非同期並列ゲーム木探索での効果的な計算ノード割り当て

横山 秀[†1,a)]　金子 知適[†2]

本稿では，性能が低い安価なネットワークで接続された計算機環境を前提としたゲーム木の並列探索において，局面の分担を改善する手法を提案する．局面を分担する並列探索は，マスタがゲーム木を管理し，計算ノードが各々一つの担当局面を非同期に探索し続けることで並列化効率を実現する手法である．担当局面を決定するためにマスタが作成するゲーム木を適切に成長させることが，無駄な探索を避けるために重要である．本稿で提案するゲーム木の成長手法は，ゲーム固有の知識によらず汎用性を持つという点と，予備探索が不要であるため一手 1 秒といった思考時間が短い環境でも性能を出しやすいという点で従来手法より優れている．実際に，チェスを題材に提案手法を実装し，対局実験を行ったところ，計算ノード数 8 以上では逐次探索よりも高性能であることを確認した．

Efficient Assignment of Computation Nodes in Asynchronous Parallel Game-Tree Search

SHU YOKOYAMA[†1,a)]　TOMOYUKI KANEKO[†2]

Asynchronous parallel game-tree search methods are effective ways to improve the playing strength by utilizing many computing nodes connected in low-cost network systems. This paper presents a method of making an improved plan for the assignment of computing nodes. In our framework, the master node manages the game-tree and makes an assignment based on the game-tree. Then, each computing node asynchronously searches the best move and evaluation for a position assigned to the node. To reduce the search overheads, the master's game-tree should be grown appropriately so that a better move has more computing nodes assigned in the corresponding sub game-tree for the move. We present two improvements over existing assignment; one is independence from game-specific knowledge and the other is the efficiency that makes the asynchronous parallel search framework suitable even for short time matches. We applied the proposed method to a top-level chess program, and evaluated the playing strength via self-plays. We confirmed that a program incorporated the presented method plays better than the original one when the number of the computing nodes are greater than or equal to eight.

1．はじめに

チェスのような完全情報ゲームを，min-max 原理に基づく木探索を行ってプレイするプログラムの，非共有メモリ環境での並列化手法について議論する．強いプログラムを実現するためには，ルール上与えられた持ち時間の範囲内でゲーム木をより深く探索することが重要である．一つの計算機のみでは限られた範囲しか探索できないため，深く探索するには複数の計算機を並列に動作させることが有効である．

min-max 木探索の逐次実行では α-β 枝刈りなどの工夫を用いた探索の効率化がなされるが，それらのほとんどは探索順序に依存している．このため並列化の際には，無駄な探索が必然的に生じる．また，計算資源間での情報伝達が必要であることが，特に非共有メモリ環境では性能に影響する要因となる．

GPS 将棋[1]では，高性能のネットワークを必要としない並列探索手法が採用されている．この方式では，ゲーム木の葉となる局面を計算機に割り当てて分担させることによって並列化を行う．

分担を決定するためのゲーム木をうまく構築することで，有用な探索をするワーカを増やすことができる．GPS 将棋ではゲーム木の構築のためにゲーム依存の知識やヒューリスティックを用いたが，本稿ではゲームの種類によらない手法を提案し，チェスへの応用を行った．

2．関連研究

既存の並列化手法である PVSplit[2]や YBWC[3]では，部分的な探索で得られた α-β 窓を残りの部分の探索に利用している．また，TDSAB[4]では局面のハッシュテーブルに基づき探索を分散化している．これらの手法では，無駄な探索の軽減が可能である反面，計算資源間の通信が頻繁に発生する．このため，Ethernet で構築された LAN のよう

†1 東京大学教養学部
College of Arts and Sciences, The University of Tokyo
†2 東京大学総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo
a) yokoyama@graco.c.u-tokyo.ac.jp

な，通信遅延の大きいネットワークで接続された計算機群を用いた非共有メモリの並列探索では，通信オーバーヘッドが大きくなり性能を発揮しにくいと考えられる．

また，APHID[5]は非同期な探索を行うことでタスクの粒度を大きくし，通信オーバーヘッドの影響を軽減するアルゴリズムである．

3．局面を分担する並列探索

本稿では，現局面から着手される可能性の高い局面を一局面ずつ，計算ノード（以下「ワーカ」という．）に割り当てることで並列探索を行う，GPS 将棋で行われている方式[6]について議論する．この並列化手法の効果は，各ワーカが現局面よりも先の局面を起点に探索することで逐次よりも深い手までを探索することができることから生じる．

