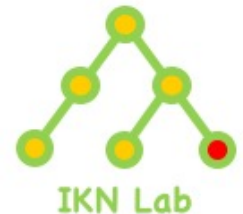


最大クリーク発見問題の 共有メモリ型 並列アルゴリズム

小島教寛

北海道大学 大学院情報科学院



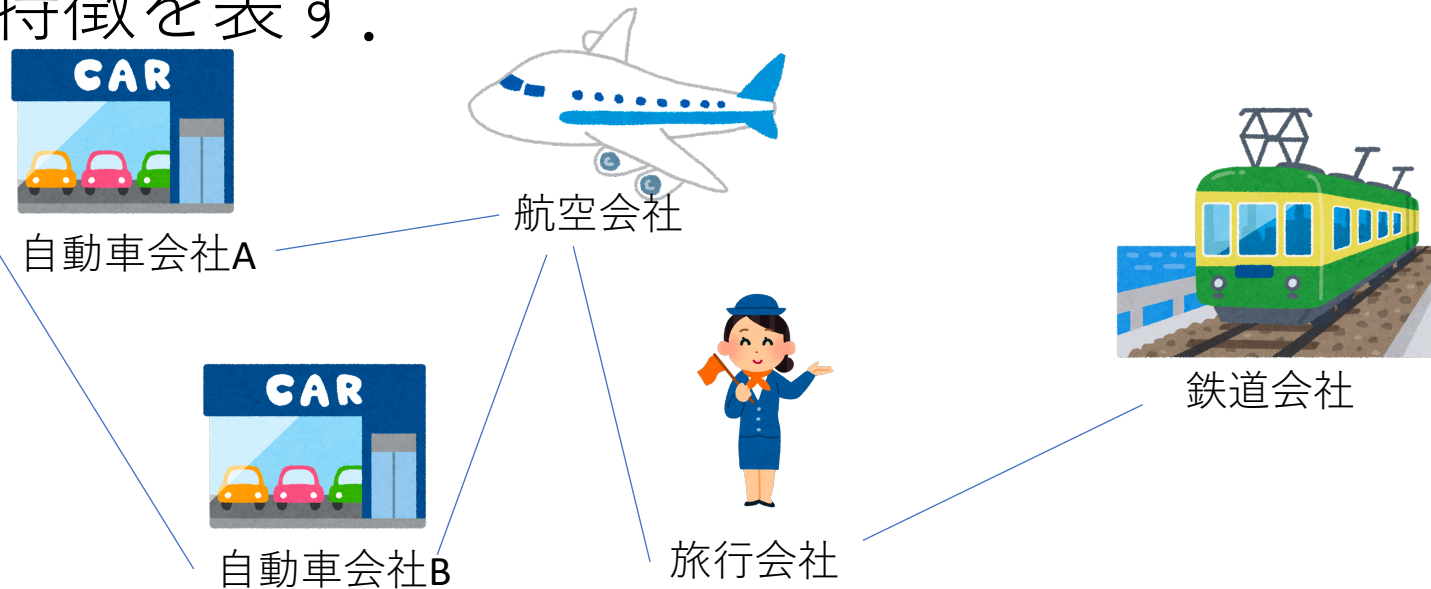
このスライドは, [githubのobatakyoukan/paper](#)に挙げています.

発表内容

- 準備
 - 従来手法MCT
 - 並列化
- 提案手法MCTP
- 実験

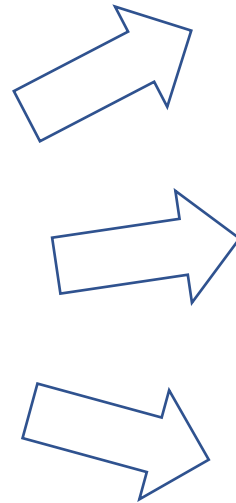
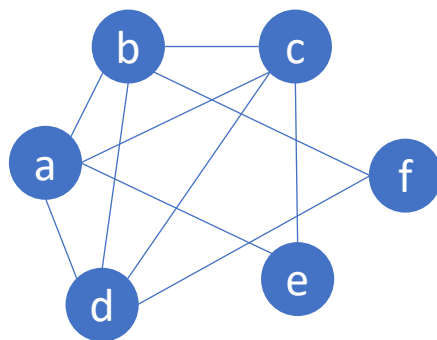
研究背景

最大クリーク問題(Maximum Clique Problem)は、幅広くの応用分野に利用でき、例えば、株式市場におけるマーケットグラフの最大クリークは、似た振る舞いをする最大のグループという重要な特徴を表す。



