ターン制ストラテジーのための状態評価型深さ限定モンテカルロ法における消極的行動の抑制

藤木翼[†1]　村山公志朗[†1]　池田心[†1]
{s1310062,kosiro_murayama,kokolo}@jaist.ac.jp

将棋などと異なり，ターン制ストラテジーでは全ての駒を任意の順序で動かすことができ，合法手数が膨大になることが探索の障害となる．ターン制ストラテジーのための探索手法としては，攻撃に優れた攻撃行動探索や，防御に優れた DLMC などが存在している．本稿ではこれら 2 つの手法を紹介し，これらを組み合わせた新たな手法を提案する．対戦実験では，提案手法が DLMC に対して 77%の勝率と大きく性能が向上したことを示す．

Depth-Limited Monte-Carlo with a State Evaluation Function for Reducing Negative Behavior

TSUBASA FUJIKI[†1]　KOSHIRO MURAYAMA[†1]
KOKOLO IKEDA[†1]

In turn-based strategy games, all the pieces can be moved in any order in one turn. In this article, we introduce an application of UCT and the previous method DLMC in turn-based strategy games. Moreover, two methods of improving the negative behavior of DLMC were proposed. As the result, the winning ratio of our program against DLMC increased to 77% in battle experiments.

1. はじめに

これまで，チェスや将棋などの古典的なゲームのコンピュータプレイヤに関する研究は幅広く行われてきた． しかし，「一回の手番に複数の駒を動かせる」タイプのターン制ストラテジーと呼ばれるゲームは広く遊ばれているにも関わらず，新しいゲームであることや，探索空間の大きさ，ルールの煩雑さなどもあって，学術的研究は少数が行われているのみである．

ターン制ストラテジーの既存ゲームとしては Battle of Wesnoth，Final Fantasy Tactics，ファミコンウォーズなどが挙げられるが，コンピュータプレイヤは人間の中級者とハンデなしで戦える強さにはほど遠い．ゲーム内ではハンデでその弱さを補っているが，対等に戦いたいという要求には応えられていない．

そこで本稿では，自然さや楽しさといったものの前にまず“強い”コンピュータプレイヤ（AI と略す）を作成することを目的とする．

対象とするゲームは研究用ターン制ストラテジー「TUBSTAP」[1][8]とする．本稿では，過去に提案された手法である深さと着手を限定したモンテカルロ法（Depth-Limited Monte-Carlo : DLMC）[2]や，標準的な探索手法である UCT にターン制ストラテジー用の工夫を加えたもの，攻撃行動探索などを紹介し，それぞれの手法について改めて記述する．その上で，DLMC が消極的な行動を取りやすいという点を改善する 2 つの新しい手法を紹介し，性能評価のため過去に提案された AI との対戦実験を行う．

†1 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology

2. 対象ゲーム

2.1 TUBSTAP のルール

既存ターン制ストラテジーの要素は内政要素やキャラクタの成長要素など数多い．

TUBSTAP（図 1）はその中でも重要な要素である複数着手性に注目し，不完全情報性や占領・生産などの要素を排除した二人零和有限確定完全情報ゲームである．将棋などと比べると複雑ではあるが，既存のターン制ストラテジーに比べると非常にシンプルなゲームとなっている．以下に重要なルールを抜粋する．尚，紹介するこれらの要素は既存のターン制ストラテジーであれば殆どが持っている共通の要素である．

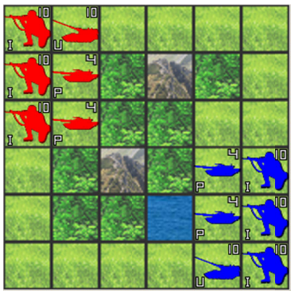

図 1　TUBSTAP のプレイ画面（先手左手側）

R1. 駒

戦闘機(F)，攻撃機(A)，戦車(T)，対空戦車(R)，歩兵(I)，自走砲(U)の 6 種類の駒を用いる．これらの駒間には相性が存在する．相性は後述する HP に与えるダメージ量に影響する．

R2. マップ

将棋等では常に同じ盤面サイズ・駒の初期配置を用いる．しかし，ターン制ストラテジーでは通常，プレイ毎に異なる設定の駒配置や盤面を用い対戦を行う．マスには複数の種類（地形）があり，被るダメージが増減する．