この手法では，全体を統括するマスタ計算機が，探索開始時にワーカ数ぶんの葉を持つマスタゲーム木を作成し，マスタゲーム木の葉とワーカとを一対一で対応付ける．その後は，各ワーカは割り当てられた葉の局面を独立に探索し，マスタに評価値を随時送出する．受信した評価値を記録したマスタゲーム木を min-max 木として扱うことで，全体の最善手を決定する．例外的な状況を除き，着手決定までマスタゲーム木の形は変更されない．

3.1 複数手の集約

一局面から取りうる全ての合法手を別個のワーカで扱うとすると，浅い段数でも多くのワーカが必要になってしまう．このため，有用性の低いと思われる手は「その他」として集約した葉とし，それらの手からの更なる変化は全てこの葉に割り当てられた単一のワーカで扱うことにする．

例えば，ある局面を A, B, C の 3 台のワーカで分担する状況を考える．この局面で取りうる合法手が 10 あったとして，ワーカ A は最有力の手に，ワーカ B は二番目に有力な手に連なる局面を担当する．ワーカ C は，残った 8 つの手を全て担当することになる．

なお本稿では，「その他」の手を扱う葉を，親局面を表すノードと同一視して考える．つまり，マスタゲーム木の全ての節点にワーカを割り当てると考えたうえで，葉でない節点に割り当てられたワーカは，その節点に対応する局面から取りうる合法手のうち子節点として別のワーカに割り当てられた手以外の全てを担当するものとみる．この考え方を先述の例に適用して示した図が，図 1 である．

3.2 マスタゲーム木の成長と tree pipeline

自他の着手により実際の盤面が進行した際には，マスタゲーム木を新しい現局面が根となるように再構築し，ワーカの担当局面を再割り当てする（図 2）．このとき，新しい現局面や，そこから変化する盤面を探索していたワーカ

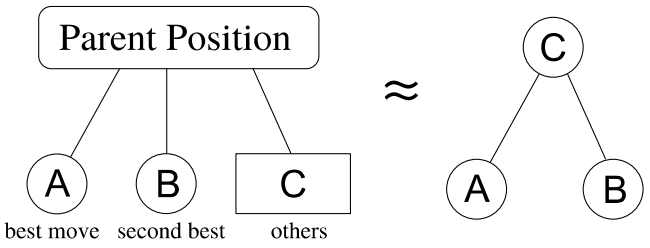

図 1　「その他」葉と親節点の同一視

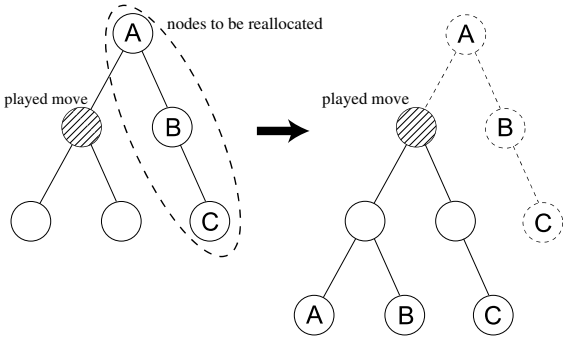

図 2　局面変化時の木の成長

の割り当ては変更せず，そのまま探索を継続させる．これらのワーカは，optimistic pondering の概念と同様に，前の一手ぶんの探索時間が追加された効果を得る．この仕組みは tree pipeline[7]として提唱されている．

一方，実際には指されなかった手に連なる局面を探索していたワーカは，探索を中止し回収される．これらのワーカに新しい局面を割り当てるために，マスタゲーム木を成長させる．

実際に指された局面を担当するワーカは tree pipeline の効果を得ることができるが，指されなかった局面のワーカは回収され，これらの探索結果は以降の探索では不要となる．よって，回収されるワーカがなるべく少なくするために，ワーカ割り当てを行うマスタゲーム木を，実局面で指される可能性の高い有力な手に多くのワーカが割り当てられるような形に構築することが重要である．このためには有力手の候補を，探索を開始する前の，マスタゲーム木を構築して各ワーカが担当する局面を決定する段階で予測する必要がある．

