情報処理学会論文誌 Vol.55 No.11 2389–2398 (Nov. 2014)

ベイジアンアプローチに基づく モンテカルロ木探索アルゴリズムの将棋への適用と評価

横山 大作[1,a)] 喜連川 優[2,1,b)]

受付日 2014年2月21日, 採録日 2014年9月12日

概要：我々は，Bayesian Approach に基づいた新しいモンテカルロ木探索アルゴリズム（MCTS）を提案し，将棋に対して適用を試みた．提案手法は，(1) 乱数を付加した評価関数によるゲーム木探索をシミュレーション試行として用いること，(2) 評価値を確率分布として扱い，Bayesian Approach に基づいてゲーム木中を伝搬させること，という 2 点から構成される．これは，将棋に代表される，従来の MCTS が有効に利用できない tactical なゲームへの適用を想定した手法である．問題領域特有の探索技法を多数利用しているトップレベルのコンピュータプレイヤを用いて実装を行い，大量の自己対戦による性能評価を行った結果，十分なリソースを利用可能であれば，提案手法によって従来のゲーム木探索を越える性能を実現できることが示された．また，提案手法の詳細な性能特性評価を通して，提案手法の設計指針に関する理解が得られた．

キーワード：ゲーム木探索アルゴリズム，モンテカルロ木探索，将棋，Bayesian Approach

An Application and Evaluation of Monte-Carlo Tree Search Algorithm for Shogi Player Based on Bayesian Approach

Daisaku Yokoyama[1,a)] Masaru Kitsuregawa[2,1,b)]

Received: February 21, 2014, Accepted: September 12, 2014

Abstract: We propose a new Monte-Carlo Tree Search (MCTS) algorithm based on Bayesian Approach to improve performance of game-tree search. It consists of two main concepts; (1) using multiple game-tree search with a randomized evaluation function as simulations, (2) treating evaluated values as probability distribution and propagating it through the game-tree using the Bayesian Approach concept. Proposed method is focusing on applying to tactical games such as Shogi, in which MCTS is not currently effective. We apply the method for shogi using a top-level computer player application which is constructed with many domain-specific search techniques. Through large amount of self-play evaluations, we conclude our method can achieve good win ratio against an ordinary game-tree search based player when enough computing resource is available. We also precisely examine performance behaviors of the method, and depict designing directions.

Keywords: game-tree search algorithm, Monte-Carlo tree search, Shogi, Bayesian Approach

1. はじめに

将棋，囲碁などに代表される 2 人ゲームは，相異なる 2 つの目的関数を持つプレイヤが互いの利益を最大化しようとする複雑な構造を持つ探索問題である．人間が楽しむエンタテイメントの目的だけでなく，人工知能分野に応用される最適化手法や，きわめて大規模な解空間を効率良く扱う分散探索計算技術の応用分野として，広く研究されてきた．

オセロ，チェス，将棋などは，局面の優位度を数値化する評価関数を用いたゲーム木（min-max 木）探索により次の指し手を求めることが通例であり，現時点ではこの手法

[1] 東京大学生産技術研究所
Institute of Industrial Science, The University of Tokyo, Meguro, Tokyo 153–8505, Japan
[2] 国立情報学研究所
National Institute of Informatics, Chiyoda, Tokyo 101–8430, Japan
[a)] yokoyama@tkl.iis.u-tokyo.ac.jp
[b)] kitsure@tkl.iis.u-tokyo.ac.jp

が最良（最強）の結果を出している．一方，囲碁に関しては乱数を用いたモンテカルロ木探索を利用したアルゴリズムが提案され [1]，急速に強さが向上している．これは，モンテカルロ木探索が比較的容易に分散計算化可能なため，コンピュータリソースを大量に利用して高速化できることも大きく影響している．この新たなアルゴリズムの成功を受けて，将棋においてもモンテカルロ木探索の適用が試みられてきたが，従来の木探索手法より良好な性能は得られていない [2], [3]．モンテカルロ木探索では，乱数を用いた適当な手順により終局まで局面を進める「プレイアウト」を多数繰り返して評価を行うが，ゲームの性質上，囲碁と異なり将棋では効果的なプレイアウトを生成することが困難であること，および多数のプレイアウトによる評価が通常の木探索の正確さに及ばないという事情によると考えられている．特に後者については，ある局面において正解といえる手順がごく少数に限られ，その正解手順をたどり続けて初めて局面に優劣の判定が付くようなゲームに関して，現在のモンテカルロ木探索手法では効率良く探索空間を絞ることができないという問題点が露呈しているといえる [4]．

我々は，Bayesian Approach に基づく探索木を作成し，乱数を加えた逐次探索をシミュレーションとして利用する，新しいモンテカルロ木探索手法を提案する．これは，将棋のような問題で有効性が十分検証されている評価関数と逐次探索を利用し，誤差を含む多数の評価値をうまく集約することでより並列実行に向いた木探索を実現することを目指すものである．提案手法を実際の将棋プレイヤを用いて実装し，探索木の形など，性能を左右する設計項目を詳細に評価することを通して，適切な設定の元ではオリジナルより有意に強いプレイヤが実現できることを確認した．

本論文の構成は以下のとおりである．2 章ではモンテカルロ木探索の将棋への応用に関する関連研究を述べる．3 章では提案手法の詳細を説明し，性能に関わる設計項目を整理する．4 章で多数の対戦による評価実験を行い，提案手法の性質と有効性を評価考察し，5 章で結論を述べる．

2. 関連研究

2.1 モンテカルロ木探索での評価関数の利用

モンテカルロ木探索は，単純に乱数による試行を繰り返すモンテカルロ法を，min-max 木の探索と組み合わせたものであり，選択的に成長させた min-max 木に従ってゲームの最善手を求める手法である [5]．近年，UCB（Upper Confidence Bound 値）を利用して絞り込みを行う UCT 探索が提案され [6]，特に囲碁において良好な性能を示し，広く使われ始めている [1]．

モンテカルロ木探索は囲碁のように精度の良い評価関数を作ることが難しい問題で特に有効とされているが，Amazons や Lines of Action（LOA）など，ある程度信頼できる評価関数が得られているゲームにおいても適用され，従来手法と同等程度の強さが得られている [4], [7]．

