グラフィックエンジンを用いたゲーム探索の高速化

田野 文彦* 近山 隆*

東京大学大学院工学系研究科*

コンピュータゲームプレイヤーでは，ゲームの進行をゲーム木と呼ばれる木で表現し，その中で有利な手を求めることを行う．ゲーム木の探索には膨大な計算が必要であり，効率よくゲーム木探索は強いゲームプレイヤーをつくるための重要な要件の1つである．また，今日グラフィックエンジン（GPU）の性能向上が著しく，画像処理だけでなくCPUで行うような汎用計算もGPUで行うことが試みられている．本研究では囲碁のモンテカルロ探索にGPUを用いることにより，9路での初期局面からのプレイアウト数毎秒9,570回で実現したことを報告する．

1 はじめに

囲碁はゲームの中でも探索空間が広く，チェス，将棋などと比べてコンピュータは人間の専門棋士より大きく劣っている．近年のGPU（Graphics Processing Unit）の性能向上に伴い，GPUを画像処理だけでなく汎用的に使おうとする試みがなされている．このようなGPUの利用をGPGPU（General Purpose GPU）と呼ぶ．GPGPUでは多数のコアを動作させるため並列的な計算の場合通常のCPUより高速に計算をすることができる．

そこで本研究ではGPGPUを用い，高速な囲碁のゲームプレイヤを構築することを目的とする．

2 関連研究

CPU以外での囲碁の実装はFPGAを用いた研究[1]がある．TLPGと呼ばれるハードウェア実装で局面の情報を3行分読みこみ，パイプライン処理によって9路盤で毎秒13,104プレイアウト，19路盤で毎秒2,055プレイアウトを実現している．

GPUで囲碁を実装したコードがメーリングリスト*1で公開されている．これによると9路盤がGPUで毎秒17,300プレイアウトで実行されている．

3 モンテカルロ法

モンテカルロ法[3]はシミュレーションや数値計算を乱数を用いて行なう手法の総称である．このゲームへの応用は，乱数を用いて決める着手に従ってゲームを進めることで勝敗を決定し，勝負を十分な回数行うことで，どの手が一番勝ちやすいのかを判断する方法である．

この手法の特徴としては，ランダムに手を生成しゲームの終了時まで打っていくプレイアウトと呼ばれる試行を行うため，評価関数が必要ないということがあげられる．したがって評価関数がつくりにくいとされる囲碁などで効果的な手法である．

4 提案手法

4.1 GPUによるプレイアウトの課題

GPUを用いた囲碁のプレイアウトについて述べる．囲碁のプレイアウトをする場合，合法手であるか，相手の石は取れるか，など判断すべき事柄は多い．通常こうした判断を計算機では条件分岐の組合せにより表現する．しかしながらGPU内のコアが異なる方向に分岐した場合，同一マルチプロセッサ内のコアは両方の分岐を通っただけの時間を消費する．この制約によって効率的なアルゴリズムが限られる．

通常CPUであれば連*2などのデータ構造を用いゲームを進めていくがGPUの性質から連のような属する石の数や連の数が状況によって変化するデータ構造を扱いにくい．また一度に処理するデータが多いほうが性能を発揮する．

* Graduate School of Engineering, The University of Tokyo

*1 Petr Baudis, "[computer-go] CUDA implementation of the per-intersection GPGPU approach", http://computer-go.org/pipermail/computer-go/2009-September/019433.html

*2 同じ色の石が縦横に連なったもの．呼吸点を共有するため便宜上ひとまとまりとして考える．

そこで本研究ではこのような事態に対処するため多数の局面の進行を同時に管理し，実行時に大きさが固定のデータを扱うために連での管理をせず局面の石の配置を基礎にプレイアウトを行う．

4.2 プレイアウトの流れ

実際の流れを述べる．局面は GPU 内のシェアードメモリに管理する．ランダムに手を選択し，合法であるかどうかを判断し，合法なら打ち，合法でなかったらもう一度ランダムに手を選択する．これを今回は 10 回繰り返し，すべて合法でなかった場合にパスをすることにした．

そして石をとる処理に移る．図 1 のような，最後に黒石を打った局面を考える．(1) では注目している白石の四方に空点があるかどうかによって呼吸点の有無を判定している．もし一か所でも呼吸点が存在すれば，その石は取られることはない．(2) で呼吸点があったという情報を隣の白石に伝えている．すると右の白石も四方を石に囲まれているが，取られることはないという判定ができる．この呼吸点の情報をすべての行，列に行きわたらせるために n 路の局面では (2) を $n-1$ 回繰り返す．今回は簡単のため自殺手は考慮していない．

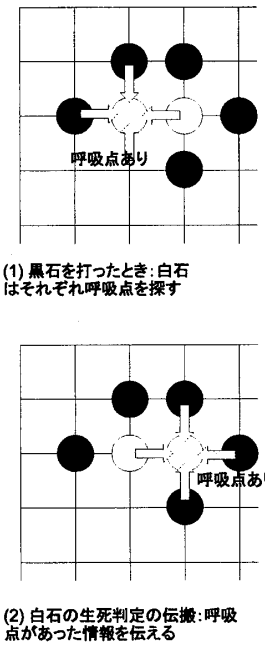

図 1 黒石を打ったときの白石の生死判定の伝搬

5 実験

プレイアウトを実施した環境は次のとおりである．

CPU Core i7 920 2.66GHz
メモリ 6GB
GPU Tesla C1060
ストリーミングプロセッサコアの数 240
プロセッサコアの周波数 1.296GHz
メモリ 4GB

GPU 用の開発環境は NVIDIA 社の CUDA[2] を用いた．9 路盤で初期局面からのプレイアウトにかかる時間を計測し，100 回の平均を算出した．GPU で一度に扱う局面数を増やしていき，単位時間当たりのプレイアウト数の変化を測る．

6 結果

結果を図 2 に示す．処理する局面数を増やすことで単位時間当たりのプレイアウト数が増加することが分かる．局面数を 40,000 と指定したときに毎秒 9,570 プレイアウトであった．

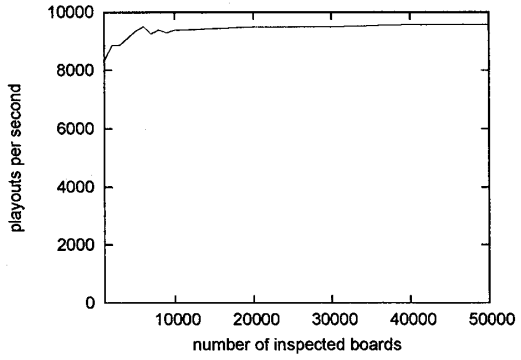

図 2 プレイアウト数と局面の処理数の測定結果

7 結論

GPU を用いた囲碁のプレイアウトを実装しアルゴリズムの工夫により FPGA を用いた先行例と大差ない性能を得た．GPU を用いた先行例に比べ，提案方式の実装はまだ最適化が不十分であるが、改善を進めれば同等の性能を得られるものと思われる．今後はさらなる高速化や UCT への組み込みなどを行いたい．

参考文献

[1] 小泉賢一，石井康雄，美添一樹，三好健文，菅原豊，稲葉真理，平木敬，『FPGA 基板を用いたモンテカルロ碁の高速化』，SWoPP2009
[2] Cuda Zone, http://www.nvidia.co.jp/object/cuda_home_jp.html
[3] Bernd Brügmann, “Monte Carlo Go”, Max Planck Institute of Physics, 1993