静的評価関数を用いたUCTの改善

中村秋吾[†]　三輪誠[†]　近山隆[†]

コンピュータゲームプレイヤにおいて、探索はその振る舞いを決める重要な処理の一つである。探索法の中でもUCT探索は知識表現が難しい囲碁のコンピュータゲームプレイヤにおいて良い結果を残している手法である。本論文では、新しいゲームであるためにまだ知識がほとんど得られていないBlokus Duoを対象とするコンピュータゲームプレイヤについて、まず$\alpha\beta$探索とBouzy's 4/0 algorithmを使用した静的評価関数を用いた方法で実装した。そして、囲碁の分野でよい結果を出しているUCT探索を用いた手法で実装を行い、$\alpha\beta$探索を用いたコンピュータゲームプレイヤとの比較を行い、UCT探索の有効性を評価した。また、Bouzy's 4/0 algorithmを使用した静的評価関数を用いてUCT探索の改善を行い、コンピュータゲームプレイヤの比較を行った。更に、UCT探索のシミュレーションによって得られる勝率は静的評価関数に3層ニューラルネットワークを用いたUCT探索を実装し、結果から考察を行った。結果としてBlokus Duoにおけるコンピュータゲームプレイヤでは静的評価関数として簡単なBouzy's 4/0 algorithmを用いた$\alpha\beta$探索よりも通常のUCT探索の方が強くなることがわかった。また、通常のUCT探索とBouzy's 4/0 algorithmを静的評価関数として加えたUCT探索では後者の方が強くなることがわかり、UCT探索に静的評価関数を加えることは有効な改善手法であることが確認できた。

Improvement of UCT using evaluation function

Shugo NAKAMURA,[†] Makoto MIWA[†]
and Takashi CHIKAYAMA[†]

In the computer game player, searching game tree is one of the important processing. The UCT search is a essential algorithm to generate good computer game players in Go. In this thesis, we made four kind of computer game players for Blokus Duo using $\alpha\beta$ search method with Bouzy's 4/0 algorithm, UCT search method, UCT search method with Bouzy's 4/0 algorithm and UCT search method with 3layer perceptron. And we compared those computer game players as an evaluation. As experimental results, UCT search method was overcame $\alpha\beta$ search. And UCT search method with Bouzy's 4/0 won UCT search method. But UCT search method with 3layers perceptron was defeated by UCT search method.

1. はじめに

コンピュータゲームプレイヤにおいて、探索はその振る舞いを決める重要な処理の一つである。探索法の中でもUCT探索は知識表現が難しい囲碁のコンピュータゲームプレイヤにおいて良い結果を残している手法である。UCT探索は知識表現の難しいゲームにおいても結果さえ分かれば探索を行うことができるという特徴を持つため、知識が得られていないゲームにおいてもUCT探索を用いることは有効であると考えられる。

本論文では、歴史が浅く、まだ知識があまり得られていないBlokus Duoを対象とするコンピュータゲームプレイヤにおいてUCT探索を用いることでどの程度の効果が得られるかを調べた。まずコンピュータゲームプレイヤを$\alpha\beta$探索とその静的評価関数にBouzy's 4/0 algorithmを用いる方法で実装した。そして、UCT探索を用いた手法でも実装を行い、$\alpha\beta$探索を用いたコンピュータゲームプレイヤとの比較を行った。また、静的評価関数で得られる評価値とUCT探索のシミュレーションにより得られる勝率には相

† 東京大学大学院新領域創成科学研究科
Graduate School of Frontier Sciences, The University of Tokyo