しかし，従来型の木探索が有効な問題領域では，評価関数以外にも探索そのものに関する多数の技術が確立されている．将棋においては，futility-pruning [8] や実現確率 [9] などの枝刈り技法，詰み専用探索の利用 [10] など，様々な工夫を利用して高性能な木探索が実現されているが，このような技術は現在のモンテカルロ木探索では利用できない．我々は，プレイアウトの代わりに従来の木探索をそのまま利用できるアルゴリズムを提案し，モンテカルロ木探索において従来技法の効果をより有効に活用することを目指している．

モンテカルロ木探索には，明確な勝ち負けが確定している局面（いわゆる「詰み」の局面）でも，そのことが探索に反映されにくいという課題点がある．これに対し，プレイアウトによる勝率の値とは別の評価値を設けて対処する，Monte-carlo Tree Search Solver [11] という技法が提案されているが，複数の評価値伝搬を併用する必要がありアルゴリズムの複雑さは増す．我々は，シミュレーションに相当する探索によって得られる評価値の分布をそのまま扱える手法を提案する．我々の手法では，明確に評価値が確定するような局面は評価値分布がある値に収束するものとして表現され，モンテカルロ木探索を行う過程において自然にその分布が考慮されるため，よりシンプルにこの問題点を解決できると考えられる．

2.2 将棋に対するモンテカルロ木探索

橋本ら [2]，佐藤ら [3] など，将棋に対してモンテカルロ木探索の適用を試みた研究が存在する．純粋にプレイアウトを用いるこれらの研究は，従来のゲーム木探索より良好な性能は得られていない．

近年の機械学習を用いた手法の成功などにより，将棋は精度の高い評価関数を利用可能になっている．竹内ら [12] は，評価値を利用し，ルート局面とプレイアウト末端の静止探索後の評価値の差を用いてプレイアウトの勝敗を定義する手法を提案した．問題集による評価では評価値を用いる効果が確認できたが，レーティングを用いた評価によると，従来のゲーム木探索に匹敵する性能は得られていない．

2.3 合議

将棋においては，近年「合議」という手法が提案された [13]．これは，探索の評価関数に異なる乱数を加えて複数回の探索を行い，多数決で指し手を選択するというシンプルな方法であるが，性能は向上することが知られている．従来の逐次探索からの変更が少ないこと，並列化方法が自明で容易なことが利点である．なお，合議は，提案手法の枠組みでは，「初期局面においてのみシミュレーションを行い，木の展開を行わない手法」と考えることもできる．

2.4 Bayesian Approach に基づく探索手法

我々は，提案する Bayesian Approach に基づくモンテカルロ木探索手法について，幅優先，UCB 値による探索制御手法，および合議手法との比較を行った検証途中結果を報告している [14]．比較結果は以下のようであった．

- すべての方式で，オリジナルとの対戦実験で 6 割程度の勝率が得られる，より強いプレイヤを作ることは可能であった．ただし，使用リソース量（探索時間）はオリジナルの数倍から数十倍を要した．
- 合議手法は多数決をとる数を増加させていっても勝率は向上しなかった．
- 幅優先制御の場合は他方式に比較して使用リソース量（探索時間）が多い．
- UCB 値による制御の場合も使用リソース量が多く，パラメータを調整しても勝率に限界があった．
- 提案手法である QSS による制御の場合，モンテカルロ木を大きくしていくと，幅優先と UCB に比較して少ない使用リソース量で同等の勝率が得られる．また，調査した範囲では勝率がリソース量増加に従って向上し，限界に達しているようには見受けられなかった．

この結果，制御方式に QSS を用いる手法が，幅優先や UCB 値を用いる方式と比べて有望であるとの結論を得た．

また，本論文の評価実験の多くは文献 [15] において途中結果報告を行ったものである．本論文は，これらの結果をふまえ，不足していた評価実験を加えるとともに，提案手法におけるシミュレーションの性質に関する評価と考察を加え，まとめた成果である．

3. 提案手法

提案手法は，モンテカルロ探索のシミュレーションとして乱数を付加した探索を用いること，Bayesian Approach に基づいて探索木中の評価値伝搬を行うこと，の 2 点から構成される．以下，詳細に説明する．

3.1 乱数を付加した探索によるシミュレーション

提案手法では，ランダムな指し手を生成するのではなく，評価関数に乱数を加えた従来のミニマックス探索を複数回行い，その探索木で得られる指し手と評価値をシミュレーション結果とする．複数のミニマックス探索により，モンテカルロ探索木のリーフでは複数個の評価値が得られる．これをそのリーフの「確率分布を持つ評価値」として扱うことにする．

乱数付加においては，合議方式と同様に，シミュレーション探索開始局面（すなわち，モンテカルロ木のリーフ局面）におけるシミュレーション回数と評価局面とをキーとした疑似乱数を生成して利用する．将棋の探索空間は合流が多いため，シミュレーション探索中に同一局面に至ったときに同じ乱数値を加えた評価値が得られるようにするためである．さらに，シミュレーション回数を乱数のキーに加えることで，モンテカルロ木の複数のリーフノードで同一局面が存在するとき，同じシミュレーション回数のときは同じシミュレーション探索結果（分布）になるような一貫性も実現できる．ただし，シミュレーション回数が異なるときは同一局面でも異なる分布となり，モンテカルロ木内で完全な一貫性を保っているわけではない．また，このような特徴が利点となるかどうかに関しては検証されていない．

乱数付加過程は，具体的には，評価局面から Zobrist hashing [16] を用いて一定長（96 bit）のキーを生成し，その分割部分列に対してシミュレーション回数を加算，乗算した一定長（48 bit）の乱数シードを生成して，疑似正規分布に従った乱数を生成し，評価局面の評価値に加算する．実行時の性能を考慮すると，あらかじめ疑似乱数列を生成してメモリ上に保存しておくなどの方法による効率化が考えられるが，本実験ではそのような工夫は施していない．

3.2 Bayesian Approach による評価値伝搬

Bayesian Approach [17] は，リーフの値に確率分布があるときに，その確率分布を考慮したミニマックス探索を実現する手法である．チェスなどでは探索に関する既存の技法が利用できないことから有効性が低いとされてきた [18]．また，囲碁を模した単純なシミュレーションでは有望な結果が得られていた [19]．

提案手法では，モンテカルロ木での評価値伝搬においてこの Bayesian Approach を利用し，リーフの評価値分布を考慮した探索木を構築する．評価値分布の伝搬を効率良く計算するために，リーフの評価値分布は離散的な確率分布であるとして複数のピンで表されるものとする．

