卒業論文

題目

GPGPUを用いた
高速な多人数ポーカーAIの実現

指導教員

大野和彦　講師

2018年

三重大学 工学部 情報工学科
コンピュータソフトウェア研究室

清水直(414838)

内容梗概

近年，ボードゲームの AI が発達している．特に，経済学や医療プランニングなどに応用することができるため，ポーカーのような不完全情報ゲームと呼ばれるものが近年研究が進められている．

しかし，有効な手を求め解を得るには，膨大な計算が必要である．そこで，不完全情報ゲームの解選択に有効なナッシュ均衡を用いた AI を作成した．だがその際に，膨大な処理時間を要した為，NVIDIA 社が提供する GPGPU 用の SDK である CUDA[1] を利用し，並列化することにより，高速な多人数ポーカー AI を実現した．その結果，勝率は従来手法より最大 3 倍も改善され，処理速度は最大 15.8 倍もの高速化を実現することができた．

Abstract

In recent years, AI of board game has been developed. Particularly, since it can be applied to economics, medical planning, etc., what is called incomplete information game like poker has been studied in recent years. However, in order to solve effective hands, enormous calculation is necessary. Therefore, AI using Nash equilibrium which is effective for solution selection of incomplete information game was created. However, since it took a huge amount of processing time at that time, we realized high-speed multiplayer poker AI by parallelizing it using CUDA which is the SDK for GPGPU provided by NVIDIA. As a result, the winning percentage was improved up to 3 times as much as the conventional method, and the processing speed could be increased up to 15.8 times faster.

目 次

図 目 次

表 目 次

1　はじめに

1.1　研究目的

近年ボードゲームの AI が発達しており，特にオセロ，チェス，将棋などのプロレベルの実力を持つものが開発されている．だが，これらは完全情報ゲームと呼ばれ，お互いの行動だけで勝敗が決まるゲームである．対してポーカーや麻雀のような確率に左右されるゲームを不完全情報ゲームと呼び，同じ確率に左右される経済学や医療プランニングなどに応用することができるため，近年研究が進められている．しかし，有効な手を求め解を得るには，膨大な計算が必要である．そこで，不完全情報ゲームの解選択に有効なナッシュ均衡を用いた AI を作成した．だがその際に，膨大な処理時間を要した為，NVIDIA 社が提供する GPGPU 用の SDK である CUDA を利用し，並列化することにより，高速な多人数ポーカー AI の実現を図った．

1.2　論文の構成

2章では背景としてポーカーAIのロジックとCUDA，GPUの並列アーキテクチャについて解説する．3章では実装方法と高速化について述べ，4章で従来手法との勝率の評価，またCPU版とCUDA版, シェアードメモリ利用版の性能比較の評価結果を示す．最後に5章でまとめを行う．

2　背景

2.1　ポーカー AI のロジック

現状のポーカー AI は期待値通りに選択するのが勝率が安定している．そこから勝率を上げるには，癖を見抜き，裏をかけるかが重要である．現在，Libratus[2] と DeepStack[3] という AI が存在する．Libratus は Carnegie Mellon 大学が開発した AI で，ポーカーのルールを簡略化して，その状態でゲームを高速に行い，戦略を学習させるアルゴリズムである．そして，ここで得られた戦術は元のルールに適用されることで強力な AI が完成する．DeepStack は Alberta 大学が開発した AI で，過去の敗北情報をすべて記録して学習させるアルゴリズムである．ここで全ての記録を管理すると速度低下の要因になる．そこで，その敗北情報を圧縮して情報送信を高速にしたり分散コンピューティングで並列に動作させて高速化を実現している．

2.2　テキサスホールデム

本研究では，日本特有のドローポーカーではなく，テキサスホールデムという世界で最も用いられるルールを扱う．全員に共通な公開場札5枚と手札2枚のうち計7枚から5枚選んで競うポーカーである．カードの交換は無く，カードの強さの読み合いが勝負の鍵である．互いに相手のカードが5枚わかっている状態なので，相手のカードの強さが推定しやすい．今回は，初期所持金は10000，BB100，掛け金制限リングゲーム方式[1]で行った．

