モンテカルロ法におけるシグモイド関数による勝敗とスコアの重ね合わせ

柴原一友[†]　小谷善行[†]

コンピュータ囲碁で近年目覚しい成果を挙げている手法としてモンテカルロ法がある．モンテカルロ法ではランダムシミュレーションの結果の勝敗やスコアの平均値の高い局面を有利な局面と判断して，手の決定を行う．ここで，勝敗とスコアでは勝敗のほうが情報の量が少ないが，囲碁では勝敗を用いたモンテカルロ法のほうが有利であることが示されている．本稿では，スコアと勝敗を用いたモンテカルロ法の性質を比較的新しいゲームである BlokusDuo 上で調査し，勝敗に比べスコアが有効であることを示す．その上で，UCT においてスコアは不向きであることを示す．スコアの持つ有効性を効率的に導入するために，シグモイド関数を用いて勝敗とスコアを重ね合わせた評価を返す関数を用いて，勝敗，スコアに代わる勝率への近似の新しい方法を提案し，その有効性を示す．また、この方法により勝敗を用いる場合に存在する最終スコアの偏りを改善できることを示す．

Combining Final Score with Winning Percentage using Sigmoid Function in Monte-Carlo Algorithm

Kazutomo Shibahara[†] and Yoshiyuki Kotani[†]

Monte-Carlo method recently has produced good results in Go. Monte-Carlo Go uses a move which has the highest mean value of either winning percentage or final score. In a past research, winning percentage is superior to final score in Monte-Carlo Go. We investigated them in BlokusDuo, which is a relatively new game, and showed that Monte-Carlo using final score is superior to the one that uses winning percentage in cases where many random simulations are used. Besides, we showed that using final score is unfavorable for UCT, which is the most famous algorithm in Monte-Carlo Go. To introduce the effectivity of final score to UCT, we suggested a way to combine winning percentage and final score by using sigmoid function. We show the effectivity of the suggest method and show that the method improves a bias where Monte-Carlo Go plays very safe moves when it has advantage.

1. はじめに

近年，コンピュータ囲碁の分野では，モンテカルロ碁が多大な成果を挙げている．局面の状況判断が難しく，有効な評価関数を作成することが困難とされた囲碁に対し，ランダムシミュレーション（PlayOut）の結果として得られる勝敗やスコアを基にして局面評価を行うこの手法は，特別な知識を必要とせずに高い局面評価を可能にした．

モンテカルロ法ではランダムシミュレーションの結果として勝敗を用いる方がスコアよりも有効であるとされている．一回あたりに得られる情報量は明らかにスコアを用いる方が多いが，スコアは外れ値による影響が大きく，勝敗よりも劣ることが示されている．しかし，勝敗を使用するシステムではゲームの勝利だけにこだわってしまうという性質があり，ゲームの展開としては面白みのないものになりやすい．

本稿では，BlokusDuo を題材としてスコアの持つ性質を調査し，その有効性を示す．その上で，モンテカルロ碁で一般的に使用されているアルゴリズムである UCT 上では，スコアは有効に働かないことを示す．そこで，勝敗とスコアをシグモイド関数によって重ね合わせることにより，スコアの有利性を反映し，勝敗を用いるために生じる性質を避け，より効果的な局面評価を実現する．シグモイド関数を適正に調整したアルゴリズムは単純に勝敗を用いるシステムに比べ，有効であることを示す．また，勝敗を用いた場合に生じる，勝利にこだわる手の指し方を改善できることを示す．また，アンケートを実施し，人間からみて面白みのある指し手を実現している傾向がみられたことを示す．

1.1 モンテカルロ法

モンテカルロ法とは，乱数を用いてシミュレーションを何度も行うことで，問題の近似解を得る手法の総称である．

古くから存在するこの手法を，Brüegmann は囲碁に適用した[1]．その方法は，ある局面の形勢を判断する

† 東京農工大学大学院 工学府
Department of Computer and Information Sciences, Tokyo University of Agriculture and Technology

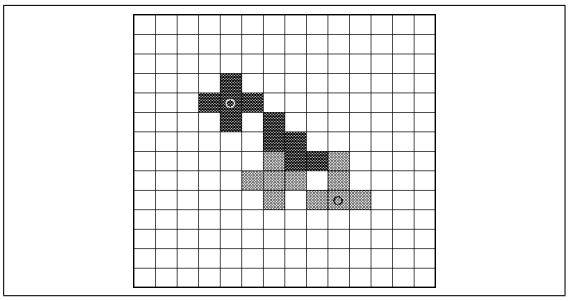

図 1 BlokusDuo

ために，その局面からランダムに手を指し続ける（ランダム対局）ことで得られる勝敗を集計し，その平均成果が高い局面を優勢な局面と判断するものである．しかし，大量のランダムシミュレーションを用いなければ精度の高い評価を得られないため，指し手の決定に時間的な制約を持つゲームでの使用は困難であった．

