疑似乱数を利用した df-pn 探索の簡易分散並列計算

保木邦仁* 伊藤毅志**
*電気通信大学先端領域教育研究センター **電気通信大学情報・通信工学科

疑似乱数を利用して、単一の既存 df-pn 探索プログラムから複数の異なるプログラムを容易に生成する手法を提案する。この手法により構成されたクライアント集合により分散並列化を行い、既存詰将棋プログラムが高速化された。生成されたクライアント集合の有効性が、GPS 将棋、Bonanza、岸本の詰将棋プログラムなど 3 つの異なる実装において確認された。

Distributed computing of df-pn search using pseudo-random number

Kunihito Hoki* and Takeshi Ito**
*Center for Frontier Science and Engineering, the University of Electro-Communications
*Department of Information and Communication Engineering, the University of Electro-Communications

This present a simple method that generates multiple different df-pn searchers from an existing searcher using pseudo-random number. Distributed computing using client sets generated by this method showed improvements in performance for solving Tsume-Shogi problems. The validity of the ensemble-generation method is confirmed by three different Tsume-Shogi programs, GPS Shogi, Bonanza, and Kishimoto's df-pn search programs.

はじめに 近年、将棋における疎結合型並列計算法の一種として乱数合議法が提案された [1]。これは、将棋プログラムの評価関数に乱数値を足すことにより複数プログラムを生成し、各プログラムの探索結果を集計して多数決などを行う手法である。この手法の仕組みは非常に簡単なものであり、プログラム間の高速な通信が原理的には必要とされない。文献によると、この手法により非並列プログラムに対して 55%から 60%程度の勝率が得られたと報告されている。また、構成されるシステムの安定性もこの手法の特徴となっている。将棋プログラムの他にも、$\alpha\beta$法をベースとした探索を行うチェスプログラムにおいてこの乱数合議法の効果が検証されている [2]。また、モンテカルロ探索を行う囲碁プログラムやトランプの大貧民においても多数決による合議法が有効であることが報告されている [3,4]。

合議法はゲーム木探索の分散並列化の 1 手法であり、$\alpha\beta$探索を分散並列化により効率化する研究は他にも多数行われている。たとえば、Brockington らの APHID [5]、岸本らの TDSAB [6]、Himstedt らの拡張予測読みに基づく手法 [7]、金子らの最善手の予測に基づく手法 [8]、投機を用いた手法が挙げられる [9]。

本研究は、乱数合議法の考えを df-pn 探索を行う詰将棋プログラムに応用できるのではないかという推論から始まった。Df-pn 探索は詰将棋プログラムで広く採用されている手法で、AND/OR 木の証明数探索を深さ優先型の探索アルゴリズムに書き換えたものである [10, 11]。現在、難題と呼ばれる詰将棋問題はほぼ解かれている [11]。それでもなお、将棋の対局においては詰将棋を高速に解くことに対する需要はあり、分散並列計算は有用であると考えられる。

囲碁・将棋プログラムと詰将棋プログラムには次の点において違いがある。囲碁・将棋では、一般に最善手を発見することは困難であり、プログラムの優劣判断も曖昧なものとなる。一方、詰将棋では詰み・不詰みの明確な結果を発見することが計算の目的となる。正解手を発見する問題という意味で、詰将棋はパズルゲームの一種と考えることができる。パズルゲームにおけるシングルエージェント探索では、異なるパラメタセットを用いたプログラムを同時実行する Dovetailing と呼ばれる方法が研究されている [12, 13]。本研究はこのような枠組みの一手法と捉えることができる。

本研究では疎結合型並列計算を目指すが、メモリ共有探索に関心のある方は文献[14]を参考にされたい。この文献では、8 並列で 3.6 倍の高速化が報告されている。また、ラインズオブアクションの df-pn 探索分散並列化の研究も報告されている [15]。この手法では、乱数を利

用しているという点において本手法と類似しているが、メモリ共有探索を念頭に置いている点が異なっている。また、df-pn 探索ではないが、AND/OR 木探索における分散並列化の研究も報告されている [16, 17]。

