Vol.2021-GI-45 No.16
2021/3/5

モンテカルロ木探索における異なるグループ化の比較

坪倉 弘治[†1,a)]　西野 順二[†1]

概要：本研究は分岐数の多い節点を持つゲームについて、ゲームの特徴に応じて異なるグループ化を行い分岐数を減少させることでモンテカルロ木探索 (MCTS) の探索精度を上げる手法、グループ化を提案し、異なるグループ化の方法を用いることによる探索性能への影響を調査する。グループ化とは、探索木のある節点の子節点をいくつかのグループに分け、各グループ毎にグループ節点を作成して親節点と子節点の間に挿入することである。これにより、下る節点を選択する前にグループにまとめた節点を選択するため探索木の分岐因子が減少する。各枝に同じ範囲の値を割り当てる TiM 木を実験ゲーム木とした実験の結果、グループ化の方法によって探索の性能が変化することが確認できた。グループ化の方法によって探索の性能が変化する理由は、平均利得が近い節点を同じグループにまとめられたかそうではないかにあると考えられる。

Comparison of different groupings in monte carlo tree search

1. はじめに

本研究は、節点のグループ化を行ったモンテカルロ木探索 (MCTS)[1][2] をするとき、そのグループ化の方法の違いが探索精度に与える効果について実験的に検討することを目的とする。これまで、戦略ゲーム (TUBSTAP) のように分岐数の多い接点を持つゲームや不完全情報ゲームにおいて、節点のグループ化と多段化が効果を上げていることが確かめられている [3]。効果がある理由として、グループ化した節点の利得の分布と平均値との関係を調べるため、子節点から先の部分ゲームの利得の平均値が分かる実験ゲーム木 TiM 木を提案し、これを用いて実験を行った結果を示す。

モンテカルロ木探索は囲碁プログラムで既存手法より高い性能を示したことから複数の AI で使用された手法である。しかし、複雑なゲームの場合分岐数が極端に多い節点を持つことがあり、モンテカルロ木探索での探索は困難である。

モンテカルロ木探索で効率的な探索を行うための改善方法は 2 つに大別できる。探索部の改良とランダムシミュレーションの改良である。前者の例は Tree Policy としてよく使われる UCT を改良した UCB1-Tuned[4] や Accerated UCT[5] である。後者の例はランダムシミュレーションを行う代わりにディープラーニングで獲得した状態評価関数を用いる「Alpha GO[6]」である。

探索部の改良のため行われる方法として、探索空間の縮小化がある。非常に数の多い状態数や行動数を減らすことで探索空間を狭めると、複雑なゲームをより簡単なゲームとして捉えることができるため、推定精度の高いモンテカルロ木探索 AI を作成できる。例えば探索空間が連続で状態空間や行動空間が有限にならないゲームでは、空間の離散化を行いゲーム木を先刈りすることで有限にまで探索空間を狭めている。例えば不確定や不完全情報なゲームでは状態の遷移先の情報や遷移する枝の情報が得られないため、ありうる可能性 (インスタンス) を複数作成し確定完全情報ゲームとみなす決定化を行うことで探索空間を狭くしている。ここで、状態数や行動数を減らすことで探索空間を狭めることを探索木の観点から見ると、節点の分岐数、分岐因子が減少していると言える。モンテカルロ木探索の元となったモンテカルロ法では、探索の上限回数としてサンプルサイズの和を固定するとサンプル数が少ないほど精

[1] 情報処理学会
IPSJ, Chiyoda, Tokyo 101–0062, Japan
[†1] 現在，電気通信大学
Presently with The University of Electro-Communications
[a)] t1931089@edu.cc.uec.ac.jp

度が良くなるため、分岐因子を減らすことはモンテカルロ法の精度の向上、つまりモンテカルロ木探索の性能の強化に繋がる。

2. グループ化

節点のグループ化はゲーム木に対してではなく探索木に対して行う。探索木とは、手番プレイヤの手番に対応するゲーム木の節点を根節点とするゲーム木の部分木をとり、その木のある節点 i から別の節点 j まで到達する行動列が複数存在する場合にその節点 j を行動列ごとに異なる節点として、j 以降の部分木を複製した木のことである。

節点のグループ化とは、探索木のある節点の子節点をいくつかのグループに分け、各グループ毎にグループ節点を作成して親節点と子節点の間に挿入することである。

これにより、下る節点を選択する前にグループにまとめた節点を選択するため探索木の分岐因子が減少する。その代わり、間に挿入されたグループ節点の分、子節点と探索木の深さが 1 増加する。また、節点のグループ化を一度行った木ではモンテカルロ木探索を二度行うので、それぞれの探索で行うプレイアウトの数は分割され、合計のプレイアウト回数は同じになるようにする。

