カーリング AI に対するモンテカルロ木探索の適用

大渡 勝己[1,a)]　田中 哲朗[2,b)]

概要：本研究では連続空間の MDP に対するモンテカルロ木探索を基にした行動決定手法を提案する．提案手法を，カーリングの戦略を議論するために作成された「デジタルカーリング」システムにおける対戦プログラムに適用した．実験の結果，単純なシミュレーション方策を用いた場合だけでなく，カーリングの知識を用いた複雑なシミュレーション方策を用いた場合にも提案手法が有効に働くことを確認した．

Applying Monte Carlo Tree Search to Curling AI

Katsuki Ohto[1,a)]　Tetsuro Tanaka[2,b)]

Abstract: We propose an action decision method based on Monte Carlo Tree Search for MDPs with continuous state space. We applied our method to agents of the UEC digital curling system, which is build for arguing curling strategies. The experimental results show that our method is effective for not only agents with a simple simulation policy, but also agents with a handmade complex one.

1. はじめに

カーリングは戦略性の高いスポーツとして知られている．複数人で行う競技ではあるが，各チームを一人のプレイヤと考えると，ターン制の二人零和ゲームとして扱うことができる．

カーリングは，連続空間のマルコフ決定過程（MDP）としても定式化できる．状態空間と行動空間がともに連続である点，行動決定後に加えられる外乱の影響によって行動後の状態が一意に定まらない点など現実世界での意思決定における困難な問題を内包している．そのためカーリングの行動決定において有効なアルゴリズムを議論することは，カーリング競技のみならず現実の困難な意思決定問題への貢献にもつながる可能性がある．

北清らが開発した「デジタルカーリング」システム [1][2] は，選手のプレーのぶれをショットに乱数を加えることで表現し，完全情報不確定ゲームとしてカーリングの戦略を扱うことを可能にしている．このシステムを用いたカーリング AI 大会がこれまでに数回開催されている [3]．

カーリングにおけるこうした困難性への対処として，加藤ら [4] は有限の候補行動を生成し，外乱を加えずに状態遷移を行って評価を求め，既知である外乱の確率密度関数に従い外乱ありの場合の評価値を計算する方法を提案している．加藤らの手法は Expectimax-search によるゲーム木探索を連続かつ不確定性のあるゲームに適用した手法であり，彼らの手法を適用したデジタルカーリングプログラム「じりつくん」はカーリング AI 大会にて上位に入賞している [3]．

他に，囲碁等の離散状態，離散行動のゲームにて成功を収めているモンテカルロ木探索の適用が考えられる．その場合にも状態空間，行動空間の連続性と行動結果の不確定性の扱いが問題になる．

本研究では特に状態空間の連続性に焦点を当て，モンテカルロ木探索をベースとして，近い空間においては行動価値も近いことを利用した探索手法を考案しその有効性を検証した．

2. 「デジタルカーリング」システム

2016 年 9 月時点でのデジタルカーリングにおける主なルールについて，物理計算部分や，実際のカーリングのルールとの差異について以下に示す．

- 石の物理的性質や石と氷の間の摩擦は常に一定かつ既知である．
- 石の運動の計算式は北清ら [1] に概要が示されている．筆者らは，衝突を考慮に入れない石の運動軌跡の形状は対数螺旋であると確認している．
- 石の衝突時の処理は汎用物理エンジンの Box2D ライブラリ [*1] を用いて行う．石の運動と衝突についての計算精度は事前に定めた秒間フレーム数に従う．
- 石を投げた瞬間に加えられる外乱の確率分布は既知である．
- 石の進行方向をブラシで擦って摩擦を小さくする「スウィーピング」は行われない．ゆえに，石を投げて外乱が加わった瞬間にショットの結果が確定する．

[1] 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo
[2] 東京大学情報基盤センター
Information Technology Center, The University of Tokyo
[a)] ohto@tanaka.ecc.u-tokyo.ac.jp
[b)] ktanaka@tanaka.ecc.u-tokyo.ac.jp

[*1] Box2D A 2D Physics Engine for Games http://box2d.org/

特に，特定の初速度ベクトルが与えられた場合の石の運動軌跡が決定的であることは実際のカーリングとの大きな違いである．実際のカーリングではスウィーピングによる不確定性や石と氷の摩擦についての不完全情報性があり，さらに氷の状態が試合中に変化する．これらがカーリングの戦略を複雑にしている．

前述のようにデジタルカーリングシステムを用いたコンピュータプログラムの大会がエンターテインメントと認知科学研究ステーション [*2] を主催として開催されてきたが，本研究で提案する手法に通ずるアイデアによって作成されたプログラムである「歩」（あゆむ）は第 1 回 UEC 杯 [*3]，第 1 回 GAT 杯で優勝している [*4]．

3. 連続空間におけるモンテカルロ木探索

3.1 モンテカルロ法

ゲーム木の探索等に用いられる行動決定手法としてモンテカルロ法がある．これは，主に乱数を利用した方策関数を用いて状態を遷移させた際に得る報酬和を行動の価値の推定として用いる手法である．

