詰将棋におけるゲーム木の並列探索とその評価

笠田洋和[†,☆] 山田雅之[†] 松波功二[†]
世木博久[†] 伊藤英則[†]

人工知能の分野で計算機上に知能を実現しようとする試みが盛んに行われてきている．しかし，人間の全知能を一度に実現することは不可能に近いから「限定した世界」をその対象としている．このような背景で従来からゲームに限定した問題解決手法についても盛んに試みられてきた．一般に，ゲームの探索問題では組合せ数の増加が避けられないことから計算量が問題となり，その解決のためにいろいろな工夫が必要である．ここではゲームとして「詰将棋」を取り上げ，並列計算機上にこれを解くプログラムを実現し，その評価について述べる．とくに，ゲーム木探索を並列に実行することによる効果，および，その処理時間と解答能力について述べる．その結果，既存の単純な評価関数による優先探索に比べ，処理時間，解答能力ともに向上することが確かめられた．

A Parallel Search Method in *Tsume-shogi* Game Tree and Its Evaluation

HIROKAZU KASADA,[†,☆] MASASHI YAMADA,[†] KOJI MATSUNAMI,[†]
HIROHISA SEKI[†] and HIDENORI ITOH[†]

In the field of artificial intelligence, realizing cognition on computers is advancing rapidly. It is impossible to realize human's whole abilities at one time. Then, we usually implement the intelligence in a limited world. Games are suitable as an example. But, the search problems of games can not avoid the combinatorial explosion of the amount of the calculations. Therefore, various methods are needed. In this paper, we use the *Tsume-shogi* (Japanese Chess Problem) game and implement a *Tsume-shogi* search program on a parallel machine. Moreover, we evaluated the solvability and the performance of the program. As a result, both the solvability and the performance are improved by using nth best-first search in parallel, compared with the simple best-first search in serial using an evaluation function.

1. はじめに

人工知能の分野で計算機上に知能を実現しようとする試みが盛んに行われてきている．しかし，人間の全知能を一度に実現することは不可能に近いから「限定した世界」をその対象としている．このような背景で従来からゲームに限定した問題解決手法についても盛んに試みられてきた．ゲームとして，バックギャモン，チェス，囲碁や将棋が取り上げられ，それぞれの特徴や難易度にあわせて，計算機で解くためのさまざまな方法が研究されてきた[1)~3)]．本論文では問題を詰将棋に限定し，詰将棋を計算機で解く際の並列処理の効果について述べる．

詰将棋は攻方が王手をかけ続け，それに対し受方が逃げる手順を考えるというパズルゲームの1種である．詰将棋を解くとは，双方が最善を尽くして詰みに至る手順を示すことである．与えられた問題に依存することにより一概に言うことは困難であるが，詰将棋のある局面に対しある程度複雑な場合では，数十手の王手が存在し，その王手それぞれに対して数十手の受手が存在する．

本論文では，与えられたある局面において，複数存在している手を並列に探索する手法について述べる．**図1** は3手詰の詰将棋の問題[4)] である．攻方の3二銀打の王手に対し，受方は同銀，または1二玉のいずれかを受手として選ぶことが可能である．しかし，同銀に対しては2二金打で詰みになり，また，1二玉に対しては2三金打の王手で詰みになる．すなわち，詰将棋の問題は**図2** に示すようなゲーム木探索問題と

† 名古屋工業大学工学部知能情報システム学科
Department of AI and Computer Science, Nagoya Institute of Technology
☆ 現在，日本IBM（株）
Presently with IBM Japan Ltd.

して考えることができる．

このような探索問題においては，木が深くなるにつれて急速に増加するノード数が問題となる．そこで通常は，評価関数による枝刈り法を用いて探索すべきノードを絞り込む．しかしながら，このような方法では，評価関数によって解答能力が大きく左右される．また一方，あらゆる局面に対して有効な優れた評価関数を作ることは実際には困難である．

詰将棋を解くプログラムはこの20年来数多く作られてきたが，1991年はじめには，13手詰くらいの問題がやっと解ける程度であった．しかし，最近，伊藤・野下らはワークステーション上でかなり優れたプログラムを作成した[5]．現在，コンピュータアーキテクチャの技術革新により，大きな処理能力をもつ並列計算機を用いることができ，従来とは違ったアプローチでこの詰将棋の問題を解く可能性がでてきた．そこで本論文では，詰将棋問題のゲーム木探索が並列に行えることに注目して，並列計算機上に詰将棋を解くプログラムを作成し，その解答能力と処理時間を評価する．

なお，並列計算機は富士通AP1000を使用し，プログラムの作成にはC言語を使用した．また，ここでは問題の詰手数は与えないものとする．なぜなら，詰手数以上の枝を調べる必要がないのでこの手数を越えた枝は「不詰」として除くことが可能となるので，詰将棋の問題の持つ本来の興味と複雑さを減少させてしまうからである．

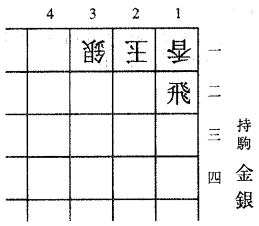


図1 3手詰の問題
Fig. 1 A 3 steps *Tsume-shogi* problem.