クリーク

無向グラフ $G = (V, E)$ が与えられたとき，任意の2頂点が隣接している頂点集合 V の部分集合 Q のことを **クリーク (Clique)** という。



a

サイズ1のクリーク

b

サイズ2のクリーク

f

c

サイズ3のクリーク

a

e

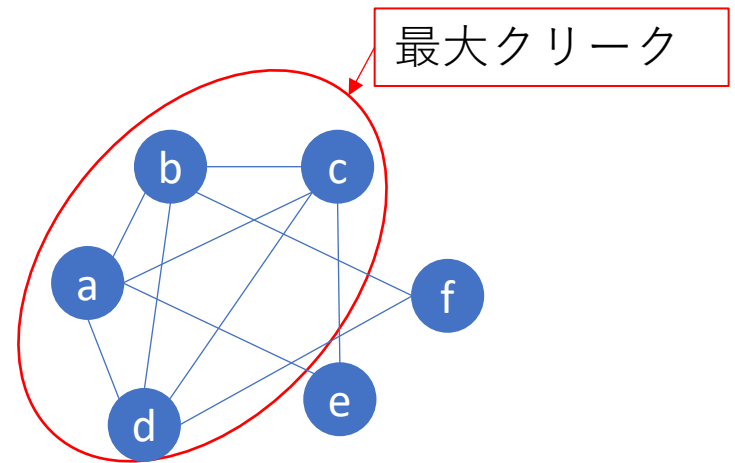
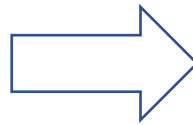
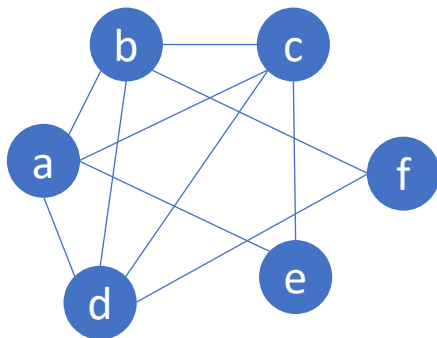
最大クリーク問題

- 入力

無向グラフ $G = (V, E)$

- 出力

任意の2頂点が隣接している V の部分集合の中で、サイズが最大のものを Q_{max} を **最大クリーク** という。



既存研究とその課題

最大クリーク発見に関する高速なアルゴリズムは、多く研究されている。その1つとして、2016年にTomitaらによってMCTアルゴリズムが提案されている[1]。

このMCTアルゴリズムは、高速に動作することが知られている。しかし、MCTの並列化によるさらなる高速化は、知られていない。

[1] Tomita, E., Yoshida, K., Hatta, T., Nagao, A., Ito, H., Wakatsuki, M.: A much faster branch- and-bound algorithm for finding a maximum clique, FAW 2016, LNCS 9711, pp.215-226, 2016.

研究目的

本研究では、MCT アルゴリズムを、その枝刈り等の効果をできるだけ損わずに、並列化することによる高速化を行うことを目指す。

主結果

- 共有メモリ型並列化した**MCTPアルゴリズム**を提案した.
- 実験では, ほとんど全てのグラフに対して, **実行時間が実際に抑えられていることを確認した.**

準備(1/2)

MCTについて

MCT

- MCT [Tomita et al.,2016]は、 **分枝限定法(B&B)**によって、最大クリークを発見するアルゴリズムである.
- 以下の工夫を用いることで、探索を効率的に行っている.
 - 彩色によるクリークの上界の概算.
 - 近似解によるクリークの下界の概算.
 - 部分問題に応じた彩色アルゴリズムの切り替え

[Tomita et al.,2016] Tomita, E., Yoshida, K., Hatta, T., Nagao, A., Ito, H., Wakatsuki, M.: A much faster branch- and-bound algorithm for finding a maximum clique, FAW 2016, LNCS 9711, pp.215-226, 2016.