関性があるものと仮定し、Bouzy's 4/0 algorithm を使用した静的評価関数と UCT 探索とを組み合わせて UCT 探索の改善を行いコンピュータゲームプレイヤの比較を行った。更に、この改善させた UCT 探索の静的評価関数に 3 層ニューラルネットワークを用いたプレイヤを実装した。これは予め時間をかけて UCT 探索のシミュレーションを行っておき、得られた各盤面とその勝率の関係を 3 層ニューラルネットワークに学習させたものであり、UCT 探索のシミュレーションを行う際に勝率の初期値として各盤面に予め設定することで、UCT 探索を改善させることを試みた。結果として Blokus Duo におけるコンピュータゲームプレイヤでは、静的評価関数として簡単な Bouzy's 4/0 algorithm を用いた $\alpha\beta$ 探索よりも通常の UCT 探索の方が強くなることがわかった。また、通常の UCT 探索を用いたプレイヤと UCT 探索に Bouzy's 4/0 algorithm を静的評価関数として加えたプレイヤでは後者の方が強くなることがわかり、UCT 探索に静的評価関数を加えることは有効な改善手法であることが確認できた。

本論文では以降、2 章で本研究に関連する研究を紹介し、3 章で本研究の手法を説明し、4 章で実験結果についてを述べ、5 章でまとめと今後の課題を述べる。

2. 関 連 研 究

本章では関連研究として 2.1 で Monte Carlo 探索を紹介し、2.2 で Monte Carlo 探索を探索効率の面で改良した UCT 探索に関して紹介する。

2.1 Monte Carlo 探索

Monte Carlo 探索は囲碁などの知識表現が難しいゲームのコンピュータゲームプレイヤにおいてよく用いられている探索法の 1 つである。将棋やチェスのコンピュータゲームプレイヤにおいては $\alpha\beta$ 探索が広く使われていて良い結果を残しているが、囲碁などの知識表現の難しいゲームにおいては盤面の評価を行うための良い静的評価関数が手に入らないため将棋やチェスなどと同様の手法では良い結果を出すことができず別の探索法が望まれていた。1993 年に囲碁の世界において、Monte Carlo 探索を用いた探索を行うコンピュータゲームプレイヤが登場し注意を引くこととなった[6]。Monte Carlo 探索はゲームに関する知識を用いた静的評価関数を持たずに探索を行うアルゴリズムで、ある盤面の状態からランダムな手を終局までシミュレートすることを何回も重ねて勝率を求めることでその盤面状態を評価する手法である。

Monte Carlo 探索の欠点としてはすべての手をランダムで選択するために、実際には選び得ないような手も選択してしまい探索の効率が落ちることが挙げられる。Monte Carlo 探索を用いた囲碁のコンピュータゲームプレイヤとしては CrazyStone が有名である[7]。CrazyStone では探索効率を上げるために、静的評価関数を Monte Carlo 探索に組み入れた手法を用いている。この CrazyStone で用いられている静的評価関数は 3×3 のパターン認識を用いた盤面評価と共に手の確率を棋譜から学習させることであり得ない手の探索を省いて短い時間で良い結果を出している。

2.2 UCT 探索

UCT 探索は Monte Carlo 探索の欠点を解消し効率の良い探索を行うよう改良した探索法である。知識表現が難しい囲碁のコンピュータゲームプレイヤにおいて良い結果を残している手法である UCT 探索はランダム性を持つ手の選択を繰り返し終局まで探索するシミュレーションと呼ばれる処理を行うことで良い手を選択する。UCT 探索ではシミュレーション中に未探索の盤面に達した場合、そこから先の探索を Monte Carlo シミュ

レーションにより終局まで探索を行い、終局で得た評価を経由した盤面の値に反映させる。また、まだ探索していない手から優先的に探索を行っていき、すべての手を1回以上探索したら、それぞれの盤面の値からUCT値Uを次の式(2)から計算し、その値が最も大きいものを次の手として選択する。

$$U = \hat{X}_i + \sqrt{2\frac{\log T}{T_i}} \qquad (1)$$

ここで$\hat{X}_i$はi番目の手を経由した際に得た報酬の和の平均値、T_iはi番目の手をシミュレーション時に通った回数、Tはシミュレーション時に現在の盤面を通った回数とする。

UCT探索は$\alpha\beta$探索などと異なり、盤面を評価するために事前知識を用いた静的評価関数を必要としない。また、UCT探索は良さそうな手を多く探索するため、探索木は一様ではなく偏ったものになるという特徴を持つ。したがって事前知識により強い静的評価関数を作ることができないゲームにおいてはUCT探索が有効であると考えられている。

