論 文 の 内 容 の 要 旨

論文題目　　大規模計算環境を活用する
　　コンピュータゲームプレイヤの研究

氏　　名　　浦　晃

並列計算環境の並列性は年々増加しており、汎用的な計算機をネットワークでつないだクラスタ環境や、必要なだけ料金を支払って利用できるクラウドコンピューティングサービスなど、大規模な計算環境はより身近なものになってきている。大規模計算環境を有効に活用するためには、様々なアプリケーションに対して、大規模計算環境を用いた研究が行われるべきである。本研究では、アプリケーションとしてコンピュータゲームプレイヤを対象に、強いコンピュータプレイヤを作るために重要な、機械学習と探索の両方の観点から、大規模計算環境を活用する手法を提案する。ゲームとしては将棋を対象とした。数百コア以上からなるクラスタ環境を用いた実験により、提案手法の多くについてその有効性を示した。

コンピュータゲームプレイヤでは、ゲームの局面の有利不利を判断するために、評価関数が用いられる。強いコンピュータプレイヤを作成するためには、精度の良い評価関数が求められるため、機械学習を用いて評価関数のパラメータを自動的に精度良く調整する研究が広く行われている。評価関数のパラメータ調整には、上級者の棋譜を用いた教師あり学習が用いられることも多い。一般に、教師あり学習では、訓練データの数が多いほど精度の高い学習が行えるが、上級者の棋譜の数には限りがある。一方で、将棋などのゲームにおいては、コンピュータプレイヤは、人間の上級者と同程度の強さを達成している。

本研究では、より精度の高い評価関数を得るために、指し手のついていない局面を自動的に生成し、それをコンピュータプレイヤに探索させることで得られる指し手と合わせて、訓練データに追加する

ことを提案する。このとき、コンピュータプレイヤは人間の上級者と同じぐらい「良い」指し手を選ぶ必要があり、そのための探索には時間がかかる。さらに、上級者の棋譜に追加して識別可能な程度の性能向上を実現するためには、棋譜と同程度の数の訓練データを追加しなければならない。したがって、この訓練データの作成には一台の計算機では時間がかかりすぎることになる。しかし、複数の局面に対する探索は完全に独立であり、容易に並列化可能であるため、大規模計算環境を有効に活用できる。訓練データに追加する訓練局面として、コンピュータの自己対戦に現れる局面、探索木のリーフノードに現れる局面、棋譜の局面からランダムに手を進めた局面の性質の異なる三種類の局面を生成した。実験には将棋プログラムの激指を用いた。生成した訓練データを上級者の棋譜と合わせて学習を行ったところ、探索木のリーフノードに現れる局面を追加した場合に、より強いコンピュータプレイヤが得られることを確認した。

学習時間は訓練データの数に比例するので、訓練データの数が増えると学習にかかる時間も長くなる。上で述べた提案により、有用な訓練データの数を増加させたとすると、それだけ学習にかかる時間も長くなることになる。学習時に用いるパラメータの調整などを考えると、学習時間は短い方が有利である。近年、大規模データの分析という需要から、機械学習アルゴリズムの中でも、データを一つ処理するごとにパラメータを更新していくオンライン学習アルゴリズムを、分散計算環境で並列化する研究が盛んに行われている。本研究では、評価関数のパラメータ調整にかかる時間を短縮するために、オンライン学習アルゴリズムの並列化手法を適用することを提案する。実際に並列学習を実装し、学習の進行の指標として学習後のプレイヤが選ぶ指し手の棋譜との一致率を用いて評価したところ、学習時間を短縮できることを確認した。将棋の評価関数の学習では、学習中に探索が実行されるため、データの入力にかかる時間よりも計算にかかる時間が支配的であり、一つのデータに対する計算時間も大きくばらつくという特徴がある。このような特徴を持つ学習手法に対する、分散並列化手法の適用可能性を示すことができた。

コンピュータゲームプレイヤの探索を高速化するために、ゲーム木探索手法として広く用いられている$\alpha\beta$アルゴリズムを、大規模計算環境を用いて効率的に並列化することを目的とする。$\alpha\beta$アルゴリズムでは、すでに終了した探索の結果を用いて、他の部分木を枝刈りすることによって探索空間を大きく削減する。しかし、最も効率の良い枝刈りができたとしても、得られた指し手が最善であることを証明するために、探索しなければならないノードが存在する。$\alpha\beta$アルゴリズムでは、できるだけそのようなノードだけを探索することが重要である。しかし、どのノードを訪問しなければならないかは当然ながら探索中には分からないため、実際の探索では探索が必要な部分木を予測してから探索を行う。しかし、あくまでも予測であるため、枝刈りされると予測された部分木の探索が必要になることもあり、逆に枝刈りされないと予測された部分木の探索が実は不要だったことが判明することもある。

