General Game Playing における 類似盤面を利用したモンテカルロ木探索性能向上の試み

上宮 佳晃[1]　横山 大作[1]

概要：General Game Playing(GGP) とは、初見のゲームを、与えられるゲームルールだけで、高い精度でプレイするプログラムを実現することを目標とした問題カテゴリである。 様々な種類の未知のゲームをプレイするため、正確な評価関数を必要としないモンテカルロ木探索 (MCTS) を用いたプログラムが主流となっている。だが、汎用の実装を用いなければならないため、シミュレーション回数を増やさずに、MCTS の効率を向上させることが求められる。本論文では、GGP において少ないシミュレーション回数で MCTS の探索効率を向上させるために、シミュレーションの過程で得られる探索経験を利用した手法の有効性を検証し、その性能について議論する。実験により、拡張する盤面に対して探索経験から一致する盤面の情報を取得し、累積報酬と訪問回数を加算する手法は MCTS の探索効率を向上させることがわかった。

Towards an improvement of simulation strategy for Monte Carlo tree search using board similarities in General Game Playing

Yoshiaki Uemiya[1]　Daisaku Yokoyama[1]

Abstract: General Game Playing(GGP) is a problem category what the goal is to create programs capable of playing the game which is never seen with high performance. Because of playing a wide variety of unknown games, using Monte Carlo Tree Search program which is not requires accurate evaluation function is main approach on GGP. However since general implementation must be used, it is needed to improve MCTS performance without increased simulation times. In this paper, to improve MCTS performance at small number of simulation times on GGP, we tested the effectiveness of the method used exploration experience which is created on simulationm and discuss the performance. By experimentation, the approach of to get information on the matching board from the exploration experience, and to add accumulated reward and visit count on expand node improves MCTS performance.

1. はじめに

これまでに研究されてきたゲーム AI の多くは、特定のゲームを高精度でプレイすることができるようなプログラムであった。このようなゲーム AI は、プレイするゲームについて開発者が研究した専門的な知識に大きく依存しており、用いられるアルゴリズムも扱うゲームに合わせて改良されている。

一方 General Game Playing（GGP）は、初見の様々なゲームに対して高い精度でプレイするプログラムの実現を目標とした問題カテゴリである。[1] 事前にプレイするゲームが決まっていないため、ゲーム固有の知識や技術に頼ることなく、プレイ中にそれらをゲーム AI が自分自身で学習する必要がある。そのため、ゲーム固有の知識や正確な評価関数を必要とせずにツリーを探索・構築することができる MCTS が GGP に特に適した手法となっている。

GGP のゲーム AI の性能向上のためには、MCTS のシミュレーション回数を少なくすることが求められる。しかし、GGP では汎用の実装にしなければならないため、少ないシミュレーション回数で MCTS の効率を向上させることが求められる。

本研究では、未知の様々な形式のゲームを扱う GGP に

[1] 明治大学大学院理工学研究科情報科学専攻
Department of Information Sciences Graduate School of Science and Technology, Meiji University

おいて、MCTS を構築する過程で得られた探索経験を利用して、少ないシミュレーション回数で MCTS の探索効率を向上させることを目指す。探索経験は、盤面表現の文字列を huffman 符号でエンコードしたビット列をキーとして、盤面の状態や報酬、訪問回数から成るハッシュ値を紐づけて作成した。MCTS の拡張ステップにおいて、有益な展開となる盤面へ遷移させるためや、拡張する盤面に情報を付加するために、拡張する盤面と一致、または類似する盤面を探索経験から取得し活用した。

探索経験を利用して ε -greedy 法を用いて拡張する盤面の選択を工夫する方法は、MCTS の探索効率を向上させることができなかった。拡張する盤面に探索経験から盤面の累積報酬と訪問回数を加算する方法は、拡張する盤面と探索経験の盤面が一致する場合には、MCTS の探索効率を向上させることができた。しかし、類似度を示すビット単位のハミング距離が離れていくにつれて MCTS の探索効率が著しく下がっていくことも示された。また、探索経験から類似盤面を取得する処理時間は、ハミング距離が 2 までの設定においては 1 割程度のオーバーヘッドに抑えられた

