モンテカルロ木探索の並列化における仮想報酬の違いによる影響

中村　考希†　　中村　篤祥††

† 北海道大学工学部情報エレクトロニクス学科　〒 060-0814 札幌市北区北 14 条西 9 丁目
†† 北海道大学大学院情報科学研究院　〒 060-0814 札幌市北区北 14 条西 9 丁目
E-mail: †{koki.nakamura,atsu}@ist.hokudai.ac.jp

あらまし　モンテカルロ木探索は、近年のゲーム AI に目覚ましい進歩をもたらした技術の 1 つであるが、限られた時間内により良い手を見つけるために並列化が行われている。並列化を行う場合、他の探索による報酬の逆伝搬が行われる前にゲーム木における展開済みのノードに対するその時点までの平均報酬と報酬取得回数から子ノードの選択を行わなければならず、選択済みではあるが報酬がまだ得られていないものをどのように扱うかにより、並列度及び出力される解の良さが変わってくる。選択済みではあるが報酬がまだ得られていないものに対する仮想報酬として何を与えるのが良いであろうか。並列化手法である MP-MCTS 法に関し、従来法であるバーチャルロスの他、期待値としてその時点までの平均的な値により仮想報酬を与える方法を考え、仮想報酬の違いによる並列度・解の良さへの影響について実験的に調査する。

キーワード　モンテカルロ木探索、並列化、ゲーム AI、バーチャルロス

1　はじめに

ゲーム AI は最近著しく進歩し、囲碁や将棋ではプロ棋士の棋力を超え、他の様々なゲームにおいも人間の最強プレイヤーを凌駕するコンピュータプレイヤーを作り上げることが可能になってきた [1]。そのような進歩をもたらした要素技術の 1 つとして、モンテカルロ木探索 [2] と呼ばれる探索手法の利用が挙げられる。モンテカルロ木探索は、現在の状態から先を深読みした次の一手を探すために用いられ、ゲームの最中に一手ごとに巨大な空間を探索するため、並列化による高速化がよく用いられる 。しかしモンテカルロ木探索には、木構造で表される探索空間において、ノード選択により辿り着いた展開済みの木の葉ノードの状態からのシミュレーションの結果を、全ての親ノードに反映するバックプロパゲーションと呼ばれる処理があり、その処理が終わらないと選択に利用される統計情報が更新されないため、元々の逐次探索と同じ探索をそのまま並列に行うことは不可能である。そこで、シミュレーション 1 回当たりの探索効率が落ちたとしても、シミュレーション回数を増やして探索速度を上げるために、各プロセスは他プロセスの探索によるバックプロパゲーションが行われる前に、それらの結果の反映前の統計からノード選択をする必要がある。

バーチャルロス [3] は選んだ子ノードに対して最悪の結果が得られたとして仮想の損失を報酬としてノードに加え、そのノードの評価値を下げ、他の兄弟ノードが選ばれやすくなるようにし、この問題を解決する 1 つの方法である。これにより複数のプロセスが同じノードを探索する可能性を下げ、並列度を上げる。逆伝搬によりノードに正確な報酬が得られたときバーチャルロスは取り除かれ、正確な報酬に置き換わる。

バーチャルロスは、選択を分散させ並列度を上げる効果がある一方、無駄な探索が増え、探索効果が落ちる可能性がある。本稿では期待値として平均的な値 (TS 確率期待報酬) を仮想報酬として加えて UCB(Upper Confidence Bound) 方策でノード選択を行う方法を考え、並列度・解の良さへの影響について、バーチャルロスと比較して実験的に調査する。

2　従来法 MP-MCTS

2.1　モンテカルロ木探索

モンテカルロ木探索はゲームにおいて、次にどの行動をとるのか決めるためによく用いられる、根付木で表わされる空間の探索手法である。ゲームにおいて探索空間である根付木は現在のゲームの状態を根ノード、親ノードの状態における行動を枝、親ノードの状態で枝の行動が行われた後の状態を子ノードとする木であり、ゲーム木と呼ばれる。モンテカルロ木探索は図 1 に示される以下の 4 つのステップからなる。

・**選択**：根ノードからスタートし、葉ノードに到達するまで子ノードを選択し続ける。多腕バンディット問題の方策（UCB (Upper Confidence Bound[5])）を用いて、以下の UCB(i) 値最大になる子ノード i を選ぶ。

$\mathrm{UCB(i)} = \frac{\mathrm{w_i}}{\mathrm{v_i}} + \mathrm{C}\sqrt{\frac{\ln \mathrm{N}}{\mathrm{v_i}}}$

