競技プログラミング課題への GPU 利用に関する考察

Consideration on using GPU for competitive programming

宮﨑諒[*1]・松前進[*1]
Makoto MIYAZAKI[*1], Susumu MATSUMAE[*1]
[*1] 佐賀大学大学院工学系研究科知能情報システム学専攻
[*1] Department of Information Science, Graduate School of Science and Engineering, Saga University
Email: 17573020@edu.cc.saga-u.ac.jp

あらまし：近年教育の一環として競技プログラミングが開催されている．今後は GPU を用いたコンテストの発展も考えられる．本研究ではよく使用される Grundy 数などを例に，GPU で並列化する際の問題点などを考察する．Grundy 数とは組み合わせゲームにおける公平ゲームにおいて算出できる数である．さらに特定の条件を満たすゲームならば，Grundy 数を用いて勝敗を判定することができる．GPU を用いて従来の逐次プログラムを単純に高速化できるのか考察した．
キーワード：GPGPU，Grundy 数，競技プログラミング，高速化

1. はじめに

組み合わせゲームにおける公平ゲーム，すなわち(1)二人でやる(2)交互に手番がある(3)ランダム性がない(4)無限に続かない(5)情報はお互いにすべてわかっている，の 5 点が満たされているゲームにおいて適切に状態を定義すると，各状態の Grundy 数を算出できる(1)．Grundy 数とは，今の状態から一手で遷移できる状態の Grundy 数に含まれていない最小の非負の整数である．さらにそのゲームが「全ての Grundy 数が 0 の時にゲームが終了した場合にそのプレイヤーは敗北する」という条件を満たすならば，各状態における Grundy 数の XOR の値によって勝敗を判定できる．

Grundy 数を求める実装法として，動的計画法を用いた逐次プログラムの実装法が存在する．しかし我々が調べた範囲では，並列プログラムを用いた実装法はなかった．そこで我々は GPGPU を用いて従来の逐次プログラムを並列化し，高速化することを試みた．

2. コインゲーム

本研究では以下のようなゲームにおいて Grundy 数を用いて勝敗を判定するプログラムを実装した．このゲームは文献(1)に記載されているものである．

“Alice と Bob が次のようなゲームをします．k 個の数字 a_1,a_2,...,a_k が決まっています．最初 n 個のコインの山があって，それぞれの山 i には x_i 枚のコインがあります．Alice と Bob が交互に 1 つの山を選び，そこからコインをとります．一度に取るコインの枚数は a_1,a_2,...,a_k のいずれかと等しくなければいけません．最後のコインを取った方が勝ちです．最初は Alice の番です．共に最善を尽くすとした場合，どちらが勝つでしょうか．ただし a_1,...,a_k は必ず 1 を含むとします．”

3. 逐次プログラムの実装法

逐次プログラムは文献(1)に記載されているものを実装した．このプログラムではコインの枚数を状態とし，動的計画法を用いて Grundy 数を求める．具体的には以下の手順である．

1. x_i の中で最大の要素 max_x を求める．
2. 1 から max_x まで順に Grundy 数を求めていく．
   2.1 std::set を用いて集合 s を用意する．
   2.2 集合 s に一手先の Grundy 数を insert する．
   2.3 0 から順に集合 s の count を参照し，count が 0 だった場合はその数を Grundy 数として記録する．
3. 各状態の Grundy 数の XOR を計算し勝敗を判定する．

4. 並列化

GPGPU を用いて逐次プログラムを並列化するため，CUDA を用いて実装した．その際の手順を以下に示す．

4.1 std::set から配列への置き換え

3 章の手順 2.2 はループを用いて順に一手先をみていく．そのループの各周回を並列化することを考える．CUDA では GPU 側で行う処理をカーネル関数として記述するが，カーネル関数内ではループ内にある std::set を使用することができない．そこで std::set で行なっていた処理を配列を用いて実装した．具体的には std::set に insert していた処理(3 章の手順 2.2)を，配列に順に格納していく処理に書き換え，std::set の count を参照して集合に含まれるかチェックする処理(3 章の手順 2.3)を，配列を sort した上で binary_search を用いて探索し配列に含まれるかチェックする処理に書き換えた．