2. 関連研究

2.1 Game Description Language（GDL）

GGP で扱うゲームルールは論理プログラミング言語の一種である Game Description Language（GDL）[4] で表現される。ゲームのルールには、各プレイヤの役割や合法手によってゲームの状態がどのように変化するのかといったことが記述されている。

GDL で表現されるゲームの盤面状態には、主に駒の種類やその駒の座標、手番、ターン数といった情報が含まれている。例えば Tic-Tac-Toe というゲームにおいて図 1 のような盤面状態は図 2 のように GDL で表現される。盤面ごとに各プレイヤは論理的なルールで記述された合法手をゲームマスターに送信する。このとき、手番でないプレイヤは noop（no operation）というゲームの状態に影響しない信号を送る。

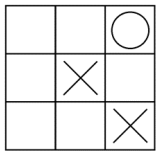

図 1 tic-tac-toe の盤面

```
( true ( cell 1 1 b ) ), ( true ( cell 1 2 b ) ),
( true ( cell 1 3 b ) ), ( true ( cell 2 1 b ) ),
( true ( cell 2 2 x ) ), ( true ( cell 2 3 b ) ),
( true ( cell 3 1 o ) ), ( true ( cell 3 2 b ) ),
( true ( cell 3 3 x ) ),  ( true ( control oplayer ) )
```

図 2 GDL で表現された盤面情報

2.2 MCTS

メモリ上に構成されるツリーの各ノードは GDL で表現されるゲームの盤面状態に対応する。またルートノードは現在の盤面状態となる。本研究での基本的な MCTS の構築は次の 4 つのステップから構成される。

(1) 選択：構築している木においてルートノードからリーフノードまでたどっていく。このとき、ノードの選択には最善手と新しい手の選択のバランスをとる必要があるため後述の UCT を用いる。

(2) プレイアウト：リーフノードからゲームの終局状態までゲームをランダムにシミュレートする。

(3) 拡張：プレイアウトでリーフノードの次に選択したノードを新たなリーフノードとして追加する。ノードの追加を 1 つにすることで、過剰にメモリを増やすことを防ぐことができる。

(4) バックプロパゲーション：訪問回数やプレイアウトで得られた報酬をリーフノードからルートノードまでバックプロパゲーション して反映させる。

2.3 UCT

Upper Confidence bounds applied to Tree（UCT）[5] は、MCTS の選択ステップにおいて探索した手の内の最善手と探索回数が少ない手の選択のバランスをとるアルゴリズムである。ノード s において以下の式の値が最大となるような子ノード a を選択するようにする。

$$UCT = Q(s,a) + C\sqrt{\frac{\log N(s)}{N(s,a)}} \quad (1)$$

ここで N(s) は s の訪問回数、N(s,a) は s においてノード a が選択された回数である。また、ノード a が未訪問の場合は UCT =∞として、すべてのノードが必ず一度は訪問されるようにする。C はパラメータであり、C を大きくすると多くの探索が可能となる。Q(s,a) は、ノード a が選択された場合の平均報酬を示している。よって第一項は最善手の探索を、第 2 項は訪問回数が少ない子ノードの探索をするようにし、パラメータ C によって二つのバランスをとるようにする。

2.4 GGP に用いられてきた手法とその性能

本項では、GGP において MCTS の探索効率を高めるために用いられてきた手法について紹介する。いずれもプレイアウトステップにおいて最善手の選択を行うために過去のシミュレーションから得られる探索経験を活用した手法となっており、MCTS の探索効率を高めている。これらの手法では目的の盤面に対して、探索経験から同じ盤面の情報を取得し利用しているが、類似盤面の情報を活用した手法とはなっていない。