R3. 着手順

各手番では，プレイヤは全ての自駒を 1 回ずつ自由な順番で行動させることが出来る．すべての駒を行動させると相手の手番となる．両者それぞれの手番をターンと呼ぶ．

R4. 勝利条件

いずれかのチームの駒が全滅した場合，駒が盤面に存在しているチームが勝利となる．ただし，ターン数に上限を設け，その上限以内に全滅条件を満たさない場合の判定条件はチーム毎の総 HP 量の差で決定される．

R5. HP

各駒は 1 から 10 の HP を持つ．攻撃を受けることで HP は減少し，0 になるとその駒は盤上から取り除かれる．

2.2 TUBSTAP の特徴

1 ターン内における複数着手性により，ターン制ストラテジーではプレイヤがとれる行動の順列（合法手）が非常に多い．例えば 1 駒あたりの平均合法手が 10，駒数が 6 とすると，1 ターン内に取れる行動の数は 7 億 2000 万通り($= 10^6 \times 6!$)にも及ぶ（同一局面に遷移するものも含む）．実際にはより駒数や 1 駒あたりの合法手数が多い場合も珍しくない．

2.3 ゲーム木の生成方針

ターン制ストラテジーは複数着手性を持つという特性上，木探索の実装方法には大きく分けて 2 通りが存在する．1 つは図 2 のように個々の駒の行動を枝として 1 ターン内の行動を数段に分けた探索木を作る方法，もう 1 つは図 3 のように 1 ターン内に行動可能な全ての駒の行動をまとめて 1 つの枝として探索木を作成する方法である．これらは根からの枝の数こそ異なるものの，1 ターンを終えた後に辿りつくノードは同じ数となり， 2.2 で説明した特徴からこれら全てのノードを調べきることは難しい．過去に提案された AI はどちらかの木の生成法を選択している．

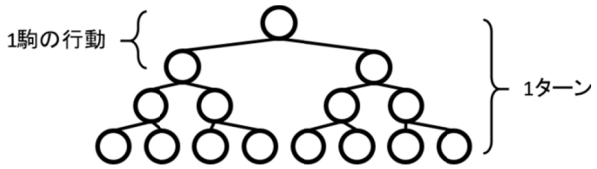

図 2　1 駒の行動を枝としたゲーム木

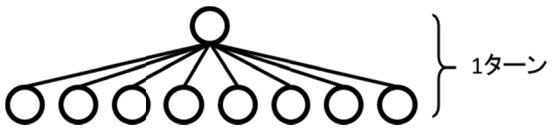

図 3　1 ターンの行動全てを枝としたゲーム木

3. 手法概略

本稿では，既存手法，提案手法を合わせ 6 つの AI を紹介し，性能評価を行う．そのため，それぞれについての立ち位置を予め述べる．性能やそれぞれの特徴などは図 4 に示す通りとなっている．これらの中で 1 駒の行動が 1 つの枝となるゲーム木による探索を行うのは UCT，UCT+Progressive Widening の 2 つであり，その他の AI は 1 ターン内の全ての行動をまとめて 1 つの枝としている．

図にある UCT，UCT+Progressive Widening，攻撃行動探索，DLMC は既存手法であり，DLMC+攻撃行動探索，DLMC+局所調整は今回提案する手法である．各手法について簡単に概要を述べる．

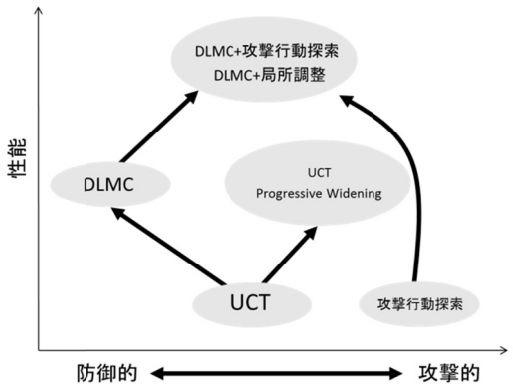

図 4　手法の関係

・UCT

1 駒の行動を 1 ノードとして探索を行う標準的な手法. 1 ターン内に駒数分の探索を繰り返し行う.