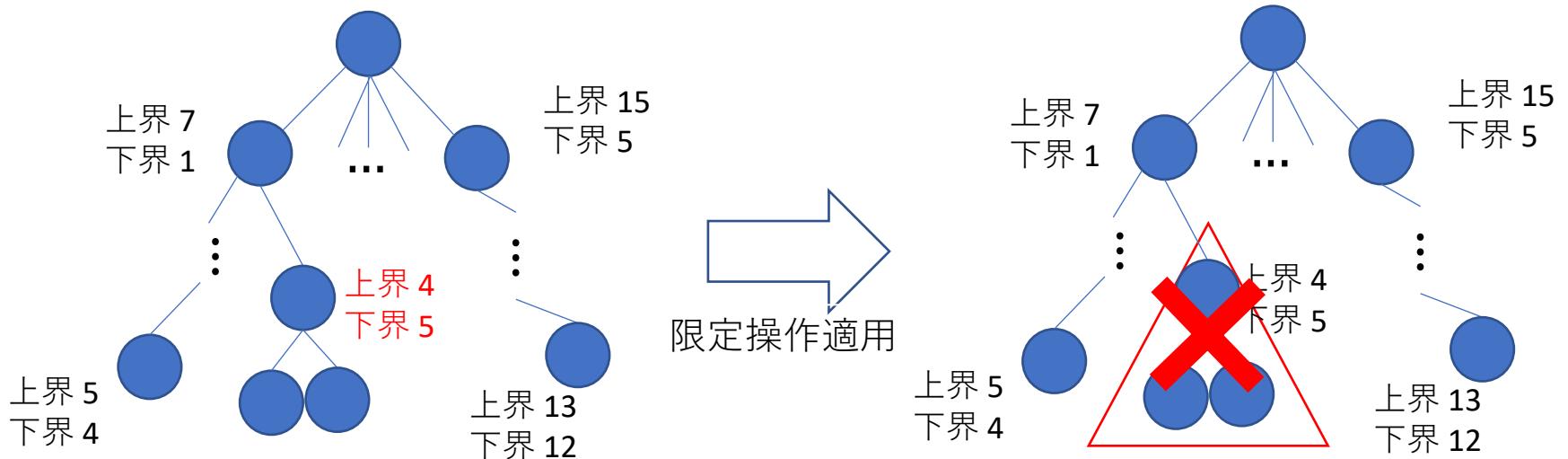
分枝限定法(B&B)

- 分枝操作(branching)

部分問題に分割する手続き. これを再帰的に行う.

- 限定操作(bounding)

部分問題の上界と下界を概算して, 最適解の候補でないものの分枝操作を打ち切る(pruning).

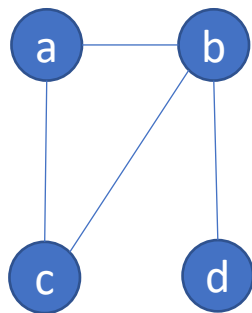


分枝操作: MCTの分枝操作手順

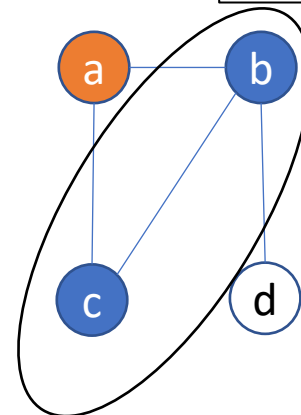
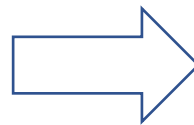
探索時に保持しているクリークを Q とし、 Q の頂点すべてに隣接している頂点集合を候補点集合とする。

分枝操作は、深さ優先探索によって、以下の操作を繰り返す。

- 候補点集合の中から、1つ頂点選び、 Q に追加する。
- Q が、現在までに見つかった中で最大のサイズるとき、暫定解 Q_{max} とする。
- 候補点集合を更新する。



aを Q に追加

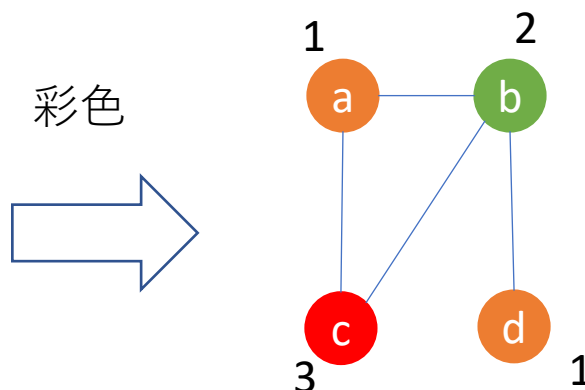
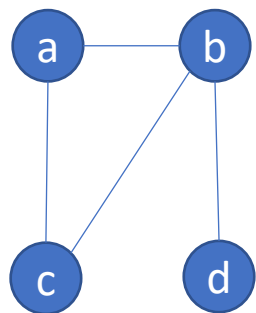


これを候補点集合として探索を繰り返す

限定操作: 彩色による上界の概算

彩色(coloring)とは、隣接する頂点に異なる正整数を振ること.