しかし，近年のマシンスペックの向上もあり，モンテカルロ囲碁は Bouzy によって一定の成果を挙げた[2]．モンテカルロ法は局面以下の平均的な価値を得るものであるが，通常ゲーム木では相手は最善手だけに依存するため，平均では正確な値を得ることができない．そこで，ゲーム木の MinMax の概念を取り入れる方法もいくつか提案された．ゲーム木の末端でモンテカルロ法を呼び出すものや[3]，確率的な枠組みで行うもの[4] などがあるが，その中でもっとも有名なアルゴリズムが UCT(UCB for Tree) である[5]．これは類似した問題である多腕バンディット問題に対して有効な UCB(Upper Confidence Bound)[6] を拡張し，ゲーム木探索に応用したものである．UCT は 2007 年に開催された第 12 回 Computer Olympiad で金，銀メダルを獲得した MoGo[7] や Crazy Stone[8] にも実装されている．

モンテカルロ法は，近年では囲碁に限らず，将棋[9] や BlokusDuo[10]，Phantom Go[11] などさまざまなゲームに実装されている．

1.2 BlokusDuo

本研究では BlokusDuo を題材としている．BlokusDuo は近年コンピュータ大会も開かれたゲームである．今回実験に使用したプログラムは，この大会で 16 チーム中 7 位になったプログラムをベースとしている． BlokusDuo は二人完全情報ゲームであり，4 人対戦ゲームである Blokus を 2 人ゲームに変更したものである．BlokusDuo で使用する盤面を図 1 に示す．

盤面は 14 × 14 で構成され，正方形が 1〜5 個つながった形のピース 21 個を，それぞれが所持する．先手は図における左上の○に重なるように自分のピースを配置し，後手は右下の○に重なるように配置する．その後も互いにピースを置きあい，最終的により多くの盤面を占有した側の勝ちとなる．ただし，自分のピース同士が頂点（角）で接するように置かなければならず，辺で接して置くことはできない．ゲーム木の大きさは大体 10^{70}〜10^{80} と試算している☆．序盤の可能手数がとても多く，1000 を超える場合もある．一方で終盤は可能手が少ないゲームである．序盤の可能性が多いため，序中盤における探索による成果が得られにくいゲームである．BlokusDuo は囲碁と同様に，盤上の領域を囲いこみ，自分だけがピースを置ける領域を多く作ることが重要である．ゲームの性質が囲碁と似ており，かつ問題のサイズが小さいため，モンテカルロ法の検証対象として好適である．先のコンピュータ大会で 4 位に入ったプログラムはモンテカルロ法を使用しており，中級者程度の実力があると思われる．

1.3 UCT

UCT は，多腕バンディット問題で成果を上げた UCB をミニマックス木の探索用に拡張した手法である．UCB は明らかに悪い手に対するランダムシミュレーションの実行を避け，良い手に多くのランダムシミュレーションを割り振ることで，試行回数の集中によるランダムシミュレーションの精度向上と，最終結果に対する最善応手手順の影響を強め，ミニマックス戦略の形を実現している．

UCB は，その手を選んだときの勝率と，その手への試行回数から，その手が持ちうる最大の価値を推定し，その値がもっとも大きい手にランダムシミュレーションを実行する．UCB の計算式はいくつか提案されているが，ここでは次の式を用いる．

$$UCB(i) = \bar{X}_i + \sqrt{\frac{2\log N}{n_i}}$$

$\bar{X}_i$ は，現在得られている手 i の勝率，n_i は現在手 i に対して実行したランダムシミュレーションの回数，N は現在実行したランダムシミュレーションの総数である．

UCT の最終的な手の選択方法としてはいくつか存在するが，ここでは平均値が最大となるものとした．UCT には様々な拡張手法があり，木の展開を有効なノードに限定することで，効率的な評価を実現する方法などがある．BlokusDuo は解析が進んでいる分野ではなく，ヒューリスティックな技法は導入しにくいこともあり，今回はプレーンな UCT を使用し，緊急度や分岐限定，枝刈り等の手法は使用していない．

2. ゲーム木における勝敗とスコアの性質

2.1 勝敗とスコア

モンテカルロ法では，まったく異なる値を使用する方法もあるが[14]，基本的にランダムシミュレーションの戻り値として得る結果として，勝敗とスコアがある．スコアは囲碁ならば占有した領域数差のように，最終局面で一意に得られる値とする．勝敗とは，手番側の勝利ならば 1，負けならば-1，引き分けならば 0 とし

☆ モンテカルロ法による試算．ランダムでは 10^{40}〜10^{50}

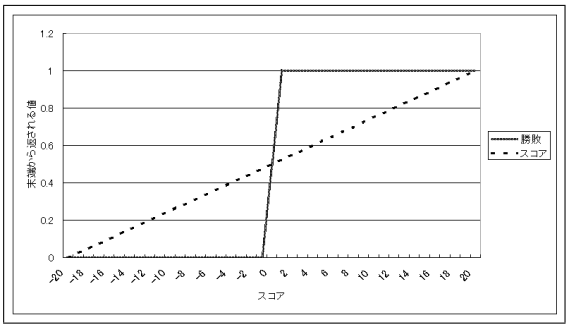

図 2 勝敗とスコア

たものであり，スコアがプラスならば 1，マイナスならば-1，0 ならば 0 と変更することで得ることができる．勝敗とスコアの関係を図 2 に示す．ここでは，スコアと勝敗は 0 から 1 までに正規化されている．

