F_031

オセロ盤面の冗長情報の破棄による探索の効率化

Improvement of Search by Reducing Redundant Information in a Position of Othello

松尾 英和† 楢崎 修二‡
Hidekazu Matsuo Shuji Narazaki

1 はじめに

これまでゲームプレイングに関して既に多くの研究がなされてきた [2]。それらの研究の多くはゲーム木をいかに高速に探索するかという点に注目していた。これは探索速度が、計算機によるゲームプレイングにおいてその強さに直結するからである。

探索の高速化として反復深化法やNull Window Searchの考え方に基づいたPVS [1] など $\alpha\beta$ 探索法の枝刈りをより多く発生させるアルゴリズムがこれまでよく研究されてきた。しかし枝刈りの手法の一つである置換表を用いて枝刈りを発生させるアルゴリズムの研究は筆者らが調べた範囲では存在は確認できたものの、その数は少なかった [3]。本論文では多くのプログラムに利用されていながらあまり研究されなかった置換表の性能を向上させる手法について提案する。それにより全体の探索速度の向上を図る。

2 関連研究

2.1 置換表による探索の高速化

置換表とは盤面とその探索結果を記憶するための表である。置換表による探索の高速化とは新しい盤面を探索する度に探索結果を記憶して、再びその盤面を探索する際に記憶していた探索結果を用いてその盤面より深い探索を省略する、事実上の枝刈り手法である。置換表は探索速度を向上させるための表なのでデータ構造にはハッシュ表が多く使われる [3]。

置換表の性能を表す指標には置換表から参照を試みたときの成功率が適当である。なぜなら、この成功率を高くすることは一定の記憶領域でより多くの枝刈りを発生させたと言えるからである。本論文ではこの成功率をヒット率と呼ぶ。

2.2 定石表による知識の利用

ゲームプレイングの分野では置換表のように盤面の評価値を記憶しておいてそれを利用する定石を用いる手法がある。

両者は盤面とそれに対する評価値を記憶しておいて参照する点では同じである。しかし置換表の目的は探索の高速化である点に対し、定石を用いる手法の目的は正確な評価値がわからない序盤で人間の知識を利用してプログラムを強くすることである。そのため置換表には正確な値が記憶するべきである。それに対して定石表 (以降定石を記憶する表をこう呼ぶ) では (もともと人間の知識は不正確なので) 値が厳密に正確である必要はない。

2.3 囲碁における定石利用の効率化手法

囲碁において定石表を用いる際にマスク処理を行いヒット率を向上させる方法がある [4] (以降この方法を囲碁におけるマスク処理による定石利用の効率化手法と呼ぶ)。この手法は定石を判定する上で不必要な石の色情報をマスクがけして破棄し、その石をDon't Careとして扱い、記憶している一つの盤面を多少違う複数の盤面に適用させるものである。その結果、定石表がヒット率を向上する。現在一部の囲碁プログラムではこの方式で定石を利用している。

具体的な手順は以下のようになる。まず定石を定石表に登録する処理を示す。なおこの作業はプログラムの実行前に行う。

i 盤面を評価する上で不必要な石をある基準で決める。この基準は人間の知識による
ii 求めた石から実際に盤面にマスクをかける
iii マスクをかけた盤面をキーにして (人間の知識による) 次に打つ手の位置情報を定石表に登録する

次に定石表から参照する際の手順を示す。なおこれらは実行中に行い、初めてその盤面を展開するときに行う

I 盤面を評価する上で不必要な石をある基準で決める この基準は人間の知識による
II 求めた石から実際に盤面にマスクをかける
III マスクをかけた盤面をキーにして定石表から同値の盤面を探し、見つかれば記憶していた次に打つ手の位置情報を利用して手を打つ。

3 度外石を用いた枝刈りの効率化アルゴリズム

3.1 度外石とは

前節の手法の記憶している一つの盤面を多少違う複数の盤面に適用できる性質をオセロの置換表に導入したアルゴリズムが本論文で提案する度外石 (どがいし) を用いた高速化手法である。

両者の共通点は盤面にマスクをかけて、その石をDon't Careとし、置換表でのヒット率を向上させる点である。両者の最も大きな相異点は前述の手法を (人間の知識を利用しているため) 適用する定石表は厳密に正確である必要は無かったが、度外石を用いた高速化手法を適用する置換表は記憶している評価値が正確なので、その値を壊さない工夫が必要である点である。

3.2 度外石である条件

盤面上のある石が度外石である条件、すなわちマスクをかけて石の色情報を破棄しても置換表の評価値が正確である石の条件は以下の2点である。

† 長崎大学生産科学研究科
‡ 長崎大学工学部

1. その石の色 (黒、白) が以後変更できない
2. その石がある為に、他の石を返す可能性がない

まず 1. の条件を満たしていると今後のどのような展開をしてもその石の色が変わらないことが保証できる。それよりその石に限っては探索結果の色がわかる。次に 2. の条件を満たしているとその石の色が白か黒かであることが今後の展開に影響を与えない。