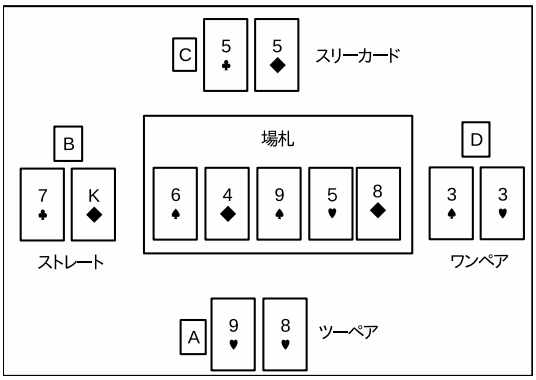

fig. 2.1: 4人対戦の役の決まり方の例

[1]所持金が10000で掛け金最低額が100からで掛け金の上限が決められており，最後に所持金が高い者の勝利というルール．

2.3　ナッシュ均衡

ゲーム理論の一つであり，プレイヤー全員が互いに最適な戦略を選択しており，これ以上自らの戦略を変更すると，自分の利益が減る互いの戦略の組み合わせのことをナッシュ均衡と呼ぶ．全ての戦略を比較し，逐次消去していくことにより全てのプレイヤーが次の一手を変更すると損をするという唯一の解にたどり着く．不完全情報ゲームでは，一部状態が不確定で次の一手の可能性も膨大になるので，時間をかけて１つ１つ比較していくか，相手の戦略の統計を取り，モデル化することにより，選択肢を絞っていくことで混合戦略ナッシュ均衡を求めることができる．

fig. 2.2: ナッシュ均衡の例 (囚人のジレンマ)

2.4 GPGPU

2.4.1 GPU

GPU とは, 画像処理を専門とする補助演算装置である. 一般的に, リアルタイムの画像処理は非常に高負荷な作業であるが, 処理内容そのものは単純であるためハードウェア化に向いており,CPU に比べて高性能化が著しい. この GPU の演算資源を画像処理以外の汎用計算に応用する技術を GPGPU という.

2.4.2 スレッドとブロック

スレッドは GPU でのカーネル実行の最小単位である. CPU ではそのコア数とほぼ同数のスレッドが動作するのに対し, GPU では数万〜数十万のスレッドを並列に動作させることにより, 高性能を達成している.

ブロックはスレッドをまとめたものであり,1 つのブロック当たり最大 1,024 スレッド格納できる. x 方向,y 方向,z 方向に 8 スレッドずつ, つまり「1 ブロックに $8 \times 8 \times 8$ スレッド」と 3 次元的表現で管理することができる. また,「1 ブロックに 512 スレッド」「1 ブロックに 16×16 スレッド」と 1 次元的,2 次元的にまとめることも可能である.

2.4.3　シェアードメモリ

NVIDIA 社の GPU は並列処理に用いる大量のデータを効率よく扱うために階層構造のメモリを持つ．CUDA を用いた GPGPU を実装する上で重要となるものが, グローバルメモリ, シェアードメモリの 2 つのメモリである．グローバルメモリは最大で 12GB と容量が非常に大きいが, レイテンシが非常に大きい．これに対して, シェアードメモリは容量が小さいが, アクセスが非常に速く処理の高速化において重要である．

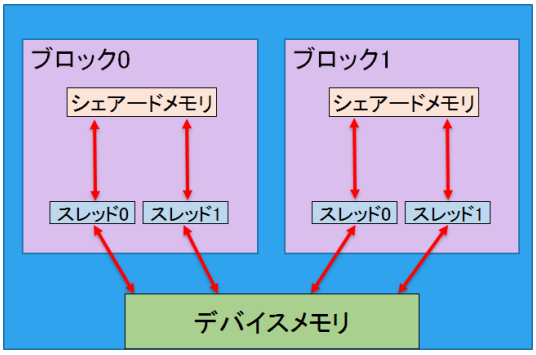

fig. 2.3: GPU の並列処理アーキテクチャ

2.4.4 CUDA

CUDA は NVIDIA 社より提供されている GPGPU 用の SDK であり，ユーザーは C 言語を拡張した文法とライブラリ関数を用いて GPGPU プログラムを開発することができる．CUDA プログラムでは, まず CPU 側 (ホスト) で必要なデータを作成し，GPU 側 (デバイス) に転送する．次にホストが GPU に実行させる関数 (カーネル) を起動させ，デバイスがカーネルを実行し，処理を開始する．最後に結果をホストに転送し，デバイスから受け取ったデータをホストが処理することで終了する．また,CUDA にはビルトイン変数というものが用意されており, 各スレッドは固有の ID を持ち, 配列の添え字にその ID を表す組み込み変数を利用することにより個々の要素を並列に処理することが可能である.