3.2 モンテカルロ木探索

モンテカルロ木探索はモンテカルロ法にバンディット問題の考え方を適用した手法であり，モンテカルロ法においてシミュレーションを繰り返し行う際に同一の状態に複数回到達することを利用して，それぞれの状態において各候補行動ごとにシミュレーションで得られた報酬を記録し，その情報を利用してシミュレーションの質を動的に向上させる．

代表的な手法として，各状態で UCB1 値 [7] を計算してバンディット問題を解く UCT[8] がある．モンテカルロ木探索はこれまで離散的なゲームである囲碁やバックギャモン [9] で成果を上げている．特にバックギャモンはカーリングと同じく不確定性を含んでいるゲームなので，カーリングでもモンテカルロ木探索が有効であると期待される．

本研究で扱うカーリングは状態空間，行動空間ともに連続であるが，モンテカルロ木探索を連続空間に適用するには多くの困難な点がある．以下では，どのような困難な点があるかと，提案する対処法について述べる．

3.3 行動空間の連続性への対処

カーリングのように行動空間が連続である場合にモンテカルロ木探索を行うには，行動候補が無限に存在することになり，離散化したり行動関数の近似を行う必要に迫られる．連続な行動空間におけるバンディット問題については UCBC[10]，Hierarchical Optimistic Optimization（HOO）[11] などの手法が提案されている．

UCBC は行動空間を有限個数の区間に分割して離散的な行動候補と捉える手法である．さらに HOO では行動区間の分割を再帰的に行い UCT として最適行動を探索する．HOO は著者が第 1 回 UEC 杯 [12] においてルートノードにおける最適行動の探索に利用したが本研究では扱っていない．

本研究においては，この行動の連続性については扱わず，1 つの行動カテゴリに対して 1 つの速度ベクトルを生成することに留めた．

状態遷移が確率的である場合に，モンテカルロ木探索のシミュレーション内で連続空間から行動を最適化する先行研究として，Yee らの Kernel Regression UCT[13] がある．

この手法では，ある行動に対して近傍の離散的な行動の評価を確率密度関数で畳み込むことで行動価値を推定する．さらに行動候補の生成個数を段々と増やすことで連続空間から良い行動を探索する．Yee らはカーリングにおける行動決定にこの手法を適用し有効性を示している．

3.4 状態空間の連続性への対処

カーリングにおいては行動空間のみならず状態空間も連続である．

連続空間中で状態遷移に乱数が加えられる想定でのモンテカルロ木探索の実装の先例としては Double Progressive Widening（DPW）[14] がある．この手法は本来連続的に分布する遷移後の状態を離散的にのみ生成し，その数を増やしていくことで，1 つ 1 つの遷移先の状態への到達回数を増やしつつ漸近的に全ての状態遷移を試す手法である．

DPW は通常のモンテカルロ木探索では解けない課題に対して一定の成果を上げているが，一方で「似た」状態において行動の価値が近い値をとるという性質を使っておらず，扱う対象によっては非効率であると考えられる．

それに対して本研究では，似た状態のシミュレーション結果を利用して新奇な状態における行動価値を近似することを考える．本研究では最終的に UCB 値を離散的な行動それぞれに対して計算するため，過去のシミュレーションで（状態，行動）対を試した回数とその際に得た報酬の和を似た状態における結果を用いて近似的に算出することを目指す．

その方法として，行動空間に対する HOO のように，状態空間を有限個数の区間に再帰的に分割することを考える．カーリングの状態空間は有界であるため，状態空間を偏りの小さい分割方法で再帰的に分割し続けることで分割された状態空間はそれぞれ 1 点に収束する．このような分割に

[*2] http://entcog.c.ooco.jp/entcog/

[*3] デジタルカーリング公式サイト [2] の「UEC 杯 第 1 回大会」のページにソースコードが公開されている．

[*4] 第 1 回 GAT 杯優勝の「歩」が本研究の手法と異なる主な点としては，エンド末端評価を点差によって調節している点，石によって状態分割の幅を変化させている点，ルートノードのショット生成にヒューリスティクスを用いている点，ルート直下のみ状態評価関数を用いている点，後述する状態木の根となる状態集合の範囲を限定して，状態木を多数生成している点がある．

よって有界な空間を階層的に捉えると図 **1** のようになる．図 1 では状態空間が 1 次元であり，分割の基準として上界と下界の平均値を利用した場合を示した．本論文中では以後図 1 のような状態空間の構造を「状態木（State Tree）」と呼ぶ．

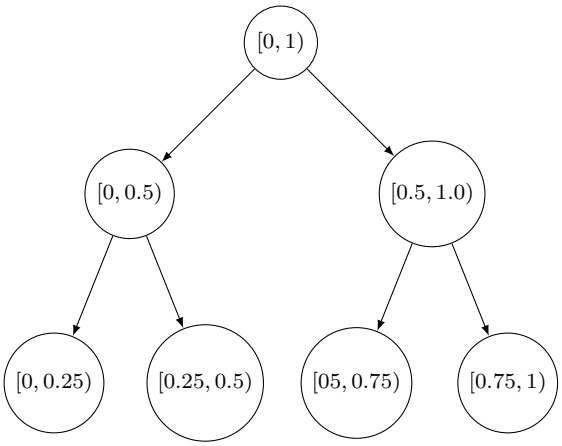


図 1 状態木の構造（状態空間が 1 次元の場合）

状態木の各ノードはそれぞれ範囲を持った状態集合を表し，各ノードに対する子ノードは自身の状態空間をさらに細かく分割した状態集合に対応する．本研究においては木という表現を状態空間の階層的な構造の意味で使用させていただく．