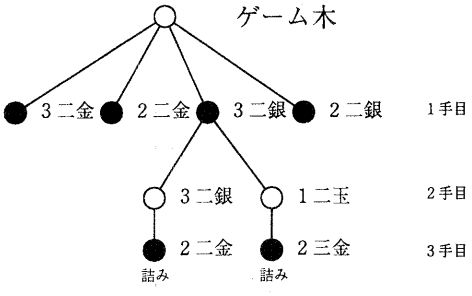


図2 ゲーム木探索
Fig. 2 A *Tsume-shogi* game-tree search.

2. 評価関数と探索方法

この節では，王手を現在の局面から静的に評価する評価関数とそれにもとづく並列探索方法について述べる．

2.1 即詰を考慮した評価関数

ここでの評価関数はある王手が有効か否か（詰みに至りそうか否か）を評価するのに用いる．すなわち，王手に対する受手の総数をその王手の評価値とし[5]，詰将棋の性質上，受方の手数の総数（自由度）が小さくなるという意味でこの値が小さいものほど詰みに至る可能性が大きくなると判断する．

このような評価方法に，さらに次の条件を付加する．すなわち，即詰（次の一手で詰む）になりそうな受手はこの数には含めないことにする．例えば，**図3**に示すような局面において，玉は王手，2三銀に対して五つの場所に逃げることが可能である．しかし，持駒に金があるので，D以外の場所（A，B，CおよびE）に逃げると即詰となると考えられる．このような場合の評価関数値は5でなくDのみの1とする．

2.2 ゲーム木全体のノードの評価

ある局面で，複数の王手の各々の評価値を2.1節で述べた方法で求めた後，MAX/MIN法を用いてゲーム木全体のノードの評価値を求める．MAX/MIN法では，王手葉ノード以外のノードの評価値を下位から順に次のように計算する（**図4**参照）．

攻方は受方にとって最も不利な手を打つことより，受手ノードの評価値は一つ下の王手ノードの中で評価の最も良い値とする（図中2手目参照）．

受方は攻方にとって最も不利な手を打つことより，王手ノードの評価値は一つ下の受手ノードの中で評価の最も悪い値とする（図中1手目参照）．

2.3 並列優先探索

攻方の局面で，ある王手が詰みに至る手であるため

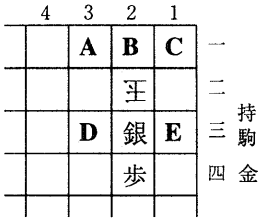


図3 即詰を考慮した王手の評価
Fig. 3 An evaluation function with taking consideration into *Sokutsume*.

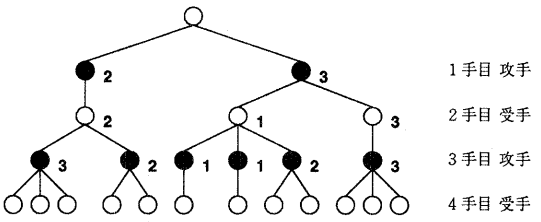


図4 MAX/MIN 法に基づくノードの評価
Fig. 4 An evaluation of nodes based on MAX/MIN method.

の条件は，その王手で詰みとなるか，または，その王手に対するすべての受手について詰みに至る手があることである．このことより詰将棋のゲーム木は王手ノードからなる OR 木と受手ノードからなる AND 木により構成される AND/OR 木である．

ここでは，ゲーム木全体のノードを 2.2 節で述べた方法で評価した後，詰みに至る手順を並列に探索する．並列度（セル台数）n で探索する場合は，評価の良い順に最大 n 個の王手葉ノード以降の探索を各セルに対して要求する．これを以降では並列優先探索法（Best-First Search in Parallel: BFSP）と呼ぶ[6]．

3. 並列プログラムの構成

ここでは，2 節で述べた評価関数を用いて並列優先探索法を並列計算機上に実現するプログラムの構成方法について述べる．このプログラムは一つのホストプログラムと同一の複数のセルプログラムから構成される．