3　実装方法

3.1　先行研究の問題点

先行研究の AI には問題点が 2 つあり，1 つ目は学習時の改善は行われているが，探索時の改善が行われていない，その両方を組み合わすことでより高性能な AI ができると考える．2 つ目は，多人数対戦の機能を持ち合わせていないことである．多人数対戦が困難な理由としては，カードの状態の組み合わせ，各プレイヤーの行動の組み合わせ，座る位置で役職も違うため，その組み合わせなどが原因となっている．

3.2　概要

多人数対戦ゲーム，特に不完全情報ゲームの解を求める際に有効なのが，ナッシュ均衡の解を選択することである．ポーカーでは一部のポジションの際，降りるかどうかの選択にナッシュ均衡を当てはめることはある．しかし，状況が固定されているので扱いにくい．そこで本研究では，他の状況でも扱えるように改良を行った．しかし，これを行うと，多人数になればなるほど膨大な情報と計算が必要となる．そこで GPGPU による並列化を行い，高速化することで AI を実現した．

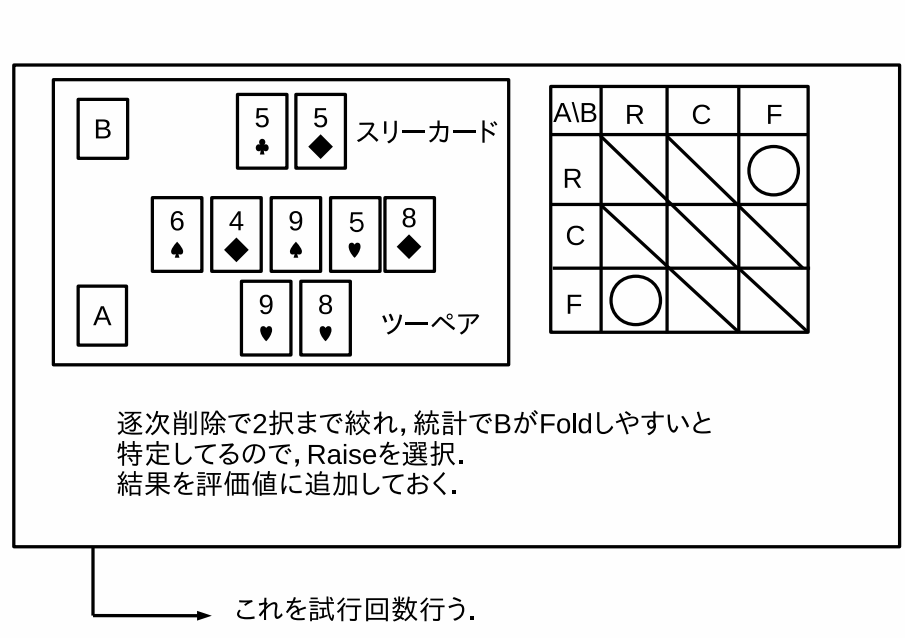

fig . 3.4: 解選択の例

3.3 実装

ポーカーにナッシュ均衡を当てはめる際に，まず各プレイヤーの行動の統計を取る．この統計の種類は多いほどいいが，多いと更に処理が複雑になるので今回はVPIP[2]などの10種類を収集した．この値は1試合ごとに更新していく．解選択関数ではプレイヤーの手札を乱数で決め，ナッシュ均衡表を作成する．不完全情報ゲームではこの表は，１つ１つ比較をして最善手を導く．ですが，今回は各プレイヤーの次の一手の組み合わせを統計を用いて列単位や行単位で逐次削除していく．そして，最も

[2]初手にコール以上の行動をした確率，ゲーム参加率

評価値の高い手を選択し，この動作を指定回数実行し，有効な一手を導出する．その際，対戦人数や試行回数が増えると処理時間も増える．そこで GPGPU による並列化で高速化を図った．

3.3.1 CUDA 版

解選択関数をデバイス側にも用意し，デバイス用の関数を新しく作成した．デバイス用の変数については，ホスト側→デバイス側への入力として現在の盤面の状態と各プレイヤーの統計，デバイス側→ホスト側への出力としてシミュレーションの結果を格納する配列を用意し，デバイスメモリに割り当てた．試行回数分のループを削除し，ブロック数:16 × 16, スレッド数:m/16 × n/16 で起動するようにした．また，試行回数はスレッド数をブロック数で割る m × n となる.