・UCT+Progressive Widening

UCT では 1 ターン全ての行動を満足に探索しきれないため，Progressive Widening を使用する.

・攻撃行動探索

攻撃行動による順列のみ探索する手法．移動行動は別のルーチンによって処理する．攻撃行動のみを探索するため, 攻撃的な着手が強い. 一方で, 防御的な着手は移動ルーチンに依存する.

・DLMC

陣形を重視するため，全駒による行動を 1 つのノードとして探索を行う手法．ただし，膨大な探索空間の中で，ランダムに選んだ一部のノードのみ評価を行う．また，各着手の評価はシミュレーションによって行う．この際，終局までのシミュレーションを行わず， 途中で打ち切る．打ち切った局面に対して状態評価関数を用いることで着手の評価とする．着手の生成方法が原因となり，防御的な行動を取りやすい性質を持つ．また，UCT と異なり選択されたノードは全駒の行動を含むため，1 ターンにおける探索は 1 回でよい.

・DLMC+攻撃行動探索

攻撃に優れた攻撃行動探索，防御に優れた DLMC の 2 つを用い，それぞれの得意分野を利用することで総合的に優れた手を選択する手法.

・DLMC+局所調整

DLMC の探索対象に攻撃行動が含まれやすいように調整する手法.

4. 既存手法

以下に本稿で用いる記号を示す.

s ： 盤面の状態.
S : 全ての盤面の状態sの集合.
a : 1 つの駒の行動
A(s) ： 状態sの合法手となる全ての行動aの集合.
$s'(s,a)$ ： 状態sで行動aを行った次状態.
　　行った場合，行動可能な駒が残っている.
$p=\{a_1,a_2,\dots\}$:1 ターン内全ての駒による行動の順列.
P(s) ： 状態sの合法手となる行動の順列pの集合.
$s'(s,p)$ ： 状態sで行動の順列pを行った次状態.
　　行った場合，相手の手番となる.
$g(s)$ ： 状態評価関数による状態sの評価値.

4.1 UCT

UCT はモンテカルロ木探索の一種であり，探索の指標に UCB1 値（式 1）を用いる．囲碁や将棋など様々なゲームでしばしば用いられる手法であり，ターン制ストラテジーであっても有望な手法である可能性が高い．しかし，ターン制ストラテジーは 1 ターンの候補手が膨大であるという理由から，単純に UCT による探索を行ったとしても，強い AI を作ることは難しい．一方で，過去に UCT の枝刈りを行う事で AI の性能が向上する事が確認されている [3]．そこで枝刈りの代わりとして Progressive Widening を用いて探索空間を狭める.

尚， 1 ターンの行動全てを 1 つの枝とするゲーム木を用いる場合，UCT としての利点が生かせないため，1 駒の行動を 1 枝として実装する.

$$\mathrm{UCB1}=\bar{x}_j+\sqrt{\frac{2\log n}{n_j}} \qquad \text{（式 1）}$$

$\bar{x}_j$: あるノード j の勝率
n_j:あるノード j のシミュレーション回数
n:シミュレーション回数の合計

4.1.1 Progressive Widening

Progressive Widening（PW と略す）[4]とは，ある局面の訪問回数 n に応じて，有望な着手だけを探索する手法である．本稿では，攻撃行動を優先して探索させる為 PW を導入する.

具体的には，式 2[5]に従って探索する候補手を増やしていく．追加する候補手は，攻撃行動を重複無くランダムに選ぶものとする．攻撃行動を全て追加した場合，移動行動をランダムに追加する.

$$候補手 = \frac{\log\frac{n}{40}}{1.4} + 2 \qquad (n < 3000)$$

$$候補手 = \frac{\log\frac{n+2000}{45}}{1.2} - 11 \quad (n > 3000) \qquad （式 2）$$

n:シミュレーション数

4.1.2 シミュレーションの詳細

TUBSTAP における駒の行動は大きく分けて単純に移動するだけの行動と攻撃を含む行動の 2 通りがある. 1 つの駒が持つ行動のうち攻撃する行動は移動するだけの行動に比べると平均的には非常に少ない．そのため，完全にランダムなシミュレーションを行うと指定ターン内に勝敗が決せず，引き分けで終わることが多々発生する．これはチェス等でも指摘される点である．では攻撃を優先してシミュレーションを行えば良いかというと，今度はシミュレーション中で安易あるいは無謀な攻撃が増加するために，探索結果としてそれらを期待する待ちの行動が多くなってしまう．