図 1 はシミュレーションの回数とそこで得られる評価値分布を示している．シミュレーション回数が 1 回のときには，得られた評価値（v_1）に δ だけずれを加えた値にも小さな確率を与えた疑似的な分布を生成する．またシミュ

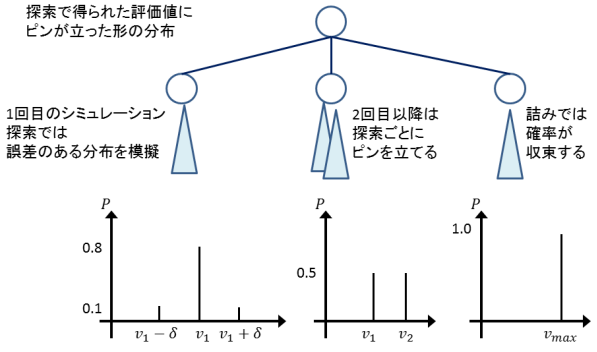

図 1 シミュレーション結果による評価値分布の作成手法

Fig. 1 Probability distribution of evaluated values constructed from simulations.

レーション数が 2 以上のときは，$v_1 \pm \delta$ の点の疑似的な確率は削除し，それぞれのシミュレーションで得られた評価値（v_1，v_2 など）が，出現回数に従った確率で得られる確率分布であるとする．今回の実装では，1 回目のシミュレーション（v_1）についてはオリジナルの評価値に乱数誤差を加えない探索を使用し，2 回目以降のシミュレーション（v_2 など）にはオリジナル評価値に乱数誤差を加えて探索を行っている．

なお，将棋の場合，探索を通して局面の勝ち負けが確定していることを判定することが可能である．これを反映するため，シミュレーションの結果，詰みだと判断され勝ち負けが確定した場合には，評価値が誤差を持たない，1 本のピンで表現される分布になるとした．

3.3 探索木の展開制御

提案手法における探索木の展開順序の制御，終了判定などのアルゴリズムを**図 2** に示す．また，探索制御に用いているパラメータの説明を**図 3** に示す．モンテカルロ木のリーフの中で最も興味があるノード *refine_p* を選択し，シミュレーション回数が設定された一定回数（*Simnum*）未満の場合はシミュレーションを行い，*Simnum* 以上に達した場合にはそのノードを展開してモンテカルロ木を成長させる，という動きが基本となる．

Playdepth はプレイヤの強さをおおむね規定することを期待した深さパラメータであり，提案手法ではおおむね *Playdepth* の長さだけの PV（Principal Variation）が最終的に得られることを期待した制御を行っている．*PVth* は，得たい PV 長をどの程度延長するかを定めるパラメータとなる．ここで，PV は，双方の手番が最良評価値の手を選択すると仮定したときに初期局面の手番側のプレイヤが最良の評価値を得られると考えられる経路を指す．通常のミニマックス木と異なり，本提案のモンテカルロ木探索ではノードの評価値は確率分布を持つため，評価値分布の期待値が手番側にとって最も有利になるような子ノードを選択する，と仮定したときの初期局面からの最良経路を提案手法での PV と定義する．

本論文で検証を試みている手法は，Bayesian Approach で提唱されている，ルートノードの確率分布に対して最も影響を与える度合いが大きいリーフノードを選択して展開する，という QSS（Q Step Size）による探索制御アルゴリズムである．現在想定している最善手を上回る手があるときの確率重み付け変化量 Q^1：

$$Q^1 = \int_0^\infty P^1(q) q dq$$

（ここで，$P^1(q)$ は「真の最善手が現時点で想定される最善手 (1) を q だけ上回る確率」である）に対し，モンテカルロ木のリーフノードそれぞれが与える変化量を QSS と呼び，

```
[探索のメインループ]
while root の終了条件が満たされない:
   refine_p ← find_refine()
  if  refine_p のシミュレーションが Simnum 未満:
     refine_p でシミュレーション追加
  else:
     refine_p を展開
  for p in [refine_p から root まで上る]:
     p の確率分布をアップデート
     p が詰まされているなら別の応手を探す*1
   root で U_all を求める
   QSS を root からリーフまで再計算

[(シミュレーション or 展開）対象ノードの選択]
function find_refine():
  if U_all が一定回数（100 回）以上変化していない:
    return PV のリーフ
  if U_all が閾値（Uall_th）以下:
    return PV のリーフ
   leaves ← モンテカルロ木リーフノード
   leaves を QSS の大きいものから降順にソート
   leaves を先頭 1/10 のみ残す
  for p in [ leaves ]:
    if p の深さ +Simdepth < Playdepth:
      return p
  // leaves のシミュレーションがすべて Playdepth 到達
  return PV のリーフ

[展開処理]
- root の場合は全幅, それ以外では max(12−depth×2, 3)
に従う数の子ノードを展開対象とする
- Simdepth の深さの通常のアルファベータ探索を行い，
必要な展開幅の数の候補手を生成して，モンテカルロ木の
子ノードとして追加する．

[終了条件]
以下の条件のいずれかが満たされたら終了
- root の確率分布が収束（詰み・詰まされの場合）
- PV のリーフノードが Simnum 以上のシミュレーション
を行い，depth(PV)+Simdepth >= Playdepth+PVth
となる
```

図 2 探索制御アルゴリズム
Fig. 2 Algorithm for controlling search.

$$QSS_L = \sum_j p_j \left| Q_L^1(j) - Q^1 \right|$$

として定義される．ここで，$Q_L^1(j)$ は現在考慮しているノード L の評価値が j に定まったときの Q^1，p_j はノード L での評価値 j の確率を表す．各リーフノードの QSS は，図 2 に示したように，あるリーフ *refine_p* の評価値分布が変化したときに，影響するノードをルートからたどって再

*1 モンテカルロ木が全幅で展開されていないため，詰まない応手が他に存在しないかチェックする必要がある．

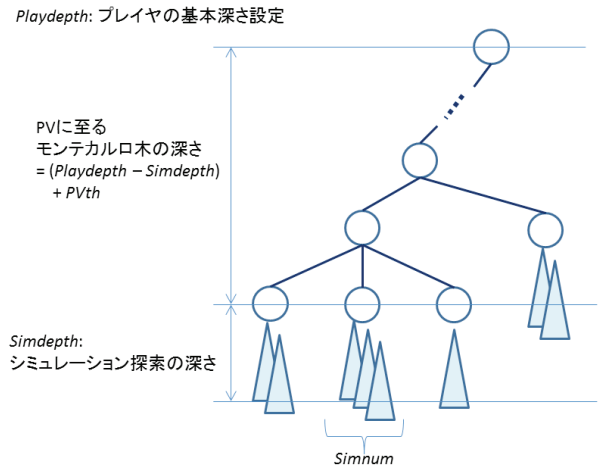

図 3 提案手法における探索制御パラメータ
Fig. 3 Key parameters of proposed method.

