The 32nd Annual Conference of the Japanese Society for Artificial Intelligence, 2018

GPGPUを用いた強化学習エージェントの並列進化シミュレーション

Parallelization of evolution of reinforcement learning agents using GPGPU

千賀 喜貴 *1 Yoshiki Senga　森山 甲一 *1 Koichi Moriyama　武藤 敦子 *1 Atsuko Mutoh　松井 藤五郎 *2 Tohgoroh Matsui　犬塚 信博 *1 Nobuhiro Inuzuka

*1名古屋工業大学 大学院工学研究科 情報工学専攻
Department of Computer Science, Graduate School of Engineering, Nagoya Institute of Technology

*2中部大学 生命健康科学部 臨床工学科
Department of Clinical Engineering, College of Life and Health Sciences, Chubu University

GPGPU is a parallel computation technology using GPU that has huge number of processor cores for parallelly calculating colors of pixels on a monitor. Owing to its parallel performance, GPGPU is being used for multiagent simulation that contains multiple independent but interdependent agents. In a previous work, we used GPGPU to parallelize many runs of reinforcement learning agents for calculating their fitness in a simulation of evolution. It speeded up the simulation surprisingly. However, the evolution part was sequentially run in CPU and the communication between CPU and GPU happened in every generation. Hence, this work uses GPGPU to parallelize the evolution part in addition to the fitness calculation. It makes the simulation even faster due to parallelism and the reduction of latency between CPU and GPU.

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って計算時間が増加するという問題がある．この問題の解決方法として，画像処理等に用いられる GPU の計算性能を利用して並列に計算を行う GPGPU という分野が注目されるようになった．GPU は CPU に比べ一つのコアの計算性能は低いが数が多く，多数のコアで並列処理をすることで CPU 単体で計算するよりも高速な計算を行うことができる．GPGPU では計算に必要なデータと命令を CPU から GPU に転送する．CPUGPU 間のデータ転送は低速でボトルネックとなりやすい [1]．GPU 内のデータ転送は CPUGPU 間のデータ転送よりも数倍大きなメモリバンド幅で行われる．本研究では，2 人ゲームにおける強化学習の適応度計算を GPGPU によって並列実行するシミュレーション [2] を対象として，逐次計算している進化計算の部分を GPGPU によって並列化ならびに，シミュレーション全体を GPU 上で実行することによる CPUGPU 間通信を削減の 2 つにより速度向上を図れるか検証した．

2. General-Purpose computing on GPU (GPGPU)

2.1 GPU

GPU とは画像処理用のプロセッサであり，主にディスプレイに表示するピクセルの色を並列に計算することに使用される．CPU と比較すると，CPU は一般的に 4〜64 のコアを有していて，コア数は少ないがコア一つあたりの計算性能は高く連続的な処理に適しているのに対し，GPU はコアあたりの計算性能は低いがコア数が 1000 以上と非常に多く，並列的な処理に適している．大量のデータを VRAM と呼ばれる GPU のメモリに転送し多数のコアで並列処理することで，CPU 単体で処理するよりも高速に演算を行うことが可能である．一方で，GPU は 1 つの命令で複数のデータを処理する SIMD(Single Instruction Multiple Data) 形式で並列実行を行うため，分岐処理が存在する場合分岐する複数の命令を同時に実行することができない．また，CPU と GPU の間のデータ転送は GPU 内のデータ転送に比べ低速である．そのため実際の GPU を使った計算時間はデータの転送速度やその回数にも依存する．

連絡先: 千賀　喜貴，名古屋工業大学工学研究科情報工学専攻，名古屋市昭和区御器所町，y.senga.381@stn.nitech.ac.jp

2.2 GPGPU

GPGPU とは，高い演算能力を持つ GPU の演算領域を画像処理以外の数値計算に応用する技術である．GPGPU 用の開発環境として 2008 年に NVIDIA から CUDA が発表された．これにより，計算時間の膨大なプログラムの一部を GPU に委任することで，全体の処理効率を上げることが可能となった．その直後に GPGPU 言語として Khronos Group が提供する OpenCL が発表された．CUDA が NVIDIA 製の GPU に特化しているのに比べ，OpenCL は NVIDIA や AMD の GPU，Intel CPU のオンボードグラフィックスなど多くの環境下で動作し様々なデバイスに対応が可能であることが特徴である．