そこで本稿では攻撃可能な駒が存在する状況ならば必ず攻撃を行う方策と完全にランダムな行動を行う方策の 2 つを混合して用いている（前者が 8 割，後者が 2 割）．

尚，これらのシミュレーション手法には様々な改善が考えられるが，探索手法の比較という観点から本論文の実験ではすべての手法で同じシミュレーション手続きを用いている．

4.1.3 予備実験：PW の効果

PW の効果を確認するために予備実験を行った．実験条件は以下の通りである．

・マップは図 1 と同じものを用いる
・先手後手 500 戦，計 1000 戦
・AI は UCT，UCT+PW の 2 種

また，本稿で用いる UCT は 1 ターン内に行えるシミュレーション数を固定とし, 1 回の行動に使用するシミュレーション数は駒数で割った数を用いる．例えば，操作可能な駒が 6 駒存在し，1 ターン内に行えるシミュレーション数が 6000 回であるとすると，1 ターン内に 6 回の行動が行われる為，1 回の行動に費やすシミュレーションは 1000 回となる．

それぞれのパラメータはシミュレーション数 6000 とした．計算時間はほぼ同一となっている．結果を表 1 に示す．尚，勝率は引き分けを勝利の半分として算出している．

表 1　UCT+PW の UCT に対する戦績

勝利数	引き分け	敗北	勝率
982	18	0	99.1

表 1 から UCT+PW の勝率が十分に伸びていることが確認できる．UCT+PW の式 1, 2 は調整していないため，性能を向上できる余地は大きく残されている．

4.2 攻撃行動探索

村山らは攻撃する行動に注目した手法を提案した[1]．以下に詳細を示す．

【攻撃行動探索】

I. 状態sにおける長さℓ以下の攻撃行動のみで構成される順列を全て列挙する．攻撃行動が存在しない場合と長さℓ以降の行動は特定のルーチンに従った行動を順列に追加し 1 ターン分の順列とする．

$$\mathrm{P}'(\mathrm{s}) = \{p_1, p_2, \ldots, p_x\} \subset \mathrm{P}(\mathrm{s})$$

II. $\mathrm{P}'(\mathrm{s})$に含まれる順列pから遷移する次状態$\mathrm{s}'(\mathrm{s}, p)$を状態評価関数$g$により評価し，最も評価の高い順列$p$を選択する．

$$p^* = \underset{p \in P'(\mathrm{s})}{\operatorname{argmax}}\, g\big(\mathrm{s}'(\mathrm{s}, p)\big)$$

I において長さℓ以下の攻撃行動にのみで構成される順列しか探索を行わないのは計算量爆発を防ぐ為である．たとえ攻撃行動のみに限定したとしても全駒が偶然攻撃行動をとれる場合，計算量の爆発が起こりえる．そこで事前に探索する順列の長さを制限しておくことで，計算量の爆発を防いでいる．

この手法は図 5 のような駒の行動順が重要となる局面で上手く動作する．ここで大文字小文字はチームを表しており，数字は残り HP，×印は移動できないマスを表す．駒 U（自走砲）は移動しなければ距離 2-3 までの位置に攻撃することができ，

駒 P（戦車）は移動後も攻撃が可能だが隣接する駒にしか攻撃できない．そのため，*P* を先に動かしてしまうと *u* を攻撃できずターンが終了してしまう．最善となるのは *U* で先に *p* を攻撃し *P* で *u* を攻撃する場合である．攻撃行動探索は攻撃行動のみ 2 手目以降を読み切るためこういった局面でも上手く探索が行える．ただし，上手く動作するのは攻撃が行える場合のみである．

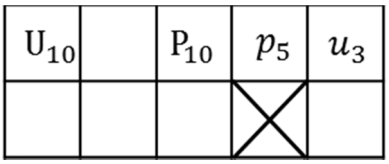

図 5　攻撃行動探索がうまく働く局面の例
（先手：大文字）

図 6 のような状況は攻撃行動探索で上手く探索できない例である．*U* は隣接する *p* に対して攻撃できず, u に対しては攻撃できるが戦果が低く反撃を受けてしまう．また，*U* の位置取りが邪魔なため *P* も p に対して攻撃できない．このような状況であれば，先に *U* を退かした上で，*P* で *p* を攻撃するのが最善となる．