2.4.1 Move-Average Sampling Technique(MAST)

2008 年に AAAI の GGP の大会で優勝した Cadiaplayer に搭載されていた探索制御手法である [6]。終局状態からルートノードへと報酬と訪問回数がバックプロパゲーションされる際に、出現した各合法手 a に対する平均報酬 Q(a)

を更新し記録しておく。よって、ゲームの状態に関係なく良い結果をもたらす合法手は、最終的に平均報酬が高くなる。そして、プレイアウトステップにおいて合法手の選択をする際には、式 2 のギブスサンプリングを用いて平均報酬が高い手を選択する確率が高くなるようにする。

$$P(a) = \frac{e^{Q(a)/\tau}}{\sum_{b=1}^{n} e^{Q(b)/\tau}} \quad (2)$$

このとき P(a) は、あるプレイアウトの盤面状態において合法手 a を選択する確率であり、Q(a) は合法手 a による平均報酬である。また τ はパラメータである。[7] では、ランダムにプレイアウトを行う MCTS プレイヤに対して MAST を用いた MCTS プレイヤは Checkers では 54.83%、Othello では 58.67% 、Breakthrough では 88.67% の勝率を示していた。

2.5 N-gram Slection Techinique(NST)

効率的なプレイアウトを行う方法として、プレイアウトで選択した手およびその平均報酬を記録し、それを用いたシミュレーション戦略が考えられてきた。出現した単語に基づいて次の単語を予測する統計モデルで有名な N-gram は、コンピュータゲームにおいても様々な研究で利用されている。2012 年に GGP の大会で優勝した Cadiaplayer は、よりゲームプレイの精度を向上させるために MCTS のシミュレーション戦略で N-Gram selection Technique（NST）[10] という技術を使用している。NST では、平均報酬が記録された長さ 1,2,3 の連続した手のシーケンスを用いてプレイアウトの推測を行う。これらのシーケンスは MCTS で終局状態に到達した後、シミュレーションで出現した長さ 1，2，3 のすべての手の組合せを抽出することで形成される。

プレイアウトの過程では、どの手を選択するかを決定するために N-Gram が使用される。ノード N から次の手 n を選択する際に、長さ L=1,2,3 のシーケンスを決定する。このとき L ＝ 1 のシーケンスは、n 単体、L ＝ 2 のシーケンスは、n と 1 個前のノードである N、L ＝ 3 のシーケンスは、n と 1 個前のノード N と 2 個前のノードとなる。それぞれのシーケンスには平均報酬が存在し、三つのシーケンスの平均報酬の非加重平均を用いて n のスコアを計算する。全ての合法手に対してこのスコアを計算し、これらのスコアを ϵ -greedy 法に適用してどの手を選択するか決定する。

3. MCTS の探索効率向上に向けた検討

本研究では MCTS の拡張ステップにおいて、探索経験に含まれる盤面情報を以下の二つの方法で活用した。

- 拡張する盤面の選択
  次に遷移することができる盤面の中から平均報酬の高い盤面を探索経験から選択し、 ε -greedy 法によってその盤面を優先して拡張する方法。
- 拡張する盤面への情報付加
  拡張する盤面に、探索経験に保存されている盤面の累積報酬と訪問回数を加算する方法。

これらの利用方法に対して、拡張する盤面と探索経験に保存した盤面との類似度の距離を 0 から 2 まで変化させ、類似度の距離以下の盤面情報を利用した時の性能をそれぞれ検証した。本項では、探索経験の作成や二つの手法を実装した GGP プレイヤについて説明を行う。

3.1 探索経験

GGP では、様々な未知のゲームに対して適用できる汎用的な手法が必要となる。ゲームによって盤面の文字列が異なるため、huffman 符号の手法を用いて、盤面の文字列表現からハッシュキーとなるビット列を作成し、盤面情報と紐づけて探索経験を作成した。