しかし実際にこの定義どおりに度外石を求めると非常に計算時間がかかるので、ある石が度外石であるか判断する際には、その石の回り 8 方向に (盤面の端まで) 空白がないかどうかで判断する。回りに空白がなければその石は前述の 1. 2. の条件に従っている。条件をより厳しく設定すれば度外石は増えその効果は増すが、石を判定する為の処理が複雑になることと、条件の定義が難しく本実験ではこの条件で近似した。

3.3 度外石を用いた前述の高速化アルゴリズムの手順

以下に度外石を用いた高速化手法の手順を示す。まず置換表に登録する際の手順を示す。なお囲碁におけるマスク処理による定石利用の効率化手法と違いこの処理は実行中に行う。これはその盤面の探索結果が判明したとき、すなわちバックトラックする直前に行う。

i 前節の度外石の条件に従い度外石を求める
ii マスクをかけるその際マスクをかけた箇所の黒石、白石の個数の差を記憶しておく (黒にとっての最善手を求めたいのであれば、黒石の数 – 白石の数 とする)
iii 探索結果 (勝敗の情報ではなく、先手後手双方の石差) から上記の石の差を引く。引いた後の値が、度外石でない部分の評価値である
iv マスクをかけた盤面をキーにして iii で求めた値を置換表に登録する

次に置換表から参照する際の手順を示す。これはある盤面を探索する際にまず行う処理である。

I 盤面上の度外石を求める
II マスクをかける。その際マスクをかけた箇所の黒石、白石の個数の差を記憶しておく
III マスクをかけた盤面をキーにして置換表から同値の盤面を探し見つかれば、記憶していた値を II の値に足す。この和は度外石を含めた盤面全体の評価値である。見つからなければ通常の探索を行う。

4 実験とその結果

4.1 実験の条件

度外石を導入して従来の手法との比較を行う。以下に実験の条件を載せる。なお計算機の環境を変更しても探索時間にしか影響を与えないので、結果を示した表 1 では探索時間のみ導入前を 1 として比率で表した。

1. 探索時間の都合上 47 手目を初期盤面として利用し、それらはランダムに作成した
2. サンプル数は 300 とする
3. $\alpha\beta$ 探索法の完全読みを行い、手の順序付けは行わない。また前向き枝刈りも行わない
4. 導入前、導入後の両方に置換表は利用し、ハッシュ値の衝突対策はチェイン法とする

表 1: 度外石導入前と導入後の比較

	導入前	導入後
探索した盤面数	2520182	1393337
探索時間 (導入前を 1 とする)	1.00	0.57
参照した盤面数	408511	572708
登録した盤面数	766146	540625
ヒット率 (%)	16.21	41.10

5. 置換表に入れる盤面の総数は 100MB 分までとする

置換表を使いきった場合は全てのノードを破棄してから登録を行う。全てのノードを捨てる理由は昔のノードは参照する確率が低いからである。また置換表で使うハッシュ表に必要な配列は使用するメモリには含めていない。ハッシュ表の配列の大きさは 999983 個とした。

4.2 実験の結果

表 1 より探索した盤面数はほぼ半減し、探索時間も同程度減少している。また探索する盤面数全体は減っているにも関わらず、参照する盤面数は 40%程度増えている。そのため置換表のヒット率は 25%程度向上し置換表の効率は大幅に向上したと言える。ただし探索した盤面数の減少に比べ、登録した盤面数の減少は小さいものとなった。以上の結果から提案手法を導入し置換表のヒット率を向上させ、それによって全体の探索速度が向上したといえる。

5 おわりに

本提案手法を導入し置換表の効率を向上させ、その結果探索速度を向上させた。

今後の課題として以下を挙げる。

- 多くの探索の高速化手法を導入しているオープンソースのソフトに度外石を導入してどの程度の効果があるか検証する
- 度外石とする石の増加による高速化と、判定の為のオーバーヘッドとのトレードオフについて評価する

参考文献

[1] G. Goetsch and M.S. Campbell. *Expriments with the Null-Move Heuristic*, pp. 159–168. 1990.

[2] 松原仁. 人工知能学辞典, AI 応用:ゲーム, pp. 881–898. 共立出版, Dec 2005.

[3] 長井歩, 今井浩. 詰将棋を解くプログラムにおける効率的なハッシュの利用法について. 情報処理学会 研究報告, pp. 21–26, June 2001.

[4] 清慎一, 山下宏, 佐々木宣介. コンピュータ囲碁の入門. 共立出版, Nov 2005.