計算される．この QSS が大きいリーフを重点的に展開することが基本戦略となる．

また，探索終了条件について，元の Bayesian Approach では U_{all} という指標が一定以上小さくなったところで探索を打ち切るという終了条件を用いていた．U_{all} は現在最善手と考えられている手を指したときの期待値変化であり，

$$U_{all} = <\rho_{root}> - <\rho_1>$$

と定義される．ただし，ρ はノードの評価値の分布であり，$<\rho>$ はその期待値，ρ_1 はルートで最善手と考えられている子ノードの分布を表す．ここで，PV 長が *Playdepth* 程度出力されることが望ましいと考えたため，本論文では U_{all} が小さいだけでは探索終了とせず，現在の PV のリーフノードを展開する，という手法をとっている．また，リーフノードの 1/10 が *Playdepth* に到達した場合も，それ以上 QSS による最良優先探索を続けず，現在の PV ノードを展開するようにしている．さらに，モンテカルロ木の展開幅（子ノードの数）についても，図 2 の［展開処理］に示したように，*root* ノードの場合は全幅，それ以外のノードは *root* からの深さ（*depth*）が深くなるごとに次第に展開幅が小さくなるように制御を行っている（ただし，最小でも展開幅 3）．これらの打ち切り条件，展開幅制御に用いられているアドホックな制御は，ある程度現実的な探索時間で結果を返すために加えられたものであるが，その妥当性について詳細な検討は行っていない．

3.4 性能を左右する設計項目

本手法の探索戦略においては，大きく以下の項目に性能を左右する設計項目，およびトレードオフ点が存在すると考えられる．

シミュレーションに加える乱数分布 加える乱数が小さい場合はシミュレーション結果が変化せず，シミュレーションとしての働きが弱くなるが，乱数を大きくすると探索の精度に影響が及び，妥当なシミュレーション結果が得られなくなるため，何らかのトレードオフ点があると考えられる．

シミュレーション結果を用いた評価値分布の作成手法 Bayesian Approach では，リーフに評価値の確率分布があるミニマックス木を容易に扱うために，確率分布をいくつかのピンによって表現する．そのため，シミュレーションを複数回行った結果からこの確率分布を作成する何らかのモデルが必要になる．特に，リーフにおけるシミュレーションの回数が少ないときには，評価値の信頼度が低いことを適切に表現するような手法が必要となる．提案手法では，図 1 のように，1 回めのシミュレーション結果について得られた評価値 v に対し $v \pm \delta$ の点にピンが存在する確率分布を作成し，誤差の存在を表現している．

シミュレーション探索部分の大きさ（*Simdepth*） 本手法は将来並列化による高速化が可能であると期待されるが，その際，1 回のシミュレーションにかけるリソース量が問題となる．本手法では，個々のシミュレーション部分は逐次に実行され，複数のシミュレーションを同時に実行することで並列化が行えると考えられる．並列化アルゴリズムの詳細は今後検討する必要があるが，大まかには，`find_refine()` で複数のリーフノードを返すなどして複数の *refine_p* を選び，そのリーフノード群に対するシミュレーションを並列に実行する方式が考えられる．ここで，1 回のシミュレーションに利用するリソース量を多くすると，シミュレーションの精度は向上するが，並列実行されるタスクの粒度が大きくなり，並列化が可能な部分を減らしてしまう可能性がある．

なお本手法において，シミュレーション部分を最大まで大きくし，ルートノードでのみシミュレーションを行うようにすると，既存手法である合議 [13] による制御にきわめて近いアルゴリズムとなる．

シミュレーション数閾値（*Simnum*） シミュレーションがある程度精度良く局面評価を行えると期待できるため，従来のシミュレーションよりは少ない回数でモンテカルロ木のリーフの評価が収束し，より少ないシミュレーション数で展開を行う方が効率が良い可能性がある．

本論文では，将棋を対象問題として本アルゴリズムの実装を行い，多数の対戦を行ってこれらの項目に対する評価を試みる．

4. 評価

4.1 実験環境

提案手法を実装し，オリジナルのプレイヤとの対局を多数回行うことで手法の性質と有効性を評価した．実装にお

いては，コンピュータ将棋プレイヤ「激指*2」を利用し，その評価関数に模擬正規分布乱数を加えてシミュレーションを作成した．激指は実現確率打ち切り探索など既存の探索技法を多数導入した実用的なプログラムであり，世界コンピュータ将棋選手権などで優秀な成績を収めている．

テストでの勝率測定は，実戦棋譜の 31 手目の局面からランダムに 500 局面を選択し，その局面から先後を入れ替えて 1 回ずつ，合計 1,000 対局によって求めた．オリジナルプレイヤの設定は得られる PV 長がおおむね 12 になるようなものとし，提案手法のプレイヤもそれに相当するよう *Playdepth* を 12 に設定した．それぞれの 1,000 対局にはおおむね 500〜1,000 時間程度の CPU 時間を要した．

また，以後の実験結果すべてにおいて，勝率については 95%信頼区間をエラーバーとして示してある．

なお，本手法においてモンテカルロ木の確率伝搬過程の消費時間は，確率分布のピンの数に大きく依存する．確率分布を持つピンの値をある程度丸める，あるいは複数のピンの分布を小数のピンで近似してピンの数を減らす [17]，などの効率化手法が考えられるが，今回の実験ではこのような効率化は行っていない．ピンの持つ値は評価関数の値と同じく，歩を 100 点としたときの 1 点を最小刻みとしている．

4.2 乱数分布の影響

シミュレーション探索に付加する正規分布乱数の分布を変化させて評価を行った．*Simdepth* を 8，*PVth* を 4 に固定し，乱数の標準偏差 σ を変化させたときの勝率推移を**図 4** に示す．各系列は *Simnum* を 3，5 と設定した場合である．なお，グラフの見やすさのためにそれぞれの実験結果の点はわずかにずらしてプロットしてある．また，激指においては評価値 100 が歩一枚の駒価値に相当している．

σ が 800 以下程度の領域では，勝率はそれほど大きく変化していない．しかし，全般的には σ が大きくなると勝率が下がる傾向にあるように見受けられ，1,500 程度まで大きくなるとはっきりと差が現れる．