4. カーリングに対するモンテカルロ木探索の適用

本節では特に本研究でカーリングに対して適用した手法の詳細について述べる．

カーリングの状態空間に対する状態木を作成するにあたって，エンド内の手数，石の数や，ハウス内の中心からの石の先手，後手の並び順が異なる場合には行動の相対的な価値も異なるであろうと考え，これらの情報によって別々の状態木に分け，それぞれの状態木は連続値のみを扱うとした．

連続空間の分割基準は図 1 と同じく各次元の座標中の値の上界と下界の平均値によって行った．ここで，状態空間が D 次元のときに全ての要素の上界と下界の平均で分割すると，状態木の分岐数は 2^D になる．

カーリングにおける状態空間は，プレーエリア内のそれぞれの石の絶対的な位置座標によって表現できる．本研究では 1 つの石の 1 方向の座標を 1 次元とし，プレーエリア内の石の数 n_{stone} に対して状態木の分岐数を最大で $2^{n_{stone}}$ とするナイーブな実装とした．ただし同じ色の石は区別する必要はないため，常に分岐数 $2^{n_{stone}}$ とはならない．

エンド内の最終投後の状態を除いた状態における最大の石の数は 15 個であるので，状態空間は最大 30 次元，状態木の分岐数は最大 2^{30} となった．

状態木の実装において，状態空間を完全に表現するためには状態木の深さは ∞ であるが，探索開始時点では有限の深さに絞り，状態集合への到達回数が増えたもののみ展開する．本研究では初期状態では根ノードだけが存在する実装とした．

状態木のノードを展開する条件は，木の先端の状態集合ノードに到達した回数が状態木の根ノードからの分割回数 d ごとの閾値 $N_{ex}(d)$ に達した場合とする．$N_{ex}(d)$ は d に対して単調増加な関数とし，木が深くなるにつれ展開が抑制されるようにする．

シミュレーション結果の報酬の逆伝播の際には，経験した状態を含む状態集合を葉ノードから根ノードまで辿り，全てのノードに行動の選択回数を 1 回分加算し，報酬和に観測された末端報酬を加える．

逆にシミュレーション中の任意の状態での着手の選択においては，状態木のうち目的の状態が含まれる葉ノードから根ノードに記録されている結果を合算して使う．このとき，より末端のノードの結果の寄与度を大きくする重み付けによって結果が合算されるように，状態木の根ノードからの分割回数 d のノードに記録されている結果に，d に対して単調増加な関数 $w(d)$ によって重み付けを行い，より細かく分割された情報集合に記録された結果を重視させる．

この $w(d)$ を用いて，状態 x における行動 a の仮の到達回数 $\tilde{n}(x,a)$ と仮の報酬和 $\tilde{r}(x,a)$ を以下のように求める．ただし S を状態木上で状態 x を範囲内に含んでいる状態集合 s の集合とし，$d(s)$ は状態木の根ノードから s に対応するノードまでの分割回数とする．$n(s,a)$ と $r(s,a)$ は状態木のノード上に記録されている行動 a の到達回数と報酬和を表す．

$$\tilde{n}(x,a) = \sum_{s \in S} w(d(s)) n(s,a) \tag{1}$$

$$\tilde{r}(x,a) = \sum_{s \in S} w(d(s)) r(s,a) \tag{2}$$

さらに状態 x への仮の到達回数は，状態 x で生成された行動全体の集合 A として

$$\tilde{n}(x) = \sum_{a \in A} \tilde{n}(x,a) \tag{3}$$

と求める．ここで，ある状態集合 s に含まれる状態 x_0，x_1 があり，状態 x_0 にて生成される行動集合と状態 x_1 で生成される行動集合が異なるということは起こり得るので，$\tilde{n}(x) = \sum_{s \in S} w(d(s)) n(s)$ とはできないことに注意されたい．

このように求めた $\tilde{n}(x,a)$ や $\tilde{r}(x,a)$ を利用して，状態 x における行動 a の UCB 値を計算することも可能であるが，状態木の展開が進むと重み付けによって $\tilde{n}(x,a)$ は実際の到達回数より速いスピードで増加するという問題点がある．

そこで $\tilde{n}(x) > 1$ のとき $n(x) < \tilde{n}(x)$ となる単調増加な関数を $\tilde{n}(x)$ に適用した値を $n(x)$ とする．

この変換での減少率を $\tilde{n}(x,a)$ と $\tilde{r}(x,a)$ に掛けて UCB1 値の計算に用いる $n(x,a)$ と $r(x,a)$ を計算する．

$$n(x,a) = \tilde{n}(x,a)\frac{n(x)}{\tilde{n}(x)} \quad (4)$$

$$r(x,a) = \tilde{r}(x,a)\frac{n(x)}{\tilde{n}(x)} \quad (5)$$

$n(x)$，$n(x,a)$，$r(x,a)$ を用いて UCB1 値 [7] を計算し，この値が最大となる行動 a_{try} を選んでシミュレーションを進める．

$$a_{try} = \underset{a \in A}{\operatorname{argmax}} \left(\frac{r(x,a)}{n(x,a)} + C_{UCB}\sqrt{\frac{2\ln n(x)}{n(x,a)}} \right) \quad (6)$$