3.1 ホストプログラム

ホストはセルの稼働状況を常時管理している．ホストは待ち状態にあるセルに対して，ある王手以降の探索を要求する．与えられた王手以降の探索によりセルが生成した部分木をセルから受信し，詰将棋の問題全体のゲーム木を構築していく．これらを以下に説明する．

（1）最初，ホストは与えられた問題の局面におけるすべての王手と，求められたすべての王手に対して各々の受手をすべて調べる．

もし，与えられた問題が 1 手詰の問題なら，処理はここで終りである．この時点で解答が得られているからである．

もしそうでなければ，評価関数によって各王手を評価し，評価値の良いものから（受手の数の少ないものから）順に，待ち状態セルに対し探索を要求する（**図 5** 参照）．

（2）ホストは，セルによって生成されたゲームの部分木を受信すると，自分自身が持っている現時点での全体のゲーム木にこれを取り込み連結し，新たなゲーム木を再構築する（**図 6** 参照）．なお，このときゲーム木の各ノードの評価値を MAX/MIN 法で再計算する．

ここで，AND 分岐の一方で詰まないと判断された場合，または，OR 分岐の一方で詰むと判断された場合は，これらの分岐の兄弟ノード以下は無意味な探索部分木となるから，これらの部分木の探索が割り当てられているセルが存在すれば，その処理を中断するメッセージをホストから該当セルに送る．

（3）ホストは待ち状態セルが m 個あれば，再構築されたゲーム木の評価値の良い順に m 個の王手葉ノードの探索要求を待ち状態セルに出す（**図 7** 参照）．

ホストは解答が得られるまで（2），（3）の手順を

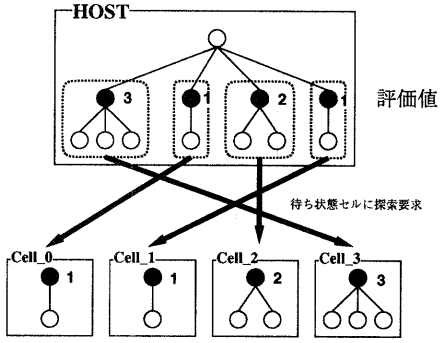


図5 ホストからセルへの探索要求（初期）
Fig. 5 Search requests from host to cells (a first time).

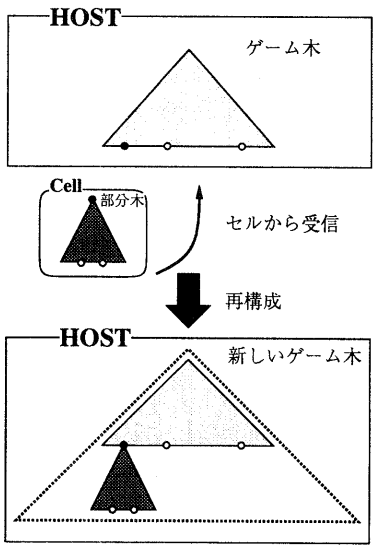


図6 新たなゲーム木の再構築
Fig. 6 Recalculate and reconstruct a new game-tree.

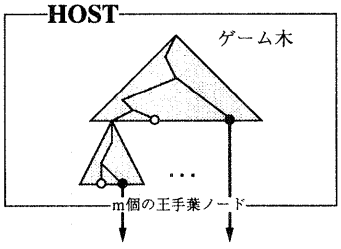


図7 待ち状態セルへの探索要求
Fig. 7 A search request to a free cell.

繰り返す.

3.2 セルプログラム

セルでは先に述べた評価関数を用いて横型最良優先探索を行う. セルは,

(i) 受けとった王手以下が詰みであると判断できるまで,
(ii) 受けとった王手以下が逃れ(不詰)であると判断できるまで,
(iii) 生成した部分木の大きさがあらかじめ定められた量に達するまで,
(iv) ホストから処理中断のメッセージが到着するまで,

のいずれかの条件を満たすまで処理を続ける. (i), (iii) の場合は探索した部分木を, (ii), (iv) の場合は処理を終了したことだけをホストに送信し, 新たな探索要求を待つ.

4. 実 行 例

BFSP を実現したプログラムの実行例について述べる. 具体的な問題の一例として伊藤果問題集 22 番の問題[7]を取り上げる. この問題は, 打ち歩詰を防ぐために角の不成が 4 回続くというものである. 以下に問題の初期局面(図 **8**), 約 43 秒で得た解答の表示(図 **9**) およびそのときのホストと各セルの稼働状況(平均セル稼働率 42.04%, 図 **10**) を示す. なお, 図 9 の各列は左から順に手数番号, 駒の移動前と移動後の位置, 駒の種類, 駒成り, 獲得駒, 評価値(詰に至る手なら-999 と表示する), その手のある部分木の番号とその手のノード番号, を表す.

