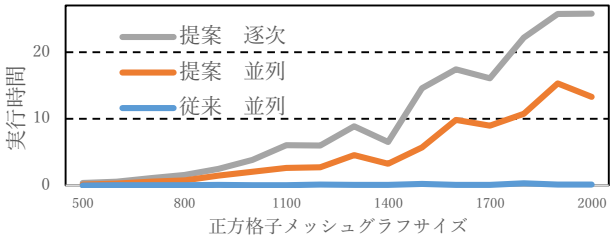


図6 提案手法の並列性評価

5. まとめ

本稿では, 極大独立集合問題における並列性と解の精度を両立させた手法を提案し, その有効性を評価した. 提案手法の解の精度は従来に比べて最大約16%向上し, かつスレッド並列化することで最大約50%時間短縮を実現した. しかし, 従来手法の並列版の実行時間はサイズを大きくしても1秒以内であるのに対し, 提案手法は並列版でも最大15秒という結果となった. 今回の実装において, 優先度の更新という処理は, スレッド間の競合が起きないように排他制御で実行している. このことがオーバヘッドとなっていることが原因であると考えられる. 今後の課題として, 解の精度を保ったまま, 実行時間をさらに従来手法に近づけさせることが重要である. 例えば, 最初に独立点として選ぶ頂点を増やすことが考えられる. その際, どの程度の個数を設定すると並列性と解の精度のバランスがとれた結果となるかをサイズごとに分析を行っていく必要がある. また, 今回は規則的なメッシュのみを対象としているため, 今後は不規則な構造のグラフを対象としていくことも重要である.

参考文献

[1]野村卓矢, 藤井昭宏, 田中輝雄: “Smoothed Aggregation に基づく AMG 法における分散アグリゲートの集約による通信の最適化” 情報処理学会研究報告 Vol.2013-HPC-139 pp.1-7(2013)

[2]藤井昭宏, 西田晃, 小柳義夫: “領域分割による並列 AMG アルゴリズム” 情報処理学会研究報告 Vol.2002-HPC-091 pp.31-36(2002)

[3] James.Blannnick, Yao.Chen, Xiaozhe.Hu, Ludmil.Zikatanov: Parallel Unsmoothed Aggregation Algebraic Multigrid Algorithms On GPUs” Numerical Solution of Partial Differential Equations Theory, Algorithms, and Their Applications Vol.45 pp.81-102(2013)