3.1.1 盤面のエンコード

初めに、GDL で表現された盤面情報から駒の位置を示す座標の情報だけを取り出し、空白マスの情報も含めて座標順にソートした文字列に変換する前処理を行う。そして、事前に行われる 500 回分のシミュレーションで発生した盤面情報から文字ごとに出現頻度を計測し huffman 符号でビット列にエンコードを行う。得られたビット列は 64 の倍数の長さになるように、適宜 0 でパディングを行う。ビット列の長さが 65 以上となった場合は、64 ビットごとに分割し、XOR 演算で複数のビット列を一つにまとめる。

例として図 1 の盤面のエンコードを示す。空白を表す文字を'#' として、事前のシミュレーションによって huffman 符号が'x'=10, 'o'=11, '#'=0 と定められたとする。GDL の文字列で表現された盤面状態 (図 2) を前処理によって図 3 に変換する。そして huffman 符号に従ってビット列に変換し、64 桁になるように 0 でパディングを行うことで、盤面のエンコーディングが完了となる。

```
(1,1,#),(1,2,#),(1,3,#),(2,1,#),(2,2,x),(2,3,#),
(3,1,o),(3,2,#),(3,3,x)
```

図 3 図 2 の GDL で表現された盤面情報を前処理して得られる文字列

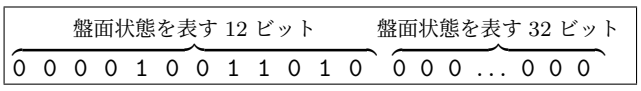

図 4 図 3 の文字列を huffman 符号でエンコードして得られるビット列

3.1.2 探索経験の作成

MCTS のバックプロパゲーションの過程で探索経験を作成していく。選択及びプレイアウトの過程で発生した盤面に対して、3.1.1 の方法でハッシュキーを作成し、そのシミュレーションで得られた報酬や訪問回数とその盤面状態をハッシュ値として探索経験に保存をしていく。すでに同じハッシュキーがある場合は、対応するハッシュ値の報酬と訪問回数を累積していく。

3.1.3 盤面の類似度

本研究では、64 ビットで表現されるハッシュキーを用いて盤面の類似度を判定する。類似度を表すために、二つのハッシュキーの異なるビット数を数えるハミング距離を用いる。二つのハッシュキーが完全一致の場合は、異なるビット数は 0 であり、ハミング距離は 0 となる。よって、異なるビット数が少ないほどハミング距離の値が小さくなり、二つの盤面は似ていると判定される。任意の局面に対し、指定したハミング距離以下の類似盤面は一定の時間で検索できる。

図 1 と図 5 の場合であれば、○のコマが一マス分ズレている違いがあるが、それぞれ盤面の文字列をエンコードすると図 6 のようなビット列となる。二つの異なるビット列の数は 2 となるため、ハミング距離は 2 となる。

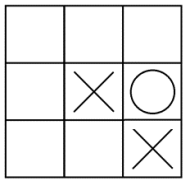

図 5 図 1 から一マスコマがズレた tic-tac-toe の盤面

図 1：000010011010（盤面状態のビット列） 000...000（0 のパディング）
図 5：000010001110（盤面状態のビット列） 000...000（0 のパディング）

図 6 図 1 と図 5 の盤面の文字列をエンコードして得られるビット列

3.2 GGP プレイヤ

プレイヤの構築のためにスタンフォード大学のロジックグループが標準化した GGP のプラットフォーム [2] を使用した。GGP ライブラリを用いて MCTS によるプレイヤを作成し、MCTS の探索の効率を向上させるために以下の二つの方法を検討する。