勝敗とスコア，どちらを用いるべきかという点に関しては，モンテカルロの試行回数に依存すると考えられる．スコアの方が持っている情報量は多いが，分散もまた大きく，試行回数が少ない場合，一部の局面の結果を強く引きずってしまう．囲碁においては勝敗を用いたほうが遥かに高い性能を持つことが示されており[15]，近年のモンテカルロ碁では勝敗を用いるのが主流である．また，勝敗とスコアの重み付け和を用いることでわずかながら性能向上があることも示されており[15]，スコアの持つ情報は有益であることも示されている．しかし，勝敗とスコアの性質に対し，深く調査した例は筆者の知る限りでは存在しない．

Crazy Stone もまた勝敗を用いているシステムであるが，その指し手にはある特徴があることが示されている[8]．それは，自分が優位な局面では無理をしない安全な手を選び，不利な局面では好戦的で危険な手を選ぶというものである．そのため，勝利する場合は僅差であることが多く，負ける場合は大敗を喫する場合が多い．これは，勝敗を用いて計算しているために，勝つか負けるかの判断部分に偏重するためであるとされている．この性質は既存のアルゴリズムでは得にくい性質であり，モンテカルロ碁の強さの主要な理由であるとされている．

2.2 勝敗とスコアの性質比較実験

実際のゲームにおいて，勝敗とスコアの性質を比較する実験を行った．対象は BlokusDuo の 16 手以降に限定した．こうすることで，モンテカルロの試行回数を多く実行できるようにし，また終盤に限定した適用との比較をできるようにした．比較するのは等回数のモンテカルロ (Normal)，UCB，UCT であり，ランダムシミュレーションの試行回数は 10^3〜10^7 とした．ただし，UCT はメモリの都合上 10^6 までとしている．実験は，スコア対勝敗とし，試行回数に応じて勝率と最終スコアの平均がどのように変化するかを調べた．対戦数はそれぞれ 1000 回としている．BlokusDuo におけるスコアは，マスの占有数の差であり，全ピースを盤上に配置したときの占有数である 89 を最大として正規化している．実際には相手は指せる手があるときはパスできないため，89 の差がつくことはないが，ルール上可能な最大占有数が不明なため，89 を用いている．比較した勝率の結果を図 3 に，平均最終スコアの結果を図 4 に示す．図 3 は，勝率が 0.5 を超えている場合，その設定ではスコアを用いたほうが勝敗を用いるより有効であることを意味している．

まず，Normal と UCB について比較してみる．Normal では試行回数が 1000 回の場合，スコアの勝率が勝敗を大きく下回っている．これは，回数が少なく統計的に安定しない状況であるため，スコアが安定した結果を得られないことが原因と考えられる．一方で UCB は重要な情報へと回数を集めることができるために，スコアによる方法が安定している一方で，いまだ勝敗が有効となるほど安定していないためと考えられる．回数が増えていくと，Normal はスコアが統計的に優位になる．UCB は 1 万回の時点では勝敗が安定した結果を得ることにより，スコアの優位性が小さくなるが，その後スコアの優位性が少しずつ高くなる．勝率の変化では小さい変化であるが，平均最終スコアはこの変化がはっきりとみてとれる．

UCB と UCT の比較をしてみると，回数が少ない時には両者に大きな違いはない．これは，UCT は初期段階では UCB と同等の動作をしているためと考えられる．しかし，回数が増えるに従い，UCT はスコアの性能が徐々に悪化する．これは UCT が木構造で情報を処理するに従い，末端では情報の数が少なくなっていくため，常に安定しない状況が生まれていると考えられる．この結果，UCT は回数が増加してもスコアの恩恵を受けにくくなると考えられる．しかし，別実験の結果，試行回数 10 万回におけるスコアを用いた UCB と勝敗を用いた UCT とでは UCT がわずかに優位な強さを持っていた．UCT は有効な方法であるが，スコアの優位性を効率的に利用できていないことがわかる．よって，スコアの優位性を効果的に導入することが重要であると考えられる．

3. 勝敗とスコアの重ね合わせ

我々は UCT に対するスコアの優位性を効率的に導入するために，モンテカルロ法で末端から返される値として，勝敗とスコアを重ね合わせた値を使用する方法を提案する．すでに単純な重み付け和による方法は一定の成果を示しているが，我々はシグモイド関数を用いることで，より統一的で汎用的な枠組みを提供する．また，スコアを用いることで，勝敗によるモンテカルロ法に存在した，勝ち負けにとらわれる性質が改善されることを示す．