5. 評 価

ここでは, 即詰を評価関数に考慮することによる効果と BFSP を実現したプログラムの (i) 処理時間と並列効果, (ii) 処理粒度と処理時間, および (iii) 解答能力について評価する.

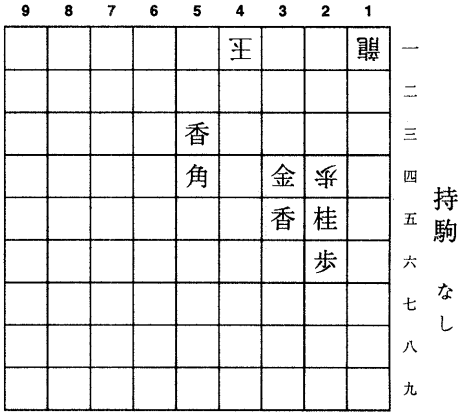


図8 伊藤果問題集 22 番初期局面
Fig. 8 The Itoh's No.22 *Tsume-shogi* problem.

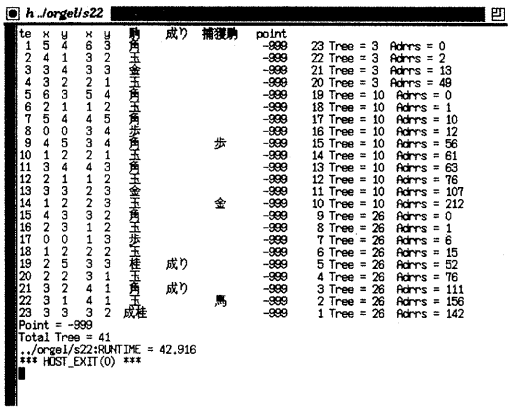


図9 伊藤果問題集 22 番解答
Fig. 9 The answer of the Itoh's No.22 *Tsume-shogi* problem.

5.1 即詰を評価関数に考慮することによる効果

即詰を考慮した評価関数と考慮しない評価関数を使用したときの処理時間を内藤国雄問題集[4]を用いて比較する. ただし, この実験は評価関数自身の有効性を調べることが目的であるので, 単一プロセッサ上での最良優先探索方法に基づいたプログラムで行った. ここでは単一プロセッサとして AP1000 のホストである SUN-4/630 MP を用いた. また, 解けた問題の処理時間の合計を総時間とする. 処理時間はプログラムの初期化時間(ホストや各セルにおいてプログラムがセットアップされる時間や詰将棋の初期局面データを読み込む時間)を除いた実際の探索処理に要した実時間のみとする. 処理時間が 10 分を越える問題は解けないものとした. 結果を**表 1** に示す.

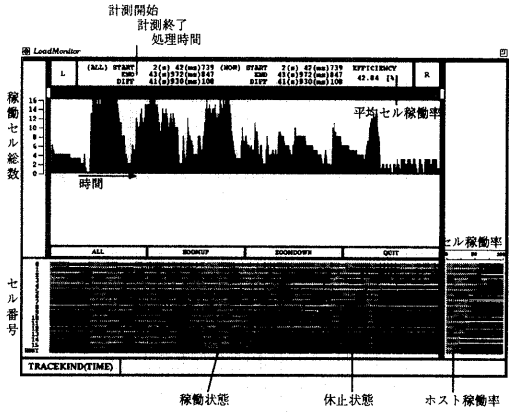


図 10 伊藤果問題集 22 番セル稼働状況
Fig. 10 The cell's busy trace for the Itoh's No.22 *Tsume-shogi* problem.

表 1 即詰を考慮することの効果
Table 1 The effect of *Sokutsume*.

評価関数	正解数	総時間（秒）
即詰あり	180	284.807
即詰なし	177	872.418

5.2 処理時間と並列効果

BFSP プログラムの処理時間と並列効果についての評価は，(a) 内藤国雄問題集と (b) 伊藤果問題集[7]について実験した結果について述べる．なお，ここで単一プロセッサ上で最良優先探索方法に基づいたプログラムによる処理時間と AP1000 上で BSFP に基づいたプログラムの処理時間を比較する．5.1 節同様，単一プロセッサとして AP1000 のホストを用いた．このホストによる結果については表，図中には「ホストのみ」として表記した．

(a) 内藤国雄問題集

内藤国雄問題集は 1 手詰（5 問），3 手詰（15 問），5 手詰（36 問），7 手詰（52 問），9 手詰（30 問），11 手詰（31 問），13 手詰（10 問），15 手詰（1 問）の計 180 問からなる．この実験では，処理時間が 10 分を越える問題は解けないものとした．**表 2** および**図 11** に処理時間と並列効果を示す．

