

GPGPU を用いた2人ゲームにおける 強化学習の高速化

黒木是治^{*1} 森山甲一^{*1} 武藤敦子^{*1} 松井藤五郎^{*2} 犬塚信博^{*1}

^{*1} 名古屋工業大学 ^{*2} 中部大学

1 はじめに

マルチエージェントシミュレーションの分野では、エージェント数の増大に伴い計算時間が増加する問題がある。そこで近年、画像処理に用いられる GPU の並列計算に特化した性質によって汎用計算を行う GPGPU を利用し、エージェントが並列にタスクを実行することでシミュレーション全体の高速化を行う研究が進められている。しかし、GPU で実行するエージェント自体は単純なものが多く、強化学習を行う等複雑なものはほとんど見られない。そのため本研究では複雑なエージェントを複数体用いるシミュレーションに2人ゲームにおける強化学習を選び、GPGPU を用いたいくつかの手法で高速化し、比較を行った。

2 対象環境

本稿では2人ゲームにおける強化学習のシミュレーションとして囚人のジレンマゲームを用いた効用利用 Q 学習の進化計算 [1] を扱う。囚人のジレンマゲームは2人2行動ゲームの一種であり、2人のプレイヤーがそれぞれ協調行動または裏切り行動を選択し、その組み合わせから利得を得るゲームである。効用利用 Q 学習は、環境から得られた報酬 r に対して各エージェントがそれぞれ生成した主観的効用 u を Q 学習の報酬として用いるものである。文献 [1] では、 u は3次関数

$$u \equiv u(r) \equiv ar^3 + br^2 + cr + d$$

で表され、係数 a, b, c, d をまとめて染色体として遺伝的アルゴリズム (GA) で求める。シミュレーション全体の流れは以下の通りである。

1. N 個の染色体をランダムな値で作成し、それを元に初期エージェントを作成する (N はエージェント数を表す)
2. N 体の内2体のエージェントに囚人のジレンマゲームを行わせ、報酬から効用を計算し、エージェントが Q 学習を行うことを有限回繰り返す

Acceleration with GPGPU for reinforcement learning on two-person games

Yoshiya Kurogi^{*1}, Koichi Moriyama^{*1}, Atsuko Mutoh^{*1}, Tohgoroh Matsui^{*2} and Nobuhiro Inuzuka^{*1}

^{*1}Nagoya Institute of Technology, Nagoya 466-8555, Japan

^{*2}Chubu University, Kasugai 487-8501, Japan

3. 2 を1回の試合とし、全てのエージェントの組み合わせで行う
4. 報酬の合計を評価値として遺伝的操作を実行し次世代のエージェントを作成
5. 2~4 を指定した世代数繰り返す

各エージェントは1回ゲームを行うたびに Q 学習により報酬を最大化しようとする。また、試合ごとに Q 値は0に初期化される。そのため各試合は独立している。

3 GPGPU を用いた効用計算の高速化

従来手法では、上記シミュレーションの4, 5の効用計算を、CPUの単一スレッドを用いエージェント数分繰り返し処理で行っているために、数時間程度の計算時間を要している。それにより Q 学習の係数や GA のパラメータを変更した実験回数が制限される。そこで、それぞれの試合が独立していることを利用し GPGPU を用いて総当たり戦を並列実行することを試みた。効用計算の高速化にあたり CPU ではなく GPU を使う理由として、計算コア数の多さが挙げられる。CPU の計算コアは一つの計算性能が高いが、数は2~72と少ない。逆に GPU の計算コアは一つの計算性能が CPU に比べ低い、数は2000以上のものもある。マルチエージェント環境下で独立した2人ゲームの試合を行う場合、エージェントの組み合わせ数が多くなるため、計算コアが多い GPU で並列処理をするほうが高速化につながる。そこで、GPGPU を使った高速化手法を2つ提案する。

Single-Matching (SM) 始めに全てのエージェントの組み合わせ数だけスレッドを作成する。次に各スレッドに割り当てられた固有 ID から対戦するエージェントの ID の組を計算する。全エージェントの組み合わせを生成した後、並列に総当たりの繰り返し囚人のジレンマゲームを行わせる。

