G-27　局面評価値による重み付けを利用した反復深化法の並列処理手法
Parallel Processing of Iterative Deepening with Weighted Evaluation Value

関根 敦史 † 前川 仁孝 † 伊與田 光宏 †
Atushi Sekine Yoshitaka Mekawa Mistuhiro Iyoda

1. はじめに

詰将棋問題の最良優先探索 [1][2] による解法では, 長手数の問題を解くことが可能であるが, 先手の指し手を全て探索しないため, 正解手順を必ずしも求めることはできない. 一方, 深さ優先探索による解法 [1][3] では, 正解手順を必ず求められるが, ゲーム木の探索深さが増えるにつれて, 探索する節点数は指数関数的に増加するため, 処理時間は膨大になる. そこで本稿では, 正解手順を短時間で得るため, 根節点における子節点を後手の指し手の数である「玉の自由度」[3] に基づき局面評価をして, 子節点の重み付けにより評価の高い子節点以下を深さ優先探索を用いて、集中的に並列探索する手法を提案する.

2. 局面評価値による重み付け手法

詰将棋問題を深さ優先探索で解く場合, 反復深化法を用いるのが一般的である [4]. 詰将棋問題は, 通常詰手数を与えないため, 反復深化法における閾値の探索深さは, 1, 3, 5, ··· と, 2 ずつ増加させながら解く. 局面評価値による重み付け手法では, 根節点における子節点に探索の優先順位を付けるため, 評価関数を用いて子節点に 1 度だけ評価値を付ける. 評価値には葉節点における後手の指し手の総数である「玉の自由度」を用い, α-β 法 [4] により根節点の子節点に評価値を与える. 図 1 に深さ 3 のゲーム木における α-β 法による局面評価の例を示す. 節点と節点を結ぶ線は指し手, 節点間にある矢印は節点展開の順序, 葉節点の数字は玉の自由度を表す. 先手番節点では子節点の内, 評価値の高い (小さい) 節点の評価値を選択し, 後手番節点では評価値の低い (大きい) 節点の評価値を選択する.

探索の効率を考えた場合, 評価値の高い根節点における子節点を集中的に探索することが, 効率的であると考えられる. このため, 評価値に基づき各子節点に重み付けを行う. 重み付けを行うことで評価値の高い節点, つまり, 正解手順のある可能性の高い節点を集中的に探索することが可能になる. ここで, 合計値を根節点の各子節点が持つ

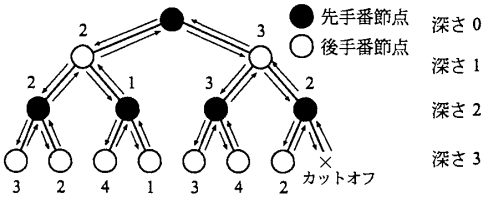

図 1: α-β 法による局面評価

値と定義する. 評価値の高い節点を低い節点より探索回数を多く設定するために, 根節点の各子節点が現在の探索深さで正解手順を得られずに反復処理を終了した場合, 局面評価値が高いほど小さくなるように式 (1) のような値 W を合計値に加える.

$$W = 1 + \text{重み} \times (\text{局面評価値} - 1) \qquad (1)$$

その合計値を基にして, 合計値の最も小さい子節点を選択して探索する. W は重み付けをしない場合 1 になるように, 重みと局面評価値をかけた値になるようにした. また, この場合通常の反復深化法を適用した深さ優先探索になる. 合計値の同じ子節点がある場合は, 最も昔に探索された子節点を選択する. つまり, 評価値の高い子節点は低い子節点に比べて, 正解手順を得られずに探索を終了した場合, 合計値に加える値は小さくなり, 多く探索される. 局面評価値の重み付けによる探索節点の選択方法を使用 PE 数 1, 重み 1 の条件下で図 2 を用い説明する. 合計値の初期値は全て 0 である (状態 0). 初めは評価値の高い節点 A で深さ 1 の探索を行い, 終了すると合計値に W の 1 を加える. 次に節点 B で深さ 1 の探索を行い, 終了すると合計値に W の 2 を加える. 続いて節点 C で深さ 1 の探索を行い, 終了すると合計値に W の 3 を加える (状態 1). 節点 A, B, C の合計値が 0 ではなくなったので, 以降は合計値の小さい節点を選択して探索する. 合計値の最も小さい節点 A で深さ 3 の探索を行い, 終了すると合計値に W の 1 を加え, 合計値が 2 となる (状態 2). 最も昔に探索された節点 B で深さ 3 の探索を行い, 終了すると合計値に W の 2 を加え, 合計値が 4 となる (状態 3). 以下正解手順を得るまで繰り返す.