3. 準備

本章では本研究で使用する，囚人のジレンマゲーム，効用利用 Q 学習，囚人のジレンマゲームを用いた効用利用 Q 学習の進化計算の説明を行う．

3.1 囚人のジレンマゲーム

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 A と B がそれぞれ C(協調) または D(裏切り) の行動を選択し，その組み合わせから表 1 の利得 $T, R, P, S \in \mathbb{R}$ を得る．囚人のジレンマゲームでは，利得 T, R, P, S 間に以下の関係を持つ．

$$T > R > P > S$$

B が C を選んだ場合，A は C を選ぶと得られる利得は R，D を選ぶと得られる利得は T なので A は D を選択したほうが

表 1: 囚人のジレンマゲームの利得表

$A\backslash B$	C	D
C	R, R	S, T
D	T, S	P, P

良い．B が D を選んだ場合，A は C を選ぶと得られる利得は S，D を選ぶと得られる利得は P，よって A は D を選択したほうが良い．以上の 2 つより A は B がどの行動を取るかに関わらず，行動 D を選択することが最適な選択になる．B にとっても同じのため行動の組み合わせが (D, D) となり，結果として両者は P の利得を得ることになる．しかし，仮に両者が C の行動を取った場合，それぞれは R の利得を得る．$R > P$ より個人が得る利得も増えるため，両者にとって良い結果となる．お互いが協調しあって大きい利得を得ることが望ましいのだが，自分の利得を追求するとお互いに裏切り合うという望ましくない結果になることからジレンマと呼ばれている．また，今回扱うのは囚人のジレンマゲームを繰り返し行う「繰り返し囚人のジレンマ」というゲームがである．繰り返し囚人のジレンマゲームでは以下の条件が加わる．

$$2R > T + S$$

この条件を加えることでどちらかが D を選択する状態が繰り返されるよりも互いに C を繰り返したほうが利得が大きくなる．

3.2 効用利用 Q 学習

エージェントはある時刻 t において状態 $s_t \in S$ を知覚し，方策 π に基づいて行動 $a_t \in A(s_t)$ を選択する．ここで S は環境の取りうるすべての状態を，$A(s_t)$ は状態 s_t の時に取りうる行動の集合を表す．行動後にエージェントは報酬 $r_{t+1} \in \mathbb{R}$ を受け取り，新たな状態 s_{t+1} を知覚する．強化学習の一種である Q 学習 [3] は，最適方策 π^* における行動価値を Q^* として行動価値関数 Q を Q^* に近づけるように以下の更新式に基づいて学習を行う．

$$Q_{t+1}(s,a) = \begin{cases} Q_t(s_t,a_t) + \alpha\delta_t & \text{if}(s,a) = (s_t,a_t) \\ Q_t(s,a) & \text{otherwise} \end{cases}$$

$$\delta_t \equiv r_{t+1} \max_{a \in A(s_{t+1})} Q_t(s_{t+1},a) - Q_t(s_t,a_t)$$

上記の α は学習率と呼ばれるパラメータで $0 < \alpha \leq 1$ の値を取り，γ は割引率と呼ばれるパラメータで $0 \leq \gamma < 1$ の値を取る．δ_t は TD (Temporal-Difference) 誤差と呼ばれ，$Q_t(s,a)$ が π^* における Q^* に近づくに連れて 0 に近づくものである．Q 学習の初期段階では $Q_t(s,a)$ が $Q^*(s,a)$ とは異なった値であるが，エージェントがすべての状態への訪問とすべての行動の選択を無限回繰り返し，α_t が以下の条件を満たす時，この誤差は 0 に収束する．$n^i(s,a)$ を i 回目に $(s,a) = (s_t,a_t)$ となった時の t とする．

$$\sum_{i=1}^{\infty} \alpha_{n^i}(s,a) = \infty, \sum_{i=1}^{\infty} [\alpha_{n^i}(s,a)]^2 < \infty, \forall s,a$$