このようなマスタゲーム木の形は，ワーカを回収した後に，これらを再割り当てして木を成長させる処理において，回収したワーカをいずれの節点からの成長に投入するかで決定される．ある節点から成長させるとは，節点が探索している手のうち最も有力な一つの手を新しい節点に分割し，別のワーカに担当させることを意味する．

3.3 GPS 将棋での大規模実証と本稿での改善点

本節で説明した並列探索手法を採用した GPS 将棋が，700 台の計算機を利用して行われた将棋対局でプロ棋士に勝利するなど，大規模環境での実証で成果をあげている．

GPS 将棋では，マスタゲーム木の形を決定する方法として，既存棋譜の解析結果をもとにしたヒューリスティックな手法がとられた[1]．また，有力候補手の予測のためには，前局面までの探索成果に加え，本探索に先立ち広く浅い予備探索を短い時間だけ行った結果や，ゲーム依存の知識に基づき算出した局面の実現確率が利用された[8]．

本稿では，ゲーム木の作成について理論的な背景を考察し，ヒューリスティックやゲーム依存の知識を用いず，予備探索を排することで探索時間を有効活用する手法を提案する．

4．マスタゲーム木成長手法の提案

マスタゲーム木のうち実局面の進行と一致した部分のワーカは，一致する深さまでは局面が進行しても回収されない．このため，有用な手を特に深く読むような形をしたマスタゲーム木を構築することで，早期に回収されるワーカを減らし，有効な探索をワーカに長く続けさせることができる．

ここでは，マスタゲーム木中の成長させる節点を選択するにあたって，成長後のゲーム木において，各節点と実際の局面進行とが一致する最大深さの期待値を最大化する手法を提案する．これにより，有用な手を平均的には最も深く読むような木が構築できる．

4.1 深さの期待値を最大化させる成長方法

ある節点が，局面進行とマスタゲーム木が一致する最深節点になることを，本稿では「節点に到達する」と呼ぶ．いま最大化したい期待値は，到達節点の深さの期待値と言い換えることができる．

3.1 節で述べたように，マスタゲーム木中の葉でない節点は，その節点に対応する局面で取りうる合法手のうち，子節点で独立に探索されていない手による局面のことを表している．局面進行がこのような手と一致したときには，葉でない節点に到達したと考える．なお，このようなある局面からの複数の手を扱う節点については，深さを局面自体の深さと同一視して考えた．このような節点には，多くのケースでは多数の手がまとめられているため，一手深い局面とはいえないためである．

マスタゲーム木の節点の集合を V として，節点 $v \in V$ の深さを d_v，v に到達する確率を p_v とすると，最大化したい期待値はこれらの積の総和であるから，回収済みワーカのゲーム木への割り当ては以下の最大化問題として表現される：

$$\text{maximize} \sum_{v \in V} d_v p_v$$

節点を 1 つ追加して木を成長させることにより得られる期待値の増分は，追加した節点への到達確率と等しい．ある節点から，到達確率が x であるような新たな節点を分割することを考えると，元の節点の到達確率は分割により x だけ減るが，新たな節点（到達確率 x）は元の節点よりも深さが 1 大きいため，全体の期待値の増分は x となる．

局面が進行することで到達確率が増すことはないから，ゲーム木内の任意の節点の到達確率は，その節点を根とする部分木の全ての節点よりも大きい．

ここで，回収したワーカ数ぶんの節点を一つずつマスタゲーム木に追加してゆくと考えると，毎回の追加において深さの期待値が最大となるようにすることで最大化問題の解を得ることができる．つまり，分割により作成可能な節点のなかで到達確率が最大となるものを，次々に作成してゆけばよい．

この証明は背理法による．この提案手法を用いて成長させることで作成された木を T とする．同じ元の木から，T の成長と同じワーカ数を用いて成長させた，T と異なる木 $T'(|T| = |T'|)$ が最大の期待値 $\sum d_v p_v$ を与えたと仮定する．提案手法で作成した T の節点を作成順に調べ，T' に存在しない節点のうち初めに見つかったものを v^* と呼ぶ（T と T' は異なる木なので，このような v^* は必ず存在する）．木 T' で v^* の代わりに作成された節点を根とする部分木内の節点は，到達確率が v^* よりも低い．そのうちの一つを回収して v^* を成長させることで期待値は大きくなるが，ここで元の木 T' は最大の期待値を与える木を仮定しており，背理法の仮定に反する．