証明数と反証数　Df-pn 詰将棋探索においては証明数 (pn) 及び反証数 (dn) が重要な役割を果たす [18]。各節点において、これらの数は次の 5 つのルールにより決定される。①. 終端 OR 節点の場合（王手がかからない不詰み局面）では pn=∞、dn=0。②. 終端 AND 節点の場合（王手回避のない詰み局面）pn=0、dn=∞。③. 末端節点の場合 pn=1、dn=1。④. 内部 OR 節点の場合 pn=min pn_i、dn= $\sum_i dn_i$ ⑤. 内部 AND 節点の場合 pn= $\sum_i pn_i$ 、dn=min dn_i。但し、pn_i や dn_i は i 番目の子節点の証明数と反証数である。④番目のルール適用例を図 1 に示す。OR 節点においては証明数が最小の子節点（図 1 中の a）を、また AND 節点においては反証数が最小の子節点が選択される。これらのルールを再帰的に適用して到達する末端節点は most proving 節点と呼ばれる。証明数や反証数は、問題を解くために新規展開が必要な節点数の理論的な下限である。一般に、これらの下限値は問題の解きやすさに関する良い指針であると期待される。しかし、これらの下限値と実際に要する展開節点数には直接的な関係はない。従って、図 1 に示された例では、子節点 a と b の証明数の差は僅かであり、子節点 b 以下を展開しても良さそうである。実際に①~⑤の展開節点の選択則に修正を加えて計算効率を改善する方法は並列及び逐次探索両方で報告されている [14, 15, 19, 20]。

提案手法　図 1 での子節点 a と b それぞれをを異なるプログラムに、プログラム間の通信を行うことなく（局面表を共有することなく）選択させることを目指す。各プログラムで証明数及び反証数の数え方が適度に異なるクライアント集合構成のため、上述のルール③を以下のものに変更する。

③a 以前の反復深化にて訪れたことのある末端節点では前回と同じ値を pn と dn に割り当てる。

③b 新規末端節点の場合、疑似乱数により末端初期値を変更

確率 p で pn の初期値に 1 を足す。dn は変更無し。

確率 p で dn の初期値に 1 を足す。pn は変更無し。

確率 $1-2p$ で pn と dn 両方変更無し。

分散並列化はマスタースレーブモデルにより行い、異なる疑似乱数系列を利用した複数のスレーブを同時に走らせて、マスターとルート局面及び探索結果を通信

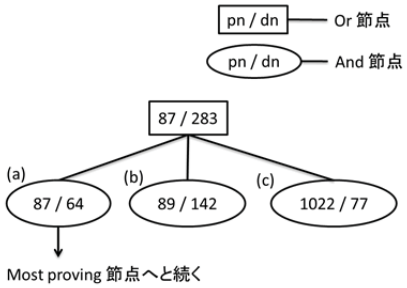

図 1　AND / OR 木における証明数（pn）と反証数（dn）概念図 OR 節点には 3 つの子節点があり、前回の反復計算でこれらのノードを訪れた際の証明数、反証数の結果が図中 a, b, c のように得られたとする。このような場合には、この OR 節点の証明数は最小値である a の 87、反証数は 64 + 142 + 77 = 283 となる。

する。マスターはクライアントの 1 つから結果を受け取り次第、これを出力する。また、スレーブの一つにオリジナルプログラムを使用することにより、通信不通や通信処理に要する時間を無視して、オリジナルに対する性能劣化を防ぐことが可能となる。同一末端局面にて同じ値を割り当てる仕組みは、Zoblist Hashing を行うプログラムではこのハッシュキーを用いて容易に実装可能である。

実験方法　実戦局面として 1970 年から 2009 年の名人戦 460 局から手順前後は考慮せず同一局面を除外した 25,020 局面、詰将棋問題として将棋図巧と将棋無双から不詰み作品を除外した 193 図面を用意した。