最適方策における行動価値関数 Q^* が既知の場合，状態 s における最適な行動 a^* を以下のように表せる．

$$a^* = \underset{a' \in A(s)}{\text{argmax}}\, Q^*(s,a')$$

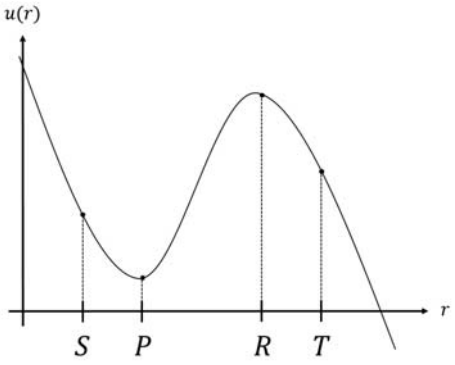

図 1: 効用導出関数 $u(r)$ のグラフのイメージ

エージェントが学習途中にこのような行動選択を行う場合，エージェントが行わない行動，未探索の状態が発生し，Q_t が局所解となる可能性がある．これを避けるため，ϵ-greedy 法などの手法を用いて行動選択を行う．ϵ-greedy 法は確率 $1-\epsilon$ で最大の Q_t を持つ行動を，確率 ϵ でランダムな行動を選択する．
効用利用 Q 学習 [4] では，エージェント内部に何らかの情動機構の存在を仮定し，その出力である主観的効用 u を Q 学習の報酬として利用する．以下では，情動機構として環境から得られる客観的報酬 r による関数を想定し，それを効用導出関数と呼ぶ．

3.3 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

本稿では 2 人ゲームにおける強化学習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 学習の進化計算を扱う [4]．このシミュレーションは囚人のジレンマゲームを繰り返し行うエージェントの情動機構の進化を疑似的に表現することを目的としている．報酬 r に対する効用を $u(r)$ とする．囚人のジレンマゲームの報酬 T, R, P, S に対して図 1 のように

$$u(R) > u(T) \quad \text{and} \quad u(S) > u(P) \quad \text{and} \quad u(R) > u(P)$$

となれば，協調行動 C が優位になる．この条件を表現する効用導出関数を

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

と仮定し，この関数の係数 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．シミュレーションの流れは以下のようになる．N はエージェント数，G は世代数を表す．

1. N 個の染色体をランダムな値で生成し，それを元に初期エージェントを生成する．
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数繰り返す．
3. 2 を一回の試合として，全てのエージェントの組み合わせで行う．
4. 各エージェントが得た報酬の合計をそれぞれの適応度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する．
5. 2〜4 を指定した世代数繰り返す．

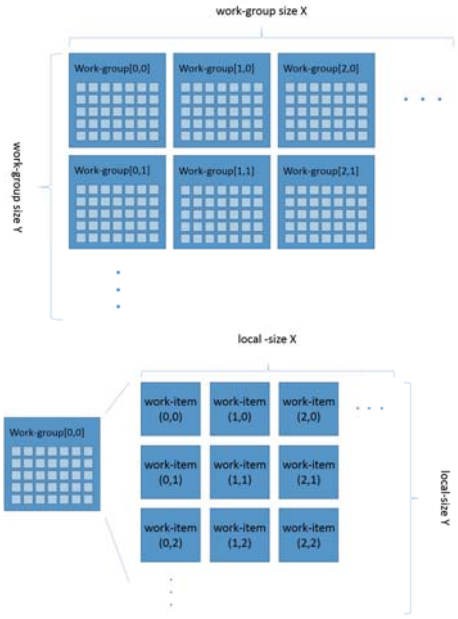

図 2: work item および work group 内の構造概念図

4. GPGPU を用いたシミュレーションの並列化

黒木ら [2] は 3 章で述べたシミュレーションの繰り返し囚人のジレンマゲームを行う部分を GPGPU により並列化した．しかし，GA は CPU 上で逐次計算されるためエージェント数が増えることで実行時間も大幅に増加する．そこで，GA についても GPU 上で並列に処理して実行速度の向上を図る．また，先行研究では遺伝子操作を実行し次世代のエージェントを作成する度に GPU に染色体データを転送し，囚人のジレンマゲームを実行した後その結果を CPU に返して適応度を計算する必要がある．これにより，先行研究では GA を実行する世代数の 2 倍の数 CPU と GPU で通信する．GA を並列化して一連のシミュレーション全体を GPU 上で実行することで CPUGPU 間の通信回数を削減することができる．GA を並列に処理することと、それによりシミュレーション全体を GPU 上で実行するという 2 つの観点から実行速度の向上を図る．本研究では先行研究同様 GPGPU のためのフレームワークとして OpenCL[5] を採用した．OpenCL では，以下のような work item，work group といった仕組みが扱われる．work item，work group の構造概念図は図 2 のようになる．