4.2 局面進行との一致確率の推定

提案手法を実験するにあたって，ゲーム木の節点までの手順と実際の局面進行とが一致する確率（到達確率）は，マスタゲーム木での兄弟間の評価値順位に基づき算出した．逐次探索と並列環境では評価値順位と実現確率の関係は変わらず，また全ての局面で同一と仮定し，各手番での評価値順位に対応する確率を掛けあわせた値を，到達確率とした．

逐次探索での評価値順位と実現確率の関係を調べる予備実験を，オープンソースのチェスプログラム Stockfish DD[*1]を用いて行った．これは，後述する対局実験でワーカプログラムとして採用したものと同一である．既存棋譜に現れる局面それぞれからの探索を，以下の Stockfish へのコマンドにより実行した：

```
setoption name MultiPV value 10
go depth 15
```

ここでは，手の候補を第 10 位まで求めるように設定したうえで，15 手先までの深さの木を探索した．この深さは，最善手のみを得る探索であれば 1 秒程度で実行可能な

[*1] https://s3.amazonaws.com/stockfish/stockfish-dd-src.zip 2013 年 11 月 30 日公開版

表 1　評価順位と実現確率

順位	局面数	割合	GPS 将棋
1	597	0.5472	0.4821
2	193	0.1769	0.1817
3	96	0.0880	0.0886
4	57	0.0522	0.0573
5	32	0.0293	0.0394
6	27	0.0247	0.0297
7	23	0.0211	0.0174
8	14	0.0128	0.0145
9	9	0.0082	0.0090
10	12	0.0110	0.0081
11 位以下	31	0.0284	--

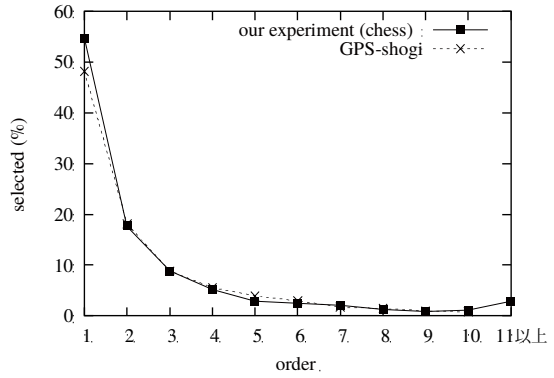

図 3　評価順位と実現確率の関係

ものとして選んだ．

確率計測に用いた既存棋譜は，1996-1997 年に行われた Deep Blue 対 ガルリ・カスパロフ の全 12 試合 1,091 局面を利用した．ワーカプログラムの癖の影響を排除するため，Stockfish 以外の棋譜，特に着手の多くが真の最善手であることが期待できる強力なプレイヤーの棋譜を選んだ．

この結果，棋譜中の局面をワーカプログラムが第何位と評価したかをまとめたのが表 1 および図 3 である．上位 10 位までを探索した結果に列挙されなかった手は，「11 位以下」としてまとめた．また，GPS 将棋が他プログラムの棋譜を用いて，1 秒の探索で同様の実験を行った[8]結果を併せて掲載した．

なお単調性を確保し下位での確率を定義するため，実際の木の成長では，9 位までは予備実験で得られた数値をそのまま利用し，n 位 $(n \geq 10)$ の確率を $(n+1)^2 \,/\, 1.7827$ とした．

4.3　シミュレーション

提案手法を用い，マスタゲーム木の形をシミュレーションした．N=8 のときの結果を図 5，N=32 を図 6 に示す．なお図中の節点内の数字は，上段が兄弟節点内での評価値順位を，下段が節点への到達確率を表す．ワーカ数 N の木は，ある局面に対して，N-1 個のワーカを投入して成長させたものにあたる．

GPS 将棋では，評価順位と実現確率の関係は前節の予備実験と似た結果となっていたが，このことから手の分割幅は 2 程度が適していると予想されていた[9]．これと比較すると本シミュレーションでは，ワーカ数 N が大きい場合において，浅い局面の分割幅がより広くなっている．