3.2.1 Expand node selection プレイヤ (ES プレイヤ)

拡張する盤面の選択に探索経験の情報を活用する。構築している MCTS のリーフノードからプレイアウトを行う際に、リーフノードの次に遷移することができるいくつかの盤面に対して、指定したハミング距離以下の類似盤面を探索経験から取得する。複数取得した場合は、類似盤面に紐付けて記録されている平均報酬の選択確率で盤面を決定する。そして、 ε-greedy 法で選択した盤面へ遷移させるか決定する。

3.2.2 Expand node with extra information プレイヤ (EE プレイヤ)

MCTS の拡張ステップにおいて、拡張するノードに対して探索履歴から指定したハミング距離以下の類似盤面の情報を取得する。そして、拡張するノードに対して元々与えようとしていた報酬と訪問回数に加えて、取得した類似盤面に紐付けて記録されているそれらも加算して与えることにする。

4. 実験

本項では、MCTS, UCT を元に構築した MCTS プレイヤに対して、3.2.1 の拡張する盤面の選択の工夫を行う ES プレイヤと、3.2.2 の拡張する盤面に対して情報を付加する EE プレイヤをそれぞれ対戦させ、構築したプレイヤの性能を検証する。

4.1 実験設定

4 種類のゲームを先行と後攻を交互に変えて 500 回対戦を行う。本研究では、各手法が MCTS の探索効率を高めるのに効果的であるかを確かめるために、一手当たりの時間を考慮せずに全てのプレイヤのプレイアウト回数を 500 回に統一し、構築する MCTS のツリーのノード数が同じになるように実験を行う。

4.2 使用するゲーム

MCTS プレイヤと ES プレイヤとの対戦では、Connect Four, BreakThrough, TTCC4 の対戦を行い、MCTS プレイヤと EE プレイヤとの対戦では以下の全てのゲームの対戦を行なった。各ゲームのルールと本研究における特徴を以下に示す。

- `Connect Four`
  重力付き四目並べ。6 × 8 の盤面にいずれかの列を選択すると自分のディスクが落下し積み重なる。自分のディスクを縦・横・斜めいずれかに 4 つ直線状に並べたプレイヤの勝ちとなる。
  ゲームを通して自分が選択できる合法手が 8 手以下であり、終局状態が 48 ターン以内となるため、同じプレイアウト回数では他のゲームと比べてシミュレーション速度が早く、構築される MCTS の探索精度も高くなる。
- `BreakThrough`
  6 × 6 の盤面に各プレイヤの手前 2 行にポーンを並べた状態からゲームを始める。毎ターン自分のポーンを進めていき、先に相手側の端の 1 行にポーンを到達さ

せたプレイヤの勝ちとなる。敵の駒の位置が斜めでないと捕獲できない。

- `TTCC4`
  チェス、チェッカー、Tic-Tac-Toe, Connect Four を組み合わせたゲームである。各プレイヤはチェスのポーン、ナイト、チェッカーのキングを所有し、毎ターン 5 × 5 の盤面に駒を配置または移動させるか、真ん中 3 列に Connect Four のようにディスクを落とすことができる。中央 3 × 3 の範囲で 3 つの駒を縦・横・斜めのどれかで連続して置いたプレイヤの勝ちとなる。コマの種類が多いため、作成される huffman 符号の種類も多くなる。よって、コマの場所が一マス違うことで似ている盤面でも huffman 符号により作成されるハッシュキーが大きく異なる可能性がある。
- `Connect Five`
  五目並べ。8 × 8 の盤面に各プレイヤーは交互にコマを自由においていく。自分のコマを縦・横・斜めいずれかに 5 つ直線状に並べたプレイヤの勝ちとなる。
  一手当たりの合法手の数が多いため、MCTS があまり深く構築されない。
- `Sheep and Wolf`
  二人のプレイヤーに羊と狼の役割が割り振られる。8 × 8 の盤面に各プレイヤの手前 1 行に対して、狼はコマを真ん中に一つ、羊は狼のコマの列と被らないようにコマを一マス空けて等間隔に 4 つ並べた状態からゲームを始める。互いに斜めに移動することしかできず、羊は前進しかできないのに対して狼は後退することができる。狼のコマが羊のコマと壁に囲まれることで動けなくなると羊側の勝利となり、狼が羊のコマのどれか一つより奥に進んだ場合には狼側の勝利となる。
  ゲームを通して盤面全体に対して空白マスの多いゲームとなるため、盤面のハッシュキーを作成すると、同じ文字列が連続して現れる割合が多くなる。