4.2 CUDA を用いた実装

4.1 で書き換えたループ処理の各周回に CUDA の各スレッドを割り当て並列に処理させる．

5. 評価実験

5.1 実験環境

CPU は 3.40GHz Intel Xeon CPU E3-1245 v3，GPU は NVIDIA GeForce GTX TITAN Black を用いた．OS は Ubuntu 16.04LTS を用いた．CUDA は Ver 8 .0，グラフィックスドライバは Ver384.111 を用いた．

5.2 実験内容

2 章で述べたコインゲームについての, 文献(1)の逐次実装, std::set を配列に置き換えた逐次実装, CUDA を用いて並列化した実装に対して時間計測を行った．パラメータは以下のように設定した．

1 つの山のコイン枚数：1~10000 からランダム
山の数：1~1000000 からランダム
k の最大値=1~1000,1~10000,1~60000 からランダム

5.3 実験結果

各実行時間を表 1 に示す.

表 1 各実行時間(micro second)

k の最大値	文献(1)	set→配列	CUDA
1000	758239	472455	670289
10000	9.18691e+6	6.12269e+6	6.31066e+6
60000	5.92586e+7	4.00037e+7	4.00901e+7

CUDA を用いて並列化したループ部分のみを計測した結果を表 2 に示す.

表 2 ループ部分のみの比較(micro second)

k の最大値	set→配列	CUDA	CUDA による高速化率
1000	1.13	2.01	0.56
10000	13.81	2.13	6.48
60000	77.04	2.23	34.54

5.4 考察

各実行時間の結果としては std::set を配列に置き換えた逐次プログラムが最も高速であり，CUDA を用いた並列化による高速化はできなかった(表 1).

CUDA を用いての並列計算を行うには GPU と CPU 間でのデータのやりとりが必要となる．その処理に時間が掛かってしまい実行時間の短縮ができなかったのではないかと筆者は考察した. GPU と CPU 間でのデータのやりとりを計測に入れず，並列化したループ部分のみを計測したところ，k の最大値が 10000,60000 と設定した場合 CUDA を用いて並列化したプログラムの方が高速であった(表 2). この結果から並列化したことによる実行時間短縮を，GPU と CPU 間でのデータのやりとりによる実行時間増加が上回り，結果として高速化に失敗していると考えられる．CUDA を用いたプログラムでは，並列化していることによりループ回数が増加しても実行時間の増加は逐次プログラムと比べ緩やかである．その結果ループ回数が増加するに従ってループ部分の並列化による高速化率が増加している．ループ回数を増加させ続け，並列化による実行時間短縮が，GPU と CPU 間のデータのやりとりによる実行時間増加を上回れば，全体として高速化できるのではないかと考える．

6. まとめ

本研究では Grundy 数を求める逐次プログラムに対して CUDA を用いて単純に高速化できるのか考察した．今回作成した並列プログラムでは高速化はできなかった．

今後の課題としては今回の実験結果を踏まえた上での高速化の実現である．具体的には std::set を配列に置き換え，CUDA を用いてループを並列化した今回の方針を改善しての高速化，std::set を配列に置き換えずに直接 CUDA を用いて並列化した高速化を検討している．

参考文献

(1) 秋葉拓哉， 岩田陽一， 北川宜稔:“プログラミングコンテストチャレンジブック”，第 2 版, 1989
(2) 前山和喜， 牧野潔夫， 落合竜也:“約数ニムのグランディ数”, 第 175 回ファジィ科学シンポジウム第 3 回工学院大学数理セミナー共催講演論文集，pp.1-5（2016）