5．有力手予測手法の提案

マスタゲーム木を成長させるにあたっては，ある局面を重点的に探索するために，一つの節点から次善手やその子を含めた複数の子を成長させることがあるため，次善手以下の有用性を判定する必要がある．

最善手は前局面までの探索までで既に得られ，マスタゲーム木に反映されているが，次善手を探索結果から得ることはできない．前局面までに行われてきた，一ワーカ内での min-max 原理に基づく局所的な探索では，次善手以下は最善手より不利な手であることが確定した段階で α-β 法により枝刈りされるため，評価値が正しく得られておらず，他の手との間で順位付けできない．次善手以下の評価値を正しく得る探索を行うと，最善手のみを探索する場合に比して α-β 法で枝刈りできる局面が減り，探索効率が下がる．このような，次善手以下も確定させるための低効率な探索が，GPS 将棋では予備探索として 1 秒間行われていた．

そこで本稿では，現局面までの探索で得られた，各ワーカ内に蓄積されているゲーム木の深さを利用することで，予備探索を行わずに，通常の探索で得られた情報のみをもとに有力手の予測を行う手法を提案する．

次善手以下の有用性は，局面予測に用いるには近似的に評価できれば十分であるため，本手法では探索効率を下げることなく得られる指標として，一ワーカ内の局面表に保存された探索木の深さを利用した．ワーカ内の探索木は，ワーカの担当局面をルートとした局所的な探索で得られた min-max 木である．有用でない手は浅い探索で枝刈りされ，枝刈りされずに深くまで探索された手ほど有用である可能性が高いと予想される．

探索木全体から，探索木の深さの大きい順に節点を取り出したものは，探索木と同じ根を持つ部分木となる．

5.1　作成された木

提案手法により得られる木のサンプルとして，以下の Stockfish コマンドを実行することで初期局面を根として一千万節点を探索した後に，探索木の中から，探索深さが上位である節点 32 個を抽出した．

```
position startpos moves
go nodes 10000000
```

結果は図 7 の通りとなった．節点の上段は深さの値，下段は駒の動きを表している．

6．チェスでの対局実験

第 4 節および第 5 節で提案した手法の効果を確認するため，これらを実装したチェスプログラムで対戦を行った．

マスタゲーム木を再構築する動作過程を，図 4 に示した．マスタゲーム木の成長では，まず第 4 節の手法により，成長前のゲーム木中の節点のうち，どの節点に何台のワーカを割り当てるかを決定した．その後，ワーカ内探索木内の局面のうち探索深さが上位のものを，そのワーカの節点に割り当てた台数ぶんまで取得し，これをそのまま節点以下の部分木の形として採用した．よって，第 4 節で求めた木の形が一つの節点から 2 つ以上の節点を成長させるものであった場合，求めた木の形と実際の木は，このような成長節点以下の部分木においては必ずしも同一にはならない．第 4 節での確率計算は合法手の数といった局面に固有の情報を使用していないため，ゲーム木の末端部分の形を決める際には，第 5 節で提案したワーカ内の局所的なゲーム木の利用が有効であると予想される．

ワーカプログラムには，1 スレッドで逐次探索を行う Stockfish を採用した（4.2 節と同一のもの）．Stockfish は標準入出力を通じ，テキストベースの UCI (Universal Chess Interface) プロトコル[*2]を用いて制御するチェス思考エンジンであり，Netcat を用いて標準入出力をネットワーク通信に振り替えることでマスタとの通信を行う．また，探索する手を限定する `searchmoves` 機能を備え，複数の手を集約した節点の探索に対応する．このように Stockfish には，大規模な改造を加えることなくワーカプログラムに転用できる利点がある．

第 5 節での提案手法に用いるため，Stockfish には探索した深さが上位 32 位までの節点を，最善手の探索結果とともに定期的に出力するよう改造を施した．

6.1 対戦相手

本手法の並列化効果を逐次実行の場合と比較するために，同等のハードウェア上で逐次実行する，ワーカプロセスで用いるものと同じ Stockfish を対局相手とした．ただし，本提案手法では相手の思考時間中も探索を継続する pondering を行っていることから，のべ計算時間の両者比をワーカ数の比と同じにするために，対戦相手のプログラムも pondering を行うようにコマンドを発行した．