ただし、w_i は子のノード i を訪問（選択）したときの累積報酬、v_i は子のノード i の訪問回数、N は親ノードの訪問数、C は定数とする。

・**展開**：ノードの訪問回数が閾値を超えたとき、子ノードを全て展開する。

・**シミュレーション**：到着した葉ノードの状態からランダムプ

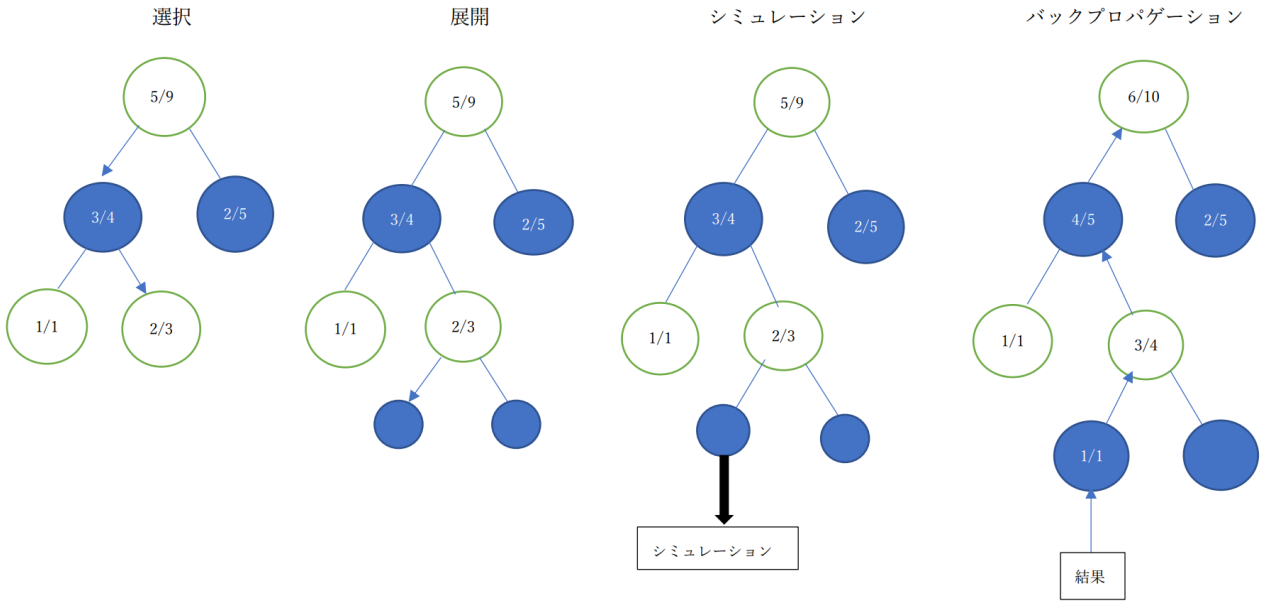


図 1　MCTS 4 つのステップ

レイを始め、勝敗の結果を得る。近年ランダムプレイにより事前に学習した深層ニューラルネットワークの出力で代用することが多い。

・**バックプロパゲーション**：シミュレーションの結果を根ノードまで伝達し、ノードの累積報酬を更新する。

2.2　モンテカルロ木探索の並列化

モンテカルロ木探索の木並列化とは、一つの木に対して複数の探索を同時に行う方法である。各探索は最も望ましい子ノードを選らんで処理を進めるが、他の探索が選択したノードにおいて、それに対する逆伝播が行われる前ノードに対する結果を知らずに選択しなければならない。このとき複数の探索が同じノードを選んでしまうと想定より並列度が上がらない。バーチャルロスはある探索がノードを選んだ後、その探索で得られた結果が最悪だったと仮定し、一時的にノードの累積報酬に加える。これにより UCB の値が下がり、他の探索は他のノードを選びやすくなり、探索が分散される。ノードに結果が返ってきたとき、バーチャルロスは取り除かれて正しい累積報酬に更新される。バーチャルロスを使うとき、選択する際の UCB は以下の式を使う。

$$\mathrm{UCBvl(i)} = \frac{w_i}{v_i + t_i} + C\sqrt{\frac{\ln(V+T)}{v_i + t_i}} \qquad \text{(A)}$$

ただし t_i は子ノード i 以下の部分木を現在探索しているプロセスの数、T は親ノード以下の部分木を現在探索しているプロセス数であり、V は親ノードの逆伝播済の訪問回数とする。

2.3　MP-MCTS

2.3.1　Transposition-table Driven Scheduling

Transposition-table Driven Scheduling (TDS)[6][7] は探索の並列化手法の一つであり、ハッシュ値を用いた負荷分散並列化手法である。並列に行う各プロセスを計算ノードと呼び、各