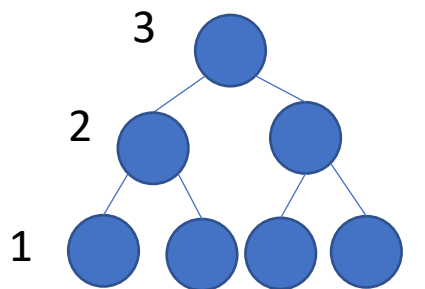
彩色に使われた最大の整数が k のとき、サイズ k を超えるクリークは存在しない.



3を超えるクリークは、存在しない.

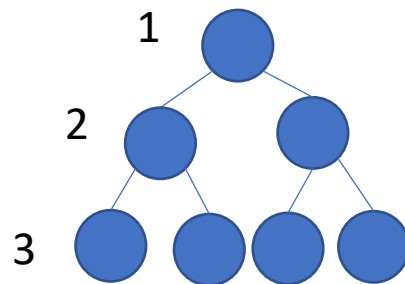
彩色方法の切り替え

根に近いときには，時間をかけて，探索空間を削減し，葉に近いときに，短時間の彩色アルゴリズムを適用することで高速化を期待できる．



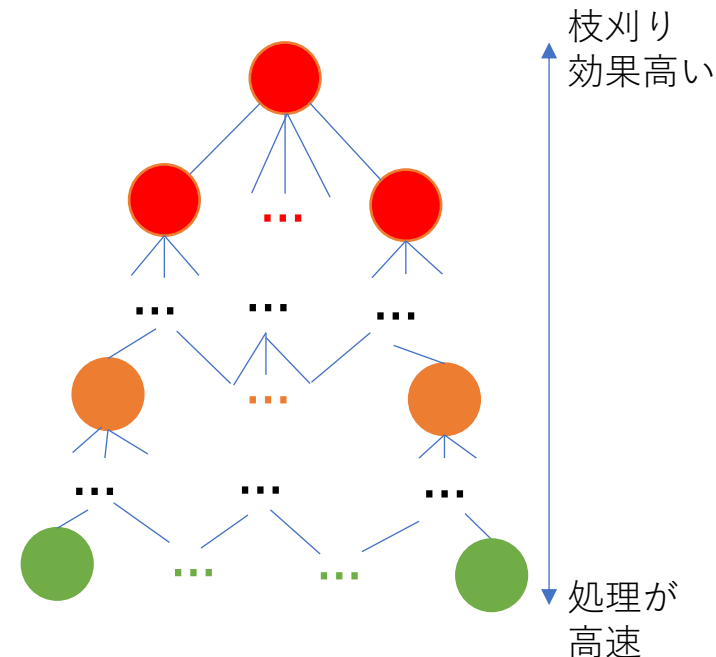
$$3 + 2 * 2 + 1 * 4 = 11$$

根の処理が重く
葉の処理が軽い



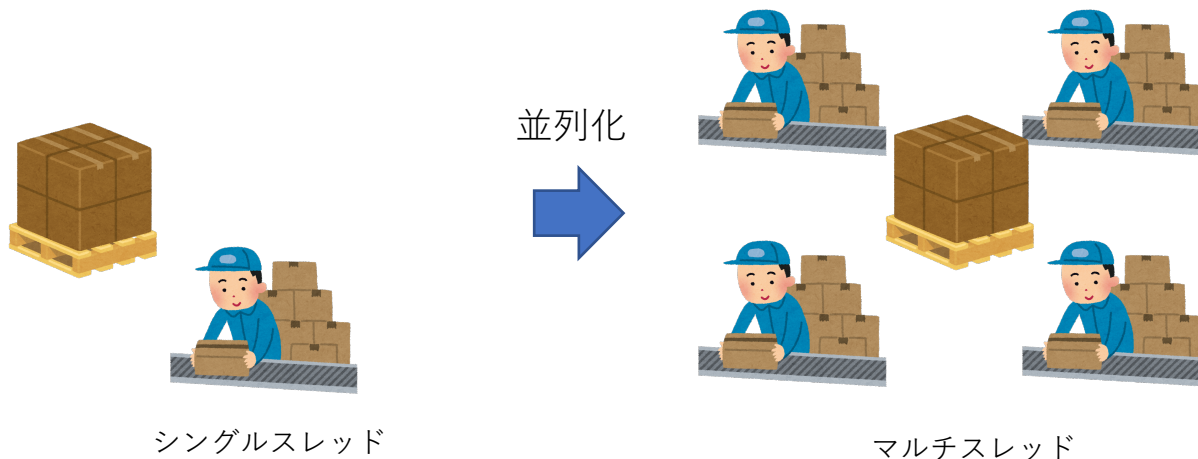
$$1 + 2 * 2 + 3 * 4 = 15$$

根の処理が軽く
葉の処理が重い



共有メモリ型並列化

共有メモリ型並列化とは，メモリを共有する形で複数の演算ユニットで，同時に演算を行うことによって性能を引き出す並列化手法.