本実験では，相手が思考している時間じゅう，現在の盤面，つまり直前の自分の着手を実施した後の局面を起点とした探索を行わせる．この探索により次節で述べるハッシュテーブルが更新され，これを次回の自分の探索時に引き継ぐことで pondering の効果が得られる．相手が指しうる手全てを考慮に入れた pondering を行う点が，相手の指す手候補を 1 つに絞った探索をする optimistic pondering と異なる．

[*2] http://wbec-ridderkerk.nl/html/UCIProtocol.html

- subtree(v) は v の部分木（v を含む）
- v^+ は，v から成長する節点

マスタゲーム木の節点集合：V
着手された手の節点：v_m
回収する節点 $V_{\mathrm{del}} \leftarrow V \setminus \mathrm{subtree}(v_m)$
$V \leftarrow \mathrm{subtree}(v_m)$
仮の木 $V_{\mathrm{temp}} \leftarrow V$
for each ($v \in \mathrm{V}$) {
　v の成長に費やすワーカ数 $w_v \leftarrow 0$
}

repeat $|V_{\mathrm{del}}|$ times {
　$v_g \leftarrow \mathrm{argmax}_{v \in V_{\mathrm{temp}}}$ 到達確率(v^+)
　$V_{\mathrm{temp}} \leftarrow V_{\mathrm{temp}} \cup \{v_g\}$
　$v_{\mathrm{a}} \leftarrow V$ のうち，v_gの直近祖先
　$w_{v_a} \leftarrow w_{v_a} + 1$
}

for each ($v_r \in V$) {
　if ($w_{v_r} > 0$) {
　　$V_{\mathrm{grow}} \leftarrow v_r$を担当するワーカの探索木から，探索深さが上位$w_{v_r}$までの節点を取得
　　$V \leftarrow V \cup V_{\mathrm{grow}}$
　}
}

図 4　マスタゲーム木成長の処理内容

6.2 Stockfish のハッシュテーブル

ワーカプロセスおよび対戦相手の Stockfish は，探索結果をメモリ上のハッシュテーブルに保存し，次の手以降の探索でも活用する．前の局面までの探索木は，現局面からの探索木を部分木として含むことが期待されるので，ハッシュテーブルの情報を用いることで，α-β 窓決定のための反復深化探索のうち，最初のいくつかの深さ制限探索を省略できることが考えられる．

ハッシュテーブルの容量は設定で変更可能だが，本稿の実験では 32MiB に固定した．なお，局面のエントリ一つのサイズは 16 バイトであるから，最大約 210 万局面を格納できる．

ハッシュテーブル保持の効果を確認するため，ハッシュテーブルを保存する仕様そのままのプログラムと，一手ごとに内部のハッシュテーブルを消去するように改造したプログラムとを対戦させた．なお対戦では Intel Xeon X5690

を双方のプログラムに 1 コアずつ用いた．双方とも一手あたりの思考時間を 1 秒に固定し，pondering は行っていない．

この結果，未改造のプログラムは 297 勝 42 敗 565 分（全 904 試合，勝率[*3]64.1%）の成績をあげた．これは，前の手までに探索された結果を利用したことによる効果と考えられる．

6.3 対局実験の実施条件

マスタゲーム木を管理するマスタプログラムを，TCP サーバーとして作成した．このプログラムは，ワーカに割り当てる局面を指示し，またワーカから評価値の情報を受け取ってマスタゲーム木に反映させ，最終的な指し手を決定する役目を担う．マスタプログラムとワーカプログラムは Ethernet で接続されている別個の計算機で動作させ，SSH トンネリングによる TCP/IP 通信を行った．この通信路で短いメッセージをやりとりする際に要したラウンドトリップタイムは 1ms ほどだった．

ワーカプログラムのための均等な計算機を多数確保することが困難だったため，一機のワークステーション（Intel Xeon E5-4650 32 コア）に，CPU コア数を上回らない数の逐次探索プロセスを作成し，これらをワーカノードとしてそれぞれを独立でマスタと通信をさせることで，疑似的な分散計算機環境とした．

対局全体をチェスアプリケーションの Xboard[*4]が統括し，勝敗の判定や棋譜の記録を行う．マスタプログラムおよび対戦相手となる Stockfish は盤面の情報や着手を UCI プロトコルで入出力し，Xboard との情報交換にはチェスプロトコル変換プログラムである PolyGlot 1.4w29[*5]を介した．