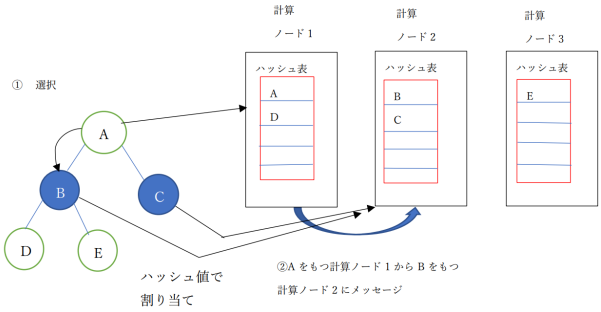


図 2　TDS のノード分散

計算ノードはハッシュ値で割り当てられるゲーム木のノードの部分集合に対する処理を担当する。ノードが新たに作成されるたびにハッシュ値に基づいて計算ノードに割り当て、割り当てられた計算ノードが持つハッシュ表に木のノードの情報を保存する。多くの場合子ノードは異なる計算ノードに割り当てられているため、子ノードの選択を行うたびに他の計算ノードと通信を行う。

2.3.2　TDS-UCT

TDS を用いたモンテカルロ木探索を TDS Upper Confidence bound applied to Trees (TDS-UCT) という。[8]TDS-UCT は以下の 4 つのステップからなる。

・**選択**: 根ノードにおいて UCBvl が最も高い子ノードを選び、担当計算ノードは選んだ子ノードの情報を持つ計算ノード（ハッシュ値により取得）に選択メッセージを送る。例えば図 2 のルートノードがノード B を選んだとき、計算ノード 1 から計算ノード 2 に選択メッセージが送られる。計算ノード 2 はメッセージを受け取ると、ハッシュ表にあるノード B の情報を基に

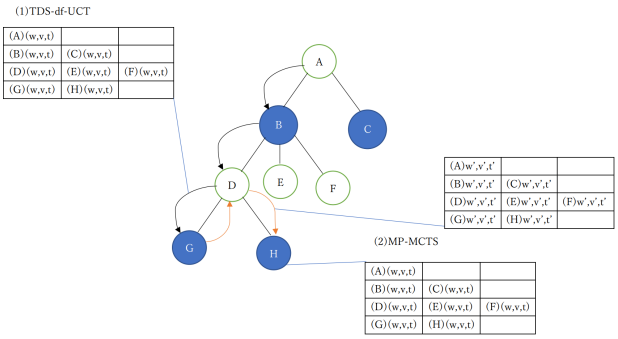


図 3　TDS-df-UCT と MP-MCTS

ノードを選択し、ハッシュ値により担当計算ノードを取得、その計算ノードに選択メッセージを送る。これを葉ノードに到達するまで繰り返す。

・**展開**: 選択メッセージを受け取ってノードが葉ノードだったとき、通常のモンテカルロ木探索同様展開する。

・**シミュレーション**：葉ノードの情報を持つ計算ノードが通常のモンテカルロ木探索同様シミュレーションする。

・**バックプロパゲーション**：シミュレーションが終わると葉ノードは親ノードを持つ計算ノードにバックプロパゲーションメッセージを送る。これをルートノードまで繰り返す。バックプロパゲーションメッセージを受け取ったノードは報酬統計情報を更新する。

2.3.3　TDS-df-UCT

TDS-UCT は通信で処理を進めるため、浅いノードほどバックプロパゲーションメッセージを受け取る機会が増えて通信競合が起こり、性能が制限される。TDS depth first UCT(TDS-df-UCT)[9] は、UCT が一度とても有望な部分を見つけるとプレイアウトが頻繁にそこで行われる経験則から TDS-UCT に深さ優先探索を取り入れた手法である。TDS-df-UCT は各ジョブメッセージに今まで通ってきたパス中にあるノードとその兄弟ノードの通過時の報酬統計情報を持つ。シミュレーション後、ノードの累積報酬と訪問回数を更新し、更新された累積報酬から計算した UCB 値が、対応するメッセージ内の報酬統計情報から得られる他のノードの UCB 値より小さいときのみバックプロパゲーションを行う。小さくない場合は他の子ノードを選択して次の探索に入り、バックプロパゲーションをまとめて行う。

例えば図 3 のノード G でシミュレーションをした後、D に一度バックプロパゲーションし (1) の履歴を確認、D の UCB が最も高ければ続けて H を探索する。これによりバックプロパゲーションする回数が減り、通信の競合が起きにくくなり、TDS-UCT より性能が向上する。