4.1 行動の価値に事前知識を用いる場合の行動決定

式 6 では探索開始時点では行動の価値について事前知識を持ち合わせていないことを前提としてシミュレーション中の行動を選択したが，事前の学習やヒューリスティクスによって予測した行動価値の情報を合わせて利用することも考えられる．

離散的ドメインにおいてシミュレーションの結果と行動価値予測の合算法として囲碁における Chaslot ら [15] の手法などが提案されているが，本研究においては事前の行動価値予測を用いた softmax 方策における選択確率を，その行動を選んだ場合の勝率と見なす．行動 a の事前学習による行動価値の値を $V_{pre}(a)$，softmax 選択における温度を T，行動価値予測の信頼度に当たる行動選択回数を N_{pre}，エンド末端での予測勝率をシミュレーション末端報酬に変換する関数を $WPtoR()$ として，事前の行動価値予測を用いる場合の状態 x における行動 a の到達回数の修正値 $n'(x,a)$ と報酬和の修正値 $r'(x,a)$ を以下のように求める．

$$n'(x,a) = n(x,a) + N_{pre} \quad (7)$$

$$r'(x,a) = r(x,a) + WPtoR(\frac{e^{\frac{V_{pre}(a)}{T}}}{\sum_{b \in A} e^{\frac{V_{pre}(b)}{T}}})N_{pre} \quad (8)$$

本研究では事前に予測する行動価値について，ある状態における候補行動の間での相対的な比較のみ可能な値であり，値域が $(-\infty, +\infty)$ という前提の下で以上の式で計算した．事前学習の結果を加算する他の手法との比較は行えていない．

5. 行動決定手法

5.1 ルートノードでの候補ショットの生成と評価

ルートノードにおける候補ショットの生成は，加藤ら [4] の手法を参考にして目標到達座標を離散化した．加藤らの手法においては，主にドロー系のショットとしてハウス内とその前方に等間隔で目標到達座標の点を生成する．さらにテイクアウト系のウェイトの大きいショットとして，遠方に一列の目標到達座標の点を生成する．加藤らの手法 [4] では 1336 個の目標到達点を生成している [*5]．本研究では加藤らの離散化から範囲を一部変更し，また大会の外乱ルールの変更に合わせて離散化の幅を変更した．石の半径 l に対して，ドロー系のショットの座標を幅 $(\frac{l}{2}, 2l)$ で 63 × 31 個の点を配置し，ヒット系のショットは幅 $(\frac{l}{4}, \infty)$ で 189 × 1 個の点を配置した．目標到達座標が計 2142 個でショットの回転方向が左右の 2 種類あるため，候補ショット数は合計で 4284 個 [*6] となった．

シミュレーションは，これらの点を目標到達座標とするショットに外乱を加えずに行い，それぞれの点に報酬を記録する．外乱を加えた候補ショットの評価は，加藤らの手法に倣い，近傍の点の外乱を加えない評価を外乱の確率密度関数によって畳み込むことで計算した．

目標到達座標から実際の到達座標のずれの分布については，正規分布とすると説明されている [1] が，本論文執筆時点においては，公式の物理演算システムの実装は速度ベクトルの各要素の差に依存している．しかしこの分布の確率密度関数を到達する位置座標系に変換する上ではヤコビアンの計算が複雑であったため，本研究では公式サイトにおける説明の通り正規分布として計算している．そのことによる誤差は検討の余地がある．

5.2 ルートノードでのシミュレーション割り振り

本研究ではルートノードのシミュレーション割り振りは全て均等に行った．つまり，4284 個の候補ショットをランダムに並べ替えて順にシミュレーションを行うことを 1 サイクルとし，これ繰り返すという手法とした．より効率的な探索を行うには，評価の高い候補手の周辺に優先的にシミュレーションを割り振ることが必要であろう．

5.3 シミュレーション中のショットの生成

本研究ではシミュレーション中の方策については 2 パターンを用意した．一方は**表 1** に示した条件で生成したショットのカテゴリ集合から一様ランダムに選ぶシンプルなシミュレーション方策，もう一方は，著者のカーリングについての知識を用いてヒューリスティックに生成したショットカテゴリ集合から，事前に過去のデジタルカーリングの試合棋譜における行動を分類してロジスティック回帰により学習したシミュレーション方策である．表 1 では 1 つのカテゴリにつき，右回転と左回転のショットを 1 つずつ生成する．

[*5] ただしこれは最終的なショット選択の対象となる点のみに絞った個数で，実際は外乱の考慮のため周囲の点も探索の対象としており，より多くの点の評価を求めていると考えられる．

[*6] 外乱の考慮のため評価するショットも含まれた個数であり，最終的な選択の候補となったショット数は 2880 であった．

表 1 シンプルなショット生成

生成条件	ショットカテゴリ	生成数
すべての局面	ハウス中心へのドロー	2
ハウス内に相手の石	相手の最も内側の石をヒット	2

後者のヒューリスティック方策のショットカテゴリ生成の詳細については複雑な分岐を含むため，論文中では詳しい説明を省かせていただくが，基本的には得点で勝っている場合，特に先攻時には相手の石を減らすことを最優先に考え，主に後攻では No.1（ハウス中心に最も近い石）を自分の色にしつつ大量得点を狙うようなショットカテゴリを生成している．1 つの状態に対するショットカテゴリの生成数は平均で 8 程度であった．[*7]