ここでセル使用台数 4 の場合，ホストのみの結果と比較してほとんど同じような値であるが，セルの CPU クロックが 25 MHz であるのに対しホストの CPU クロックが 40 MHz とプロセッサの性能差があること☆とこれらの問題に対する並列探索の必要性が小さい，すなわち，評価関数の値が妥当であるため評価値の悪いノードを並列に探索しても効果がなかったことに起因している．また，使用台数 16 よりも 12 台の方がわずかに速いが，セル数増加による通信量の増加にも係わらず，先と同様に，並列探索の必要性が小さいため並列効果が十分出なかったことによる．なお，通信量の影響については 5.3 節で詳細に述べる．

☆ 単純にクロックで比較するとホストのみの処理時間は，セルプロセッサ一台における約 450 秒に相当する．

表 2 内藤国雄問題集における並列効果
Table 2 Results of Naitoh's (the number of cells vs the total processing time).

セル数	解答数	総時間（秒）
4	180	278.847
8	180	107.234
12	180	79.087
16	180	81.901
ホストのみ	180	284.807

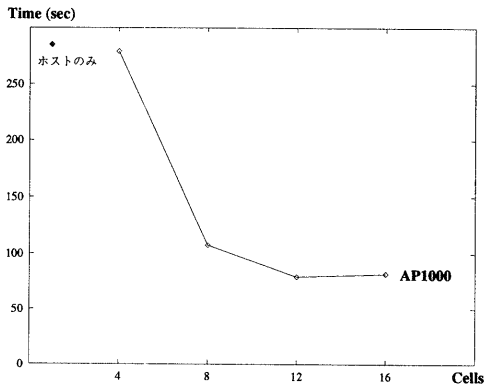


図 11 内藤国雄問題集・使用セル数–総時間
Fig. 11 Results of Naitoh's (the number of cells vs the total processing time).

(b) 伊藤果問題集

伊藤果問題集は，9 手詰（4 問），11 手詰（3 問），13 手詰（6 問），15 手詰（10 問），17 手詰（4 問），19 手詰（3 問），21 手詰（2 問），23 手詰（6 問），25 手詰（1 問），計 39 問からなる．ここでは処理時間が 10 分を越える問題は解けないものとした．**表 3** および**図 12** に処理時間と並列効果を示す．

内藤国雄問題集とは異なり伊藤果問題集では，ホストのみに対してセル 4 台のときも並列効果があがっている．また，セル 12 台に対して 16 台のときも並列効果が上がっている．このような結果の相違は両問題集における問題の種類の違いにより生じた．すなわち，伊藤果問題集の問題では評価値の悪いノードを並列に探索することによる効果があったことと，内藤国雄問題集に比べ詰め手数の長い問題が多いことによる．一概には言えないが詰め手数の長い問題では，探索する

表3 伊藤果問題集における並列効果
Table 3 Results of Naitoh's (the number of cells vs the total processing time).

セル数	解答数	総時間（秒）
4	38	956.673
8	39	626.494
12	39	485.965
16	39	420.449
ホストのみ	39	1417.504

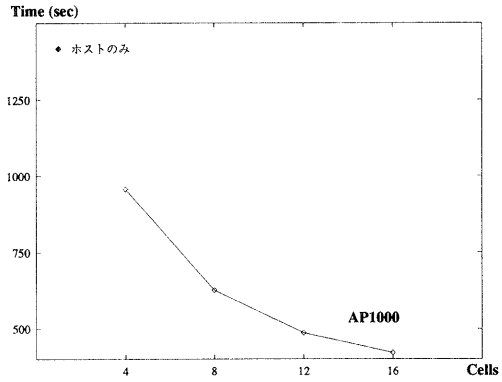


図12 伊藤果問題集・使用セル数－総時間
Fig. 12 Results of Itoh's (the number of cells vs the total processing time).

ゲーム木は深さのみでなく横幅も大きいため，並列に取り扱えるノードの数が多い．そのため並列効果が上がると考えられる．

ここで使用セル台数4の場合，解くことができなかった問題は31番13手詰（図**13**参照）である．この問題の初手は11通り存在する．このうち，正解である王手（4一銀不成）に対して2.1節で述べた評価関数は10番目の評価を与えている☆ため，使用セル台数が4の場合は正しい初手が制限時間内にセルに割り当てられず処理が中断されている．なおホストのみでは約355秒，8台使用時では約84秒，12，16台使用時では約1.6秒で解いている．なお，使用セル台数16のときのホストと各セルの稼働状況（ホスト稼働率25%，平均セル稼働状況57.56%）を図**14**に示す．