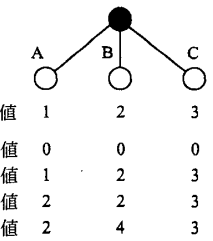

図 2: 探索節点の選択方法の説明図

3. 並列処理手法

使用する PE 数が根節点の子節点より少ない場合, 木分割法 [5] と同等に根節点の子節点 1 つに 1PE を割り当てる. 割り当てられた PE を MP(マスター PE) とする. 使用 PE 数が根節点の子節点より多くなった場合, 評価値の高い子節点に割り当てられている MP は, 割り当てられずに余った PE を探索に使用する. 割り当てられずに余った PE を SP(スレーブ PE) とする. SP は割り当てられた

*† 千葉工業大学 情報科学部 情報工学科
Department of Computer Science, Chiba Institute of Technology

MP が展開した深さ 3 の節点に次々と割り当てられ, 深さ 3 の節点を根節点として探索を行う. MP は SP と同様に深さ 3 の節点に自分自身を割り当てる. つまり, 深さ 2 の節点を根節点として木分割法を行う. もし, 深さ 3 の節点数がその根節点に割り当てられた MP, SP 数より多い場合, 処理の終了した MP, SP が探索されていない節点を探索する. 図 3 に使用 PE 数 5 における PE 割り当て手法を示す. 深さ 1 の節点 A, B に MP0 と MP1 が割り当てられ, 評価値の高い節点 A では, MP0 は探索で SP0, SP1, SP2 を使用する. 節点 B に割り当てられている MP1 は, 1PE で節点 B 以下を探索する. 節点 C の子節点である節点 E, F(深さ 3 の節点) を展開したら, MP0 は自分自身と SP0, SP1, SP2 を節点 E, F に割り当てる. この場合は深さ 3 の節点数が 2 なので, 2PE はアイドル状態となる. 節点 C が詰になった場合, 節点 D においても同様に節点 G, H に割り当てる.

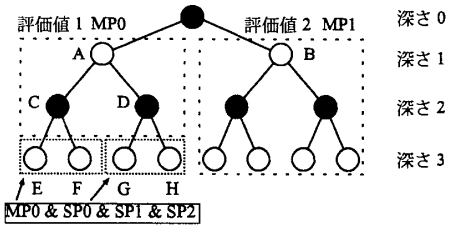

図 3: PE 割り当て方法の説明図

4. 性能評価

重み付けと PE 数の関係を調べるために, 3 種類の重み付け (0, 0.2, 0.4) において処理時間を比較した. 実装環境は共有メモリ型並列計算機の Sun Enterprise 4500, CPU は Ultra SPARCII-s 400MHz が 14 台, メモリ 4GB である. 対象問題には「塚田詰将棋代表作」[6] を使用し, 5〜21 手詰の計 195 問を解いた. なお, 無駄合い判定と中合いはしていない. 深さ 5 の α-β 法での局面評価は, 塚田詰将棋代表作 195 問中 187 題 (約 95.9%) の問題で, 根節点の正解手順である子節点に最も良い評価値を与え, 残りの 8 題 (約 4.1%) においても次に良い評価値を与えたことから, 局面評価値を得るための計算には, 深さ 5 の α-β 法を使用した. また, SP は評価値の最も高い節点での探索に使用した.