方策の学習については以下に記す．方策は線形の行動価値関数を利用した softmax 方策である．各ショットカテゴリに対応する特徴ベクトルの要素について**表 2** にまとめた．石の絶対的な位置，相対的な位置関係については離散的な分類を行っている．

本研究で用いたような評価要素とは異なり，位置関係をより連続値に近い形で用いて局面評価関数を計算する加藤ら [4] の手法もあり，行動価値関数についても連続値のまま計算可能な評価要素が望まれる．

学習では各状態での行動選択確率の交差エントロピーの合計を誤差関数とし，Stochastic Gradient Descent によってパラメータの最適化を行った．以下に学習時のパラメータを示す．ここで i は繰り返し回数，ϕ_j は特徴量ベクトルの j 番目の要素，$Var()$ は分散を表す．

- 学習率 $\frac{0.00005}{Var(\phi_j)}\left(0.01+\frac{0.99}{\sqrt{i+1}}\right)$
- 温度 1
- L1 正則化係数 0
- L2 正則化係数 0.0000001 $\left(0.01+\frac{0.99}{\sqrt{i+1}}\right)$
- バッチサイズ 256
- 総繰り返し回数 50

教師として使用した棋譜は UEC 杯，GAT 杯のルールのものをいずれも含み，主に著者の環境における 10 エンドの対戦で生成したものである．そのうち過去のデジタルカーリング大会の上位プログラムである「歩」「じりつくん」「GCCS」「CSACE」の行動決定機会の中で，大量得点差が付いた場合と最終エンドの中終盤は教師としての正当性が疑わしいと考えて除き，それ以外のうち 184 万局面を学習に用いた．これらの状態でヒューリスティックなショットカテゴリ生成を行い，そのうちどのショットが教師のショットに近いかを分類し，正解を定めた．

ショットの分類は以下のように行った．棋譜に記録された外乱が掛かる前のショットについて，まず「右回転か左回転か」で分ける．次に，実際に物理演算によって局面を進め，他の石とのコンタクトがあったかどうかで分ける．一方で，ヒューリスティックに生成したショットカテゴリも，それぞれに回転方向と，他の石とのコンタクトを目的とするショットかどうかの情報を保持しており，この 2 要素で分類する．

[*7] ショットカテゴリ生成や行動価値関数については http://www.tanaka.ecc.u-tokyo.ac.jp/wp/ohto/2016/09/26/ に公開している実験のコードをご覧いただきたい．

表 2 行動価値関数の評価要素

評価要素	パラメータ数
現時点でのそれぞれの石の位置	1248
行動に関係する石と それ以外の石の相対的な位置関係	39936
2 つの石に関係する行動において 目的の 2 つの石同士の関係	408
現在の石配置でエンド終了の 場合に得る得点と行動の種類の関係	240
現在の石配置から 相手の最もハウス中心に近い石を除いて エンド終了した場合の得点と 行動の種類の関係	240
計	42072

ここまでの分類において，対応するショットカテゴリが無くなった場合には，正解なしとして学習は行わない．用いた 184 万局面の中には正解なしの状態が約 $\frac{1}{4}$ 程度含まれていた．複数の行動カテゴリが教師の行動と同一の分類に属した場合には，以下の条件で正解のショットカテゴリ $a_{similar}$ を決定した．

$$a_{similar}=\underset{a\in A}{\operatorname{argmin}}\left\{2(v_x(a)-V_x)^2+(v_y(a)-V_y)^2\right\} \tag{9}$$

ここで A は生成したショットカテゴリ集合，a は 1 つのショットカテゴリ，$v_x(a)$ と $v_y(a)$ は a から，盤面情報を元に速度ベクトルに変換した各要素であり，一方で V_x と V_y は教師のショットの速度ベクトルの各要素である．石を投げる点から見て横方向の速度成分 $v_x(a)$ と V_x の差を奥行き方向の速度成分の差よりも大きく扱うためにこのような式としたが，特に性能についての評価は行えていない．

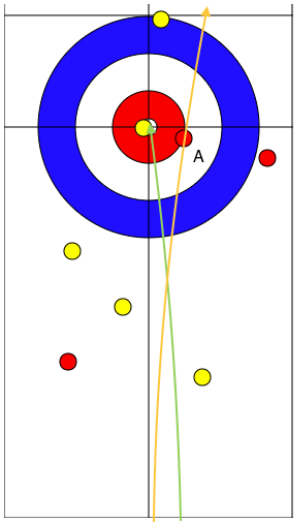
本研究で用いたショットの分類法は生成したショットカテゴリによって変化するものであり，ショットカテゴリ生成のパターンごとに新たに分類を行う必要がある．

この手法の問題点としては，全く違う意図のショットに分類されうる点がある．例として，教師のショットがハウス前方へのガードショットであり，ショットカテゴリ生成においてコンタクトを目指さないものとしてハウス中心へのドローのみ生成していたとすると，教師のショットはハウス中心へのドローというカテゴリで扱われてしまう．

