Nested Monte Carlo Searchの ぷよぷよへの適用

齋藤　晃介[1,a)]　三輪　誠[3]　鶴岡　慶雅[2]　近山　隆[2]

概要：
ランダム性を持つゲームでは，プレイヤーが未来に得る情報が確率的に決まる．そのようなゲームの中から，ぷよぷよという落下型パズルゲームに着目する．ぷよぷよは，人間のプレイヤーは将来の完成型を考えて行動を選択しており，また完全にランダムな探索が無駄になりやすいために先読みが難しいゲームである．本論文では，確定完全情報パズルゲームで有効性を示された Nested Monte Carlo Search をぷよぷよに適用する手法を提案し，そのアルゴリズムの振る舞いを調査した．その結果，今回の設定では計算時間に見合うだけの有意な性能は示せなかった．

Application of Nested Monte Carlo Search to Puyo-puyo

Kosuke Saito[1,a)]　Makoto Miwa[3]　Yoshimasa Tsuruoka[2]　Takashi Chikayama[2]

Abstract:
Indeterminate games are games in which outcome of plays has probabilistic nature. Among such games, we focus on Puyo-puyo, which is a popular tile-matching video game. Puyo-puyo is a difficult game, because human players determine each action in such a way that it will lead to a good completed form in the future and a completely random search often comes out to be wasteful. In this paper, we propose a method for applying Nested Monte Carlo Search, the effectiveness of which has been shown in logical perfect information games, to Puyo-puyo and investigate its behavior. As a result, we can not show the efficiency which reflects time to calculate.

1. はじめに

不確定ゲームとは未来に得られる情報が確率的に決まるゲームであり，例としては未来に落ちてくるブロックが分からないぷよぷよ，テトリスなどのパズルゲームが挙げられる．その中でもぷよぷよというパズルゲームは不確定ゲームとして研究に適したゲームであるが，その研究の論文はテトリスなどの他のパズルゲームと比較しても非常に少ない．

[1] 東京大学工学部電子情報工学科
Department of Information and communication Engineering, The University of Tokyo
[2] 東京大学大学院工学系研究科
Graduate School of Engineering, The University of Tokyo
[3] マンチェスター大学コンピュータ科学科
School of Computer Science, The University of Manchester
[a)] saito@logos.t.u-tokyo.ac.jp

ぷよぷよが研究に適しているという理由については次の二つが挙げられる．まず，この種のパズルゲームと他のゲームとの差として，ルールおよび操作方法が非常に単純なことである．このことは問題を解くためのプログラムを簡単なものとし，研究が容易になる．次に，ぷよぷよと他のパズルゲームの差として，先読みの難しさがある．ぷよぷよは同じ色のブロックを 4 つ以上接触させることで消し，消えたブロックの上に積んであったブロックが落下することによってさらに消えていくことで連鎖を発生させるゲームである．このゲームは操作性の似ているゲームであるテトリスと比べて探索ノード数が遥かに多い．また，連鎖数を増加させるためには何手もの先読みをしなければならず，間違った行動をとるとそれを修正することが非常に難しくなっている．以上の理由から，ぷよぷよは他のゲームにない特徴を持ったゲームである．

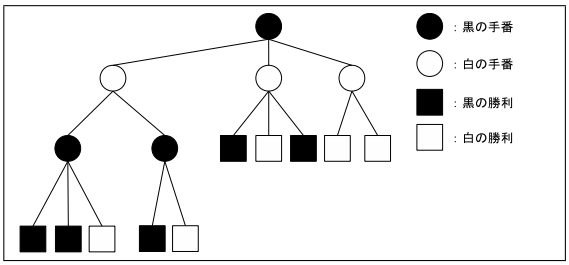

図 1　モンテカルロ木探索

ぷよぷよの探索は非常に状態数が多く，無駄なノードが多いため，単純なモンテカルロ木探索では計算コストと成果を両立させることが難しい．Nested Monte Carlo Search という確定完全情報ゲームに対して提案された手法 [1] は複数のモンテカルロ探索を入れ子にして，上位が再帰的に下位のモンテカルロ探索を呼び出すという特徴があり，より有望な手を早く発見することができると期待できる．

そこで本論文では，Nested Monte Carlo Search をぷよぷよに適用し，連鎖数を指標としてそのアルゴリズムの振る舞いを調査する．

2. 関連研究

2.1 モンテカルロ木探索

モンテカルロ法は可能な行動の中からランダムに一手選び，終了条件まで状態を進めていくシミュレーションを繰り返すことによって問題の解を探索する手法であり，これと木探索を組み合わせたアルゴリズムがモンテカルロ木探索である．ランダムに終局までプレイすることをプレイアウトと呼ぶ．囲碁を例にとると，モンテカルロ木探索はまず各候補手に対してプレイアウトを行い，その結果を集計して有望な手に多くのプレイアウトを割り当てる．さらに，各候補手でのプレイアウトの回数がある閾値を越えたとき，図 1 のようにプレイアウトを開始する手を 1 つ深くするという特徴がある．これによって，有望と判断された部分だけ探索木が成長することになる．