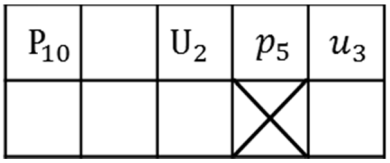

図 6　攻撃行動探索が失敗する局面例
（先手：大文字）

また，図 1 のような初期局面でも上手く動作しない，このような初期局面は移動ルーチンしか動作しない為である．図 1 のような局面を得意とするのは次に紹介する DLMC となる．

4.3 DLMC

Depth-Limited Monte-Carlo[2]は図 7 に示す通り 2 つの特徴がある．1 つは膨大なノード（合法手となる順列）の中で極一部しか探索を行わない事．もう 1 つはモンテカルロ手法をベースとしているが，シミュレーションは途中で打ち切ることである．シミュレーションの打ち切りはボードゲーム Amazon において優れた結果を示している手法である[6][7]．尚，UCT+PW に打ち切りを加える事も行ったが，単純な実装では性能が落ちてしまう為本稿では扱わない．

【DLMC】

I.　順列pをランダムにm個サンプルする．

$$P'(s) = \{p_1, p_2, \dots, p_m\} \subset P(s)$$

II.　$P'(s)$に含まれるpによる次局面$s'(s,p)$を d ターン先までシミュレーションし，状態評価関数$g(s)$により評価を行う．これをn回繰り返し，評価値の合計を測定する．

$h_i(s): S \to S$　i 回目のシミュレーション．
d ターン進める．

$$g_n(p) = \sum_1^n g(h_i(s'(s,p)))$$

III.　$P'(s)$の中で評価最大の行動p^*を選択する．

$$p^* = \underset{p \in P'(s)}{\operatorname{argmax}}\, g_n(p)$$

DLMC ではシミュレーションによる局面の評価を 1 ターン内の全ての駒の行動が終了した局面に限定している．これにより，全駒の動作が決まった状態を評価するため，駒間の相互位置を評価しやすい．しかし，全駒を動作させた場合，遷移する局面は膨大な数となる．そこで，探索空間を減らすため，ランダムな前向き枝刈りを用いている．これにより高速な動作が可能となっていが，良い手の見落としのリスクが大きく存在する．

ランダムな着手を用いる DLMC は移動するだけの行動を取りやすく，防御的な行動を取りやすい．これは攻撃する行動より移動する行動の方が多いためである．このような特性から図 1 のような初期局面において上手く動作するが，図 5 のような局面には弱い．また，図 6 のような状況であれば，DLMC は一旦引く手を取りやすく，その後の相手の行動次第で打開できる状況に持ち込みやすい．

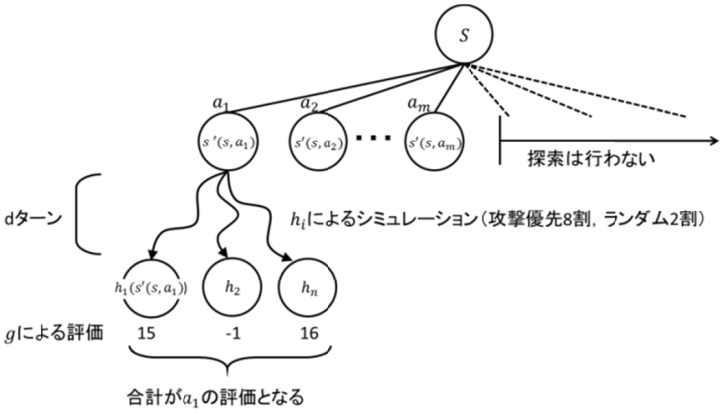

図 7　DLMC の概要図

4.3.1 状態評価関数

打ち切った後に使用する状態評価関数 g(s)は重要な要素ではあるが, 1 回のシミュレーション毎に使用するため， DLMC ではシンプルな状態評価関数を用いている．

$$g(\mathrm{s}) = (\mathrm{M} - \mathrm{E}) \times \mathrm{B}$$

$$\mathrm{B} = \begin{cases} 2, & \text{if M} = 0 \text{ or E} = 0 \\ 1, & \text{else} \end{cases}$$