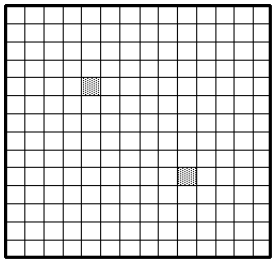
UCT探索の問題点としては、シミュレーション中に未探索の盤面に達した場合、そこから先の探索をMonte Carlo探索同様にランダムで行うため、実際にはあり得ないような手も探索してしまう可能性があるという点がある。囲碁のコンピュータゲームプログラムのCrazyStoneではMonte Carlo探索において、静的評価関数を用いてあり得ない手の探索を省いて短い時間で良い結果を出しているため、UCT探索においても静的評価関数を用いることで効率のよい探索が行えると考えられる。

Sylvain GellyとDavid Silverの研究では9路碁におけるコンピュータゲームプレイヤでUCT探索を改良するいくつかの手法についてが提案されている[5]。まず、UCT探索のシミュレーション時において、手の選択を完全なランダムではなく線形関数を用いた静的評価関数によって手の選択を決める確率分布を変化させることでUCT探索による対戦成績が上がることが示されている。また、UCT探索のシミュレーション時における各盤面の評価の初期値は通常ランダムであるが、これを線形関数を用いた静的評価関数を用いて盤面の評価値を得てそれを初期値として用いることで、既に一定数のシミュレーションを行ったものと同等の効果を得ることができ探索効率を挙げることができることが示されている。これは静的評価関数で得られるジェネリックな評価値とUCT探索のシミュレーションで得られるローカルな評価値は相関性があるという仮定に基づいている。

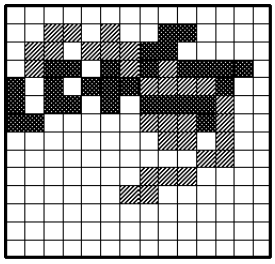
3. Blokus Duoを対象としたUCT探索の改善

本研究はBlokus Duoにおける強いコンピュータゲームプレイヤを作成することを目的としている。

Blokus Duoとは2000年に登場した新しいゲームであり、縦横14マスの格子状に区切られた盤上に、1〜5個の正方形をつなげた21個の違う形のピースの角と角をつなげて配置していき、ゲーム終了の時点で多く置いた側が勝つという二人・零和・有限・確定・完全情報を満たした陣地を取り合うルールのゲームである。初手は決められた枡を覆うようにして置かなければならない（図1(a)）。また2手目以降は盤面に置いた自分のブロックの角のみに接するようにして次のブロックを置かなければならない（図1(b)）。先手が用いるブロックの色はパープル、後手が用いるブロックの色はオレンジと分けられている。Blokus Duoの特徴としては新しいゲームであるため知識がまだほとんど存在しないことや盤面が変化しやすいことが挙げられる。また、Blokus Duoにおいては強いコンピュータゲームプレイヤがまだ存在してい



(a) Blokus Duo の初期盤面。マークの位置を覆うようにしてそれぞれの初手を置かなければならない。



(b) 対戦中の盤面例。自分のブロックの角のみに接するように置いて繋げていく。

図 1 Blokus Duo の盤面

ない。

したがって Blokus Duo は囲碁のように UCT 探索による効果が得られやすいゲームであると考えられる。そこで本研究では Blokus Duo を対象として、まず $\alpha\beta$ 探索とその静的評価関数として簡単な Bouzy's 4/0 algorithm を用いたプレイヤを実装する。そして通常の UCT 探索を用いたプレイヤを実装しそれらの比較を行う。次に UCT 探索に Bouzy's 4/0 algorithm を用いた静的評価関数を組み込んだプレイヤを実装し比較を行う。更に UCT 探索に組み込む静的評価関数を 3 層ニューラルネットワークで表現したプレイヤを実装し比較を行う。