3.3.2　シェアードメモリの利用

関数内で多用される各プレイヤーの統計と解選択時に使う評価値をシェードメモリに格納する．それらの格納先をシェアードメモリ内の競合を避けるためにスレッド ID を $threadidx.x + threadidx.y*n$ で求めた．スレッド数分ずらすことでシェアードメモリにシミレーション結果を格納し，使用した．

4　評価

4.1　実験方法

まず，今回作成した AIa と，従来手法の AIb と対戦を行い，比較を行った．AI の割合，AIa の勝率，AIa の勝率と人数に対して平均勝率 (2 人なら 50，4 人なら 25) との比を示した表を Table1 に示す．

また，次に CPU 版，CUDA 版，シェアードメモリ版の 1 試合の平均思考時間と CPU 版との時間の比を示した表を Table2 に示す．実行環境としてメモリ:2GB，CPU:Intel(R)Pentium(R)CPU G640 2.80GHz，GPU:Tesla K20c を搭載した計算機を使用した.

表 4.1: 従来手法 AI に対して提案手法 AI の勝率 (%)

人数	AI の割合	AIa の勝率	勝率の比
2 人	a，b × 1	66	1.32
4 人	a，b × 3	37	1.48
6 人	a，b × 5	29	1.73
8 人	a，b × 7	33	2.64
10 人	a，b × 9	30	3.00

表 4.2: 1 試合の平均思考時間 (秒)

人数	高速化段階	思考時間	対 CPU 比
2 人対戦	CPU 版	30.20	—
	CUDA 版	13.27	2.28
	シェアードメモリ	12.48	2.41
4 人対戦	CPU 版	87.47	—
	CUDA 版	26.54	3.30
	シェアードメモリ	23.38	3.74
6 人対戦	CPU 版	172.48	—
	CUDA 版	31.18	5.53
	シェアードメモリ	27.20	6.34
8 人対戦	CPU 版	332.00	—
	CUDA 版	31.46	10.55
	シェアードメモリ	26.76	12.40
10 人対戦	CPU 版	541.87	—
	CUDA 版	40.80	13.28
	シェアードメモリ	34.28	15.80

4.2　考察

表 4.1 の結果を見ると，多人数対戦ほど勝率の比を改善することができた．あまり改善できなかった 2 人対戦の場合は，先行研究と組み合わせるとより強力になると推測される．

表 4.2 結果を見ると，10 人対戦の場合，最大 15.8 倍もの高速化を実現できたが，少人数の場合だと大きな速度改善には繋がらなかった．これは大人数の場合より少人数の場合のほうが処理が単純なためだと推測される．

5　おわりに

本研究ではナッシュ均衡を用いた多人数ポーカープログラムを作成し，勝率は従来手法より最大3倍改善された．また，CUDA化により並列化し，高速化を図った．更にシェアードメモリを用いることで更なる高速化を図った．本手法を用いることにより，CPUに比べて15.8倍程度の高速化を実現することができた．今後の課題としては，処理がより複雑になる掛け金無制限式や麻雀等への応用が挙げられる．

謝辞

本研究を進めるにあたり、ご指導を頂いた卒業論文指導教員の大野和彦講師に感謝致します。また、日常の議論を通じて多くの知識や示唆を頂いたコンピュータソフトウェア研究室の皆様に感謝します。

参考文献

[1] https://developer.nvidia.com/category/zone/cuda-zone. NVIDIA Developer Zone.

[2] Sandholm, T. (2017, August). Super-Human AI for Strategic Reasoning: Beating Top Pros in Heads-Up No-Limit Texas Hold'em. In Proceedings of the 26th International Joint Conference on Artificial Intelligence (pp. 24-25). AAAI Press.

[3] Bowling, M. (2017). DeepStack: Expert-Level Artificial Intelligence in No-Limit Poker." arXiv preprint. arXiv preprint arXiv:1701.01724.