探索木の性質から、節点から伸びる枝と子節点は一対一対応が成り立つため、子節点のグループを作成する代わりに枝 (行動) のグループを作成することもできる。子節点のグループと枝 (行動) のグループにも同様に一対一対応が成り立つため、今後節点のグループ化や枝 (行動) のグループ化のことを単にグループ化と呼ぶこととする。グループ化を行った探索木を探索するモンテカルロ木探索をグループ化モンテカルロ木探索 (Grouped Monte Carlo Tree Search)、グループ化 MCTS(Grouped MCTS) と呼ぶ。

2.1 多段のグループ化

一度グループ化を行った後、さらにグループ化を行うこともできる。グループ化を多段に行っていくことでさらに分岐因子が減少し、その分子節点と探索木の深さが増加する。グループ化を行った後の分岐数がすべて等しいとすると、元の節点の分岐数を m、グループ化の回数を n として、グループ化後の分岐数 m' は、

$$m' = \sqrt[n+1]{m}$$

となる。

一方、グループ化によって 1 ターンあたりの深さが増えるためその分意思決定の回数が増え、各意思決定で行うことのできるプレイアウト回数は減る。n 回グループ化を行うとすると意思決定は $n+1$ 回行う必要があるため、それぞれの意思決定のため $n+1$ 回モンテカルロ木探索を行う。したがって、各モンテカルロ木探索で可能なプレイアウト回数 l' は、元々のプレイアウト回数を l としてグループ化を行った後のプレイアウト回数がすべて等しいと仮定すると、

$$n' = \frac{l}{n+1}$$

となる。

分岐数とプレイアウト回数の変化を比較すると、プレイアウト回数は $\frac{1}{n+1}$ という速度で減少しているのに対し、分岐数は $n+1$ 乗根という著しい速度で減少している。モンテカルロ木探索の元となったモンテカルロ法を考えると、その性能は根節点の分岐数 m と探索で可能なプレイアウト回数 n に影響され、$\frac{n}{m}$ が大きくなるほど性能が向上することを考えると、プレイアウト回数より分岐数が非常に早く減少することは性能の向上に貢献すると考えられる。

3. ゲーム木のランダム生成

グループ化による MCTS の性能の変化を検討するため、分岐因子が膨大な複数の異なるゲーム木を用意する。ゲーム木は、葉節点以外の節点が持つ子節点の数が N で、葉節点の深さが揃っている完全 N 分木とする。ゲーム木の枝に対応する行動 a は $a \in \{0, 1, \ldots, N-1\}$ と非負整数で表されるとする。終端節点の利得を利得関数 $f : \mathbb{R} \to [0, 1]$ で与える。根節点から終端節点までの行動列 $a_0, \cdots, a_{d-1}$ が与えられたとき、行動列を N 進数の数とみなして整数値に変換する。終端節点の値 x_t に対し $f(x_t)$ をその利得とする。このゲーム木を実験ゲーム木と呼ぶ。

4. グループ化の方法による差異の評価実験

以下に示す、実験ゲーム木の欠点を解消する TiM 木を提案し、これを用いてグループ化の方法の違いについて検討する。

- ゲーム木の深さが一定だと先手後手の一方だけがゲーム最後の行動をすることができる。
- ゲームの終端が一定の深さでしか登場しない。
- 異なるグループ化方法の比較実験を行うに当たり、人間から見て明らかに優れた手法である/誤った手法であることがわかることが満たされた木が必要。

4.1 TiM 木

作成した実験ゲーム木を TiM 木 (Termination in the Middle Tree) と呼ぶ。TiM 木は次のように作成する。

各枝に同じ範囲の値 (例 : [-0.5,0.5]) を割り当て、枝を下る毎に値の和を取る。その和が 1.0 以上になる/-1.0 以下になるならばその時点で先手プレイヤの勝ち/負けとする。ゲームが終わらなくなるのを防ぐためゲーム木は特定の深さで打ち切り、和の正負で勝ち負け (0 なら引き分け) を決める。

図 1 に TiM 木の例を示した。分岐因子は 5 で各枝に [-0.5,0.5] 範囲の値を等間隔で割り当てる。一番上の節点が

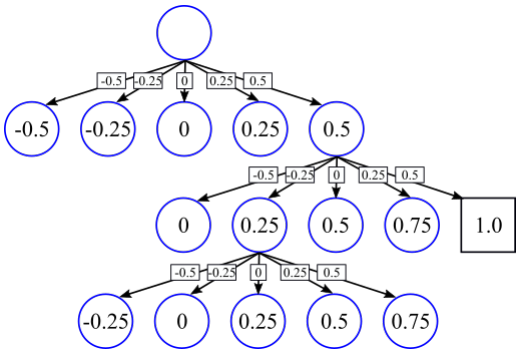

図 1　TiM 木の例

根節点で、節点の中に枝の値の和が示されている。深さ 2 にある黒い正方形の節点は枝の値の和が 1.0 なので、先手プレイヤの勝ちとなり終端節点となる。それ以外の青丸の節点は非終端節点である。このようにゲームの終端となる深さが一定ではないことが分かる。