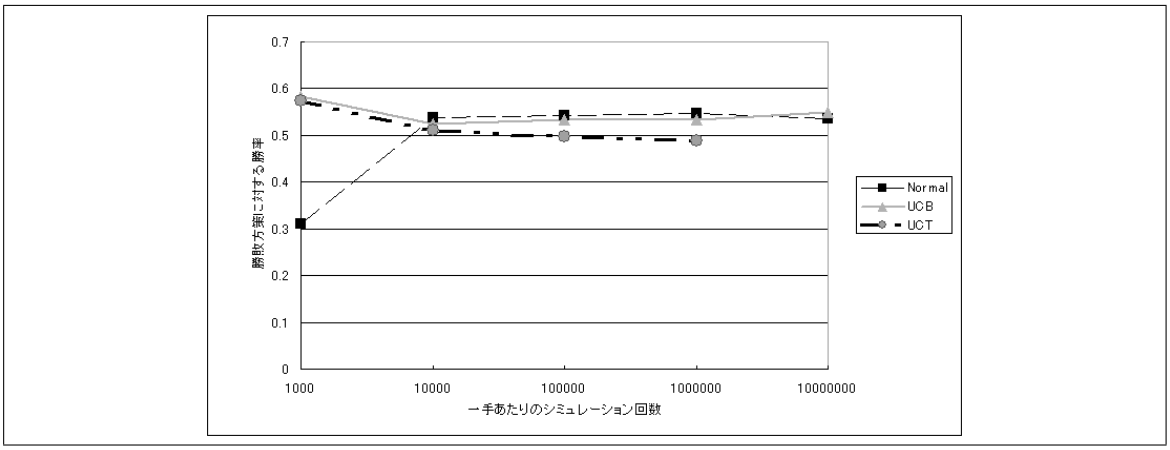

図 3 勝敗方策に対する試行回数ごとの勝率の変化

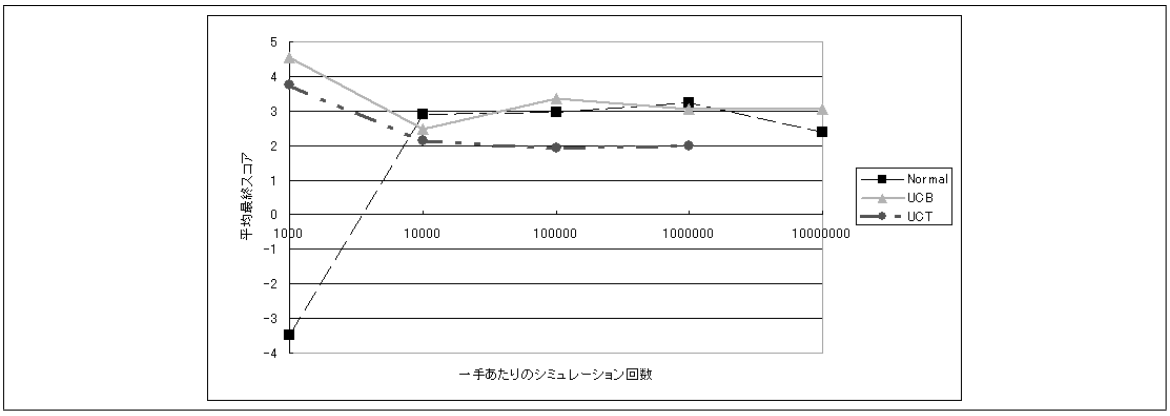

図 4 勝敗方策に対する試行回数ごとの平均最終スコアの変化

3.1 意義と効果

スコアの持つ情報を生かしながら，勝敗に近い値を得る．この問題に対し，我々はシグモイド関数によって両者の重ねあわせを行う方法を提案する．シグモイド関数は，二値の変数を確率へと回帰分析するロジスティック回帰分析において使用されているモデルでもある．スコアを引数としたシグモイド関数は勝率の性質と似通った性質を持つため，近似する関数として好適である．勝率と似ている性質を利用して，評価関数の特徴の二値変数にシグモイド関数を適用して学習する方法も提案されている[16]．シグモイド関数のグラフを図 5 に示す．

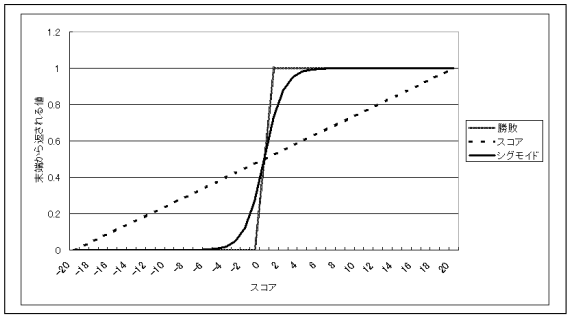

図 5 シグモイド関数

シグモイド関数は以下の式で定義される．

$$f(x) = \frac{1}{1 + \exp^{-kx}}$$

ここで x は終了局面でのスコア，k は定数であり，図 5 では k=1.0 である．k の値が大きくなるほど，スコア 0 付近における傾きは急になり，勝敗に近づく．スコアを代入したときのシグモイド関数の値をランダムシミュレーションの戻り値に使用することで，スコアを勝敗へと近似する．