- work item: OpenCL における処理対象データの最小単位．この work item に対してカーネルと呼ばれるプログラムが実行される．work item 内でのみ共有できるプライベートメモリと呼ばれる記憶領域を持つ．
- work group: work item をいくつかのグループにしたもの．work group 内で共有できるローカルメモリと呼ばれる記憶領域を持つ．

GA の並列化手法として島モデルが知られている [6]．本研究では GPGPU による GA の並列化に島モデルを使用した．島モデルとは，染色体集合を島と呼ばれるいくつかの部分集合に分配し，それぞれの島で並列に GA を実行するとともに，

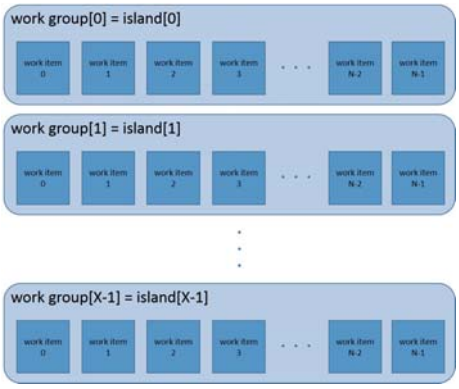

図 3: 島構造の概念図

一定世代数実行するなどの条件を満たした後，各島から一部の染色体を選んで移動させる手法である．本研究では，島の数を work group 数，島の染色体の数を work group 内の work item 数として並列化していく．work group 内は 1 次元の要素数 N 個とし，work group の概念図は図 3 のようになる．シミュレーションの流れは以下のようになる．

1. シミュレーションに必要な遺伝子情報，パラメータを GPU に転送する．
2. GPU 上で繰り返し囚人のジレンマゲームを実行し，その後染色体を島に順番に分配していく．
3. 適応度を元にルーレット法により島内の染色体を選択する．
4. 選択された染色体を交叉率 P_c で交差し，次世代の遺伝子とする．
5. 次世代の遺伝子それぞれに対し，突然変異率 P_m で突然変異を実行する．
6. 繰り返し囚人のジレンマゲームを実行し，次世代のエージェントの適応度を計算する．
7. 前回の移動から K 世代が経過していたら，適応度を元にルーレット法により選択された島内の染色体数の 5%の染色体をを隣の島の同じインデックスの染色体に上書きする．
8. 2〜6 を一定世代数繰り返す．
9. シミュレーション終了後に結果を GPU から CPU に転送する．

黒木らは全ての囚人のジレンマゲームが終了した時点で結果を CPU へ返し，CPU 上で遺伝的操作をした後に次世代のエージェントを GPU に転送し，囚人のジレンマゲームを行わせたが，本研究では上記のようにシミュレーション全体が GPU 上で実行される．

5. 実験

実験に使用したサーバーのデバイス情報は表 2 のとおりである．また，以下の実験において囚人のジレンマゲームの報酬を $T = 5$, $R = 3$, $P = 1$, $S = 0$ と設定し，Q 学習のパラメータを $\alpha = 0.25$,$\gamma = 0.5$,$\epsilon = 0.05$ 遺伝的アルゴリズムの交叉確率，突然変異確率を $P_c = 0.9$, $P_m = 0.01$ と設定する．

表 2: サーバーの CPU と GPU

ハードウェア	名称	個数
CPU	Intel Xeon E5-2650 v4	2
GPU	GeForce GTX 1080	1

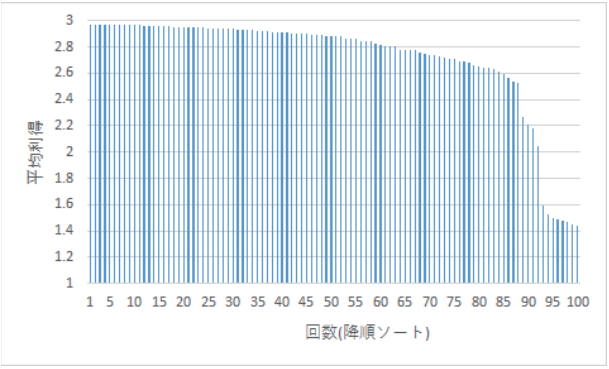

図 4: 平均利得

5.1 妥当性検証実験

まず，提案した手法の実行結果が正しいものであるかについての検証を行う．先行研究と同様に文献 [4] に従い，シミュレーションのエージェント数を 100，島内の染色体数を 100，島数を 1，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 と定めてシミュレーションを 100 回行った．相互協調が起こるとされる平均利得 2.7 以上の回数を見る．