M:自チームの駒の HP の総和（駒 I は 0.2 倍）
E:敵チームの駒の HP の総和（駒 I は 0.2 倍）
B:ボーナス係数

ここで B は勝敗が決した際に，対戦中の場合と区別し，評価値を倍にするための係数である．勝利であれば自チームの方が敵チームよりも HP の総和が大きいためより良い評価となり，敗北であれば敵チームの方が自チームよりも HP の総和が大きいためより悪い評価となる．

また, 駒 I の HP を 0.2 倍して使用しているのは，I の攻撃力が非常に弱く，活躍があまり期待できないためである．他の駒に重みはかけられていない．

5. 提案手法

攻撃行動探索は攻撃的な手を取ることに優れている一方で，防御的な手を取れない．特に図 1 のような状況で陣形を取ることが出来ない．DLMC であればこのような状況であっても陣形を取ることが可能である．しかし，図 5 のような詰め行動は非常に弱くなる．

これら 2 つはお互いに明確な欠点を持っており，補える立場にある．そこで， DLMC に攻撃行動探索を組み合わせることで性能の向上を図る．

5.1 攻撃行動探索と DLMC の併用

単純な手法ではあるが DLMC の着手に攻撃行動探索の最善手を組み込む事で性能の改善を図る．

【提案手法 1: DLMC+攻撃行動探索】

I. 順列pをランダムに m 個サンプルする．

$$\mathrm{P}'(\mathrm{s}) = \{p_1, p_2, \dots, p_m\} \subset \mathrm{P}(\mathrm{s})$$

II. 状態sを攻撃行動探索し，その中で最も評価の高い順列p'を求める．さらに行動p'を$\mathrm{P}'(\mathrm{s})$に加える．

$$\mathrm{P}'(\mathrm{s}) = \{p_1, p_2, \dots, p_m, \boldsymbol{p'}\} \subset \mathrm{P}(\mathrm{s})$$

III. $\mathrm{P}'(\mathrm{s})$に含まれるpによる次局面$\mathrm{s}'(\mathrm{s},p)$をｄターン先までシミュレーションし，状態評価関数$g(s)$により評価を行う．これをn回繰り返し，評価値の合計を測定する．

$\mathrm{h}_i(s): S \to S$　i 回目のシミュレーション．
d ターン進める．

$$g_n(p) = \sum_1^n g(h_i(s'(s,p)))$$

IV. $\mathrm{P}'(\mathrm{s})$の中で評価最大の行動p^*を選択する．

$$p^* = \underset{p \in P'(\mathrm{s})}{\operatorname{argmax}}\, g_n(p)$$

この手法は特定の局面𝒔における良い手の見逃しを防ぐ事を目的としている．図 8 にあるように攻撃行動探索にとって最も評価の高い 1 つの順列を DLMC の着手に加える事で動作している．

攻撃行動探索による最善手と DLMC のランダムな合法手がそれぞれ別個𝒔にシミュレーションによって評価される為，図 1 や図 5 のような状態どちらの状況にも対応することが出来る．

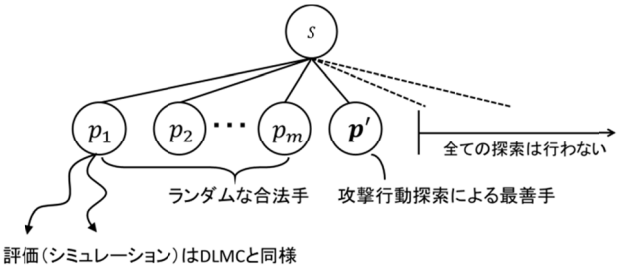

図 8　DLMC+攻撃行動探索の概要図

5.2 局所調整

局所調整では DLMC でサンプルした解のうち有望なものを更に選別し，そのうえで最後の数駒の行動に攻撃行動探索を用いて変更する．

【提案手法 2:DLMC の局所調整】

I. 順列pをランダムにm個サンプルする．

$$\mathrm{P}'(\mathrm{s}) = \{p_1, p_2, \dots, p_m\} \subset \mathrm{P}(\mathrm{s})$$