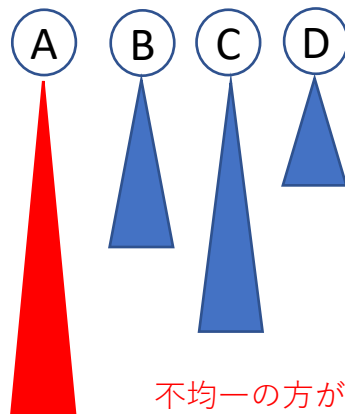
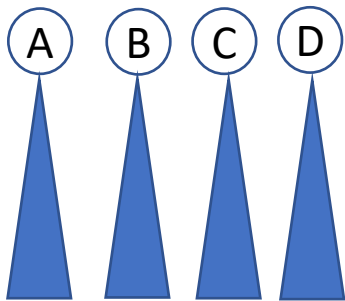
準備(2/2)

並列化について

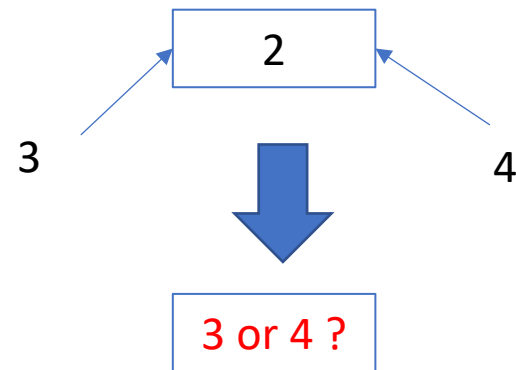
並列化の課題

並列化には，以下の点を気をつける必要がある．

- 仕事の量を他のスレッドと均等に分けること．
- 複数のスレッドに変化させられる可能性があるデータを，排他的に制御をすること．



不均一の方が大きい処理が発生する．



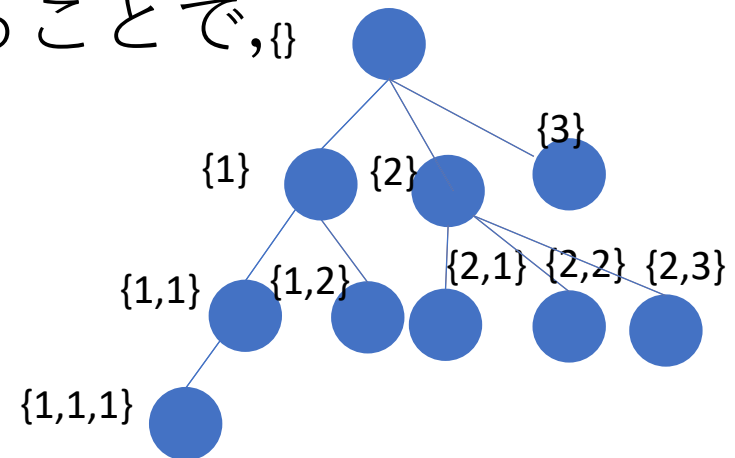
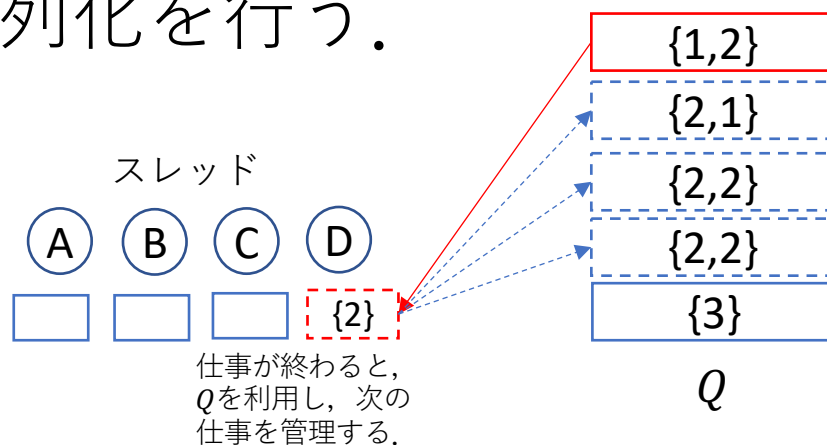
提案手法MCTP