2.3.4　MP-MCTS

TDS-df-UCT はメッセージでのみ履歴を保持し、浅いノードの履歴はそのノードに対して行われるバックプロパゲーションの回数が減るほど更新されづらくなる。そのため最新の結果を他のプロセスに伝達するのに時間がかかり、バックプロパゲーションを頻繁にスキップし、木を広く浅くしてしまう。これを改善するために massively parallel Monte-Carlo Tree Search (MP-MCTS)[8] は図 3 の (2) のように上位ノードの通過時の報酬統計情報をメッセージとノードが持つ。これにより各計算ノードは上位ノードの最新の通過時の報酬統計情報を保持することができ、浅いノードまでバックプロパゲーションが行われなくとも、各ノードから最新の通過時の報酬統計情報を使うことができる。これにより少しメモリの消費量を増やすだけで TDS-UCT の過度なバックプロパゲーションのスキップを抑えながら通信の競合を減らすことができる。そしてシーケンシャル実行時に近い木が得られ、並列探索の性能が大きく向上する。

3　TS 確率期待報酬による仮想報酬

モンテカルロ木並列化探索のバーチャルロスは選択済みで報酬がまだ得られていないノードに仮想的な損失を与えるが、それが無駄な探索を増やしている可能性がある。そこでバーチャルロスではなく、その時点までの平均報酬が期待値となるような仮想報酬の与え方を考える。

TS 確率期待報酬はベータ分布 B (1,1) を期待報酬（期待勝率）の事前分布とした Thompson Sampling[4] を用いる方法である。各ノードの期待報酬の分布を一様分布 B (1,1) とすると事後分布はベータ分布で表現でき、パラメータ α = 勝った回数 + 1　, β = 負けた回数 + 1　のベータ分布 $B(\alpha, \beta)$ となる。この分布からのサンプリングにより得られた値を仮想報酬として利用する。ある探索 A がノード j に到達して子ノード i を選んだとき、子ノードの勝敗数からこの β 分布によって値を生成して記録する。別の探索 B がノード j に到達したとき子ノード i の UCB の値は式 (A) により計算する。ただし

w_i = 子ノードを選んだとき勝った回数 + 子ノード i に対する仮想報酬の和

である。そして探索 B の分の TS 確率期待報酬による仮想報酬を記録し、以降の処理を進める。探索 A に対するバックプロパゲーションが行われたとき、探索 A 分の仮想報酬は取り除かれ、正しい結果を更新する。

4　実 験 方 法

本稿ではバーチャルロスと TS 確率期待報酬による仮想報酬のどちらが並列化探索に適しているかを調べるため、バーチャルロスを用いた MP-MCTS と TS 確率期待報酬を用いた MP-MCTS で 8 × 8 のオセロ対戦をする。どちらもオセロの盤面を入力として木を展開、探索を行う。選択の際使う UCB の値は (A) を採用し、UCB1[10] と同じ値 C = $\sqrt{2}$ とした。シミュレーションはそれを行うノードの盤面から完全ランダムに手を打ち続けるのを終わるまで繰り返し、終わった時点での白黒の数を比較して勝敗を決め、シミュレーションの結果とする。報酬は [0,1] に設定し、勝ったとき 1, 負けたとき 0, 引き分けの

	TS 確率期待報酬	バーチャルロス	引き分け
コア 2	96	103	1
コア 4	101	98	1
コア 8	90	108	2
コア 16	88	112	0
コア 32	77	123	0
コア 64	83	117	0

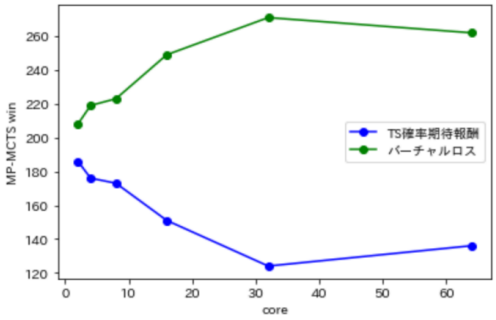
(a)TS 確率期待報酬を用いた MP-MCTS が先攻

	バーチャルロス	TS 確率期待報酬	引き分け
コア 2	105	90	5
コア 4	121	75	4
コア 8	115	83	2
コア 16	137	63	0
コア 32	148	47	5
コア 64	145	53	2

(b)バーチャルロスを用いた MP-MCTS が先攻

	TS 確率期待報酬	バーチャルロス	引き分け
コア 2	186	208	6
コア 4	176	219	5
コア 8	173	223	4
コア 16	151	249	0
コア 32	124	271	5
コア 64	136	262	2

(c)合計 400 戦の勝敗



(d)勝ち数のグラフ

図 4　TS 推定期待報酬とバーチャルロスのオセロ対戦

とき 0.5 となるようにした。実装には python3.7 を使い、通信には python 用の MPI ライブラリである mpi4py を採用した。MP-MCTS は [8] の openreview のサイトからソースコードをダウンロードし、それを改造することによりオセロ用のコードを実装した。