5.3 処理粒度と処理時間

ホストからの探索要求（Job）に対して，セルが実行する処理の粒度☆☆を変化させた時，総時間がどのように変化したかを示す．5.2節と同様に（a）内藤国雄問題集と（b）伊藤果問題集について実験を行った．この結果をそれぞれ図**15**と図**16**に示す．

ここで（a），（b）双方とも使用セル台数8，16のいずれの場合もおおむねU字型の特性を持つ結果を得た．この理由として，処理時間は各セルのプロセッサ性能と通信バス性能に依存するが，(1) 処理粒度が小さい場合はゲーム木分割が頻繁に行われ分割損による通信量が増加し，かえってオーバヘッドとなり，(2) 処理粒度が大きくなると与えられた王手ノードが正解または不正解のノードであることの判定が遅くなり，その処理に費やされる時間粒度が大きくなることが原因として考えられる．

5.4 解答能力

ここでは並列処理により解答能力が向上することを示し，また，その結果を伊藤・野下[5]らの *Ito* および *T2* のプログラムの結果と比較する．与える問題は「続・詰むや詰まざるや」[8]に収録されている200題とし

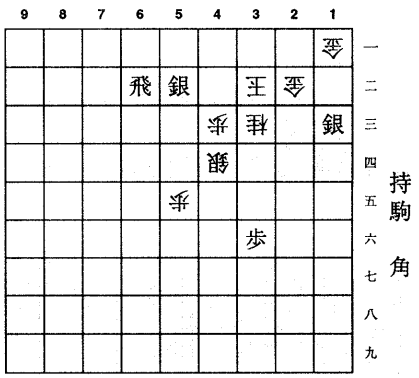


図13 伊藤果問題集・31番13手詰
Fig. 13 The Itoh's No.31 *Tsume-shogi* problem.

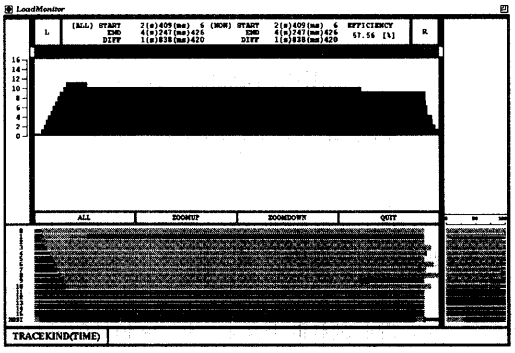


図14 セル稼働状況（使用セル台数16）
Fig. 14 The cell's busy trace for the Itoh's No.31 *Tsume-shogi* problem (16 cells).

☆ 同一の評価値を持つ王手ノードの順位づけは局面の駒リスト順位による．

☆☆ セルは与えられた王手葉ノード以降を探索し部分木を生成するが，この大きさを制限すれば当然一回の探索要求に対するセルの処理量をも制限することになる．ここでは厳密性に欠けるが生成する部分木の大きさを処理粒度として見なしている．

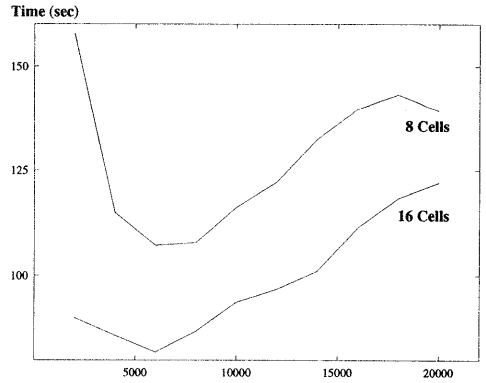


図15 内藤国雄問題集・処理粒度－総時間
Fig. 15 Results of Naitoh's (the size of Job vs the total processing time).

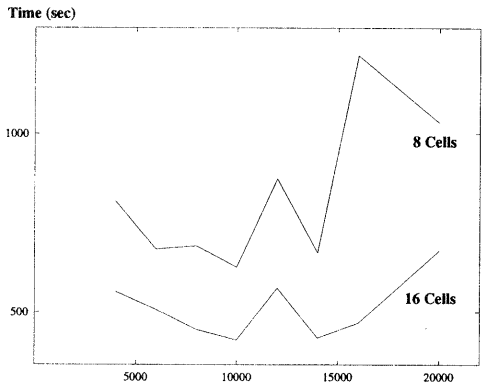


図16 伊藤果問題集・処理粒度－総時間
Fig. 16 Results of Itoh's (the size of Job vs the total processing time).