4.2　実験設定

4.2.1　方法 1：順に分ける

枝が $e_1, \ldots, e_n$ まであるとして、各枝に例えば [-0.5,0.5] の値が割り振られているとすると、左端 e_1 に行くほど負けに近づき、右端 e_n に行くほど勝ちに近づく。なので、左端から順に分割するようなグループ化が最良な方法に近いと考えられる。方法 1 では近い値の節点をグループにまとめているので、グループごとに平均をとってもグループごとに平均値が異なる。

4.2.2　方法 2：両端から分ける

方法 2 ではグループごとの平均値が同じになるようなグループ化をする。それは両端から順に節点を取り出しグループ分けする方法である。

例えば、$e_1, \ldots, e_{16}$ の枝があったとして、これを前記の方法で 2 つのグループに分けると、

$$(e_1, \ldots, e_4, e_{13}, \ldots, e_{16}), (e_5, \ldots, e_{12})$$

となり、それぞれをさらに 2 つのグループに分けると、

$$((e_1, e_2 e_{15}, e_{16}), (e_3, e_4, e_{13}, e_{14}))$$
$$((e_5, e_6, e_{11}, e_{12}), (e_7, e_8, e_9, e_{10}))$$

となる。各枝に割り当てられた値が等間隔であるとする。各グループごとの平均値を見ると同じ深さにあるグループについて平均値が同じになっていることがわかる。その代わりに、左に行くほど分散が大きく、右に行くほど分散が小さくなっている。

評価実験は実験ゲーム木で行い、4.4 節の方法でグループ化を行う GrpSeq と、4.4 節の方法でグループ化を行う GrpEdge、ランダム AI の 3 つの AI を用い、GrpSeq と GrpEdge、GrpSeq とランダム AI、GrpEdge とランダム AI の 3 組で対戦を行った。

- AI の探索手法は UCT 探索とした。
- 対戦は先攻後攻 500 戦ずつ計 1000 戦行う。
- UCT 探索の探索回数 (プレイアウト回数) の制限を変化させ、$\bar{v}$ の変化を調べた。
- ゲーム木のサイズは分岐因子 65536、深さ 4 とした。終端節点の数は $65536^4 \approx 5 \times 10^{19}$

4.3　実験結果

実験の結果を横軸に探索回数の制限であるプレイアウト回数、縦軸にゲームが終局したときの終端節点の利得の平均値 $\bar{v}$ をとったグラフにプロットした。その図を図 2 に示す。図 2 を見ると、GrpSeq はランダム AI との試合で非

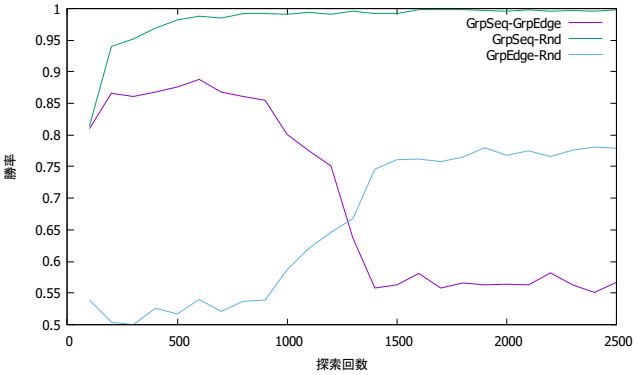

図 2　異なるグループ化と探索回数による実験結果

常に高い勝率を示し、探索回数 100 回では勝率が 0.814 と比較的低かったのが 600 回ほどからはほとんど 1 に近い率になっている。一方 GrpEdge は探索回数 900 回まで勝率が 0.5 から 0.55 の間にとどまり、それから少しづつ勝率を伸ばして 1400 回から 0.75 ほどで横ばいになっている。GrpSeq と GrpEdge の試合に関してはそれぞれの勝率の差の変化に影響されていることがわかる。GrpEdge のランダム AI に対する勝率の上昇に伴い GrpSeq の GrpEdge に対する勝率は下がり、GrpEdge のランダム AI に対する勝率が横ばいになると GrpSeq の GrpEdge に対する勝率も横ばいになっている。