5　結　　果

まずバーチャルロスを用いた MP-MCTS と TS 推定期待報酬を用いた MP-MCTS で 8 × 8 のオセロ対戦を、両者のコア数を 2,4,8,16,32,64 にしてそれぞれ 400 戦行った。探索にかけた時間は全てのコア数で 1 秒とした。その結果を図 4 に示す。なお図 4-(a)(b)(c) のバーチャルロスや TS 確率期待報酬の列の数字は, それらを用いた MP-MCTS の勝利数を表す。また一手打つのにかける探索時間は 1 秒とした。図 4-(d) の図は縦軸に勝利数、横軸に並列計算に用いたコア数を示している。

図 4-(c) から分かるように、全てのコア数で TS 確率期待報酬がバーチャルロスを用いた MP-MCTS の勝利数を下回った。特にコア数が増えるほど勝率の差は広がっていった。図 4-(a) から分かるように TS 確率期待報酬が先攻のときはコア数の増加しても勝率の差はあまり変化していないが、図 4-(b) から TS 確率期待報酬が後攻のときコア数の増加とともに勝率の差が増えている。このことから TS 確率期待報酬が後攻のときは特に、コア数の増加が TS 確率期待報酬を用いる MP-MCTS の探索の性能向上につながっていないことが分かる。

次にバーチャルロスを用いた MP-MCTS とバーチャルロスも TS 確率期待報酬も用いない (UCB)MP-MCTS、TS 確率期待報酬を用いた MP-MCTS と UCB の MP-MCTS でそれぞれ 8 × 8 のオセロ対戦を行った。探索にかけた時間は全てのコア数で 1 秒とし、両者のコア数 2,4,8,16,32,64 にしてそれぞれ 400 戦行った。その結果を図 5 に示す。なお図 5-(a)(b) のバーチャルロスや TS 確率期待報酬、UCB の列の数字はそれらを用いた MP-MCTS の勝利数を表す。

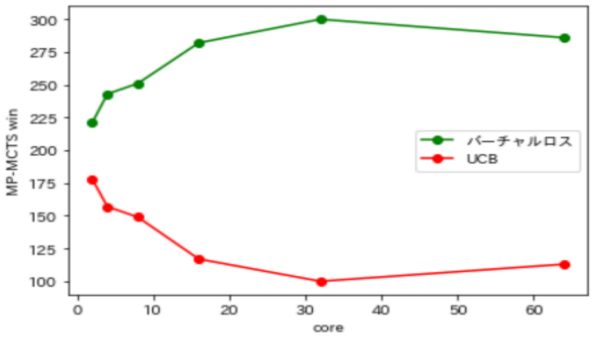
まず図 5-(a) について、バーチャルロスを用いた MP-MCTS が全てのコア数で UCB を用いた MP-MCTS の勝利数を上回った。特にコア数 2〜32 ではコア数を増やすほどバーチャルロスの勝利数が増えている。図 5-(b) について、TS 確率期待報酬を用いた MP-MCTS も全てのコア数で UCB を上回った。ただしバーチャルロスと違ってコア数が増えても顕著な勝率増加はしていないことから、バーチャルロスほどのコア数増加と性能向上のつながりは見られなかった。以上図 4、5 から並列探索の性能はバーチャルロス、TS 確率期待報酬、UCB の順に良いことが分かった。

6　考　　察

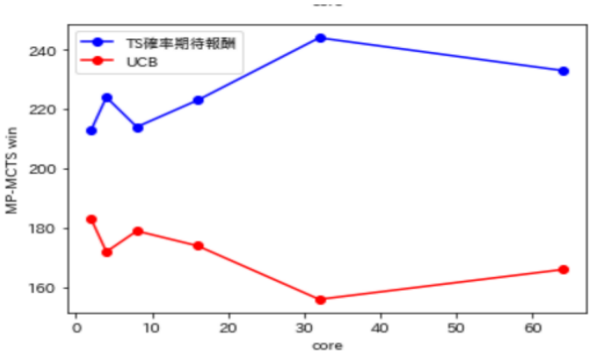
性能の差がどこから来るかを調査するため、盤面・打つ手が決まっている対局で次の手の探索を各 MP-MCTS で行った。各 MP-MCTS は先攻側の探索をシミュレーションが 100 回行われるまで実行し、探索の結果に関わらずあらかじめ設定した手を出力して盤面を進める。後攻側は探索をせず決めた手を出力して盤面を進め、これを一局終わるまで行う。これを 10 回行い、全ての探索における深さ 1 のノードの訪問回数 (=v+t) と探索にかかる時間を測った。そのうち、最も訪問回数が多い