最大クリーク発見の省メモリアルゴリズム

基本方針

深さ優先探索を優先順位付キュー Q を利用した探索アルゴリズムに修正する。探索数を増加させないため、優先順は、探索順序を利用する。

各スレッドは、 Q の top を取り出し、次に探索する情報を Q に入れることで、並列化を行う。

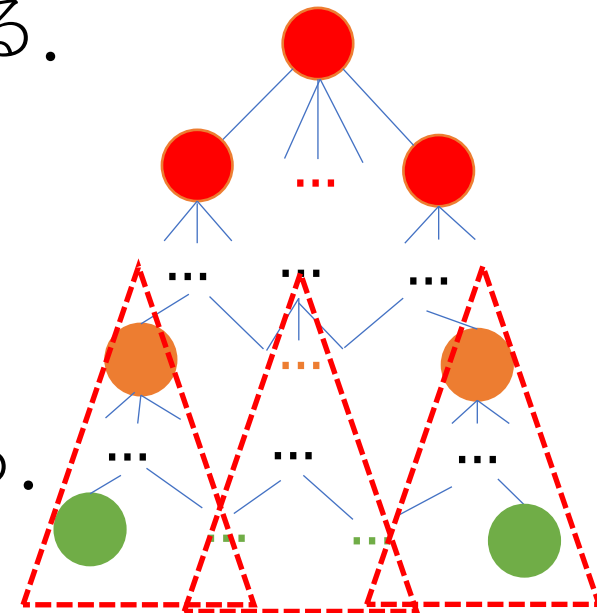


探索方針の切り替え

複数のスレッドに対して、1つの Q しか持たず、同時にアクセスできないので、排他制御が必要になる。

Q へのアクセスする回数が、膨大になるので、排他制御に多大な時間が必要になる。

これを削減するために、
根に近い問題のみ、 Q を利用し、
そうでない場合、
 Q を利用せず探索を行うことにする。



実験

実験

- アルゴリズム: MCTとMCTPを隣接リストで実装し, 近似解アルゴリズムでは, 2005年にKatayamaらが提案したKLSアルゴリズムを使用した[2].
- 実験データ: (1) DIMACS ベンチマークセットのグラフ¹を用意した.
- 環境: PC(OS: Ubuntu 20.04.2 LTS, AMD@Ryzen threadripper 3960x) 上で, 全てのプログラムは, g++ 9.3.0 -O3 -lm -fopenmpでコンパイルを行った.

[2] Kengo Katayama, Akihiro Hamamoto, and Hiroyuki Narihisa. An effective local search for the maximum clique problem. *Information Processing Letters*, Vol. 95, No. 5, pp. 503–511, 2005.

¹http://iridia.ulb.ac.be/~fmascia/maximum_clique/DIMACS-benchmark
2021/5/7

実験結果(実行時間)

ほとんどのグラフにおいて、高速化されたことが確認できた。

入力グラフ			実行時間(秒)		
グラフの名前	グラフサイズ	解	MCT	MCT48	MCT/MCT48
C250.9	250	44	335.7	11.9	28.1
C2000.5	2000	16	17167.0	1133.7	15.1
p_hat1000-3	1000	68	34214.6	1063.2	32.2
brock800_4	800	26	666.5	26.6	25.0
keller4	171	11	0.02	0.09	0.23

まとめ

まとめ

- 最大クリーク発見する並列化可能なアルゴリズムを提案した.
- 実験において, MCTPはMCTに比べて, ほとんどのグラフに対して, 実行時間が減少することが確認できた.
- 今後の課題
 - 他のアルゴリズムに提案手法を組み込むことで, 並列化を実現すること.
 - より多くのコアを持つPCでの高速化を確認すること.

ご静聴ありがとうございました.

補足資料

OpenMP ライブラリ

OpenMPとは、共有メモリ型マシンで並列プログラミングを可能にするAPIで、FORTRAN, C/C++から利用できます.

- ディレクティブを挿入するだけで並列化できる.
- コアごとにロードバランスを取り易い.
- 比較的粒度の小さい計算でも並列の効果が得られる.
- 自動並列化できる.
- 導入への敷居が低い