4.3 実験結果

4.3.1 Expand node selection の評価

初めに、類似盤面を利用するか決める際に用いられる ε-greedy 法の最適なパラメータを調べるために、Connect Four で ε の値を 0.0〜1.0 まで変化させて実験を行った。表 1 に結果を示す。Connect Four において、 ε の値が大きくなるほど類似盤面を利用する割合は低くなり、元の MCTS の性能と変わらなくなるので勝率は 5 割近くなった。しかし、 ε の値を小さくし類似盤面の利用する割合を高くしても、ES プレイヤは勝ち越したものの性能が格段に上がることはなかった。また、性能差が現れなかった ε =0.2 に対して他のゲームでも対戦を行なったところ表 2 の結果となり、全ての対戦において符号検定での P 値が 5%以上となったため、拡張する盤面の選択に探索経験の情報を活用する本研究の手法は、MCTS の探索効率をあげることはできなかった。

表 1 Connect Four における MCTS プレイヤに対して ε の値を変化させた時の ES プレイヤの成績

ε	勝ち	負け	引き分け	符号検定 P 値
0.2	249	241	10	0.752
0.4	231	260	9	0.206
0.6	234	255	19	0.366
0.8	254	241	15	0.55

表 2 MCTS プレイヤに対して ε =0.2 の時の ES プレイヤの成績

	勝ち	負け	引き分け	符号検定 P 値
Connect Four	249	241	10	0.752
Break Through	244	256	0	0.623
TTCC4	266	231	3	0.127

4.3.2 Expand node with extra information の評価

対戦結果を表 3 に示す。探索経験から得られるハッシュキーと拡張する盤面のハッシュキーとのハミング距離が 0 の時、すなわち完全一致の場合は Connect Five を除いて全てのゲームにおいて P 値が 5%を下回り有意差ありとなった。しかし、ハミング距離が離れた盤面を活用していくに従って勝率は減少した (図 7)。このことから、拡張する盤面に類似する盤面が持つ、報酬や訪問回数を単純に加算するだけでは、MCTS の探索効率をあげることができないと考えられる。Connect Five での性能が上がらなかった理由としては、他のゲームと比べて一手当たりの合法手の数が多いため、一回のシミュレーションで取得した探索経験をハミング距離 0 の完全一致で再度活用する機会が少ないためだと考えられる。

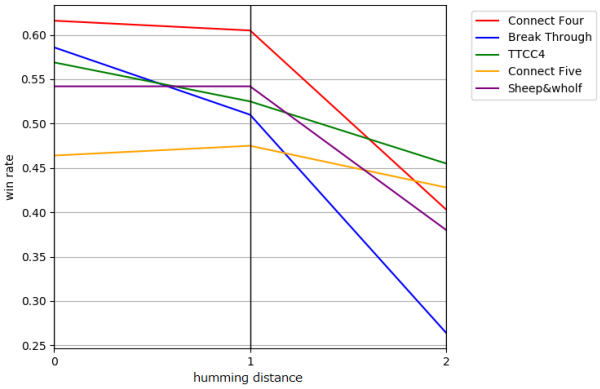

図 7 ハミング距離に対する各ゲームの勝率の推移

次に、一手ごとにどれくらい探索経験から類似盤面が見つかるかについて計測を行なった。Connect Four において、ES プレイヤと MCTS プレイヤを引き分けとなる 48