Df-pn 詰将棋プログラムには、ソースコードが公開されている GPS 将棋と Bonanza を使用した [21, 22]。これらの実装には大な違いがあり、2 プログラムで実験を行うことは手法の汎用性を示す一つの指針になると期待できる。GPS 将棋の df-pn 詰将棋は本格的なものであり、DFPN+や岸本の GHI 対策が実装されている [14, 23]。一方、Bonanza の df-pn 詰将棋は実戦で役に立てばよいという簡易的なものであり、GHI 問題対策には長井の∞-1 の閾値を用いる手法を模倣しているが [10]、GHI 問題を回避しきれないことが指摘されている。さらに、歩角飛と 2(8)段目の先手（後手）香車不成りは生成しない。背尾の証明駒 [24]、固定深さの探索 [20] は両プログラムで実装されている。Bonanza においては香車をなる手を不正に生成する不具合が知られていて、これを修正したものを用いた。簡単のため、以後この修正を施したものをオリジナルの Bonanza と呼ぶ。p の値は、十分探索節点数にばらつきを与える小さい値を予備実験により調べて設定した。それぞれ、GPS 将棋では 0.05、

Bonanza では 0.02 である。

詰将棋問題においては GPS 将棋に加えて岸本の詰将棋プログラムの結果を示す [25]。ソースコードは手に入らないが、岸本章宏氏（東工大）の厚意により実験結果を得た。この実験では実装されている末端評価関数値に 1 を足す方法は適切ではないと考え、末端評価関数の初期値に 2 を掛けるよう③b を修正し、p の値には 0.02 を使用した。

提案手法により生成されたクライアント集合の性能を測る指針として、

$\chi = 1 -$ クライアント集合の最小探索節点数 / オリジナル探索節点数

を使用する。本実験ではクライアントの一つにオリジナルを使用した。これにより、χ は 0 から 1 までの値を取り、大きいほど集合としての性能が良いとみなす。

実験結果　図 2 に実戦局面における GPS 将棋と Bonanza の 6 クライアントによるχ値の散布図を示す。実験に使用した Intel Xeon 5680 の下で 1×10^7 節点以上探索した 10 局面より逐次探索の毎秒探索節点数の平均値を求めると、両プログラムとも 30 万程度であった。1×10^8 を探索節点数の上限とした。各クライアント使用メモリは GPS 将棋が 2.5GByte、Bonanza が 1.5GByte 程度である。この制限内で解けた場合は全クライアント全局面で詰み・不詰みの結果は一致していた。χ値が 0.001 以下のデータは示されていないが、GPS 将棋の 10%の詰みと 17%の不詰み、Bonanza の 18%の詰みと 32%の不詰み局面がχ値 0.001 以下に該当する。GPS 将棋では DFPN+の実装により末端節点初期証明・反証数が 1 ではない。また、用いた p 値も異なっている。従って単純な比較はできないが、この結果を見る限りでは GPS 将棋の方が、提案手法が効果的に働くと考えられる。詰む局面でχ値がほぼ 0.1 以上になっている点に着目されたい。

表 1 に、Bonanza を用いた 10 クライアント集合による分散並列計算の結果を示す。局面にはオリジナルプログラムが 1 秒から 100 秒程度で詰みと判定可能な局面を選んだ。分散計算は同室内の 1000BASE-T イーサネット接続による LAN 環境により行った。実際の計測時間により算出された時間短縮率とχ値に一定の相関がみられる。

表 2 は図巧・無双の図面の内、並列化が有用と思われる探索節点数上位 10 局面での結果を示す。オリジナルプログラムを含むクライアント数は 8 とした。GPS 将棋と岸本の詰将棋プログラム両方において、半数以上の