このモンテカルロ木探索を用いたプログラムによってコンピュータ囲碁は大きく強化されたことが知られている [2]．

2.2 Nested Monte Carlo Search

Nested Monte Carlo Search は Cazenave が提案した，有望なノードをより深く探索していく手法であり，ランダムシミュレーションや Upper Confidence Bound Applied to Trees (UCT) などの既存手では良い解が得られなかった Morpin Solitaire というパズルゲームで高い性能が報告されている [1]．それに All Move As First (AMAF) という手法を加えて Morpin Solitaire で世界記録を出した研究もあり [3]，学習の要素を加えた Nested Rollout Policy Adaptation という手法も提案されている [4]．図 2 に

```
NMCS(level, node):
    if level == 0:
        while num_children(node) > 0:
            node = child(node, random())
        return score(node)

    else:
        bestscore = 0
        while num_children(node) > 0:
            for chilren i of node:
                temp = child(node, i)
                result = NMCS(level-1, temp)
                if result > bestscore:
                    bestscore = result
                    next = i
            node = child(node, next)
    return bestscore
```

図 2　Nested Monte Carlo Search の疑似コード

Nested Monte Carlo Search の疑似コードを示す．

Nested Monte Carlo Search は次のように大まかに 3 段階に分かれている．

(1) まずネストレベル n を定義する．$n \geq 1$ では現在のノード（レベル n）の子ノード（レベル $n-1$）を全て再帰的に探索し，$n = 0$ に到達するとランダムシミュレーションを開始する．
(2) 子ノード（$n-1$）の探索が全て終了すると，その中から一番評価値が高いものを選び，それを親ノード（n）とする．
(3) 最終局面に到達するまで（1），（2）を繰り返す．

ネストレベル n における探索はレベル $n-1$ での探索よりも多くの場合良い結果を与えることが知られている [1]．

2.3 ぷよぷよに関する研究

一人用ゲームの探索アルゴリズムとして，A*アルゴリズムなどが広く用いられている．しかし，A*アルゴリズムはある程度正確な評価関数がなければうまく動かず，またゴールノードを定めなければ使用できない [5]．ぷよぷよではゴールノードが存在しないため，A*アルゴリズムを適用することはできない．そこで，本節ではぷよぷよに対する他の手法を紹介する．

ここで，ぷよぷよで用いる対象クラスを定義する．

- プレイ人数：1 人のプレイヤーが 1 つの盤を持つ．
- 盤とマス：盤は n 個のマスからなり，有限かつ離散的であるとする．ここでは，重力のある長方形格子状の盤を用いる．
- 石：1 つのマスは，空であるか，1 つの石が占めるものとする．石には複数で有限の種類または状態が存在するものとする．プレイヤーは盤の全てのマスの情報を得られる．
- 配石：プレイヤーには 1 つまたは複数の石からなる配石が与えられる．ここでは配石の種類はランダムに決められ，数手先の配石は予告されている．

2.3.1 ポテンシャル最大化アルゴリズム

ポテンシャル最大化アルゴリズムとは，ぷよぷよにおい

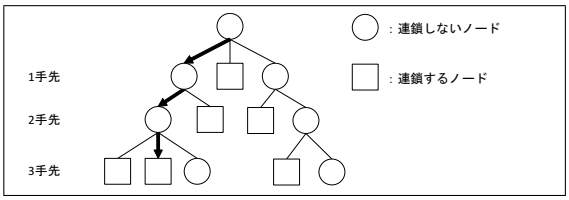

図 3 ポテンシャル最大化アルゴリズム

て既に配石の予告されている次の 3 手先までを全探索し，葉ノードでの評価値が最も大きくなる手を選ぶ手法である [6]．このアルゴリズムを用いると，場合によっては 10 連鎖程度を組むことができることで注目された．

(1) 可能な行動を 3 手先まで全探索する．
(2) 特殊な場合を除いて，1 手目で連鎖を開始する手を除外する．
(3) 2，3 手先に連鎖を開始できる行動のうち，評価値が最大であるものを選択し，そこに至る 1 手目の行動を採用する．

図 3 に探索例を示す．通常のぷよぷよでは 3 手先までの配石が分かっているので，3 手先の局面を全て列挙することができる．基本的により大きな連鎖を組むことを目指すため，通常 1 手目で連鎖を開始する手は除外する．ただし，次の特殊な場合には選択することもある．