表 3 MCTS プレイヤに対して拡張ノードに情報を付加する EE プレイヤの成績
(太字は統計的に優位な結果を示す)

	距離 0 のみ				距離 1 以下				距離 2 以下			
	勝ち	負け	引き分け	符号検定 P 値	勝ち	負け	引き分け	符号検定 P 値	勝ち	負け	引き分け	符号検定 P 値
Connect Four	**300**	**187**	**13**	3.45e-07	**296**	**193**	**11**	3.68e-06	**198**	**293**	**9**	2.09e-05
Break Through	**293**	**207**	**0**	1.39e-04	255	245	0	0.68	**132**	**368**	**0**	8.34e-27
TTCC4	**282**	**214**	**4**	2.59e-03	**260**	**235**	**5**	2.81e-03	**225**	**270**	**5**	4.79e-02
Connect Five	232	268	0	0.12	237	262	1	0.283	**214**	**286**	**0**	1.47e-3
Sheep and Wolf	271	229	0	6.66e-2	271	229	0	6.66e-2	**190**	**310**	**0**	8.95e-8

手目まで対戦させ、探索経験を使用した回数をハミング距離ごとに計測した結果を図に示す。図 8 よりハミング距離が 1 の時は探索経験からあまり情報を取得していないことがわかる。そのため表 3 より、ハミング距離 1 以下の探索経験を活用した時の性能は、ハミング距離 0 のみの場合と比べて探索効率が格段に落ちることがなかったのだと考えられる。

また、ハミング距離 2 の時の探索経験からの取得回数はハミング距離 0 の時と比べて約 4〜10 倍の取得率であった。本研究では、GDL で表現される盤面状態の文字列から huffman 符号を用いてハッシュキーを作成しているため、ハミング距離が小さければ似ている盤面状態であると仮定して実験を行なった。しかし、huffman 符号を用いた盤面の文字列のエンコードでは、ある座標の空白マスにコマが置かれた場合はその座標以降のビット列がズレる可能性がある。例えば TTCC4 において、コマの種類が自分と相手のを区別して 8 種類あるため、一部のコマの huffman 符号の桁数が 4 桁になる場合がある。空白マスの huffman 符号は 0 の一桁なので、空白マスにコマが置かれるとそれ以降のビット列が 4 桁ズレる可能性がある。そのため、huffman 符号からエンコードしたビット列の差を用いて類似盤面を判定する本研究の手法は、あまり盤面の類似度を比較することができていない可能性がある。この時、探索経験から得られるハミング距離が 2 の盤面は、拡張する盤面の状態と類似していないものが多く、それらの情報を付加したことで MCTS の探索効率を下げてしまっているのではないかと考えられる。。

4.3.3 探索経験から類似盤面を取得する時間について

MCTS プレイヤとハミング距離 0,1,2 の類似盤面を探索経験から活用する EE プレイヤの処理速度をそれぞれ比較する。図 9 の Connect Four の盤面において、次の合法手を選択するまでの時間を計測した。MCTS プレイヤの結果を基準としてハミング距離 0,1,2 の実行時間比率を求め、25 回分の計測時間の相乗平均の値を求めた。

結果は表 4 のようになり、どの場合においても探索経験から類似盤面を取得する分、処理時間が増えてはいるが、一手当たりの探索時間が大きく増加することはなかった。

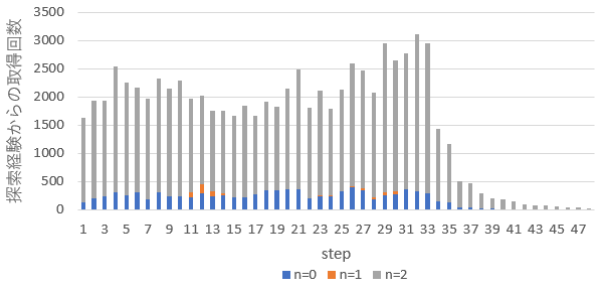

図 8 Connect Four におけるゲームの進行に対する各ハミング距離の探索経験からの取得回数

このことから、盤面の文字列をエンコードしたビット列をハッシュキーとして、探索経験から指定したハミング距離以下の盤面を取得する本研究の手法は、MCTS の探索効率の向上を著しく妨げることはないと考えられる。