複数試合を行い勝率を測る際，常に同じ初期盤面から探索を行うと最終的な対局結果が同一になる可能性があるため，ゲームのオープニング（序盤）は双方とも探索ではなく，Marc Lacrosse により作成された定跡ファイル performance.bin[*6]に記録された定跡を，PolyGlot の機能でランダムに指すようにした．

6.4 ワーカ数による強さの変化

提案手法の並列化効果を確認するため，逐次実行の対戦相手に対してワーカ数を変化させ，対局での勝率を調べた．一手あたりの時間は 1 秒に固定した．この時間は，マスタ-ワーカ間の通信に要する時間を含む．

表 2　逐次プログラムとの対戦成績

ワーカ数	勝	負	分	計	勝率 [%]
4	83	128	497	708	46.8
8	73	71	357	501	51.0
16	89	41	275	405	55.9
32	63	26	163	252	57.3

この結果，表 2 に示す対戦成績を得た．ワーカ数 32 までの範囲では，ワーカ数を増やすことによって勝率が上がったことが確認できた．

しかしながら，ワーカ数 8 で逐次実行と同程度の強さであり，これ以下のワーカ数では並列化の効果が得られていないことが分かった．本方式での並列探索では，有力な局面は新しいワーカに割り当てられるが，局面を割り当てられた直後のワーカはそれまでの探索のハッシュテーブルを利用できない．ハッシュテーブルの利用が有効であることは，6.2 節の予備実験で確かめた．総ノード数の少ない並列探索ではマスタノード木が浅くなるため，このデメリットが大きく表れたことが考えられる．

7．おわりに

局面分担による並列探索について，分担する局面を決定するためのマスタゲーム木を適切に構築し，計算ノードに探索を効果的に分配する方法を提案したうえで，これを用いてチェスをプレイするシステムを実装した．提案手法はゲーム依存の知識や試行錯誤によるヒューリスティックな調整が必要ないことから，min-max 原理により探索できる多くのゲームに対応できると考えられる．ゲームの種別によるマスタゲーム木の形の差異や，将棋に適用したときの GPS 将棋との性能の優劣については，検討すべき課題である．

実装したチェスシステムは，16 並列では逐次探索に対して大幅に勝ち越す（勝率は 55.9%）など，並列化の効果を実証できた．特に，GPS 将棋では予備探索に費やしていた時間に等しい一手 1 秒の環境での並列化に成功したのは，大きな成果である．

一方で，計算ノード数が 4 と少ない場合の性能は逐次より劣った．探索開始時のハッシュテーブル情報の有無が性能に大きな影響を与えるため，ワーカ間でハッシュテーブルの一部を共有することで改善が期待される．共有に要する通信オーバーヘッドとのトレードオフを解析し，ハッシュテーブルの情報を適切に選別する必要があるだろう．

[*3] 勝率の計算では，チェスで一般的な計算方法に従い，引き分けを 0.5 勝として算入した．以降の実験も同じ．

[*4] http://www.gnu.org/software/xboard/

[*5] http://www.geenvis.net/pg.html 2012 年 1 月 21 日公開版

[*6] http://wbec-ridderkerk.nl/html/download.htm

参　考　文　献

[1] 金子 知適, 田中 哲朗：多数の計算機を活用したゲーム木探索技術の進歩 —三浦弘行八段と GPS 将棋との対局を振り返って—, 情報処理 Vol. 54, No. 9, pp. 914–922 (2013).

[2] Marsland, T.A., Campbell, M.: Parallel Search of Strongly Ordered Game Trees, *ACM Computing Surveys*, Vol. 14, No. 4, pp. 533–551 (1982).

[3] Feldmann, R.: Game Tree Search on Massively Parallel Systems, Ph.D. Thesis, University of Paderborn (1993).

[4] Kishimoto, A: Transposition Table Driven Scheduling for Two-Player Games, M.Sc. Thesis, University of Alberta (2002).

[5] Brockington, M.G., Schaeffer J.: APHID Game-Tree Search, Journal of Parallel and Distributed Computing, Vol. 6, pp. 90–114 (1997).