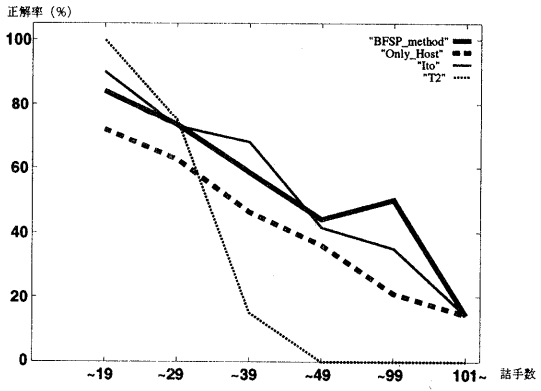


図17 「続・詰むや詰まざるや」詰手数別正解率
Fig. 17 Results of *Zoku Tsumuya Tsumazaruya*.

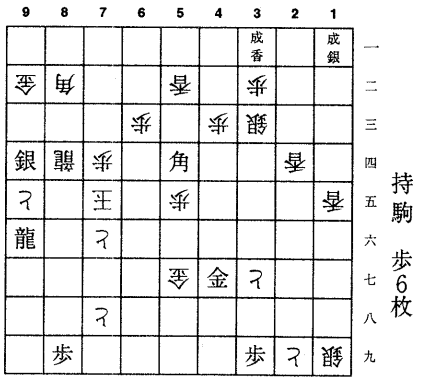


図18 「続・詰むや詰まざるや」第140番
Fig. 18 No.140 of *Zoku Tsumuya Tsumazaruya*.

た．これには江戸時代から昭和まで，有名な古典作品が多く含まれている．なお，本プログラムはAP1000上に実現しセル16台による並列処理を行い，いずれも一題30分の時間制限を与えて実験した．*Ito* および *T2* プログラムはワークステーションSPARC IPX上に実現し，単一プロセッサで処理を行い，メモリ量をオーバした時点で解答不能とした結果を用いた．

詰手数別の正解率を図**17**に示す．*Ito* および *T2* に比べ使っている評価関数にあまり工夫がないので，ホストのみのときは，それらより解答能力は劣っている．しかしながら，並列処理をした場合ホストのみに比べ全体に解答能力が向上している．そして，評価関数が比較的良くないにも係わらず，*Ito* とほぼ同等の解答能力が得られている．これらの結果より解答能力における並列処理の効果が確認できる．

なお，時間制限を1題60分とした場合，新たに23手詰めと51手詰めの問題がそれぞれ1問ずつ，52分と56分で解けた．これにより，20手以上29手以下の問題と50手以上99手以下の問題に対する正解率がそれぞれ2.3%と2.1%向上した．5.2節で述べた問題集[4),7)]の結果から使用セル台数を増加するとさらに正解率が上がる可能性があると思われる．

また，第140番131手詰（図**18**）が解いた最長手数であった．この140番の問題は265手詰が正解手順とされているが，本プログラムは131手詰☆の余詰を約27秒で見つけている．

6. おわりに

詰将棋におけるBFSP（Best-First Search in Parallel）を定義するとともにこのプログラムを実現した．またその性能を処理時間と解答能力について評価した．

☆ 本プログラムでは付録に示す無駄合処理アルゴリズムによって文献5)で133手詰とした余詰手順130手目の歩合を無駄合と判定している．

その結果，処理時間，解答能力ともに単一プロセッサ上での最良優先探索に比べ，並列化したことによる処理時間および解答能力の向上が確かめられた．短篇問題では，通常の人間のレベルをはるかに越えているものと思われる．また，中長編でもかなり速い時間で解いている．評価関数で即詰を考慮するようにすることでかなりの処理時間の向上が確かめられた．

しかしながら，解けない問題も依然かなり残っている．例えば，見かけは良いが正解ではない手が作者により巧みに埋め込まれている問題は解きにくい．今後は，解答能力と処理速度を向上させるために，このような解けない問題の原因を探り，さらに有効な評価関数やこれに基づく並列処理方式について検討する必要がある．また，さらにセルの使用台数を増やして実験する予定であるが現在の方式では，ホストの処理がボトルネックになるものと思われ，セルの構成を階層的に配置するなどの改良すべき点が考えられる．