結果は図 4 の通りである．グラフは 100 回行った実験を平均利得の降順にソートしたものである．100 回中 75 回が相互協調が起こるとされる平均利得 2.7 以上となった．文献 [4] では平均利得 2.7 以上の達成率が 8 割なので少し低い結果になったがほぼ同等の結果が得られた．GPGPU による GA の並列化は先行研究と同じように動作していると考える．

5.2 速度比較実験

従来手法と提案手法で実行速度を比較した．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 1000，K を 5 と設定し，エージェント数を 100 ずつ増やして実行速度を計測して先行研究との比較を行った．今回の実験では最初に島を一つに設定し，島ごとのエージェント数を 100 に設定して島の数を 1 つずつ増加させて実験を行った．

速度比較の結果は図 5 の通りである．全てのエージェント数において本研究の実行速度が先行研究より向上した．先行研究では，GA を実行する際に適応度計算で囚人のジレンマゲームのシミュレーションを行う．つまり GA の世代数の 2 倍の数 CPU と GPU の間でデータ転送を行う．エージェント数 N とすると一回あたりの GA では，4 つの遺伝子を含むエージェントが N 体，エージェントの総当りの対戦の報酬や対戦したエージェントの id などをを一時保存するデータが $N(N-1)$ 個，その他パラメータのデータ転送が行われる．本研究では適応度計算を含む GA が全て GPU 上で行われているので上記のデータ転送は一回である. また，本研究では先行研究で逐次計算されていた GA が並列に処理されている．本研究では GA はエージェント数と同じ数の work item 上で並列に行われ

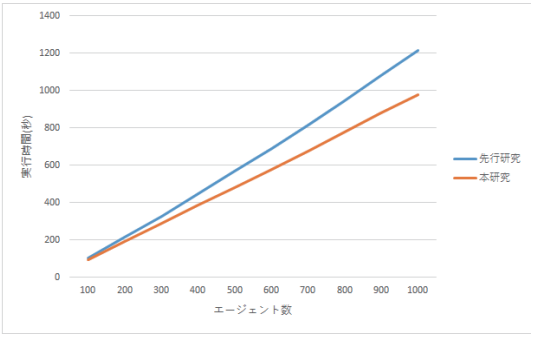

図 5: 速度比較

ており，エージェント数分並列化できていることになる．実際に CPU と GPU 間の通信速度が計測できたわけではないが，CPUGPU 間通信の回数を減らし，GA を並列化することでシミュレーションの実行速度の向上ができたと考えられる．

6. おわりに

GPGPU を用いて遺伝的アルゴリズムの並列化およびアルゴリズム全体を GPU 上で動かすことによる CPUGPU 間通信の削減により，シミュレーションの実行速度の向上が確認された．しかし，今回行ったアルゴリズム全体の並列化において，データ転送回数の削減と GA の並列化という 2 つの面において実行速度の向上が見られたが，それぞれにおいてどの程度速度が向上されたのか検証はできていない．GA を並列化することによる実行速度の向上とデータ転送回数の削減による実行速度の向上のどちらの恩恵が大きいのかの検証が今後の課題となる．

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] ゼロからはじめる GPU コンピューティング,http://www.gdep.jp/page/view/248, (2018 年 2 月 1 日閲覧)

[2] 黒木是治:GPGPU を用いた 2 人ゲームにおける強化学習の高速化 情報処理学会第 79 回全国大会，2017

[3] Christopher J.C.H. Watkins and Peter Dayan: Technical Note: Q-learning. *Machine Learning,* Vol. 8, pp. 279–292, 1992

[4] Koichi Moriyama, Satoshi Kurihara, and Masayuki Numao: Evolving Subjective Utilities: Prisoner's Dilemma Game Examples. *Proc. 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS),* pp. 233–240, 2011

[5] The OpenCL Specification Version 2.2 Revision 06, https://www.khronos.org/registry/OpenCL/specs/opencl-2.2.pdf, 2016

[6] 棟朝雅晴: "遺伝的アルゴリズム —その理論と先端的手法—" 森北出版 2008