	バーチャルロス	UCB	引き分け
コア 2	221	178	1
コア 4	243	157	0
コア 8	251	149	0
コア 16	282	117	1
コア 32	300	100	0
コア 64	286	113	1

	TS 確率期待報酬	UCB	引き分け
コア 2	213	183	4
コア 4	224	172	4
コア 8	214	179	7
コア 16	223	174	3
コア 32	244	156	0
コア 64	233	166	1

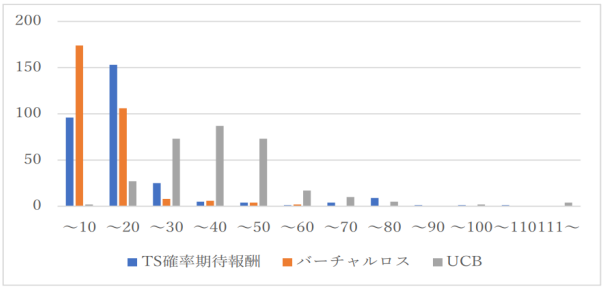


(a)バーチャルロス vsUCB



(b)TS 確率期待報酬 vsUCB

図 5　TS 推定期待報酬と UCB、バーチャルロスと UCB のオセロ対戦



	TS 確率期待報酬	バーチャルロス	UCB
～10	96	174	2
～20	153	106	27
～30	25	8	73
～40	5	6	87
～50	4	4	73
～60	1	2	17
～70	4	0	10
～80	9	0	5
～90	1	0	0
～100	1	0	2
～110	1	0	0
111～	0	0	4

図 6　ノードの訪問回数最大と最小の差

ノードと最も訪問回数が少ないノードの訪問回数の差を図 6 に、最も訪問回数が多いノードと 2 番目に訪問回数が多いノードの訪問回数の差を図 7 に示す。

またバックプロパゲーションの数が 100 回のとき一回の探索にかかる時間を測り、10 局分の集計をとって平均をとった。結果バーチャルロス 0.238585 秒/1 手、TS 確率期待報酬 0.284582 秒/1 手、UCB 0.407567 秒/1 手だった。

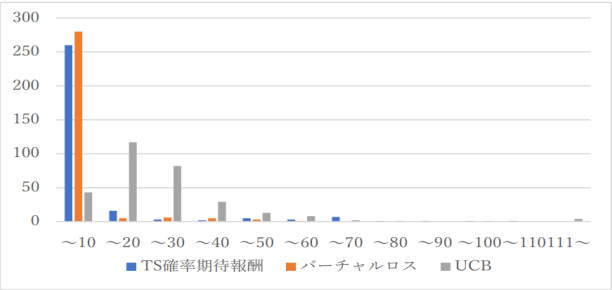
まず図 6 のバーチャルロスでは 0～10 の分布が最も多く、次に 11～20 の分布が多くなり、それ以降はほとんど分布していない。バーチャルロスの探索はほぼ全てのノードで均等に行われていることが多いことを示している。全てのノードを均等に探索することは色々な可能性を模索できるメリットがある一方で比較的見込みの少ないノードも同じ数探索することになるので、その分の探索は多少無駄である可能性がある。

次に図 6 の TS 確率期待報酬では 11～20 の分布が最も多くなり、次に 0～10 の分布が多くなっている。また 21～30 の分布も 25 とそこそこ多く、それ以外の分布も多少ある。これはバーチャルロスよりも探索が偏っていることを示していて、より良い一手を見逃す可能性が上がるが見込みの少ないノードを足切りしてその分探索を集中させている可能性がある。

最後に図 6 の UCB では 31～40 の分布が最も多く、ついで 21～30 と 41～50 が多い正規分布のような形になっている。このことから、極端に一ヵ所へ探索が集中しているかほとんど探索していないノードが存在する場合が多いとわかる。

図 7 のバーチャルロスは訪問回数の差が 0～10 の分布が最も多く、それ以外はほとんど分布していない。バーチャルロスでの最善手と次善手の探索回数はほとんど差がなく、探索が進んで訪問回数の一番多い手が最善手でなくなってもすぐに次の最善手が選べることがわかる。

図 7 の TS 確率期待報酬は 0～10 の分布が最も多く、次に 11～20 が少し分布している。また、それ以外のところにも多少分布している。このことからバーチャルロスとそこまで大差ない次善手の探索を行っていることがわかる。ただし 61～110 の分布も少しずつ存在することから次善手に完全な見切りをつけてしまうことがやや多く、それによってより良い手を見逃している可能性がバーチャルロスより高い。



	TS 確率期待報酬	バーチャルロス	UCB
～10	260	280	43
～20	16	5	117
～30	3	6	82
～40	2	5	29
～50	5	3	13
～60	3	1	8
～70	7	0	2
～80	1	0	1
～90	1	0	0
～100	1	0	1
～110	1	0	0
111～	0	0	4