表 4 MCTS プレイヤに対してハミング距離 0,1,2 の計測時間の相乗平均の比率

ハミング距離	0	1	2
計測時間の相乗平均の比率	1.11	1.03	1.13

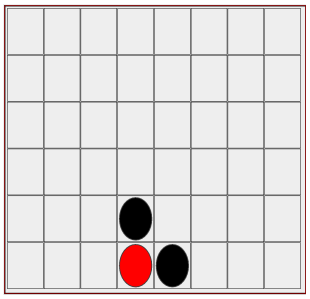

図 9 Connect Four において赤側のプレイヤの 2 手目となる盤面

5. おわりに

本研究では、少ないシミュレーション回数で MCTS の探索効率を向上させるために、MCTS を構築する過程で得られた探索経験を利用して拡張する盤面に対して工夫を行なった。探索経験は、huffman 符号を用いて盤面の文字列をビット列にエンコードしたものをハッシュキーとして

作成した。また、ビット列の差を用いて類似度の判定を行なった。

指定したハミング距離以下の盤面情報を利用する二つの方法を実装し、有効性を検討した。。ε-greedy 法を用いて拡張する盤面の選択を工夫する方法は、MCTS の探索効率を大きく向上させることができなかった。拡張する盤面に探索経験に保存されている盤面の累積報酬と訪問回数を加算する方法では、ハミング距離が 0 の時、つまり拡張する盤面と一致する盤面情報を活用する場合は MCTS の探索効率を向上させることがわかった。しかし、探索距離を大きくするにつれて MCTS の探索効率を著しく下げていることも示された。これは huffman 符号を用いて盤面の文字列をエンコードしてビット列を作成する本研究の手法では、少ないビット列の差でも盤面状態の相違が大きくなることが原因ではないかと考えられる。今後検証を行いたい。

参考文献

[1] GGP.org - General Game Playing. http://www.ggp.org/. Accessed:.

[2] GitHub - ggp-org/ggp-base: The General Game Playing Base Package https://github.com/ggp-org/ggp-base. Accessed:.

[3] Y. Bjornsson and H. Finnsson, CadiaPlayer: A Simulation-Based General Game Player, IEEE Transactions on Computational Intelligence and AI in Games, vol. 1, no. 1, pp. 4 – 15, 2009.

[4] N. Love, T. Hinrichs, D. Haley, E. Schkufza, and M. Genesereth.General game playing: Game description language specification, 2008.

[5] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In Proceedings of the European Conference on Machine Learning 2006, pp. 282 – 293, 2006.

[6] H. Finnsson and Y. Björnsson. Simulation-based approach to general game playing. In Proceedings of the 23rd AAAI Conference on Artifical Intelligence, pp. 259 – 264, 2008.

[7] H. Finnsson and Y. Björnsson. Simulation control in general game playing agents. In Proceedings of the International Joint Conference on Artificial Intelligence Workshop on General Game Playing, pp. 21 – 26, 2009.

[8] H. Finnsson and Y. Björnsson. Cadiaplayer: SearchControl Techniques. KI - Künstliche Intelligenz, Vol. 25, No. 1, pp. 9 – 16, 2011.

[9] H. Finnsson and Y. Björnsson, Learning Simulation Control in General Game-Playing Agents, in Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence. AAAI Press, pp. 954 – 959, 2010.

[10] Mandy J.W. Tak, Mark H.M. Winands, and Yngvi Bjornsson “Decaying Simulation Strategies”IEEE Transactions on Computational Intelligence and AI in Games, vol. 6, pp. 395 – 406, 2014.