以降の実験では，$\sigma = 200$ の乱数を用いている．

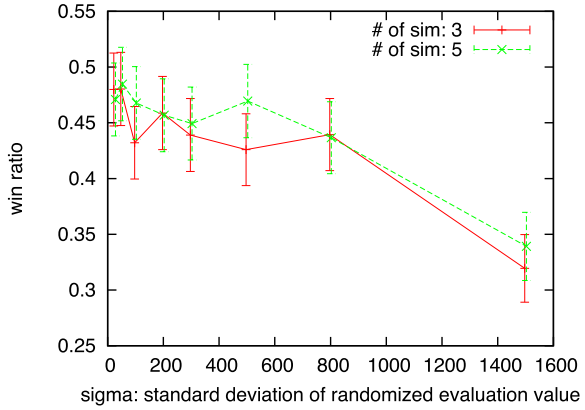

図 4 評価値に付加する乱数分布変更時の勝率

Fig. 4 Effects of additional random values for evaluated values.

*2 http://www.logos.t.u-tokyo.ac.jp/~gekisashi/

4.3 初期確率分布の影響

モンテカルロ木のリーフを展開し，シミュレーションを 1 回だけ行ったノードにおいては，誤差のある確率分布を模擬するため，シミュレーションで得られた評価値 v に対して δ だけ離れたところにピンを立てる（図 1）．この仮想的な確率分布の形の設定がどのような影響を与えるかを調べるため，δ を変化させて対局による評価を行った．

Simdepth を 8，*PVth* を 4 と固定し，δ を変化させたときの勝率変化を**図 5** に示す．各系列はノード展開に必要なシミュレーション回数の閾値 *Simnum* を 1，3，5 と変化させたときの結果である．横軸は δ であり，それぞれの系列で 25，50，100，200，300，500 と変化させているが，グラフでは見やすさのためにわずかに横方向にずらしてプロットしている．ただし，$\delta = 500$ の点のみ，$\pm\delta$ 地点の確率分布の値が 0.25 と異なって設定されているため，参考結果として掲載する．

今回の実験の範囲では，*Simnum* の設定によらず δ が小さい方が高い勝率を得られる結果が見て取れる．*Simnum* が 1 の場合，δ は 50 以下の範囲で勝率が高く，それ以上大きくした場合には勝率が低下する．一方，*Simnum* を 3 以上にした場合は，δ が 200 以下の範囲ではあまり明確な勝率変化は現れない．これは，*Simnum* が 1 より大きい場合には，シミュレーション探索の評価値に加えられる乱数（$\sigma = 200$）が最終的に得られるモンテカルロ木の性質を規定しており，そのシミュレーション結果の分布と比較して 1 回目のシミュレーションの確率分布範囲が小さい場合には，δ は最終的な結果へ影響を及ぼしにくい，という理由によるものと推察される．

また，*Simnum* を増やしても勝率にはそれほど影響が現れない，という結果も見て取れる．現在の手法について，

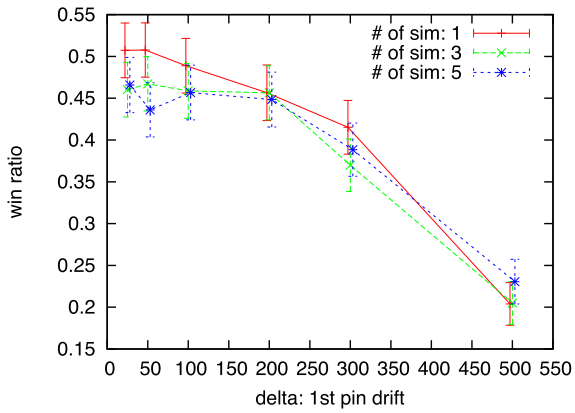

図 5 初期確率分布変更時の勝率

Fig. 5 Effects of initial probability distribution.

モンテカルロ木が初期確率分布に強く影響を受けており，シミュレーションによってあまり修正されない，という状況にあると推察される．

4.4 シミュレーション単位の影響

δ を 100，*PVth* を 4 に設定し，シミュレーション探索のサイズ *Simdepth* を変化させたときの勝率変化を**図 6** に示す．グラフの系列は *Simnum* をそれぞれ 1, 3, 5 と設定した場合を示している．また，*Simnum* を 1，δ を 500 とした設定での結果を系列 “# of sim: 1, delta 500” に示している．

Simdepth が 6 より小さい場合は勝率がほぼゼロであり，ある程度以上の規模でシミュレーション探索を行わないと，オリジナルのアルファベータ探索の強さに達しないことが分かる．また，異なる δ でもこの傾向は同様であることが見て取れる．

また，*Simdepth* を 2 から 10 までの値で固定し，*PVth* を変化させたときの勝率変化を**図 7** に示す．横軸はオリジナルプレイヤと比較した消費時間の比である．

Simdepth が 6 より小さい場合には，*PVth* を 4 から 16 まで変化させてもほぼ勝率は横ばいであり，モンテカルロ木の深さを深くしていっても効果がないことが分かるが，*Simdepth* を 8 まで大きくすると，モンテカルロ木の深さを深くすることで勝率が向上し，提案手法で効果が得られることが確認できる．ただし，木の深さを深くしたときの勝率向上の度合いは漸減していき，深さ 12 程度で向上が頭打ちになることも見て取れる．*Simdepth* を 10 まで大きくすると勝率はさらに高くなり，シミュレーションサイズは大きいほど強くなるという結果が得られた．

一方，消費時間を考えると，*Simdepth* が 10 の場合は，8 の場合と比較して同一の *PVth* 設定での消費時間が 2.5〜3 倍程度延びており，同一の消費時間を要する領域で比較すると，*Simdepth* が 8 の方が高い勝率を得られる場合もあることが分かる．つまり，使用リソースに対する効率の良さの観点では，必ずしもシミュレーションサイズを大きくした方が良いとはいえない．また，シミュレーションサイズを大きくするということは，逐次実行部分の粒度を大きくするため，並列計算の適用を考える場合にはより不利になることが考えられる．

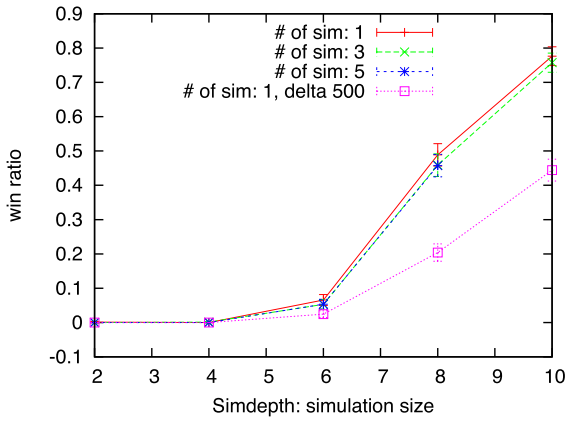

図 6 シミュレーション探索のサイズ変更時の勝率
Fig. 6 Effects of simulation search depth.