図 7　ノードの訪問回数最大と 2 番目の差

図 7 の UCB は 11～20 の分布が最も多く、次に 21～30、3 番目に 0～10 の分布が多い。このことから次善手と最善手でも探索に差ができることが多いとわかる。図 6、7 より一ヵ所に探索が集中しすぎることが多く、うまく探索を分散できていない。

MP-MCTS は計算ノードにとどいたメッセージを処理することで探索をすすめるので同じノードに探索が集中するとメッセージが 1 つの計算ノードにたまり、メッセージを処理するまで探索が止まる。したがって一手探索するのに時間がかかるほど 1 つのノードに探索が集中している可能性が高い。バーチャルロスの探索時間 0.238585 秒/1 手を基準に考えると、TS 確率期待報酬はバーチャルロスよりも 1 つのノードに探索が集中することが時間の観点からもわかる。同時に 1 秒での探索におけるシミュレーションの数がバーチャルロスよりも少ないこともわかる。UCB に至ってはバーチャルロス、TS 確率期待報酬と比べてもかなり探索が集中して（偏って）おり、シミュレーションの数も相当減っていることが分かる。

木の形を調べるために、ある盤面をそれぞれの MP-MCTS で 1 回だけ探索を行い、ノードの親子関係を調べた。探索はそれぞれの木の特徴をはっきりさせるために 2 秒行った。その結果を以下に示す。なお、並列計算に用いたコア数は 16 である。

図 8-(a) は各探索でできた木の深さに応じたノードの数で、() の中の数字は子ノードをもつノードの数である。UCB は一手終わるのにかかる時間が長いので十分な探索ができず、深さ 3 のノードを展開するに至らなかった。バーチャルロス、TS 確率期待報酬ともに深さ 3 までノードが存在するのでどちらも十分な探索は行われたといえる。一方深さ 2 のノードの数と各深さでの () の数字を比較すると、TS 確率期待報酬はバーチャルロスと比べて木の幅が狭い。一方深さ 3 のノードの数はバーチャルロスと大差ないことから、まんべんなく探索せず特定のノードの探索を進めていると分かる。図 8-(b) は深さ 1 のノードのうち深さ 3 のノードをもつノードのなかで、いくつ深さ 3 のノードを持っているかを示す。ノード A は TS 確率期待報酬、バーチャルロスともに同じくらいの数の深さ 3 のノードを持っており、ノード C は TS 確率期待報酬のほうが深さ 3 のノードの数が多い。一方でノード B とノード D は TS 推定期待報酬で深さ 3 のノードが存在せず、探索にあまり幅がない。以上より、狭く深い木を作っていることがわかる。盤面が異なれば違う傾向が現れる可能性もあるが、木の幅が狭いためより良い手を見逃しやすく、TS 確率期待報酬 MP-MCTS の性能を制限していると考えられる。

	バーチャルロス	TS 確率期待報酬	UCB
深さ 1	14(9)	14(5)	14(8)
深さ 2	87(10)	52(5)	66(0)
深さ 3	71(0)	62(0)	0(0)

(a)各 MP-MCTS のノード

	ノード A	ノード B	ノード C	ノード D
バーチャルロス	43	8	3	17
TS 確率期待報酬	47	0	15	0

(b)深さ 3 のノードをもつ深さ 1 のノード

図 8　木構造のデータ

以上より 3 つの手法をまとめると、UCB は探索が一ヵ所に集中しやすいためメッセージが停滞し、シミュレーションの数が減って浅く狭い木しか作れず色々な手を探れなくて性能を低下させる。TS 確率期待報酬は探索が特定の箇所に集中し、そのせいで木は深くなるが広く探すことがあまりできず、狭く深い木を作るのでコア数の増加が性能の向上につながらない。バーチャルロスは探索を分散させることで色々な手を探ることができ、それがメッセージの集中も防いで十分なシミュレーションの数を確保し、広く深い木をつくることができる。それがコア数の増加と性能の向上につながっている。

TS 確率期待報酬のコア数増加と性能向上がつながらなかった原因の 1 つにバックプロパゲーションの回数が少ないため探索が深いところで止まり、浅いノードを選ぶ回数が減った可能性がある。図 8 の実験と同じ盤面で各 MP-MCTS を 1 秒探索し、それを 50 回行って TS 確率期待報酬とバーチャルロスのバックプロパゲーションの回数をカウントした。その結果を以下に示す。ここでのバックプロパゲーション 1 回は探索が 1 つ上へ逆伝播する回数である。なお、図 9-(b) の「シミュレーション以外の BP の回数の割合」は BP(バックプロパゲーション)の回数の平均からシミュレーション回数の平均の差をとり、それを BP の回数の平均で割った値である。