II. $\mathrm{P}'(\mathrm{s})$に含まれるpによる次局面$\mathrm{s}'(\mathrm{s},p)$をｄターン先までシミュレーションし，状態評価関数$g(s)$により評価を行う．これをn回繰り返し，評価値の合計を測定する．

$\mathrm{h}_i(s): S \to S$　i 回目のシミュレーション．
d ターン進める．

$$g_n(p) = \sum_1^n g(h_i(s'(s,p)))$$

III. P′(s)で評価の高い上位 j 個の順列を抜き出す．

$$P''(s) = \{p_1, p_2, \ldots, p_j\} \subset P'(s)$$

IV. P″(s)に含まれる順列pの行動を末尾から k 個削除し，攻撃行動探索をよる k 個の行動から成る順列p'をpに加える．ここで 1 ターンに行動可能な駒数は u とする．

$p = \{a_1, a_2, \ldots, a_{u-k}\}$　行動の削除
$p' = \{a'_1, a'_2, \ldots, a'_k\}$　p'の生成
$p'' = \{a_1, a_2, \ldots, a_{u-k}, a'_1, a'_2, \ldots, a'_k\}$
$P'''(s) = \{p''_1, p''_2, \ldots, p''_j\} \subset P(s)$

V. II と同様に$P'''(s)$を評価する．

VI. $P'(s), P'''(s)$の中で最も評価の高い順列pを選択する．

$$p^* = \underset{p \in P'(s) \cup P'''(s)}{\operatorname{argmax}} g_n(p)$$

提案手法 2 は DLMC の中で評価の高い合法手を攻撃行動探索に調整させることで動作している．図 9 に示すように攻撃的な調整がされた合法手が j 個増えることになる．提案手法 2 において順列pにおける末尾行動を変更するのは，動作可能な駒数が減った状況であれば提案手法 1 に比べれば探索すべき局面が減り，同じ長さの順列分探索したとしても計算コストが少なくなるためである．また，経験則となるが 1 ターンの全行動の内，末尾において攻撃をかける事が良い合法手となる場合が多いように思える．例えば図 6 のような状況で最善となる合法手は攻撃行動を全行動の内の末尾において行う事で得られる．

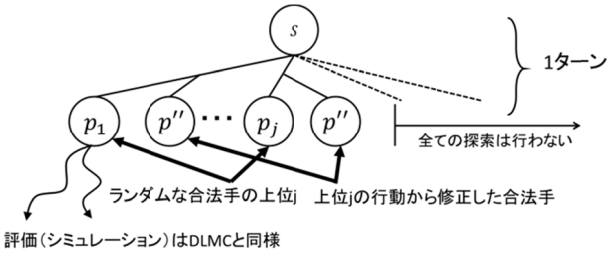

図 9　DLMC+局所調整の概要図

6. 対戦実験

本稿で紹介した UCT+PW，DLMC，DLMC+攻撃行動探索，提案手法 1，提案手法 2，提案手法 1+2 のそれぞれを用いて対戦実験を行う．実験条件は 4.1.3 と同様とし，計算時間を揃える為，パラメータは表 2 の通りとした．1 ターンの実行時間は Core i5 3.3GHz，メモリ 8GB の PC を用いて約 3 秒程度としている．尚，表には存在しないが DLMC と提案手法において使うシミュレーションの深さ d は 2，提案手法 1 において攻撃行動探索が調べる順列の長さℓは 4，提案手法 2 の抜き出す順列数 j は 10，書き換える行動数 k は 4 とした．また，提案手法 1+2 は提案手法 2 のⅠ～Ⅲを実行した後に提案手法 1 のⅡ～Ⅳを動作させる形で成り立たせ，サンプル数とシミュレーション数以外のパラメータはそれぞれと同じものを使う．

対戦結果を表 3 に示す．数値はそれぞれ，勝利数－引き分け数－敗北数となっており，括弧内は勝率を表す．

結果より，提案手法がそれぞれ DLMC と UCT に大きく勝ち越している事がわかる．また DLMC に比べると自己対戦における引き分け数が減っており，防御的な行動を取りやすかった問題点を解決できている．UCT+PW の方が DLMC に比べて提案手法に対する勝率が良いのは，提案手法が DLMC に比べて改善している点が攻撃的な部分である為，元々攻撃的な部分が強い UCT+PW に対する勝率は DLMC からあまり伸びなかったものと考えられる．