Average-Matching (AM) 始めにエージェントの数だけスレッドを作成する。次に1スレッド内に1つのエージェントを生成した後、各スレッドの固有 ID と現在の試合数から相手エージェントの ID を計算する。相手エージェントを生成した後、繰り返し囚人のジレンマゲームを行わせることを以下の回数分行う。

$$(\text{各スレッドの試合回数}) = \frac{(\text{エージェントの組み合わせ数})}{(\text{エージェント数})}$$

表 1: 従来手法との比較

手法	実行時間 (sec)
従来	11033.2
SM	50.7
AM	1204.5

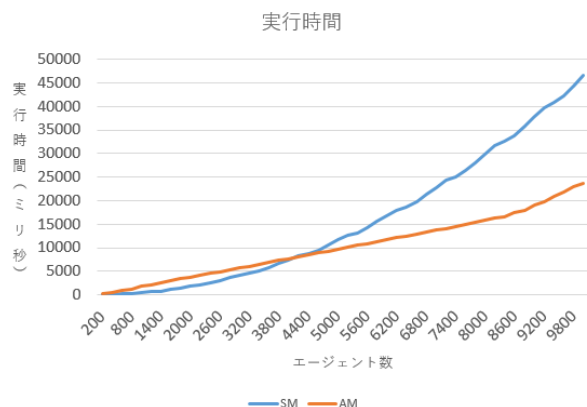


図 1: 実行時間測定結果

4 速度比較実験

実験準備 シミュレーションのエージェント数を 100, 繰り返し囚人のジレンマゲームを行う回数を 1000 回, 世代数を 10000 回と定め, シミュレーション全体の実行時間を計測し従来手法との比較を行った. なお, GPU 側のプログラムの実装は OpenCL [2] で行った. また, 実験に使用した計算機の CPU Intel Xeon E5-2650 v4 を 2 台, GPU GeForce GTX 1080 を 1 台.

実験結果 速度測定結果は表 1 の通りである. 従来手法では 3 時間超かかっているシミュレーションが GPGPU で並列化したことによりそれぞれ約 50 秒, 20 分と大幅に高速化が行うことが出来た.

実験考察 各試合を独立して実行し, ループ処理の少ない SM が一番高速となっている.

5 スケーラビリティ比較実験

実験準備 繰り返し囚人のジレンマゲームを行う回数を 1000 回とし, エージェント数を 200, 400, ..., 9800, 10000 と変化させてシミュレーションを 1 世代分実行し, 効用計算に要した実行時間を計測した.

実験結果 SM, AM での実行時間とエージェント数の関係を表したのが図 1 である. エージェント数が 200 ~ 4200 では SM のほうが実行時間が短い. 4200 を超えると AM の実行時間のほうが短くなり, 最終的に約 2 倍の速度差が生じた.

実験考察 エージェント数が増加した結果 SM より AM の方が早くなる現象に関して, 全スレッドの処理の終

了同期待ちの時間がかかっているためと考えられる. 例えばエージェント数が 10000 の時, SM のスレッド数は総当りの組み合わせ数と等しいので $\frac{10000 \times 9999}{2} = 49995000$ 個と膨大な数になる. また, GPU で実行するプログラムでは計算コア 1 つあたりの計算性能が低い. ため, 分岐処理や繰り返し処理を減らすことが望ましいとされている. SM はエージェントの ID を求める箇所に分岐処理が多く必要で各スレッドの負荷が大きい. 以上より全てのスレッドが同タイミングで全て終了するとは考えにくい. ため, 待機スレッドと動作スレッドが多量に混在することとなり終了同期待ちのオーバーヘッドが生まれているのではないかと考える.

6 おわりに

GPGPU を使うことで繰り返し囚人のジレンマゲームを用いたマルチエージェント強化学習を高速化できることが確認された. その他の 2 人 2 行動ゲームに関しても同様に並列化を行うことで高速化が可能である. また, 行動数が増加しても行動と状態の組み合わせ数が増加するだけなので他の 2 人ゲームのシミュレーションでも同様の手法を用いて高速化が可能であると考えられる. しかし, ゲーム人数の増加に関しては 1 人増えるごとにエージェントの組み合わせの数が極端に増えるため, 高速化の可否については検証が必要となる. また, 今回提案した手法の差異は並列タスクの粒度であるため, エージェントの数に応じた最適な粒度を動的に決める研究等の発展が期待される. 今後は, 考察で述べた仮説についてプロファイラーなどを用い GPU に負荷がかかっている箇所から原因究明, 検証を行いたい.

謝辞

本研究の一部は, JSPS 科研費 JP16K00302, 堀科学芸術振興財団, および栢森情報科学振興財団の助成を受けて行われた.

参考文献

- [1] 森山甲一, 栗原聡, 沼尾正行 “囚人のジレンマゲームにおける主観的効用の進化”, JAWS2010 論文集 2010.
- [2] The OpenCL Specification Version 2.2 Revision 06 <https://www.khronos.org/registry/OpenCL/specs/opencl-2.2.pdf> (2017 年 1 月 13 日 閲覧)