4.4　グループ化方法の違いによる影響

GrpEdge の勝率が探索回数の増加に伴い上昇してはいるものの、どの場合においてもランダム AI に対する勝率について GrpSeq が GrpEdge を大きく上回っている。このことから少なくとも今回の実験設定の TiM 木では GrpSeq が GrpEdge より優れたグループ化方法だと言え、グループ化の方法によって探索の性能が変化することが言える。また、ランダム AI に対する勝率が急速に上がる探索回数が GrpSeq と GrpEdge で異なり、GrpSeq は 100 回から 600 回にかけて、GrpEdge は 900 回から 1400 回にかけて上がっている。この違いにより、より早くランダム AI に

対する勝率が上昇した GrpSeq は探索回数が少ない場合に GrpEdge に対する勝率が高くなる。したがって、勝率の面でよりよいグループ化は特に探索回数が少ない場合においてより有用であると考えられる。

グループ化を行ったモンテカルロ木探索、特に UCT 探索は根節点の分岐数より探索回数が少ない場合にランダム AI とほとんど変わらない挙動をするため、ランダム AI に対する勝率が 0.5 であることはその AI がランダム AI と変わらないことを表していることを考えると、そこから脱する速さが GrpSeq のほうが圧倒的に高いことがわかる。モンテカルロ木探索は近似アルゴリズムであるため、この速度の違いは近似の収束速度の違いと言える。実験の結果から、ゲーム木に対して近似の収束速度が速いグループ化と遅いグループ化が存在すると考えられる。

グループ化の方法によって探索の性能が変化する理由としては、平均利得が近い節点を同じグループにまとめられたかそうではないかにあると考えられる。モンテカルロ木探索では、平均利得 $\overline{x_j}$ を節点 j の子節点からバックアップされた利得の平均で求めている。これは各子節点 c の平均利得 $\overline{x_c}$ を、プレイアウト回数 n_c を重みとした加重平均ともいえる。これには、子節点に突出して高い平均利得を持つ節点があったとしても、それ以外の多数の節点で平均利得が低ければ、親節点の平均利得は平均化され低い数値になってしまうという問題が存在する。よって探索終了時に、本来であれば平均利得が高い子節点を持つ節点ではなく別の異なる節点が最適な節点として選択されうる。k 回のグループ化によって 1 ターン分の行動決定は $k+1$ 回、そしてモンテカルロ木探索も $k+1$ 回繰り返し行う必要があるため、もし最適手から外れた節点を一度でも選択してしまうと、それ以降の探索では決して最適手に到達できなくなってしまう。

UCT 探索においては UCB 値を利用して下る節点を決定するため、十分なプレイアウトを行えれば、平均利得の高い子節点に多くのプレイアウトを振り分けることができ、この問題も解消できる。しかし、割り当てられるプレイアウト回数が少ない場合、すなわち単純に探索時間が短いかプレイアウト回数に対して分岐因子が多すぎる場合には、十分なプレイアウトを行うことができず同じ問題が発生する。

ここで、子節点に平均利得が近い節点が集まっている場合を考えれば、平均利得が子節点と親節点との間で大きく変化することがないので、そのような問題は発生しない。そして、TiM 木では各枝に例えば [-0.5,0.5] の値が割り振られていることから、TiM 木の隣接する節点同士は平均利得が近く、離れている節点同士は平均利得が離れていると考えられる。このことから、節の方法は平均利得が近い節点をまとめているといえる。対して節の方法は子節点の両端から同じグループにまとめるため、非常に離れた節点を同じグループにまとめ、平均利得が離れた節点をまとめているといえる。したがって、グループ化の方法によって探索の性能が変化する理由は、平均利得が近い節点を同じグループにまとめられたかそうではないかにあると考えられる。

5. 結論

本研究では探索空間が広いゲーム、特に分岐因子が非常に多いゲームについて、分岐因子を削減することで MCTS の性能を向上させるグループ化を提案し、異なるグループ化の方法を用いることによる探索性能への影響を調査した。TiM 木を用いた実験の結果、グループ化の方法によって探索の性能が変化することが確認できた。また、勝率の面でよりよいグループ化は特に探索回数が少ない場合においてより有用であると示した。

グループ化の方法によって探索の性能が変化する理由は、平均利得が近い節点を同じグループにまとめられたかそうではないかにあると考えられる。

参考文献

[1] Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

[2] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, Vol. 4, No. 1, pp. 1–43, 2012.

[3] 武藤孝輔, 西野順二. ターン制戦略ゲームにおける UCT とファジィ評価の適用. 日本知能情報ファジィ学会 ファジィシステムシンポジウム 講演論文集 第 31 回ファジィシステムシンポジウム, pp. 226–229. 日本知能情報ファジィ学会, 2015.

[4] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

[5] Junichi Hashimoto, Akihiro Kishimoto, Kazuki Yoshizoe, and Kokolo Ikeda. Accelerated UCT and its application to two-player games. In *Advances in computer games*, pp. 1–12. Springer, 2011.

[6] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, Vol. 529, No. 7587, p. 484, 2016.