理想的には，プログラムの思考時の行動カテゴリ生成の結果とは関係のない形で教師のショットがカテゴリ分類され，局面情報に応じてカテゴリ間の距離を与えられることが望ましいと考えている．

ある局面に対して，学習された方策の出力を図 **2** に示した．石配置の画像出力には著者らが作成した curing_log_viewer[*8] を利用した．

確率の出力において softmax 方策の温度は実験時と同じ $T = 0.8$ としている．出力されたショットのうち 1 位，2 位のヒットショットを黄線で，3 位のドローショットを緑線で図中に示す．ラストストーンで相手の石を出せば 3 点の局面であるが，実際にヒットするショットを最優先している点，さらに手前のガードを避ける回転方向を選択している点が確認できる．



局面情報：
・黄色の手番
・最終1つ前のエンド
・同点
・ラストストーン

出力された上位 5 ショット

ショット	回転	確率
石 A をヒット	右	0.437
石 A を強くヒット	右	0.437
ハウス中心にドロー	左	0.065
石 A をヒット	左	0.024
石 A を強くヒット	左	0.024

図 2　学習したヒューリスティック方策が有効と判断したショット

5.4　シミュレーション末端報酬

カーリングは 1 エンド 16 投の繰り返しゲームであり，特にデジタルカーリングは完全情報ゲームであるため，試合終了までシミュレーションを行わずにエンド末端に報酬を与えてシミュレーションを打ち切ることが可能である．本研究においては勝率の最大化を目指すため，エンド末端での報酬は予測される勝率をベースとした値とする．

相手による違いを考慮に入れなければ，エンド末端での予測勝率は残りエンド数とその時点での点差，次のエンドで先攻か後攻かによって一意に定まる．予測勝率は実際の対戦結果から算出することが可能であるが，本研究においてはより簡単に以下のように求めている．

1. 同点で延長戦に入る場合の先手の予測勝率を決定
2. 残りエンド数や点差に関わらず 1 エンドの得点分布が一定と仮定し，得点分布の確率質量テーブルを作成
3. 最終エンドから順に予測勝率を確率質量で畳み込むことで，残りエンド×得点差の先手視点でのテーブルを作成

[*8] デジタルカーリングで公式に扱われている棋譜を再生するシステム．https://github.com/u-tokyo-gps-tanaka-lab/curling_log_viewer にて公開している．

この得点分布と延長戦勝率は著者の主観で定めており，改善の余地があるが，ルールやプログラムの戦略によって変化しうる値であり厳密に計算可能なものではない．本研究で用いた得点分布の値を**表 3** に掲載した．また延長戦の先手勝率は 0.25 と定めた．

以上に示したエンド末端にて予測勝率を定めたが，この値をモンテカルロ法の報酬として用いる場合，分散が一定でないため探索と活用のバランスが崩れるという懸念がある．そのためエンドごとに得点の分布が表 3 に従った際に予測勝率が平均 0，分散 1 となるように正規化して末端報酬として用いた．この変換が式 (8) における関数 $WPtoR()$ に対応する．

表 3　1 エンドの得点分布として定めた数値

次エンド先後交代あり		次エンド先後交代なし	
得点	割合	得点	割合
後手 8	0.000195461	先手 8	0.00000547291
後手 7	0.000781844	先手 7	0.0000547291
後手 6	0.00312738	先手 6	0.0000547291
後手 5	0.0125095	先手 5	0.000547291
後手 4	0.0390922	先手 4	0.00547291
後手 3	0.070366	先手 3	0.0156369
後手 2	0.195461	先手 2	0.0547291
後手 1	0.390922	先手 1	0.172006
		0	0.0390922

6.　実験

6.1　状態木の実装

図 1 で示した状態木の実装については様々な手法が考えられるが，本研究においては状態木の最大分岐数が大きく木構造を保持するコストが大きいと見て，現れた状態集合に対してのみノードを生成する以下の手法とした．
なお，位置座標は石を投げる点から見て横方向に x 軸，奥行き方向に y 軸を取っている．

1. それぞれの石の連続空間での x 座標と y 座標のそれぞれを小さなステップサイズの離散座標（各 15 ビット）に変換
2. 離散座標の各ビットの値によるゾブリストハッシュ（石の色ごとに別）を加算によって計算
このとき下位の $15 - d$ ビットを無視することで分割回数 d の状態集合のハッシュ値を計算できる
3. 手数，ハウス中心からの石の並びの情報を加えたハッシュ値とする
4. 開番地法によるハッシュテーブル上のエントリに到達回数や報酬和を記録

1 で石の座標を 15 ビットで表現することで，状態木における状態空間の最大分割回数を 15 回と定めた．状態集合

における各座標の範囲の幅は x 軸，y 軸ともに分割 15 回の最深部では 0.5mm 程度，状態木の根ノードでは 15.8m 程度でプレーエリア全体が収まるようにした．

また 2 において石の座標からハッシュ値への変換の際に石を区別せず先攻の石か後攻の石かのみで場合分けすることで，同じ色の石の位置が入れ替わった等価な状態集合をまとめあげることを可能とした．