図 9-(a) より TS 確率期待報酬はバックプロパゲーションの回数がバーチャルロスよりとても少ない。また図 8-(b) よりシミュレーション回数もバーチャルロスより少ない。シミュレーショ

	TS 確率期待報酬	バーチャルロス	UCB
BP の回数の平均	764.68	1378.04	398.68
シミュレーション回数平均	566.66	947.5	354.16
シミュレーション以外の BP の割合	25.90%	31.25%	11.17%

図 9　各 MP-MCTS のバックプロパゲーションの回数

ン以外の BP の割合でも TS 確率期待報酬はバーチャルロスより値が小さい。これは葉ノード以外でバックプロパゲーションが行われた割合であり、バーチャルロスより探索を根ノード付近に近づける割合が少なかったということである。このことから TS 確率期待報酬は深いノードで探索が集中しており、各計算ノードの負荷バランスが悪くなってシミュレーション、バックプロパゲーションが減り、性能が上がらなかったと考えられる。これはバックプロパゲーションを行うか否かの判断方式にも一因がある。今回 TS 確率期待報酬では 3 節の子ノードを選ぶ UCB の式を用いてバックプロパゲーションを行うか否かを判断した。これが理由で根ノード付近に探索を戻すバックプロパゲーションの回数が減った可能性が高い。探索が浅いノードに戻ってきにくくなり、幅広く探索できなくなったと考えられる。したがって TS 確率期待報酬 MP-MCTS のバックプロパゲーションを行うか否かの判断を別の適したものに置き換えることで性能の改善が行われる可能性がある。。

7 終わりに

バーチャルロスの代わりに TS を用いた仮想報酬を使うことによって MP-MCTS の性能が向上するかを検証した。既存のバーチャルロスが最も良い結果になったが、TS 確率期待報酬を MP-MCTS に適用する上での改善点も見つかった。今後は仮想報酬の値の決め方のみでなく、バックプロパゲーションを行うか否かの判断の仕方、並列化法などシステム全体としてうまく機能させ性能を向上させる方法の開発に取り組んでいきたい。

文　献

[1] Julian Schrittwieser,Ioannis Antonoglou, Thomas Hubert,Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap,David Silver. "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model". In *Nature* volume 588, p604–609 (2020).

[2] R. Coulom. "Efficient selectivity and backup operators in Monte-Carlo tree search." In *Proceedings of the 5th International Conference on Computers and Games (CG)*, p72–83 (2006).

[3] Guillaume M.J-B. Chaslot, Mark H.M. Winands, and H. Jaap van den Herik. "Parallel Monte-Carlo tree search." In *Proceedings of the 6th International Conference on Computers and Games (CG)*, volume 5131 of *Lecture Notes in Computer Science*, p60–71 (2008).

[4] Shipra Agrawal and Navin Goyal. "Analysis of Thompson Sampling for the Multi-armed Bandit Problem", In Proceedings of *the 25th Annual Conference on Learning Theory*,PMLR 23 p39.1-39.26 (2012)

[5] P. Auer, N. Cesa-Bianchi, and P. Fischer. "Finite time analysis of the multi-armed bandit problem". In *Machine Learning*, 47 p.235–256 (2002).

[6] Romein, J.W., Bal, H.E., Schaeffer, J. and Plaat, A. "Analysis of Transposition-Table-Driven Work Scheduling in Distributed Search", In *IEEE Trans. Parallel Distrib. Syst*, Vol.13, No.5, p.447–459 (2002).

[7] Romein, J.W., Plaat, A., Bal, H. E. and Schaeffer, J. "Transposition Table Driven Work Scheduling in Distributed Search". In *16th National Conference on Artificial Intelligence (AAAI' 99*, p725–731 (1999).

[8] Xiufeng Yang,Tanuj Kr Aasawat,Kazuki Yoshizoe. "Practical Massively Parallel Monte-Carlo Tree Search Applied to Molecular Design", Published as a conference paper at ICLR 2021, p1-19 (2021).(2023 年 1 月 7 日取得、https://openreview.net/forum?id=6k7VdojAIK)

[9] K. Yoshizoe, A. Kishimoto, T. Kaneko, H. Yoshimoto, and Y. Ishikawa. "Scalable distributed montecarlo tree search". In *Proceedings of the Fourth Annual Symposium on Combinatorial Search* (2011).

[10] Auer, P., Cesa-Bianchi, N. and Fischer, P. "Finite-time Analysis of the Multiarmed Bandit Problem" In *Machine Learning*,Vol. 47, No. 2-3, pp. 235–256 (2002).