重み 0.4 の時は重み 0 に比べ, 1PE 時で 195 問中 181 題 (約 92.8%), 14PE 時で 195 問中 150 題 (約 77.0%) について処理時間が短縮した. 重み付けによる探索節点の選択の効果により, 大きく短縮している例の 1PE, 14PE 時の各重み付けにおける処理時間を表 1 に示す. 表 1 のどの問題においても重みが増えるにつれて, 処理時間が短縮している. 正解手順のある根節点の子節点で評価値が高い場合にこの傾向が見られ, この場合重みの値が大きいほど処理時間が短縮される. 14PE 時では重み付けによる処理時間短縮が, 1PE 時より効果が小さくなっているが, これは塚田詰将棋代表作 195 問における根節点の平均子節点数が約 10.4 なので, 問題によっては使用 PE 数が根節点の子節点数より多くなり, 全ての子節点は常に探索されている状態になる. そのため探索節点の選択による処理時間短縮の効果が小さくなる. 1PE 時で 195 問中 14 題 (約 7.2%) について, 処理時間が増加した, または処理時間の短縮が見られなかった. 重み付けによる探索節点の選択の効果がなかった例の 1PE, 14PE 時の各重み付けにおける処理時間を表 2 に示す. どの問題でも 1PE 時で重みが 0.4 になると, 処理時間が増加している. このパターンに当てはまるの問題は, 正解手順のある根節点における子節点の評価値が, 他の子節点に比べ低い場合である. これは探索深さが増えるにつれて, 探索にかかる時間も増加するため, 正解手順の無い根節点における子節点の探索に時間がかかり処理時間が増加する. しかし, 14PE 時では重みが増えるにつれ, 処理時間が短縮している. これは, 使用 PE 数が増えたために, 重み 0 の時に比べ探索節点の選択による効果が出たためであると思われる.

表 1: 重み付け手法の効果が見られた問題の処理時間 [s]

問題番号	PE 数 1 重み 0	重み 0.2	重み 0.4	PE 数 14 重み 0	重み 0.2	重み 0.4
144	385.0	262.8	204.7	60.5	57.2	56.5
147	174.7	139.8	71.3	28.1	25.3	25.8
162	821.6	173.0	55.5	38.5	22.4	22.1
181	333.5	165.1	139.3	11.2	10.5	10.5

表 2: 重み付け手法の効果が見られない問題の処理時間 [s]

問題番号	PE 数 1 重み 0	重み 0.2	重み 0.4	PE 数 14 重み 0	重み 0.2	重み 0.4
36	17.9	17.3	68.2	0.5	0.5	0.5
125	4.2	4.1	8.1	0.7	0.6	0.6
127	9.3	9.2	14.2	3.3	3.1	3.1
166	2524	6071	16801	792.8	759.2	752.3

正解手順のある節点に最も高い評価値をつけた場合, 重みの値を大きくするほど, 1PE, 14PE 時の処理時間が重み 0 に比べ短縮した. また, 2 番目に高い評価値をつけた場合, 重みの値を大きくするほど, 1PE 時の処理時間は重み 0 に比べ増加し, 14PE 時では短縮したことがわかる.

5. おわりに

反復深化法による深さ優先探索を用いて並列処理を行う際, 子節点の重み付けにより評価の高い子節点以下を集中的に探索する手法を提案した. 探索節点に重み 0.4 の重み付けを行った結果, 1PE 数時で塚田詰将棋代表作 195 問中 181 題 (約 92.8%), 14PE 時で 195 問中 150 題 (約 77.0%) において処理時間が短縮した.

参考文献

[1] 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.
[2] 伊藤 琢巳, 河野 泰人, 野下 浩平:非常に手数の長い詰将棋問題を解くアルゴリズムについて, 情報処理学会論文誌, Vol.36, No.12, pp.2793-2799, 1995.
[3] 野下 浩平:詰将棋を解くアルゴリズムについて, 電子情報通信学会研究会報告, COMP91-56, pp.29-37, 1991.
[4] 松原 仁, 竹内 郁雄:ゲームプログラミング, 共立出版, 1998.
[5] Schaeffer, J:Distributed Game-Tree Searching, *Journal of Parallel and Distributed Computing*, Vol.6, pp.90-114, 1989.
[6] 塚田 正夫:塚田詰将棋代表作, 日本将棋連盟, 1999.