本章では以降 3.1 で $\alpha\beta$ 探索とその静的評価関数として Bouzy's 4/0 algorithm を用いたプレイヤについて説明し、3.2 で UCT 探索に Bouzy's 4/0 algorithm による静的評価関数を組み込んだプレイヤについての説明を行い、3.3 で UCT 探索に組み込む静的評価関数を 3 層ニューラルネットワークで表現したプレイヤの説明を行う。

3.1 $\alpha\beta$ 探索と Bouzy's 4/0 algorithm の静的評価関数を用いたプレイヤ

$\alpha\beta$ 探索と静的評価関数を用いたコンピュータゲームプレイヤを実装した。

静的評価関数には Gnu Go 2.X で用いられていた Bouzy's 4/0 algorithm を用いる[8]。このアルゴリズムは盤面中で陣地が影響している分布を調べるために用いられるものであり、囲碁においては効果が得られている手法である。

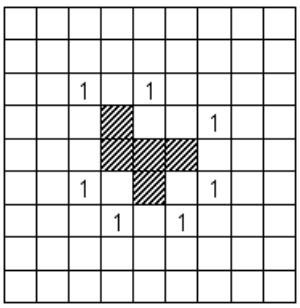
具体的には、まず初期状態として、ブロックを置ける枡を 1、それ以外の枡を 0 と表す。そして次にその上下左右の枡の数字をインクリメントする DILATION 操作を行い陣地の影響範囲を広げる。今回は DILATION 操作を 4 回繰り返し陣地の影響範囲を広げたものを自陣・敵陣それぞれの影響の分布として用いる（図 2)。

今回用いる静的評価関数ではこのアルゴリズムによって得られた影響分布の合計値を自陣・敵陣のそれぞれについて求め、自陣の合計値から敵陣の合計値を引いた値を評価値として算出している。これによって自陣の有利らしさが得られることを期待している。

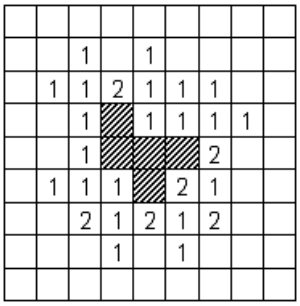
3.2 UCT 探索に Bouzy's 4/0 algorithm の静的評価関数を加えたプレイヤ

UCT 探索に Bouzy's 4/0 algorithm の静的評価関数を取り入れたプレイヤを実装した。

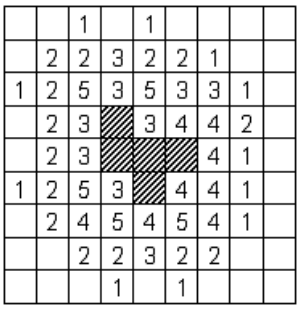
UCT 探索はシミュレーション中に未探索の盤面に達した場合、そこから先の探索をランダムで行うため、あり得ない手までも探索してしまうことがあるために探索効率が落ちるという欠点があった。しかし、これは予め静的評価関数を用いて勝率にあたりをつけることで改善することができると考えられ、実際に 9 路碁を対象としたコンピュータゲームプレイヤにおいては効果が得られることが示されている。よって Blokus Duo を対象とするコンピュータゲームプレイヤにおいても静



(a) 初期設定。ブロックを置ける枡に 1 を設定する。



(b) DILATION を 1 回施した結果。



(c) DILATION を 2 回施した結果。

図 2 DILATION による影響範囲拡大の様子

的評価関数を UCT 探索に組み込み結果の改善を図る。今回用いる静的評価関数は $\alpha\beta$ 探索によるプレイヤで用いた Bouzy's 4/0 algorithm のよる静的評価関数と同じものを使用する。

UCT 探索に静的評価関数を加えたものは、次のような手法である。

$$Q_{UCT} = \sigma(E(s_t)) \tag{2}$$

まず UCT 探索のシミュレーション中に未探索の盤面 s_t に遭遇した際、Bouzy's 4/0 algorithm を用いた静的評価関数 E でその盤面の評価を行う。その評価値をシグモイド関数 σ に通して UCT のシミュレーションで得られる勝率に対応するように変換する。これは静的評価関数で得られる評価値と勝率の相関性があるものと仮定している。そして得た値を一定回数のシミュレーションを行ったときに得られるであろう勝率と仮定して盤面 s_t の初期値 Q_{UCT} に設定し、UCT 探索の効率を上げることを試みる。