- 盤面の空きが残り少なくなり，これ以上連鎖数を伸ばせない場合．
- 連鎖数を伸ばすために邪魔なぷよを部分的に消す場合．

また，3 手先まで読んでも連鎖が発生しない場合，4 手目以降は読まずに評価値を 0 とする．これは，次の配石が予告されていないために，全ての可能性を考慮する必要があり計算量が非常に大きくなること，またそれを短時間で計算できたとしても，そこから最良の場合を選ぶのか，あるいは平均や最悪の場合を選ぶのかなど難しい判断を必要とするからである．よって，2，3 手先に連鎖を開始できる手の中で最も良い手を選び，その 1 手目を選択することになる．

3. 提案手法

Nested Monte Carlo Search は確定情報ゲームに対して提案された手法であり，未来の状態が確定していない不確定ゲームであるぷよぷよにそのまま適用することはできない．そこで，Nested Monte Carlo Search を不確定ゲームに適用する手法を提案する．

(1) 次にくる可能性のある各操作対象をそれぞれ子ノードとする．
(2) その子ノードの中からランダムにいくつかを選択し，それを探索する．

ぷよぷよでは，現在の盤面と次の 3 手先までの配石が分かっている情報である．1 手進めるとまた新しく 3 手目の配石が予告されるため，あまり長く先読みをしても無駄になりやすいと考えられる．また，人間の思考としても 4 手先以上を推測しながら連鎖を組んでいるわけではないにも関わらず大連鎖を組むことができている．

そこで，計算量を考えて先読みを打ち切る限界数を設定する．また，ぷよぷよには最終局面が存在しないため，Nested Monte Carlo Search における最終局面としては，連鎖を開始する時または設定した限界数まで読んだ時とする．

手順としては次のようなものを考える．

(1) まず，設定した限界数までの配石の予測を多数用意する．
(2) 特殊な場合を除いて，1 手目で連鎖を開始する手を除外する．
(3) ネストレベル n，プレイアウトの回数 r を設定し，Nested Monte Carlo Search を実行する．

プレイアウトの結果をどう扱うかなどについては，様々な条件の元で実験を行う．候補としては，結果の中からもっともよいものを選ぶ，平均したものを選ぶなどを考えている．

また，このように条件を設定すると，Nested Monte Carlo Search をそのまま適用しても効果が得られにくい．なぜなら，どの手を選んでも連鎖数がほとんど変わらない状況が特に序盤に多いからである．この問題はルールベースや他の手法を併用することによって改善を期待する．

4. 評価

本研究ではぷよぷよからアクション性を排除した崩珠 [7] を用いた．崩珠ではぷよぷよと違いターンが存在し，1 ターンに 1 回の行動を取ることができる．

4.1 評価設定

提案手法を評価するため，以下の 3 つのプログラムを用いて性能を比較した．

- PMS：ポテンシャル最大化アルゴリズム（2.3.1）を用いたプログラム．
- MCS：PMS で探索した各終端ノードに対し，モンテカルロ法を用いてランダムに探索するアルゴリズムを用いたプログラム．
- NMCS：提案手法により Nested Monte Carlo Search をぷよぷよに適用させたアルゴリズムを用いたプログラム．

この 3 つのプログラムに同じ乱数による配石を与え，連続する 1,000 ターンの間にどれだけの連鎖が発生したかを比較した．3 つのプログラムに加えた設定は以下の通りである．

- 共通：残りマスが 14 以下になった場合，3 ターン以内に連鎖を開始できる手のうち最大の手を選ぶ．また，途中で 2 連鎖以下を開始する手があっても探索を続け，その後の探索でそれ以上の連鎖が発生するならば