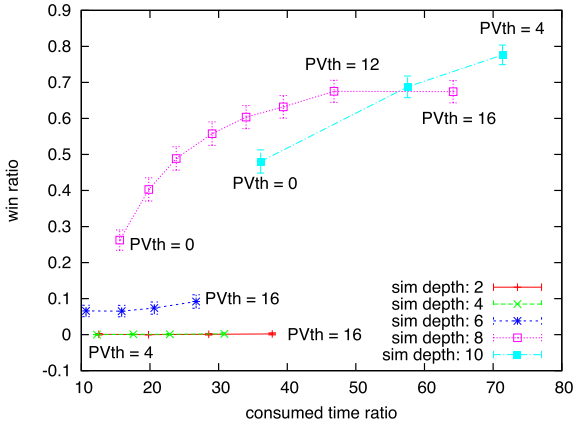

図 7 PV 長変更時の消費時間比率と勝率
Fig. 7 Effects of additional PV length compared by consumed time.

4.5 展開に要するシミュレーション回数の影響

Simdepth を 8 に固定し，*Simnum* を 1 から 7 まで変化させたときの勝率変化を**図 8** に示す．それぞれの系列は *PVth* を変化させた場合を示している．どの系列においても，*Simnum* を増加させても勝率は向上しない結果が見て取れる．全般的に，*Simnum* を 1 から 3 に変化させたときに比較的大きく勝率が下がり，それ以降は *Simnum* を増加させてもほぼ横ばいの推移を行っているように見える．*Simnum* = 1 のときは，激指の持つオリジナルの評価関数の値そのものが期待値となるような評価値分布，すなわち図 1 の v_1 が乱数誤差を加えないオリジナルの評価値そのものとなるように設定されており，$v_1 \pm \delta$ の評価値に小さな確率が存在するにしても，実質的にはモンテカルロ木のリーフの値は激指の評価関数の値が支配的であると考えられる．シミュレーション回数を増やした場合には，乱数誤

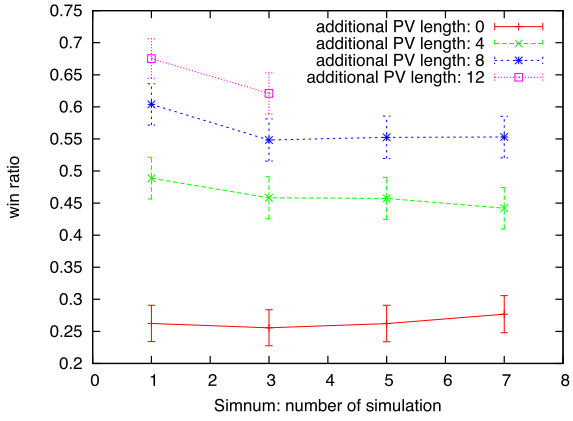

図 8 シミュレーション回数閾値と勝率の関係
Fig. 8 Effects of the threshold of node expansion.

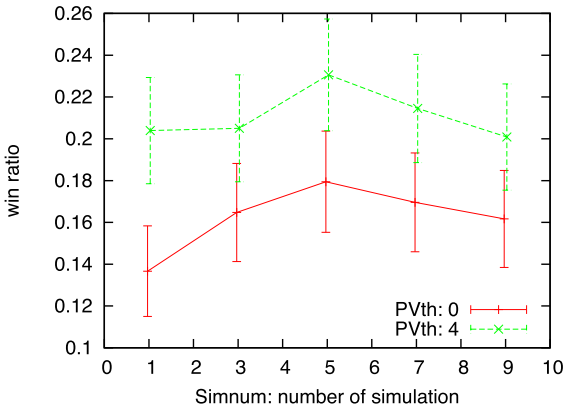

図 9 シミュレーション回数閾値と勝率の関係（$\delta = 500$）
Fig. 9 Effects of the threshold of node expansion ($\delta = 500$).

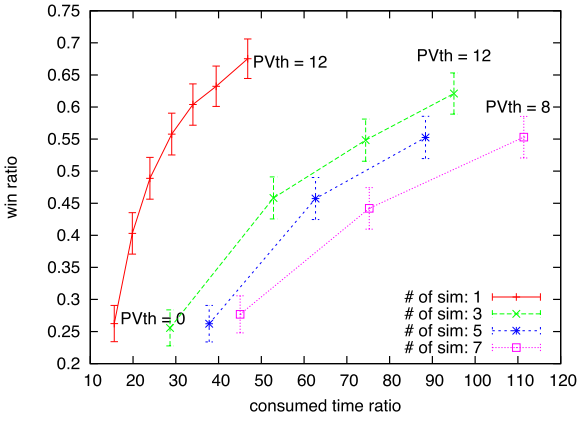

図 10 勝率と所要時間の関係
Fig. 10 Performance comparison of win ratio and consumed time under typical settings.

差を加えた評価値（v_2 など）にも高い確率が存在するような分布が用いられるようになるため，今回の実験結果にはその乱数誤差の影響が現れたと考えられる．

また，初期確率分布の δ を 500 とし，$v \pm \delta$ の点の確率を 0.25 としたプレイヤで，*Simnum* を変化させたときの勝率変化を**図 9** に示す．これは，シミュレーション 1 回目に得られた評価値の確率分布がより不確かなものとして扱われるような設定となる．*Simnum* が 5 回程度までは性能向上するが，それ以上の回数ではやはり勝率が低下している．初期確率分布の形の違いは，シミュレーション回数が増えると影響を失うため，ある程度以上のシミュレーション回数の領域では図 8 と同様の性質を示していると考えられる．シミュレーション回数が少ない領域で勝率が若干向上傾向にあることを考えると，評価値分布の構成方法を検討することで，提案手法が改善される可能性を示唆しているといえる．