3.3 UCT 探索に 3 層ニューラルネットワークの静的評価関数を加えたプレイヤ

3.2 では Bouzy's 4/0 algorithm を用いた静的評価関数を使用したが、この静的評価関数として 3 層ニューラルネットワークを用いたプレイヤを実装した。今回対象とする Blokus Duo はまだ知識表現があまり得られていないゲームであるため、適切な評価値を出力する静的評価関数をヒューリスティックな方法で作成しても良い結果がでることは期待できない。そこで、静的評価関数を 3 層ニューラルネットワークで表現し、盤面と勝率の関係を学習させることとする。3 層ニューラルネットワークは非線形関数であるため、Blokus Duo のような適切な特徴が取りにくいゲームについても効果的な静的評価関数が得られ易いと考えられる。

まず 3 層ニューラルネットワークによる静的評価関数で入力として用いる盤面の特徴は 1×1・2×2・3×3 のそれぞれのサイズのパターンを用いる。盤面中に表れるこれら全てのパターンの数についてをそれぞれカウントし、それらの数値を特徴ベクトルとして 3 層ニューラルネットワークの入力に用いる。ただし、このときのパターンは回転対象と鏡像対称のものは全て同一のものと見なす。

この 3 層ニューラルネットワークの学習に用いる学習データは、UCT 探索を用いたプレイヤ同士を対戦させて、その試合中に表れた全盤面の特徴とシミュレーションで得た盤面の勝率を対にして保存することで、盤面の特徴を入力、勝率を出力とする学習データを生成する。このとき学習データは手数ごとに

分けて保存する。また、試合中に表れた全ての盤面のうち同一盤面は省いて保存し、そのときの勝率には全ての同一盤面の勝率の平均値を用いる。このように生成した学習データを用いてそれぞれの手数ごとに3層ニューラルネットワークの学習を行う。プレイヤは手数に応じて学習させた3層ニューラルネットワークを切り替えて使用し、3.2と同様の手法でUCT探索の改善を図る。

4. 評　　価

$\alpha\beta$探索とBouzy's 4/0 algorithmの静的評価関数を用いたプレイヤ、通常のUCT探索を用いたプレイヤ、UCT探索にBouzy's 4/0 algorithmの静的評価関数を用いたプレイヤ、UCT探索に3層ニューラルネットワークによる静的評価関数を用いたプレイヤのそれぞれの強さの比較を行った。

UCT探索に3層ニューラルネットワークの静的評価関数を加えたプレイヤにて使用した学習データは、シミュレーション回数を18,000回とする設定のもとでUCT探索を用いたプレイヤ同士を対戦させて得たものである。また学習データは1,800試合の対戦を通して生成され、試合中に表れた全ての盤面のうち同一盤面を省いた53,000通りの盤面が保存された。

また3層ニューラルネットワークの学習結果を評価するため、学習データの生成過程と同様に、UCT探索を用いたプレイヤ同士を200試合対戦させ、その試合中に表れた全ての盤面のうち同一盤面を省いた6000通りの盤面の特徴とその勝率をテストデータとして保存した。

尚、実験はIntel Pentium4 3.00GHz・メモリ2GBのマシン上で行った。実装にはC++言語を用いた。

では、それぞれの手法で実装したプレイヤの比較を行っていく。まずBouzy's 4/0 algorithmによる簡単な静的評価関数を使用した$\alpha\beta$探索を用いたプレイヤと通常のUCT探索を用いたプレイヤを対戦させた。両プレイヤとも1手につき100秒として対戦させた。通常のUCT探索を用いたプレイヤは100秒につき平均約18,000回のシミュレーションを行うことを確認した。100試合対戦させた結果を表1に示す。