本研究では、大規模計算環境を用いて、$\alpha\beta$アルゴリズムを効率的に並列化するために、二つの手法を提案する。一つは、探索が必要だと予測されなかったノードを投機的に探索するために、実行が必要となる可能性に着目してタスクの優先度付けを行う手法であり、もう一つは、探索が必要となる部分木の予測を、探索途中で得られる結果を用いて動的に修正する手法である。

一つめの提案は、大規模計算環境では、探索が必要だと予測される部分木の探索だけでは、アイドル状態のプロセスが生じてしまうという問題点の解消を目的としたものである。一般に、並列に実行可能なタスクを増やすためには、タスクをさらに細かく分割すればよい。しかし、タスクの粒度を小さくしすぎてしまうと、並列化に伴う様々なオーバーヘッドが相対的に大きくなってしまい、性能上のボトルネックとなってしまう。これは、通信遅延時間が大きい分散計算環境では大きな問題となる。この問題点を解決するための方法として、探索が必要だとは予測されなかった部分木の探索を投機的に実行することが挙げられる。枝刈りの予測は外れることもあるため、探索が必要だと予測されなかった部分木であっても、投機的に探索しておくことには意味があるからである。この投機的実行により、タスクの粒度を小さくすることなく、タスクを増やすことができる。ただし、探索が必要だと予測されなかったノードの中でも、必要になる可能性が高いものと低いものが存在するはずである。本研究では、各部分木の探索が必要となる可能性を見積ることで、実行が必要となる可能性の高いタスクから先に実行することを提案する。部分木の探索が必要となる可能性を見積もる方法としては、次の二つの手法を提案した。一つは、親ノードについて、各子ノードが枝刈りされない確率を予測し、親ノードが枝刈りされない確率に乗じることにより、子ノードが枝刈りされない確率を求める手法である。もう一つは、他の部分木の探索結果を確率分布を用いて予測することで、着目している部分木が枝刈りされない確率を計算する手法である。実験では、将棋の探索を並列化し探索時間を比較したが、提案手法の有効性を示すことはできなかった。しかし、ゲーム木のある深さ以上の部分木の探索時間が全て等しいとみなしたシミュレーションによる評価では、ゲーム木の分枝数が小さい場合に提案手法は有効である可能性があることが分かった。

二つめの提案は、探索が必要だと予測された部分木を探索した後で、実は探索が不要だったと判明する可能性を減らすためのものである。逐次探索の場合は、その時点で分かっている全ての情報を用いて、次の部分木の探索を行うことができる。これに対し、大規模な計算資源を十分に活用して並列探索を行うには、逐次探索ならば得ることができたはずの他の部分木の探索結果を得る前に、ある部分木の探索を始めなければならない。したがって、探索が必要となる部分木の予測に探索途中の結果を即座に反映して、探索の必要がない部分木を無駄に探索してしまうことを避けることが重要である。ゲーム木探索では、探索が必要となる部分木の予測に、浅い探索の結果を用いることも多い。よって、並列探索でも、浅い探索の結果を、この予測にすぐに反映させることも重要である。このような予測の修正を行う手法は、すでに先行研究で実装されているが、この予測の修正の性能に対する影響は詳しく評価されておらず、また、この実装も64プロセスまででしか評価されていない。本研究では、この先行研究を参考に、情報の更新が即座に反映されるような、より大規模計算環境に適した実装を行った。性能評価では、探索が必要となる部分木の予測精度の高い人工ゲーム木と激指を用いて生成した将棋のゲーム木の二種類のゲーム木を用いた並列探索を実行した。探索が必要となる部分木の予測を浅い探索の結果を用いて動的に修正することにより、並列探索時間を短くできることを示し、その程度は、予測精度が人工木より低い将棋のゲーム木を用いた場合に大きいことを示した。しかし、理想的な線形な速度向上はまだ実現できていなかった。そこで、速度向上を妨げている原因をいくつかの要因に分けて分析し、得られた結果を概ね説明できるモデルを構築した。