謝辞 本研究を進めるにあたり貴重なご示唆を賜った米長企画（代表 米長邦雄 前名人）の方々に深謝いたします．

参考文献

1) Levy, D. and Newborn, M.: *How Computer Plays Chess*, W.H. Freeman and Company (1990).（小谷善行監訳：コンピュータチェス，サイエンス社 (1994)）．
2) Belekamp, E. and Wolfe, D.: *Mathematical Go*, A K Peters, Ltd. (1994).（吉川他訳：囲碁の算法，トッパン）．
3) 松原 仁：将棋とコンピュータ，共立出版 (1994).
4) 内藤国雄：九級から一級までの詰将棋，成美堂出版 (1992).
5) 伊藤琢巳，野下浩平：詰将棋を速く解く2つのプログラムとその評価，情報処理学会論文誌，Vol.35, No.8, pp.1531–1539 (1994).
6) Kumar, V., Ramesh, K. and Rao, V.N.: Parallel Best-First Search of State-Space Graphs: A Summary of Results, *Proc. of AAAI '88*, pp.122–127 (1988).
7) 伊藤 果：オルゴール，将棋世界 10 月号付録，日本将棋連盟 (1994).
8) 門脇芳雄：続・詰むや詰まざるや，平凡社 (1978).
9) 小谷善行，吉川竹四郎，柿木義一，森田和郎：コンピュータ将棋，サイエンス社 (1990).

付録 実現した諸機能

付録として，詰将棋の探索を効率良く行うために工夫し，実現した諸機能を挙げる．

(a) 無駄合処理

詰将棋では，飛車など遠くからの王手に対して合駒を置くのが有効か否かを調べなければならない．これを調べるためのアルゴリズムとして以下に示す柿木のアルゴリズム[9]が知られている．本プログラムでもこれを採用し実現した．

(i) 詰み上がり局面における無駄合を，次のように規定する．合駒を攻方に取られて，合駒ができず，玉が逃げられず，攻駒を取ることができないとき，その合駒は無駄合とする．

(ii) 途中局面における無駄合を次のように規定する．合駒以外の受手ではN手の詰みであるとき，合駒を取って王手を掛け（複数の手があり得る），その取った駒を使わずに，取った手を含めずに数えて，N手以下で詰めば，その合駒は無駄合である．ここで，N手の詰みを調べるときにも，上述の無駄合判定を行い，正確な詰手数を得なければならない．

(b)「成」「不成」の王手処理

王手をかける時に「成」「不成」を選択できる場合，もしこの王手の駒を取ることができるならば，この取る受手以下の局面は同一になる（受手の持駒に加えられるだけである）．そこでこの局面以下の探索木を両方（「成」「不成」）の王手ノードに共有させることで処理時間とメモリの節約を図っている．

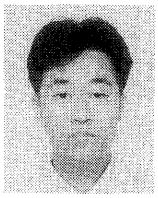
(c) 千日手処理

攻方，受方の双方が最善を尽くした結果，同一局面を繰り返す「千日手」は「不詰」とみなすことができるのでその枝を除くこととする．

(d) ガーベジコレクション

本プログラムのホストでは，いくつかの部分木の集まりで全体のゲーム木を構築するが，この部分木の格納場所（メモリ）が処理が進むにつれて消費されていく．そこで，この格納場所が少くなったときに部分木単位でガーベジコレクションを行う．すなわち，ここでは悪い評価値をもつ部分木，および前回のガーベジコレクションから一度も参照されなかった部分木を消去する．

(平成7年2月1日受付)
(平成7年9月6日採録)

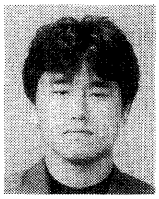


笠田 洋和

1993年名古屋工業大学工学部電気情報工学科卒業．1995年同大学院工学研究科博士前期課程修了．現在，日本IBM勤務．

山田 雅之（正会員）

1992年名古屋工業大学工学部電気情報工学科卒業．1994年同大学院工学研究科博士前期課程修了．同年同大学工学部知能情報システム学科助手．人工知能学会会員．



松波 功二

1995年名古屋工業大学工学部知能情報システム学科卒業．同年，名古屋工業大学大学院工学研究科博士前期課程電気情報工学専攻入学，現在に至る．並列処理，遺伝的プログラミング，包摂アーキテクチャに興味をもつ．

世木 博久（正会員）

1979年東京大学工学部計数工学科卒業．1981年同大学院工学系研究科修士課程修了．同年4月より三菱電機（株）中央研究所に勤務．1985年～1989年（財）新世代コンピュータ技術開発機構に出向．1992年4月より名古屋工業大学工学部知能情報システム学科助教授．工学博士．論理プログラミング，演繹データベース等に興味を持つ．電子情報通信学会，人工知能学会，ACM，IEEE Computer Society各会員．

伊藤 英則（正会員）

1974年名古屋大学大学院工学研究科博士課程電気・電子専攻満了．工学博士号取得．同年日本電信電話公社入社，横須賀研究所勤務．1985年（財）新世代コンピュータ技術開発機構出向．1989年より名古屋工業大学教授，現在知能情報システム学科所属．これまでに，数理言語理論とオートマトン，計算機ネットワーク通信OS，知識ベースシステムなどの研究と開発に従事．電子情報通信学会，人工知能学会，ファジィ学会各会員．