先手	後手	結果
UCT	$\alpha\beta$	89勝 11敗 0引分け
$\alpha\beta$	UCT	40勝 55敗 5引分け

表1 $\alpha\beta$探索とBouzy's 4/0 algorithmによる静的評価関数を用いたプレイヤとUCT探索を用いたプレイヤの対戦結果

次に、通常のUCT探索を用いたプレイヤとUCT探索にBouzy's 4/0 algorithmによる静的評価関数を加えたプレイヤを対戦させた。両プレイヤとも1手につき10,000回のシミュレーションを行わせた。100試合対戦させた結果を表2に示す。

先手	後手	結果
UCT	UCT + Bouzy	13勝 68敗 19引分け
UCT + Bouzy	UCT	59勝 28敗 9引分け

表2 UCT探索を用いたプレイヤとUCT探索にBouzy's 4/0 algorithmによる静的評価関数を加えたプレイヤの対戦結果

そして更に、UCT探索にBouzy's 4/0 algorithmによる静的評価関数を加えたプレイヤとUCT探索に3層ニューラルネットを静的評価関数として加えたプレイヤを対戦させた。両プレイヤとも1手につき1,000回のシミュレーションを行わせた。100試合対戦させた結果を表3に示す。

先手	後手	結果
UCT	UCT + NN	90勝 7敗 3引分け
UCT + NN	UCT	11勝 63敗 26引分け

表3 UCT探索を用いたプレイヤとUCT探索に3層ニューラルネットワークを静的評価関数として加えたプレイヤの対戦結果

これらの結果より$\alpha\beta$探索よりもUCT探索を用いるプレイヤの方が強くなることがわかった。そしてUCT探索にBouzy's 4/0 algorithmによる静的評価関数を加えてUCT

探索のシミュレーション時において盤面評価の初期値を予め与えることで更に良い結果が得られることがわかった。しかし、UCT 探索に 3 層ニューラルネットワークを静的評価関数として加えたプレイヤは UCT 探索を用いるプレイヤよりも弱くなってしまうことがわかった。

これらより Blokus Duo のコンピュータゲームプレイヤにおいて UCT 探索を用いることは囲碁と同様に有効であることが確認できた。また UCT 探索に静的評価関数を加えることもプレイヤを強くするための有効な手法であることが確認できた。UCT 探索に加える静的評価関数は更に改善することによってもっと良い結果が得られると考えられるが、3 層ニューラルネットワークによる静的評価関数を用いたプレイヤは UCT 探索を用いるプレイヤより劣る結果となってしまった。

そこで今回使用した 3 層ニューラルネットワークで表現された静的評価関数を評価するために、テストデータを用いて平均二乗誤差を計算した（図 3)。

これより手数が増えるにしたがって急激に静的評価関数の精度が落ちていることが確認できる。これはゲームが進行するにつれて取り得る盤面の数が増えていくため学習によって収束がしにくい状況になるためと考えられる。この問題を解決するには手数がゲームの終盤に近づくほど学習データを増やす必要がある。今回は UCT 探索を用いたプレイヤ同士を対戦させ試合中に表れた盤面を学習データとして保存したが、その方法では手数に応じて学習データの数を柔軟に増やすことが困難であるため、ある手数の盤面をランダムで生成することで学習データを出力するような方法を取るなどの工夫が必要であると考えられる。

更に先手として UCT 探索によるプレイヤ、

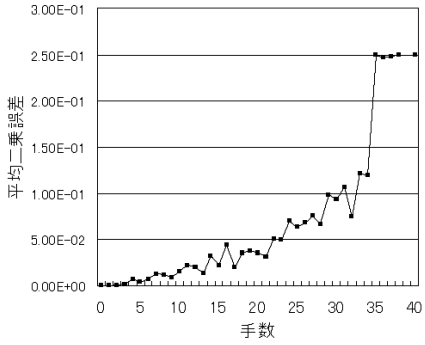


図 3 学習させた 3 層ニューラルネットワークのテストデータを用いた平均二乗誤差の測定

後手として UCT 探索に 3 層ニューラルネットワークを加えたプレイヤを対戦させたときのそれぞれの陣地の数の変化についても調べた（図 4)。これは 100 回対戦させたときの各手数における陣地の数を平均したものである。Blokus Duo において陣地の数は盤面の有利度を表す簡単な指標になると考えられる。これよりゲームの序盤から徐々に陣地の数の差が広がっていることが確認でき、序盤の段階から既に UCT 探索に 3 層ニューラルネットワークを加えたプレイヤは悪い手を選択してしまっていると判断できる。