4.6 勝率と所要時間の関係

これまでの評価結果をもとに，代表的なパラメータ設定を用いて提案手法の勝率と所要時間の関係を調べた．

δ を 100，*Simdepth* を 8 とし，*PVth* を変化させたときの勝率を**図 10** に示す．各系列は *Simnum* を 1 から 7 まで変えた場合である．*Simnum* が 1，3 の系列では *PVth* を 0 から 12 まで，それ以外の系列は *PVth* を 0 から 8 まで変化させている．横軸は対局で消費した時間をオリジナルプレイヤとの比率で示している．

いずれの設定においても，*PVth* を延ばしていくことで勝率は向上し，実験の範囲ではまだ限界には達していないように見受けられる．前述したように，*Simnum* を増やしたときには勝率がむしろ低下しており，消費時間が伸びることも考慮すると，現在の手法で *Simnum* を増やすことはあまり有効ではないと結論づけられる．

4.7 シミュレーション探索の特性

シミュレーション回数を増やしたときに性能が低下する

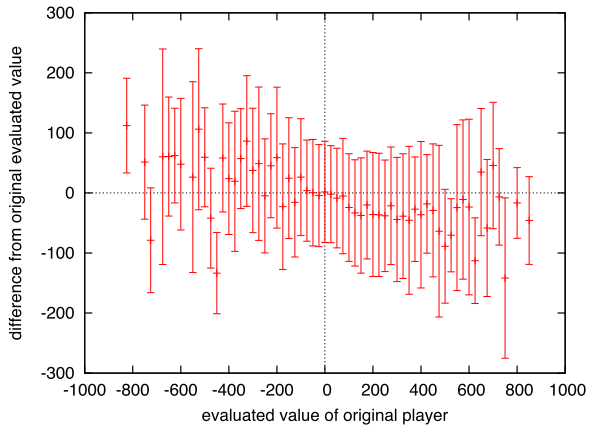

図 11 シミュレーション探索とオリジナル評価関数の探索結果のずれ（*Simdepth* = 8）
Fig. 11 Distribution of difference between evaluated values by simulation and original evaluation function (*Simdepth* = 8).

原因を探るために，シミュレーション探索で得られる探索結果値分布の性質を調査した．実戦棋譜において 20 手から 40 手目までの局面から 494 局面をランダムに抽出し，その一手後の局面と合わせた合計 988 局面を評価対象とした．それぞれの局面で，$\sigma = 200$，*Simdepth* = 8 のシミュレーション探索を 100 回ずつ行い，得られたシミュレーション結果（乱数が付加された評価関数を用いた探索木の値）とオリジナルの探索結果（乱数が付加されない評価関数を用いた探索木の値）との差を求めた結果を**図 11** に示す．横軸はオリジナルの探索結果の値であり，正の値のときに初期局面の手番側が有利であることを示すように正規化されている．初期局面をオリジナルの探索結果値 25 点ごとの刻みで分けて集計し，縦軸に木の値の差の平均と標準偏差を示している．

シミュレーション結果は，オリジナルの探索結果が正のときには負の誤差を，オリジナルが負だと正の誤差を持つような分布になっていることが見て取れる．つまり，自分

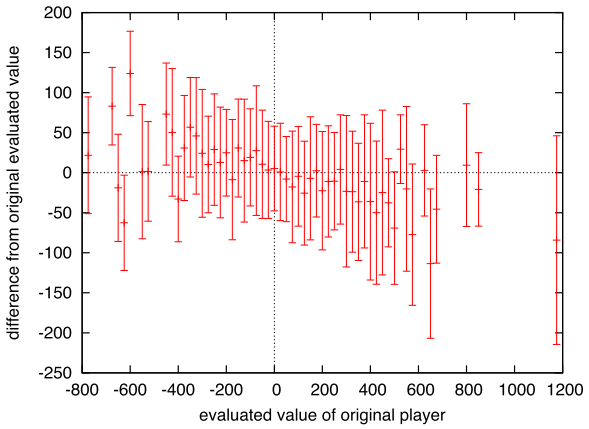

図 12 シミュレーション探索とオリジナル評価関数の探索結果のずれ（*Simdepth* = 12）

Fig. 12 Distribution of difference between evaluated values by simulation and original evaluation function (*Simdepth* = 12).

に有利な結果が得られる手についてシミュレーションはその有利さをやや割り引いて考えがちであり，不利なときには逆にやや有利に考えがちである，という傾向にあるといえる．この誤差は，最善手と次善手の値の差を小さくする方向に働き，最善手を選びにくくする結果をもたらしている可能性がある．この誤差の大きさがどの程度探索結果に影響を与えているかは検証の必要があるが，この分布はBayesian Approach の想定に沿っているとはいいにくい．

また，シミュレーションの深さを変えて同じ実験を行うと，シミュレーション深さが大きいときには前述の偏りがやや緩和されていくような傾向が見られた．例として *Simdepth* = 12 としたときの結果を**図 12** に示す．シミュレーション間のずれの分散がやや縮小しているが，偏りは弱くなりながらも依然として存在しているようである．適切なシミュレーション深さ設定への影響検証も含めて，今後の検討課題とする．

4.8 考察

本実験を通して，モンテカルロ木のリーフノードでのシミュレーション回数が 1 回のときが最も有望な性能を出し，シミュレーション回数を 3 回以上に増やしていっても効果が得られないばかりか，逆に性能低下の傾向が見られた．ただし，これは乱数を利用した合議による手法が有効であるという既知の知見 [13], [14] とはやや合致しない結果である．図 11，図 12 に示したように，シミュレーションを繰り返して得られるノードの確率分布が，手の間の評価値差を小さくする方向に偏っている可能性があり，これを考慮するような手法，あるいはこのような望ましくない性質が生じないシミュレーション作成方法を用いる必要が考えられる．既存の合議手法の研究においては，シミュレーション間での最良の評価値を採用する楽観合議手法が有効であるとの結果が得られているが，この原因がシミュレーションの評価値分布の性質と関連している可能性も考えられる．

また，今回はすべてのノードで同一の誤差を固定的に付加しているが，真の確率分布はノードごとに異なっており，その違いを反映することで探索がより精度良く行えると推察される．これは，元々の Bayesian Approach の提案でも指摘されている [17]．Baum らは局面の特徴量をもとにクラスタリングを行って確率分布を変化させたが，乱数付加したシミュレーションを繰り返すことは，このような局面の確率分布の違いを自然に反映するのかを検証し，真の確率分布がうまく推定できないような局面に限ってシミュレーションを行う，などの改良が考えられる．