提案手法の利点として，スコアと勝敗の理論的な重ね合わせを提供できる点がある．スコアの信頼性は，試行されるランダムシミュレーションの回数に依存すると思われる．多くのランダムシミュレーションを試行できるゲームでは，k をより小さく設定することで容易に応用可能となる．試行できるランダムシミュレーションの回数が局面に応じて変動しやすいゲームでは，

k を局面の状況やランダムシミュレーションの試行回数に応じて変更するだけで，より勝敗やスコアを自然な形で勝率に効率的に近似させることができると思われる．k の値は，各局面からの対局における，平均スコアと平均勝率との組を用いることで，最適な値を推定できると思われる．しかし，局面の進行度等に依存することが予想されるため，より正確な値を設定するためには，数多くのデータや特徴量が必要と思われる．本研究では自動的な設定は試みないで，各条件下においていくつかの k を試行することで調整を試みている．

また，モンテカルロ法の特徴であった，僅差で勝ち，大差で負ける性質を改善できると考えられる．勝敗だけで行う場合，勝利数の大きくなる局面を選ぶため，優勢な局面と不利な局面との落差が激しい．また，優勢な局面の間の差異が小さいため，得られた勝率に誤差が含まれやすいことが，指し手が進むにつれて勝率の緩やかな低下を招くと思われる．人間もまた，有利な局面では手を緩めて指すことがある．確実に勝利するためには重要なことであるが，有利であるという目測を誤っている場合，緩手は負けにつながりやすくなる．人間は高い評価能力によって，見落としの少ない判断を実行できるであろうが，コンピュータはモンテカルロの持っているランダム試行に起因する不確実性のため，見落としが起こりやすい．勝利局面間の相違性の薄さと，モンテカルロ法が本質的に持っている不確実性に起因する不安定さが対局相手によって徐々に咎められ，結果的に形勢を悪くさせる．これが，モンテカルロ法が僅差で勝ちやすい原因であり，同時に有利性の判断を誤った局面での勝率を低下させている原因であると考えられる．モンテカルロ法は有利な展開に持っていけるように手を指しながら，有利性を積み重ねていく手法であることから，微細な変化を正確に反映して手を決定することが重要となる．同じ勝利する上でも，よりスコアが高いほど勝ちやすい局面であることが多いため，回数の少なさに起因する誤差に振り回されない範囲で使用することで，より勝利に近づきやすい局面を選択できるようになると考えられる．

また，勝敗を使用したモンテカルロは優勢ならば守りに入り，味気ない戦いをしてしまい，劣勢ならば無謀すぎる手を好み，粘り強い反撃ができないことが考えられる．勝敗の偏りの改善は，単純な勝率の変化だけでなく，人間と戦う場合に，勝敗にこだわり過ぎない戦い方ができることから，対戦して楽しめるシステムにすることができると期待される．

3.2 提案手法の勝敗，スコアとの比較実験方法

実験では，BlokusDuo に対し，UCT を実装して勝敗とスコア，提案手法とを比較する実験を行った．実験では，k の値を変化させたもの，正規化したスコアによるものを，勝敗で算出するアルゴリズムと対戦させることで行った．対戦数は 1000 回である．一手決定するごとに試行するランダムシミュレーション数は 10 万回とした．

3.3 提案手法の勝敗，スコアとの比較実験結果

モンテカルロの結果が安定しやすい，終盤に限定した効果をみるため，手数を 16 手以降に限定した場合の実験結果を図 6 に示す．勝敗との対戦であるため，勝敗の欄は 0.5 となる．この結果，手数を限定した場合は k=0.4 の場合に勝率が約 55%となり，統計的に有意な結果を得た．また，勝利時，敗北時の平均スコアの結果を図 7 に示す．この結果，単純に勝敗を用いる場合に比べ，スコアを重ね合わせた方法は勝利スコアの平均が高く，敗北スコアの平均は若干低下している．勝率が変化しているため，母数が k によって変化していることも考慮すると，シグモイド関数を用いてスコアを混ぜ合わせることによって，既存のモンテカルロ法に見られた勝敗の偏りが改善されたことが分かる．手数を限定しない場合の実験結果を図 8 に示す．k=0.2 のときに勝率が 54%となり，こちらも統計的に有意な結果を得た．しかし，手数を 16 手目以降に限定した場合と比べると，グラフが歪んでいる．これは，序中盤の局面の難しさが影響していると考えられる．

3.4 局面進行ごとに k の値を変えた場合の実験結果

実験の結果，終盤に対して適用する場合と，序中盤に対して適用するのでは性質が異なることが分かった．シグモイド関数で使用する k の最適な値は，局面の状況に依存すると思われる．最適な k の値を使用することで，さらなる性能の向上が期待できる．

そこで，局面の進行に応じて k の値を変更して適用する実験を行った．BlokusDuo は手数と局面の進行の関係が深いため，手数で適用方法を区切ることとした．様々な組み合わせを行うことが最善だが，時間が多くかかってしまう．そこで，ある程度設定を決め打って実験することとした．用いた設定は以下のとおりである．

- 1〜7 手目:勝敗
- 8〜16 手目:k=0.5〜1.0
- 17 手目〜:k=0.4

すでにある程度性質が判明している終盤は k=0.4 に固定することにした．ここで，k=0.2 としていないのは，予備実験の結果，k=0.4 の方が優れていると判断したためである．また，予備実験の結果，序盤はスコアが有効に働きにくいことから，序盤は勝敗を使用することとした．中盤で使用するシグモイドの k の値を変更しながら，勝敗を用いた UCT と各 1000 対局対戦した結果を図 9 に示す．一手決定するごとに試行するランダムシミュレーション数は 10 万回とした．