序盤で用いる 3 層ニューラルネットワークについては、テストデータを用いた評価では適切に学習されたと判断できたにも関わらずこのような序盤から悪手を売ってしまう結果となってしまった原因としては、用いた盤面の特徴が適切ではなかったことが考えられる。Blokus Duo は囲碁とは異なり決まった形をしたブロックを置いていくゲームであるため、盤面によって置けるブロックが限られてしまう。そのためプレイヤが持っているブロックに関する情報がなければ盤面の評価は難しいと考えられる。今回は盤面の 1×1・2×2・3×3 のサイズのパターンのみしか特徴として用いなかったため、3 層ニューラルネットワークで適切な静的評価関数を表現できなかったと考えられる。

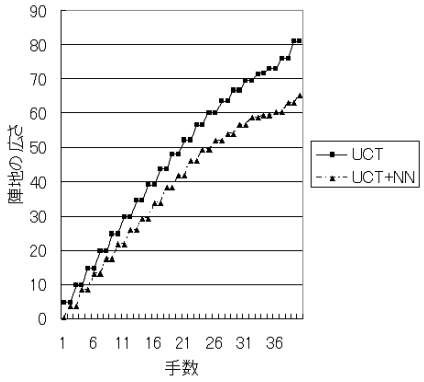


図 4 UCT 探索を用いたプレイヤと 3 層ニューラルネットワークを静的評価関数として加えた UCT 探索を用いたプレイヤを対戦させた各陣地数の変化

5. おわりに

Blokus Duo におけるコンピュータゲームプレイヤでは静的評価関数として簡単な Bouzy's 4/0 algorithm を用いたαβ探索よりも通常の UCT 探索の方が強くなることがわかった。また、通常の UCT 探索と Bouzy's 4/0 algorithm を静的評価関数として加えた UCT 探索では後者の方が強くなることがわかった。これより UCT 探索に静的評価関数を加えることは有効な改善手法であることがわかった。UCT 探索に加える静的評価関数として 3 層ニューラルネットワークを用いたプレイヤは通常の UCT 探索によるプレイヤよりも弱くなってしまったが、これは 3 層ニューラルネットワークの学習を工夫することと盤面の特徴を適切に取ることで改善の余地があると考えられるため今後の研究で改善を行っていく。

参 考 文 献

1) Levente Kocsis and Csaba Szepesvári. *Bandit based Monte-Carlo Planning* In 15the European Conference on Machine Learning(ECML), pp.282-293, 2006.
2) Sylvain Gelly and Yizao Wang. *Exploration exploitation in Go: UCT for Monte-Carlo Go* NIPS-2006, Online trading between exploration and exploitation, Whistler Canada, 8 December, 2006.
3) Rémi Coulom. *Computing Elo Ratings of Move Patterns in the Game of Go* Accepted at the Computer Games Workshop 2007, Amsterdam, The Netherlands, 2007.
4) Bruno Bouzy. *Mathematical morphology applied to computer Go* International Journal of Pattern Recognition and Artificial Intelligence, 17(2), 2003.
5) Sylvain Gelly and David Silver. *Combining Online and Offline Knowledge in UCT* International Conference of Machine Learning, 2007.
6) B. Bruegmann. *Monte carlo go* 1993.
7) Rémi Coulom. *Efficient selectivity and backup operators in monte-carlo tree search* In P. Ciancarini and H. J. van den Herik, editors, Proceedings of the 5th International Conference on Computers and Games, Turin, Italy, 2006.
8) FSF. *GnuGo Reference Manual* http://www.gnu.org/software/gnugo