[6] 田中 哲朗, 金子 知適：将棋プログラムの大規模並列実行, 情報処理学会研究報告, Vol. 2010-GI-24, No.2, pp. 1–8 (2010)

[7] Himstedt, K. : GridChess : Combining Optimistic Pondering with the Young Brothers Wait Concept, ICGA Journal, Vol.35, No.2, pp. 67–79 (2012).

[8] 金子 知適, 田中 哲朗：最善手の予測に基づくゲーム木探索の分散並列実行, 第 15 回ゲームプログラミングワークショップ, pp. 126–133 (2010).

[9] 金子 知適, 田中 哲朗：GPS 将棋とテキストプロトコルによる大規模将棋ソフトウェアの組み立て, コンピュータソフトウェア, Vol.29, No.1, pp. 75–81 (2012)

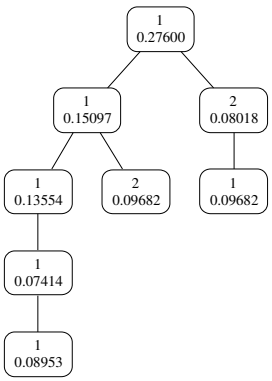

図 5　N=8 のシミュレーション木
（深さ期待値 1.470）

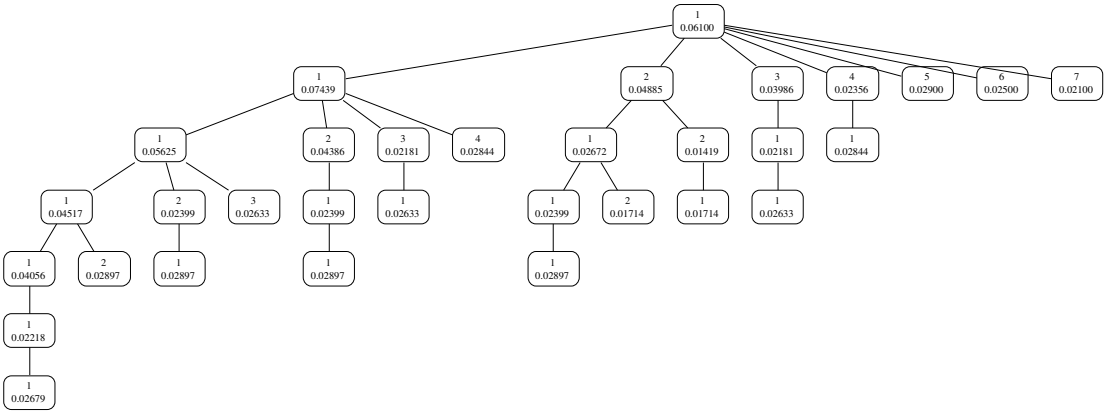

図 6　N=32 のシミュレーション木（深さ期待値 2.333）

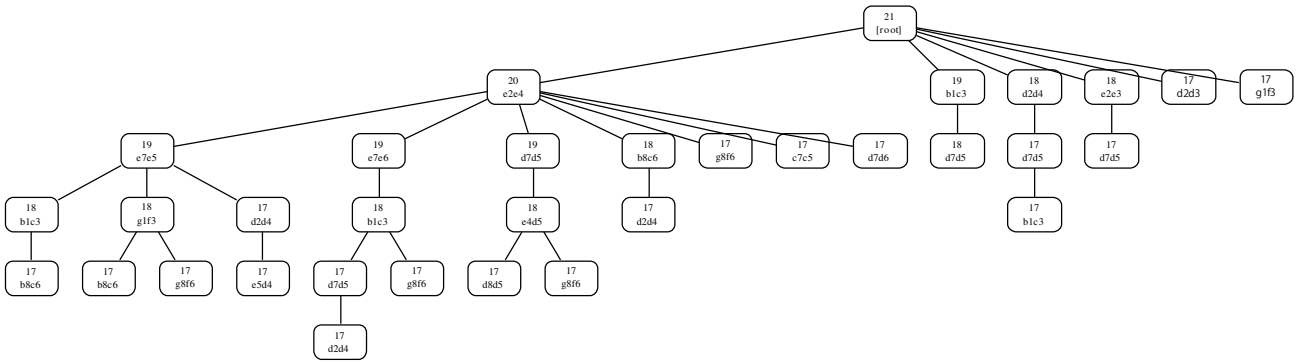

図 7　ワーカ内の探索木に基づく木（N=32，初期配置から）