一方，状態集合をハッシュ値によって見分けていることにより，低い確率ではあるが異なる状態集合が同じ場所に記録されてしまう可能性がある．本論文における実装では，ハッシュテーブルはエントリ数約 100 万の固定サイズかつエントリの上書きをせず，リハッシュ回数に上限を設けたため，容量オーバーやリハッシュ回数オーバーによって該当の状態集合のエントリを作成できない場合がある程度あり，厳密に状態木を再現できてはいない．

状態木の作成において手数に制限は設けず，到達した状態全てに対して，対応する状態木が作成されていない場合には状態木の作成を試みた [*9]．また，状態木のノードに記録された報酬情報は手数が進んでも有効であるものが含まれるが，本論文の実験においてはハッシュテーブルは行動決定機会ごとにクリアした．

次に，状態木の挙動を定めるそれぞれの関数については以下の設定とした．

状態木の分割回数 d のノードの展開閾値 $N_{ex}(d)$ は

$$N_{ex}(d) = N_{exbase} C_{ex}^{d} \tag{10}$$

とした．定数 N_{exbase} はベースとなる展開閾値であり，一方の定数 C_{ex} は展開スピードの減衰の度合いを定める．本論文の実験においては $N_{exbase} = 1$，$C_{ex} = 1.3$ とした．

状態への到達回数を合算する際の重み付けの関数 $w(d)$ は以下とした．

$$w(d) = (d+1)^{C_w} \tag{11}$$

定数 C_w が大きいと，細かく分割された状態集合の結果を重視する．実験においては $C_w = 4$ とした．

各ノードの到達回数を $w(d)$ で重み付けて算出した状態 x の仮の到達回数 $\tilde{n}(x)$ から $n(x)$ への変換は

$$n(x) = \tilde{n}(x)^{C_{mod}} \tag{12}$$

実験では定数 $C_{mod} = 0.4$ とした．

また UCB1 値の計算の際には，状態木中で初めて行う行動にも ∞ でない値が算出されるように，状態木の各ノードにて初期化時の各行動の到達回数を $\frac{1}{2}$ 回とした．

[*9] ただし特別な状態として，石の半径を l としてハウスの近辺 $2l$ とその手前側の範囲に一部でも入っている石が無い場合には空の状態と認識し，状態木とは別に結果を保存した．

6.2 物理演算

デジタルカーリング公式サイト [2] にて汎用の物理演算エンジン Box2D を利用した物理演算ライブラリが用意されているが，計算時間が長いことが難点である．そのため本研究では，デジタルカーリングで想定されている石の運動方程式と Box2D の衝突処理を参考に物理演算システムを自作して効率化を図っている．ただし，計算方法が異なるため多くの場合公式の物理演算システムの出力結果と異なる結果を出力する．

6.3 プログラムの設定

以下の実験においては，行動決定機会 1 回あたりのシミュレーション回数は，後攻のラストストーンのみ各ショット 1 回ずつの 4284 回，それ以外では全て各ショット 47 回の 201348 回で固定した．

シミュレーション中のパラメータの設定は UCB1 値の計算の際の係数 $C_{UCB} = 1$，ヒューリスティックなシミュレーション方策における softmax 方策の温度 $T = 0.8$，学習した行動価値関数の信頼度は行動選択回数 $N_{pre} = 2$ 回分とした．

以下本研究で提案した状態木（State - Tree）を利用したモンテカルロ木探索によるプログラムを State - Tree MCTS，純粋モンテカルロ法によるプログラムを Pure MC と表記する．

6.4 対戦実験

以下の 4 条件で対戦実験を行った．

1. State - Tree MCTS（シンプルな方策）
   vs Pure MC（シンプルな方策）
2. State - Tree MCTS（ヒューリスティックな方策）
   vs Pure MC（ヒューリスティックな方策）
3. Pure MC（ヒューリスティックな方策）
   vs Pure MC（シンプルな方策）
4. State - Tree MCTS（ヒューリスティックな方策）
   vs State - Tree MCTS（シンプルな方策）

試合数は先後を入れ替えて各 400 試合の計 800 試合を各対戦それぞれで行った．対戦は GAT 杯ルール [*10] で全て 2 エンド戦とした．実験プログラムが延長戦に対応していなかったため 2 エンド終了後に同点の場合は試合を打ち切った．

結果の集計にあたって，方策の学習の際に用いた過去のデジタルカーリングの棋譜から最終エンド開始時点で同点の場合の試合結果を抽出し，そこから棋譜中のプログラム群における延長戦の先手の期待勝率を求めたところ 0.219 程度であった．

[*10] デジタルカーリング公式サイト [2] の「GAT 杯 (2016)」のページを参照．

そのため対戦結果から勝率への換算を行う際には仮に延長戦があった場合に先手で延長戦に入る場合は 0.219 勝，後手で延長戦に入る場合は 0.781 勝として計算に加えた．

上記の換算によって計算された勝率が $\frac{1}{2}$ と異なるかをカイ二乗検定で検定した．

6.5 対戦実験結果

State - Tree MCTS の Pure MC に対する勝敗と換算勝率，換算勝率の先後での平均，検定の結果を**表 4** と**表 5** に示した．

同じように，ヒューリスティックなシミュレーション方策を用いたプログラムのシンプルなシミュレーション方策を用いたプログラムに対する勝敗と換算勝率，換算勝率の先後での平均，検定の結果を**表 6** と**表 7** に示した．