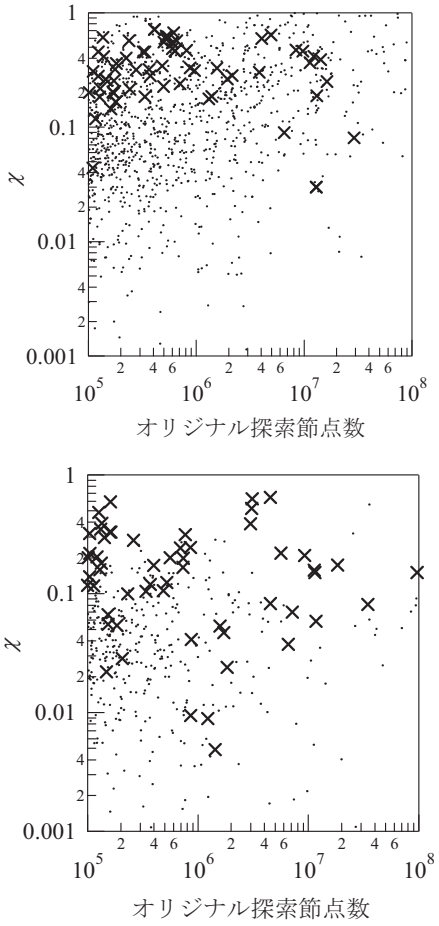

図 2（上）GPS 将棋と（下）Bonanza の探索節点数と 6 クライアントのχ値。x は詰み、・は不詰み。

表 1　Bonanza を用いた分散並列計算。探索節点数には 1,000 で割った。時間は 3 回測定の平均値。

	オリジナル 時間 (s)	節点数	10並列 時間 (s)	時間短縮 率 (%)	χ
#1	94.0	33,785	91.5	2.6	0.061
#2	57.8	23,461	59.9	-3.7	0.000
#3	47.6	18,444	40.3	15.2	0.167
#4	29.5	11,303	15.6	47.2	0.438
#5	14.8	5,640	8.1	45.3	0.446
#6	8.3	3,095	3.1	62.6	0.639
#7	3.8	1,835	3.7	2.4	0.054
#8	3.5	1,580	3.3	5.1	0.072
#9	1.9	809	1.9	1.4	0.092
#10	1.9	775	1.6	17.7	0.167

χ値が 0.2 以上になっている様子が示されている。特にオリジナルプログラムの探索節点数が最も多い図巧 8 番でのχ値は 0.7 以上である。これは、プログラムの調整が上手く働いていない局面に対して提案手法が上手くいくという可能性を示唆している。

まとめ　疑似乱数を利用して、単一の df-pn 探索プログ

表２（上）GPS 将棋と（下）岸本の詰将棋プログラムの探索節点数とオリジナル＋7 クライアントのχ値。探索節点数は 1,000 で割った値を示した。

		オリジナル探索節点数	節点数最小値	χ
図巧	8番	62,351	13,559	0.783
図巧	99番	8,033	5,310	0.339
無双	48番	6,326	5,770	0.088
無双	30番	6,186	4,791	0.225
図巧	1番	3,973	3,940	0.008
図巧	91番	3,817	2,987	0.217
図巧	9番	2,902	2,669	0.080
無双	28番	2,621	1,633	0.377
無双	94番	2,612	864	0.669
図巧	90番	2,315	2,157	0.069

		オリジナル探索節点数	節点数最小値	χ
図巧	8番	1,400,706	109,574	0.922
無双	48番	63,943	48,393	0.243
無双	100番	6,810	3,965	0.418
無双	1番	4,029	3,166	0.214
図巧	1番	3,495	3,495	0.000
図巧	23番	2,915	1,079	0.630
無双	51番	2,074	2,065	0.005
図巧	37番	1,567	880	0.438
無双	46番	1,328	846	0.363
図巧	90番	930	275	0.704

ラムから複数の異なるプログラムを容易に生成する手法を提案した。この手法により構成されたクライアント集合により分散並列化を行い、既存詰将棋プログラムが高速化されることが示された。生成されたクライアント集合の有効性をχ値により計測した。実戦の詰将棋においては GPS 将棋と Bonanza、作品としての詰将棋においては GPS 将棋と岸本の詰将棋プログラムにおいて有効性が確認された。

謝辞　GPS 将棋の使用法や実験遂行に関して助言して下さった金子知適氏、実験結果を提供し実験遂行に関して助言して下さった岸本章宏氏に謝意を表します。また、実験遂行に関して有用な議論をして下さった長井歩氏に謝意を表します。