実験の結果，最大で約 55%の勝率を得た．k の値が 0.4 から離れるほど勝率が上昇しており，局面の性質に分けた適用が効果的であることが分かる．

3.5 勝敗モデルとスコアモデルの比較アンケート

提案手法の利点の一つとして，人間にとって親しみ易いゲームの実現が挙げられる．その検証を行う

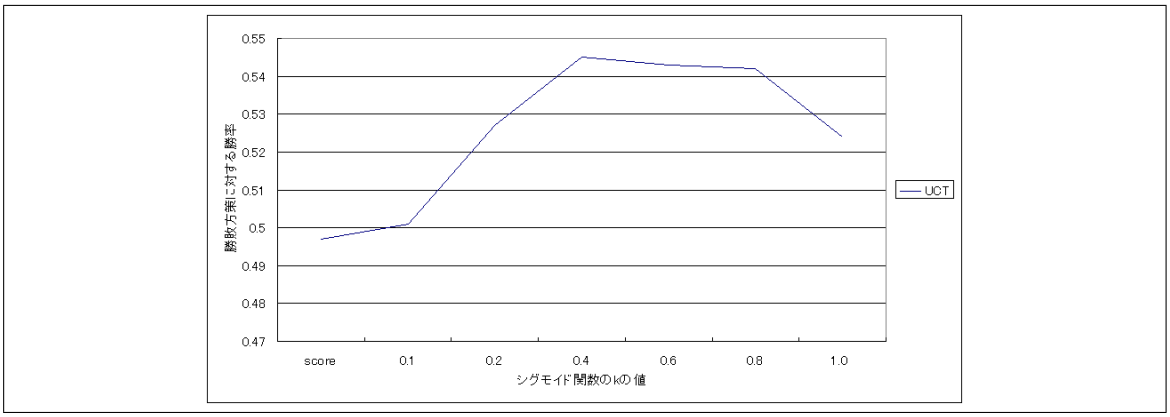

図 6 提案手法の勝敗，スコアとの比較実験結果（16 手目以降限定）

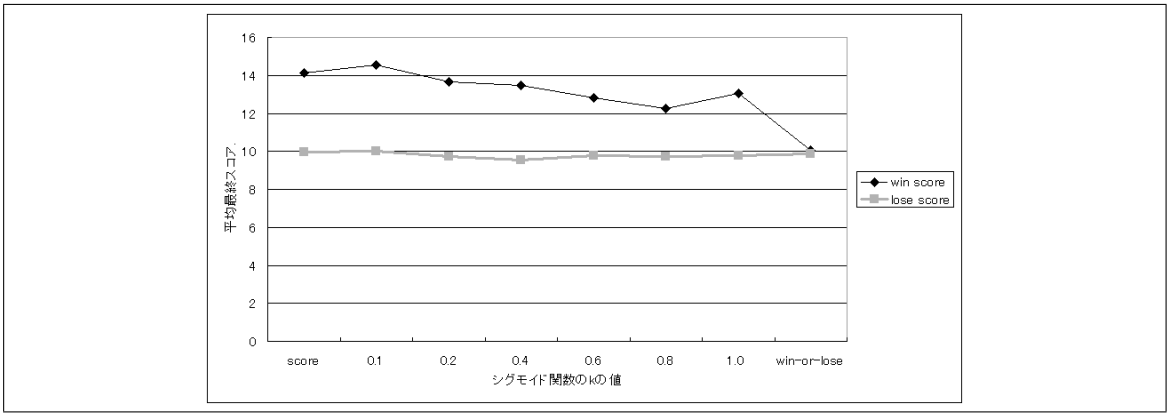

図 7 提案手法の勝敗，スコアとの平均スコア比較結果（16 手目以降限定）

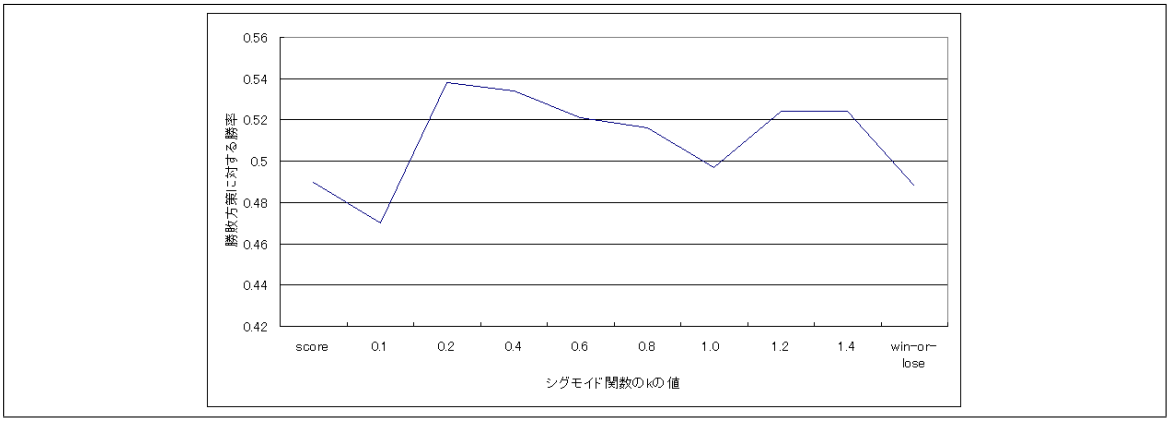

図 8 提案手法の勝敗，スコアとの比較実験結果

ため，人間に対するアンケートを実施した．被験者は BlokusDuo の経験者 5 人である．初級者が多いが，被験者のうちの一人は大会における最強の BlokusDuo プログラムに勝つことができる．

実験方法を次に示す．被験者には勝敗モデルとシグモイドモデルが戦った 12 棋譜を提示する．ここで使用したシグモイドモデルは，勝敗モデルと同じ強さのものであり，全ての手を k=1.0 で計算している．また，提示した棋譜は様々な面で（勝敗数，トータルスコア数等）偏りのないように選び出している．被験者はその棋譜を見て，以下の三点について，どちらのプログラムが優れていたかを選んでもらった．

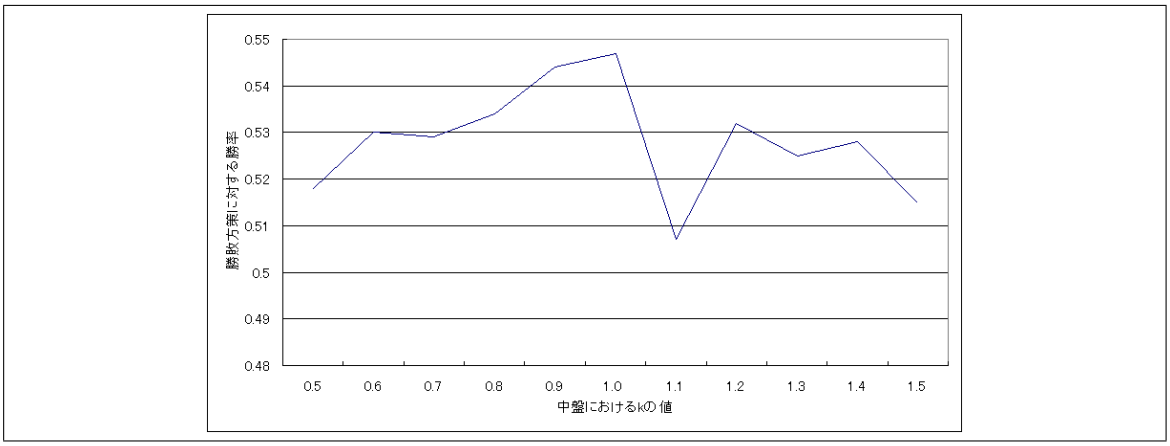

図 9 局面の進行度によって変更した場合の提案手法の勝敗方策に対する勝率

- どちらのほうが強いと感じたか
- どちらのほうが人間ぽいと感じたか