表 2　AI のパラメータ

AI名	サンプル数	シミュレーション数
DLMC	200	100
UCT+PW	-	6000
提案手法1	150	100
提案手法2	150	100
提案手法1+2	100	100

表 3　対戦結果 win – draw - loss (winrate)

AI名	vs UCT+PW	vs DLMC	vs 手法1	vs 手法2	vs 手法1+2
UCT+PW	468 – 38 – 494 (48.70)	-	-	-	-
DLMC	427 – 245 – 328 (54.95)	385 – 220 – 395 (49.50)	-	-	-
手法1	521 – 258 – 221 (65.00)	588 – 129 – 283 (65.20)	448 – 78 – 474 (48.70)	-	-
手法2	552 – 271 – 207 (65.75)	629 – 148 – 223 (70.30)	509 – 89 – 402 (55.35)	424 – 114 – 462 (48.40)	-
手法1+2	537 – 259 – 204 (66.65)	711 – 119 – 170 (77.05)	494 – 134 – 372 (56.10)	468 – 136 – 396 (53.60)	437 – 122 – 441 (48.70)

7. 人間プレイヤとの対戦

TABSTAPの元となったファミコンウォーズDS2を50時間以上プレイしたプレイヤ2人が手法1+2を用いたAIと対戦した結果，AIの4勝3分3敗という結果となった．対戦回数は少ないものの，AI側が勝ち越している．現状，対戦に用いるマップサイズは小さな物に限定している．小さなマップという前提条件さえあればAIであっても人間とまともな対戦が出来るという結果となった．

また，人間プレイヤの感想を聞くと「一度防御に周ると的確に攻めてくる」「面倒な人間プレイヤに近い」といった感想が多く聞けた．人間プレイヤが楽しむ相手としてはあまり良くないAIかもしれないが，AIの強さという面では十分な性能が出せている．

8. まとめと今後の展望

本稿では，ターン制ストラテジーにおけるゲーム木の生成方針から，DLMCの詳細，UCTのターン制ストラテジーへの適用方法，などを紹介した上で新たに2つの手法を提案した．

提案した2つの手法と過去に提案されている手法で対戦実験を行う事で提案手法の有用性を示し，UCT+PWのAIに対し最大で66.65%の勝率を示した．また，UCT+PWに勝ち越すDLMCに対しては77.05%の勝率となることを報告した．

また，現在の対戦実験で用いているマップは非常に小さいものであり，実際のターン制ストラテジーでは駒数やマップサイズはより大きなものとなる．生産などのルールを含めると現在のAIの方針では対応しきれない．そこで盤面の一部を抽出して現在のAIに与えるために局面の分割をする機構や，分割されたそれぞれの局面に整合性を持たせた上で動作させる為の統合AIなどが必要である．

参考文献

[1] 村山，藤木，池田，“学術研究用プラットフォームとしての大戦略系ゲームのルール提案”，IPSJ-GPW 2013-11-01，pp.146-153

[2] 藤木，村山，池田，ターン制ストラテジーのための状態評価型深さ限定モンテカルロ法，第8回E&Cシンポジウム,2014-3-19

[3] 加藤，三輪，鶴岡，“ターン制ストラテジーゲームにおける戦術決定のためのUCT探索とその効率化”，IPSJ-GPW 2013-11-01，pp.138-145

[4] Remi Coulom，“Computing Elo Ratings of Move Patterns in the Game of Go”， ICGA Journal Vol.30 pp.198-208，2007

[5] 美添，山下，“コンピュータ囲碁 モンテカルロ法の理論と実践”，共立出版，（2011）

[6] Kloetzer, J., Iida, H., Bouzy, B.，“The Monte-Carlo Approach in Amazons” Computer Games Workshop,2007

[7] Kloetzer, J.，“Monte-Carlo Techniques:Applications to the Game of the Amazons”, Japan Advanced Institute of Science and Technology, 2010，博情第240号 Includes bibliographical references pp. 87-92

[8] ターン制戦略ゲーム 学術用基盤プロジェクト TUBSTAP，http://www.jaist.ac.jp/is/labs/ikeda-lab/tbs