また，初期確率分布が木の形をおおむね決めてしまい，その後のシミュレーションが木の形を修正するのに役立っていないように推察される結果も得られている（図 5）．偶然得られたシミュレーション結果の影響が修正されないのは望ましくない．シミュレーション回数が少ないときの確率分布の扱い（図 9）などに改良の可能性が考えられる．

一方で，シミュレーション回数 1 回のみでも，Bayesian Approach によってオリジナルより強いプレイヤが作成できたことは有用な知見である．乱数を用いたシミュレーションを行わない場合でも，Bayesian Approach による最良優先探索を行っている部分は並列に値を求めたいノードが多数存在しており，並列探索の適用が容易であると考えられる．並列処理の適用は今後の検討課題である．

5. 終わりに

我々は，従来モンテカルロ木探索の適用が難しかった将棋を対象に，

- 評価関数に乱数を与えた探索を用いてモンテカルロ探索のシミュレーションを構成する，
- Bayesian Approach に基づき，評価値分布を反映したゲーム木を構築する，

という 2 点を変更した新しいモンテカルロ木探索アルゴリズムを提案した．オリジナルプレイヤとの対局実験を通して，提案手法は十分なリソースを利用すれば強さを向上させることができることを確認した．また，設計項目に関する詳細な評価を行った結果，シミュレーションに用いる探索のサイズはある程度以上大きくなければ有効でないこと，適切な探索サイズは消費時間との関係で変わりうること，などの特性に関する理解が得られた．

一方，シミュレーション回数が少ないときの確率分布の扱いがモンテカルロ木の性質を大きく決定しており，シミュレーション回数を増やしてもその結果があまり反映されていないという問題点も明らかになった．並列実行適用時の有用性検証も含め，今後の検討課題である．

謝辞 本研究の一部は科学研究費助成事業（26280130）の助成によって行われた．

参考文献

[1] Gelly, S., Wang, Y., Munos, R. and Teytaud, O.: Modification of UCT with Patterns in Monte-Carlo Go, Technical Report RR-6062, INRIA (2006).

[2] 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，第 11 回ゲームプログラミングワークショップ (2006).

[3] 佐藤佳州，高橋大介：モンテカルロ木探索によるコンピュータ将棋，第 13 回ゲームプログラミングワークショップ (2008).

[4] Winands, M.H. and Björnsson, Y.: Evaluation Function Based Monte-Carlo LOA, *Advances in Computer Games*, van den Herik, H. and Spronck, P. (Eds.), *LNCS*, Vol.6048, pp.33–44, Springer Berlin Heidelberg (online), DOI: 10.1007/978-3-642-12993-3_4 (2010).

[5] Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo tree search, *CG 2006* (2006).

[6] Kocsis, L. and Szepesvári, C.: Bandit based monte-carlo planning, *Proc. 17th European Conference on Machine Learning*, *ECML '06*, pp.282–293 (online), DOI: 10.1007/11871842_29 (2006).

[7] Lorentz, R.J.: Amazons Discover Monte-Carlo, *CG 2008*, pp.13–24 (2008).

[8] Hoki, K. and Muramatsu, M.: Efficiency of three forward-pruning techniques in shogi: Futility pruning, null-move pruning, and Late Move Reduction (LMR), *Entertainment Computing*, Vol.3, No.3, pp.51–57 (online), DOI: http://dx.doi.org/10.1016/j.entcom.2011.11.003 (2012).

[9] Tsuruoka, Y., Yokoyama, D. and Chikayama, T.: Game-tree Search Algorithm based on Realization Probability, *ICGA Journal*, Vol.25, No.3, pp.145–152 (2002).

[10] Kishimoto, A., Winands, M., Müller, M. and Saito, J.-T.: Game-Tree Search using Proof Numbers: The First Twenty Years, *ICGA Journal*, Vol.35, No.3, pp.131–156 (2012).

[11] Winands, M.H., Björnsson, Y. and Saito, J.-T.: Monte-Carlo Tree Search Solver, *CG 2008*, pp.25–36 (online), DOI: 10.1007/978-3-540-87608-3_3 (2008).

[12] 竹内聖悟，金子知適，山口和紀：将棋における，評価関数を用いたモンテカルロ木探索，第 15 回ゲームプログラミングワークショップ，pp.86–89 (2010).

[13] 伊藤毅志，小幡拓弥，杉山卓弥，保木邦仁：将棋における合議アルゴリズム—多数決による手の選択，*IPSJ*, Vol.52, No.11, pp.3030–3037 (2011).

[14] 横山大作：モンテカルロ木探索アルゴリズムの将棋への適用，第 17 回ゲームプログラミングワークショップ，pp.76–83 (2012).

[15] 横山大作：ベイジアンアプローチに基づくモンテカルロ木探索アルゴリズムの将棋への適用，第 18 回ゲームプログラミングワークショップ，pp.58–65 (2013).

[16] Zobrist, A.L.: A new hashing method with application for game playing, *ICCA Journal*, Vol.13, No.2, pp.69–73 (1990).

[17] Baum, E.B. and Smith, W.D.: A Bayesian Approach to Relevance in Game Playing, *Artificial Intelligence*, Vol.97, No.1-2, pp.195–242 (1997).

[18] Junghanns, A.: Are there practical alternatives to alpha-beta in computer chess?, *ICGA Journal*, Vol.21, No.1, pp.14–32 (1998).

[19] Tesauro, G., Rajan, V.T. and Segal, R.: Bayesian Inference in Monte-Carlo Tree Search, *The 26th Conference on Uncertainty in Artificial Intelligence* (*UAI*), pp.580–588 (2010).

横山 大作 （正会員）

2006 年東京大学で博士号取得．博士（科学）．2002 年同大学大学院新領域創成科学研究科助手等を経て，2009 年より同大学生産技術研究所助教，現在に至る．並列プログラミング，分散計算環境，ゲームプログラミングに関する研究に従事．

喜連川 優 （フェロー）

東京大学大学院工学系研究科情報工学博士課程修了（83 年）．工博．現在，国立情報学研究所長，東大生産技術研究所教授，東大地球観測データ統融合連携研究機構長．文科省「情報爆発」特定研究領域代表（05-10），経済産業省「情報大航海」戦略会議委員長（07-09）．データベース工学が専門．ACM SIGMOD Edgar F. Codd Innovation Award 受賞，ACM/IEEE フェロー，電子情報通信学会フェロー，本会会長，内閣府最先端研究開発支援プログラムを推進．