表 4 State - Tree MCTS の Pure MC に対する勝敗（シンプルなシミュレーション方策）

	勝	延後	延先	負	換算勝率	平均
先攻	128	34	67	171	0.423	0.617
後攻	289	43	8	60	0.811	($p = 4 \times 10^{-11}$)

表 5 State - Tree MCTS の Pure MC に対する勝敗（ヒューリスティックなシミュレーション方策）

	勝	延後	延先	負	換算勝率	平均
先攻	107	21	67	205	0.345	0.552
後攻	257	54	19	70	0.758	($p = 0.003$)

表 6 ヒューリスティックなシミュレーション方策のシンプルなシミュレーション方策に対する勝敗（Pure MC）

	勝	延後	延先	負	換算勝率	平均
先攻	140	54	54	152	0.485	0.665
後攻	287	61	17	35	0.846	($p = 8 \times 10^{-21}$)

表 7 ヒューリスティックなシミュレーション方策のシンプルなシミュレーション方策に対する勝敗（State - Tree MCTS）

	勝	延後	延先	負	換算勝率	平均
先攻	118	20	72	190	0.373	0.557
後攻	247	59	16	78	0.741	($p = 0.001$)

表 4 と表 5 より，2 条件双方で State - Tree MCTS は Pure MC に有意水準 5%で有意に勝ち越した．

また表 6 と表 7 において，ヒューリスティックなシミュレーション方策を用いたプログラムがシンプルなシミュレーション方策を用いたプログラムに 2 条件いずれも有意水準 5%で有意に勝ち越した．

7. 考察

対戦実験の結果より，本研究で提案した木探索，さらにカーリングの知識と機械学習により作成したシミュレーション方策はいずれもデジタルカーリングにおいて有効であったと考えられる．今後の課題としては連続な行動空間において行動を最適化することや状態木をより効率良く分割することが挙げられる．

参考文献

[1] 北清勇磨, 岡田雷太, 伊藤毅志：デジタルカーリングサーバーの提案と紹介, 情報処理学会研究報告, 2014-GI-31, No. 2, pp. 1 - 5 (2014).

[2] 電気通信大学. デジタルカーリング, http://minerva.cs.uec.ac.jp/curling/wiki.cgi (2016. 9. 26 閲覧).

[3] 伊藤毅志, 森健太郎：デジタルカーリング大会報告 2015 年度, 情報処理学会研究報告, 2016-GI-36, No. 2, pp. 1 - 6 (2016).

[4] 加藤修, 飯塚博幸, 山本雅人：戦略の不確実性を考慮したカーリング AI の開発, 人工知能学会研究会資料—知識ベースシステム研究会 (第 104 回), pp. 7 - 12 (2015).

[5] M. Yamamoto, S. Kato, H. Iizuka,：Digital Curling Strategy on Game Tree Search, 2015 IEEE Conference on Computational Intelligence and Games, (2015).

[6] 加藤修, 飯塚博幸, 山本雅人：デジタルカーリングにおけるゲーム木枝刈りの有効性検証, エンタテインメントコンピューティングシンポジウム 2015 論文集, pp. 412-417 (2015).

[7] P. Auer, N. Cesa-Bianchi, and P. Fischer：Finite-time Analysis of the Multiarmed Bandit Problem. Machine Learning, Vol. 47, pp. 235 - 256 (2002).

[8] L. Kocsis and C. Szepesvari：Bandit based Monte-Carlo Planning. European conference on machine learning (ECML2006), pp. 282 - 293 (2006).

[9] F. van Lishout, G. Chaslot, and J. Uiterwijk：Monte-Carlo Tree Search in Backgammon, Computer Games Workshop, pp. 175 - 184 (2007).

[10] P. Auer, R. Ortner and C. Szepesvari：Improved Rates for the Stochastic Continuum-Armed Bandit Problem, International Conference on Computational Learning Theory, Springer, pp. 454 - 468 (2007).

[11] S. Bubeck, R. Munos, G. Stoltz and C. Szepesvari：Online Optimization in X-Armed Bandits. Advances in Neural Information Processing Systems (NIPS2009), pp. 201 - 208 (2009).

[12] 伊藤毅志, 森健太郎, 北清勇磨：第 1 回 UEC 杯デジタルカーリング大会報告, 情報処理学会研究報告, 2015-GI-34, No. 2, pp. 1 - 6 (2015).

[13] T. Yee, V. Lisy, and M. Bowling：Monte Carlo Tree Search in Continuous Action Spaces with Execution Uncertainty. International Joint Conference on Artificial Intelligence. (2016).

[14] A. Couetoux, J. Hoock, N. Sokolovska, O. Teytaud and N. Bonnard, Continuous upper confidence trees. International Conference on Learning and Intelligent Optimization, pp.433 - 445 (2011).

[15] G. Chaslot, C. Fiter, J.P. Hoock, A. Rimmel and O.Teytaud：Adding expert knowledge and exploration in Monte-Carlo Tree Search, Advances in Computer Games, LNCS, Vol. 6048, pp. 1 - 13 (2009).