- どちらのほうが面白そうか

当然ながら，プログラムの処理に関する知識は与えられていない．また，棋譜を提示する順番についても配慮している．

調査の結果，次のような結果が得られた．

- どちらのほうが強いと感じたか (シグモイド: 1 - 4 :勝敗)
- どちらのほうが人間ぽいと感じたか (シグモイド: 3 - 2 :勝敗)
- どちらのほうが面白そうか (シグモイド: 3 - 2 : 勝敗)

一つ目の設問については，勝敗モデルの方が上回っているが，他の設問についてはシグモイドの方が上回った．この結果から，シグモイドモデルは，人間らしく，面白いシステムの作成に貢献していることが伺える．

シグモイドが不評だった点は，序盤における明らかな悪手であった．序盤は形勢の判断が難しいことが原因として考えられる．このことから序盤は勝敗モデルを使用するほうが良いだろう．この悪手がシグモイドの評価を大きく下げているようであり，強いと感じられない原因と思われる．また，その他の設問においても序盤の悪手が原因で勝敗の方が優れていると解答されている例があった．

一方でシグモイドモデルは劣勢での粘り強さや，着手の面白さ，意外性，攻め合いのうまさなどを評価する声が多かった．勝敗モデルにはこういった言及は見られず，シグモイドモデルが対戦の面白さを引き出していることが伺える．ちなみに，最強のプログラムに勝てる被験者は全ての設問でシグモイドが良いと回答している．

3.6 囲碁での適用結果

提案手法を囲碁に適用した．使用したシステムは単純な UCT で手を決定するシステムである．コミを 7 目半にして 9 × 9 囲碁で 400 対局した．k の値は 1.0 とし，一手あたりの試行回数は数万回程度としている．その結果，勝率は 59%となり，有意な結果を得ている．