1 小幡拓弥、保木邦仁、伊藤毅志、ゲームプログラミングワークショップ 2010、箱根、2010 年 5 月。
2 大森誠也、保木邦仁，伊藤毅志、チェスプログラムを用いた合議アルゴリズムの効果の検証、第 26 回ゲーム情報学研究会、松江、2011 年 7 月。
3 Y. Soejima, A. Kishimoto, O. Watanabe, Evaluating Root Parallelization in Go, IEEE Trans. Computational Intelligence and AI in Games, 2, pp. 278–287, 2010.
4 小沼啓、西野哲朗、コンピュータ大貧民に対するモンテカルロ法の適用、第 25 回ゲーム情報学研究会、東京、2011 年 3 月。
5 M.G. Brockington and J. Schaeffer, The APHID Parallel alpha-beta Search Algorithm, IEEE Symposium of Parallel and Distributed Processing (SPDP'96), New Orleans, Oct. 23–26 (1996).
6 A. Kishimoto and J. Schaeffer. Distributed Game-Tree Search Using Transposition Table Driven Work Scheduling, In Proc. of 31st International Conference on Parallel Processing (ICPP'02), pages 323-330, IEEE Computer Society Press, 2002.
7 K. Himstedt, U. Lorenz, and D.P.F. Möller, A Twofold Distributed Game-Tree Search Approach Using Interconnected Clusters, Euro-Par, 587–598 (2008)
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 2010、箱根、2010 年 11 月。
9 浦　晃、横山大作、近山　隆、投機を用いた並列ゲーム木探索の効率化、第 15 回ゲームプログラミングワークショップ 2010、箱根、2010 年 11 月。
10 長井歩、今井浩、Df-pn アルゴリズムの詰将棋を解くプログラムへの応用、情報処理学会論文誌、43, pp. 1769–1777, 2002.
11 岸本章宏、詰将棋を解くための探索技術について、人工知能学会誌、26、pp. 392–397、2011。
12 K. Knight, Are Many Reactive Agents Better Than a Few Deliberative Ones?, In IJCAI, pp. 432-437, 1993
13 R. Valenzano, J. Schaeffer, N. Sturtevant, K. Buro, and A. Kishimoto. Simultaneously Searching with Multiple Settings: An Alternative to Parameter Tuning for Suboptimal Single-Agent Search Algorithms. In Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'2010), pages 177-184, 2010.
14 T. Kaneko Parallel Depth First Proof Number Search, Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10), pp. 95–100, 2010.
15 J-T. Saito, M.H.M. Winands, and H.J. van den Herik, Randomized Parallel Proof-Number Search. In Advances in Computer Games (ACG 2009). Lecture Notes in Computer Science (LNCS 6048), pp. 75-87, Springer, 2010.
16 Y. Nakayama, T. Akazawa and K. Noshita, A Parallel Algorithm for Solving Hard Tsume-Shogi Problems, 19, pp. 94–99, 1996.
17 A. Kishimoto and Y. Kotani, Parallel AND/OR tree search based on proof and disproof numbers, In the 5th Games Programming Workshop, Japan, 1999.
18 L.V. Allis, M. van der Meulen, and H.J. van den Herik, Proof-Number Search, Artificial Intelligence, 66, pp. 91–124, 1994.
19 A. Nagai and H. Imai, Application of df-pn+ to Othello Endgames, In the 5th Games Programming Workshop, Japan, 1999.
20 金子知適、田中哲朗、山口和紀、川合慧、新規節点で固定深さの探索を併用する df-pn アルゴリズム、第 10 回ゲームプログラミングワークショップ、箱根、2005 年 11 月。
21 Team GPS, GPS Shogi, OSL revision 4373, url: http://gps.tanaka.ecc.u-tokyo.ac.jp/gpsshogi/ (last access: Sept. 2011).
22 K. Hoki, Bonanza v6.0, url: http://www.geocities.jp/bonanza_shogi/ (last access: May 2011).
23 A. Kishimoto and M. Müller, A Solution to the GHI Problem for Depth-First Proof-Number Search, Information Sciences, 175, pp. 296–314, 2005.
24 脊尾昌宏、詰将棋プログラムを解くアルゴリズムにおける優越関係の効率的な利用について、In the 5th Games Programming Workshop, Japan, 1999.
25 A. Kishimoto. Dealing with Infinite Loops, Underestimation, and Overestimation of Depth-First Proof-Number Search. In Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10), pages 108-113, 2010.