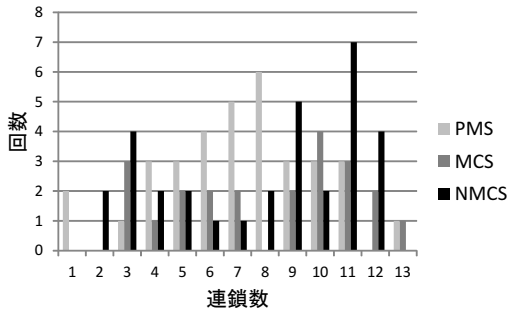

図 4　各プログラムの比較

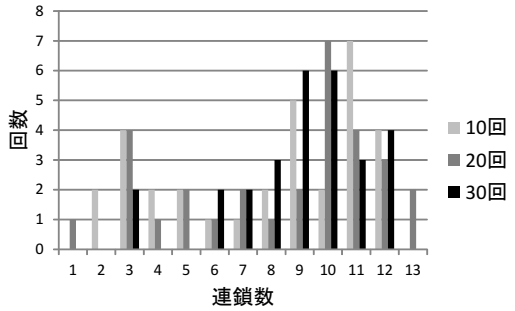

図 5　プレイアウト数による比較

その連鎖数を返す．これは，小さな連鎖を発生させてしまったが将来大きな連鎖を発生させる手のことを考えた設定である．

- PMS：2.3.1 で説明したものに加え，1 手目でも 10 連鎖以上が発生するならばその連鎖数を返す．
- MCS：まず 5 手先までの配石をランダムに 10 個生成する．3 手先まで探索しても連鎖が発生しない葉ノードに対し，生成した配石とランダムな行動を 2 手分まで 10 セット与え，連鎖が発生した場合その連鎖数を返す．
- NMCS：ネストレベル $n = 3$，プレイアウトの回数を 10 回とし，5 手先までの配石をランダムに 10 個生成する．また，PMS と同様に基本的に 1 手目で連鎖を開始する手は選択しない．

また，同様な設定の上で NMCS におけるプレイアウトの回数を 10 回，20 回，30 回と増やし，プレイアウトの回数によって性能がどう変わるかを調べた．

4.2　結果

図 4，5 に結果を示す．図 4 より，PMS よりも MCS，NMCS の方が大きな連鎖の頻度が高くなっており，平均連鎖数は PMS が 7.1，MCS が 8.0，NMCS が 7.9 と MCS と NMCS にあまり違いがなくなっている．また，計算時間を比較したところ，平均して MCS は PMS の約数十倍，NMCS は PMS の約数百倍となっていた．

また，図 5 より，平均連鎖数は 10 回が 7.9，20 回が 8.3，30 回が 9.0 と増加しているが，計算時間はプレイアウト数に比例して増加していた．

4.3　考察

評価結果から，NMCS はその性能と比べて計算時間が非常に長く，Nested Monte Carlo Search をぷよぷよに用いることは計算コストの面から見て割に合わないと考察される．この原因については次のようなものが考えられる．

- 第 3 章で述べたように，どの手を選んでも連鎖数がほとんど変わらない状況が多い序盤の影響によって，Nested Monte Carlo Search の性能が十分に引き出されなかった．
- 提案手法では Nested Monte Carlo Search をぷよぷよに適用するに当たって不十分であった．

5.　おわりに

本論文では Nested Monte Carlo Search をぷよぷよに適用する手法を提案し，そのアルゴリズムの評価を行った．結果としてはモンテカルロ法を用いたアルゴリズムと比較して，期待したほど良い結果が得られなかった．その原因と考えられる 4.3 節で挙げた問題を解決するためには，序盤に対してはルールベースまたは定石形配置法 [6] を用ることや，提案手法に加えて Nested Monte Carlo Search とモンテカルロ木探索を組み合わせた Nested Monte Carlo Tree Search [8] という手法を試みることなどが考えられる．これらについては今後の課題としたい．

参考文献

[1] Cazenave, T.: Nested Monte-Carlo Search, *International Joint Conference on Artificial Intelligence 2009*, pp. 456 – 461 (2009).

[2] 美添一樹：モンテカルロ木探索：コンピュータ囲碁に革命を起こした新手法，情報処理，Vol. 49, No. 6, pp. 686–693 (2008).

[3] Akiyama, H., Komiya, K. and Kotani, Y.: Nested Monte-Carlo Search with simulation reduction, *Knowledge-Based Systems*, Vol. 34, No. 0, pp. 12 – 20 (2012). A Special Issue on Artificial Intelligence in Computer Games.

[4] Rosin, C. D.: Nested rollout policy adaptation for Monte Carlo tree search, *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, AAAI Press, pp. 649–654 (2011).

[5] Schadd, M. P., Winands, M. H., van den Herik, H. J., Chaslot, G. M.-B. and Uiterwijk, J. W.: Single-player monte-carlo tree search, *Computers and Games*, Springer Berlin Heidelberg, pp. 1–12 (2008).

[6] 富沢大介，池田心，シモンビエノ：落下型パズルゲームの定石形配置法とぷよぷよへの適用，情報処理学会論文誌，Vol. 53, No. 11, pp. 2560–2570 (2012).

[7] 崩珠. http://www.jaist.ac.jp/is/labs/ikeda-lab/poje/.

[8] Baier, H. and Winands, M. H. M.: Nested Monte-Carlo Tree Search for Online Planning in Large MDPs., *European Conference on Artificial Intelligence*, Frontiers in Artificial Intelligence and Applications, Vol. 242, IOS Press, pp. 109–114 (2012).