4. 考　　察

実験の結果，シグモイド関数を使用することで，勝敗の場合よりも勝率を向上させることができ，有意な差があることを確認した．最大の勝率は約 55%であった．一方で，実行時間はシグモイド関数の計算だけであり，わずかな増加で抑えられている．よって，この手法は勝敗やスコアによる方法よりも効果的な方法であることがわかった．また，ランダムシミュレーション数が多くなるほど，スコアを重視する方が有効な結果となることを確認した．また，囲碁に対しても適用を行い，有意な勝率を得ている．

多くのゲームでは勝負どころや，重要な局面ではより多く探索を実行する．そのような場合，実行するランダムシミュレーション数も変更されるため，費やす時間に応じたシグモイド関数を使用することで，さらに効率的な実行が可能となると思われる．また，一つの手を決定するランダムシミュレーション中でも，始めと後でシグモイド関数を変更することも有効かもしれない．

実験の結果，終盤に近い局面ではスコアの値が有効になることが伺えた．また，シグモイド関数を局面の状況に応じて適用することにより，約 55%の勝率が得られた．モンテカルロ法では手の枝刈りやランダムシミュレーションでの手の決定などにヒューリスティックを用いる方法が提案されているが[7]，k の最適値を求めるためにヒューリスティックな情報を使用することも有効かもしれない．

人間に対するアンケート結果では，人間らしい面白い手をシグモイドモデルは実現できることを得た．た

だし，序盤においては勝敗による計算の方が明確な悪手を指しにくくなることがアンケート結果からも得られた．回答数が少ないため，断言はできないが，シグモイドは面白いゲームを実現する上で有効であると思われる．

5. おわりに

モンテカルロ法においてランダムシミュレーションの数が十分実行できる場合，スコアを用いる方が勝敗を用いる場合よりも優ることを示した．その上で，UCT はスコアの持つ情報を有効に活用できないことを示した．そこで，モンテカルロ法の戻り値として，勝敗やスコアの重ね合わせを得ることができる，シグモイド関数を用いる方法を提案した．その結果，勝敗を用いる場合と比較し，約 55%の勝率向上が得られ，有意に性能を改善することが示された．また，既存のモンテカルロ法に見られた勝敗の偏りを改善することができた．アンケートでは，より人間らしく面白い手を実現できることを示した．

今後の展開としては，ランダムシミュレーションの試行回数や局面の状況に応じた動的なシグモイド関数の傾きの設定方法や，対象とするゲームによらない，最適な傾きの計算方法の構築などが挙げられる．

参考文献

1) Bernd Brüegmann: Monte Carlo Go, Technical report, Physics Department, SyracuseUniversity, unpublished, 1993.
2) Bruno Bouzy and Bernard Helmstetter: Monte Carlo go developments, Advances in Computer Games conference (ACG-10), pp. 159-174, 2003.
3) Bruno Bouzy: Associating Shallow and selective global tree search with Monte Carlo for 9 × 9 Go, 4rd Computer and Games Conference, CG04, Ramat-Gan, Israel, LNCS 3846/2006, pages 67-80, 2004.
4) Coulom, R.: Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search, Proceedings of the 5th Computers and Games Conference (CG 2006), pp.29-31, 2007.
5) L. Kocsis and C. Szepesvari. Bandit-based monte-carlo planning: In 15the European Conference on Machine Learning(ECML), pp. 282-293, 2006.
6) P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite time analysis of the multiarmed bandit problem. Machine Learning, 47(2-3), pp. 235-256, 2002.
7) S. Gelly and D.Silver: Combining Online and Offline Knowledge in UCT, International Conference of Machine Learning, ICML 2007, Corvallis Oregon USA, pp. 273-280, 2007.
8) Rémi Coulom: Monte-Carlo Tree Search in Crazy Stone, Game Programming Workshop 2007，pp.74–75 , 2007.
9) 橋本隼一, 橋本 剛, 長嶋淳: コンピュータ将棋におけるモンテカルロ法の可能性, GPW2006, pp.195-198, 2006.
10) 中村 秋吾，三輪 誠，近山 隆: 静的評価関数を用いた UCT の改善，Game Programming Workshop 2007，pp. 36–43 , 2007.
11) Tristan Cazenave: A Phantom-Go Program, ACG2005, pp.120-125, 2005.
12) Guillaume Chaslot, Mark Winands, H. Jaap van den Herik, Jos Uiterwijk, and Bruno Bouzy: Progressive strategies for Monte-Carlo tree search, In Joint Conference on Information Sciences, 2007.
13) Sylvain Gelly and Yizao Wang: Exploration exploitation in Go: UCT for Monte-Carlo Go, Twentieth Annual Conference on Neural Information Processing Systems (NIPS2006), 2006.
14) Tristan Cazenave and Bernard Helmstetter: Combining tactical search and Monte-Carlo in the game of go. IEEE CIG2005, pp. 171-175, 2005.
15) Bruno Bouzy: Old-fashioned Computer Go vs Monte-Carlo Go, Invited tutorial, IEEE 2007 Symposium on Computational Intelligence in Games, CIG 07, unpublished, 2007.
16) 保木 邦仁: 局面評価の学習を目指した探索結果の最適制御，Game Programming Workshop 2006, pp.78-83, 2006.