講演番号 20

ゲーム木探索アルゴリズムの比較と改良

情報工学科 12番　川岸良次,　指導教員　東海林智也

Comparison and improvement of search algorithm for a game tree

KAWAGISHI Ryoji

Abstract : This study compares and improves various search algorithms which are used for a thinking routine of a computer chess application. In order to compare each search algorithm, MinMax algorithm and Alpha-Beta algorithm are used. In addition, we consider an algorithm which is improved on the base on the Alpha-Beta algorithm. By using server-client system, the searching process which does not depend on the performance of the hardware is obtained.

Key words : Game theory, Chess, Search algorithm

1. はじめに

本研究では，代表的なゲーム木探索アルゴリズムをコンピュータチェスのゲーム木に対して組み込みその探索効率を比較・改良することを目的とする．更に，探索効率がハードウェアの性能に左右されないようアプリケーションをサーバ・クライアント方式のネットワーク構成にすることで，安定した探索処理を実現する．

2. 開発環境

OS	: Windows2000
コンパイラ	: Microsoft Visual C++ 6.0

3. ゲーム木と探索アルゴリズム

3.1 ゲームとゲーム木

本研究で扱うゲームは二人零和完全情報ゲームである．「二人」は対戦する人数，「零和」は両者の損得の和が必ず零になる事を示す．そして「完全情報」は相手の情報も見えるという事を示す．

二人のプレイヤーが交互に着手し得る全ての可能性を網羅したものをゲーム木といい，二人零和完全情報ゲームの思考ルーチンはゲーム木の各ノードを評価し最善手を探索することで先読みを実現している．探索の際の計算量は先読みの深さによって増加する．図1にゲーム木の例を示す．

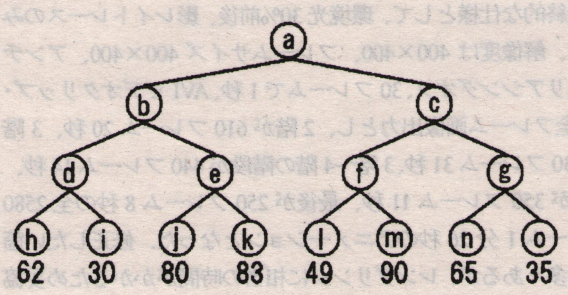

図1　ゲーム木

3.2 評価関数

評価関数とは，現在の局面が有利か不利かを数値で比較できるように評価する関数のことである．ここでは主に駒の価値によって評価を行っている．

3.3 MinMax 法

この方法は，自局面では評価が最大，相手局面では評価が最小となる着手を選択していき最善手を探索する方法である．単純に全ての局面を調べるため必ず最適解を求めることが出来るが探索効率が悪い．図1の例では全てのノードを探索して最善手順 a→c→g→n が求められる．

3.4 αβ法

MinMax 法において，αとβという二つの制限を設けて不要な探索をカットする方法をαβ法という．この方法は MinMax 法と同じ解が得られるにも関わらず探索数を減らすことができる．図2にカットの様子を示す．

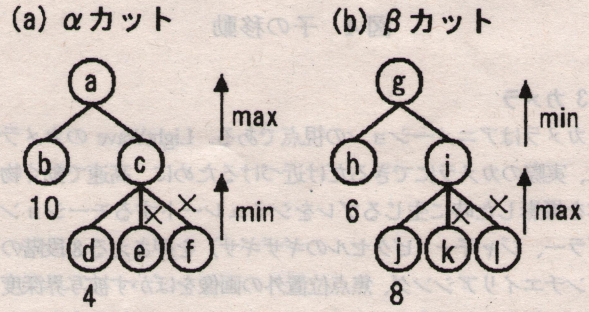

図2　αカットとβカット

図2 (a) では d が4だとわかった時点で c の値が b の値を超えないことがわかるため e，f は探索の必要が無くなる．一方，図2(b) では j が8だとわかった時点で i の値が h の値を下回らないことがわかるため k，l は探索の必要が無くなる．図1の例では k のノードがαβによってカットされる．

3.5 αβ法の改良

一度浅い探索を行うことでその時の最善手を記憶しておき，二度目以降の探索では先に求めた最善手から探索することで手の探索順序を最適に近いものにすることができる．このようなαβ法の効率化を反復深化という．

また，無駄だと思われる手を最初から探索しないことを前向き枝刈りという．今回は評価値がある一定の値を下回ったときにそれ以降のノードの探索を打ち切るものとした．これにより高速化が期待できるが，探索を打ち切ったノードの先に良い評

価を得られるノードがあったときにそれを見逃してしまうという問題が生じる.

4. システム概要

4.1 サーバ

サーバはクライアントから受信した着手が合法か非合法かを判断する. 非合法である場合, もう一度入力する旨を伝える. 合法である場合, 局面を更新しクライアントに送信する. 手番が回ってきたときは思考ルーチンが実行され合法手の中から最善手を検索し局面を更新, クライアントへ送信する.

4.2 クライアント

クライアントは自分の手番が回ってきたときに着手を入力してサーバへ送信する. 局面はサーバから受信した局面のデータを表示するだけである. 着手は移動元の場所と移動先の場所を指定する. ポーンのプロモートについては着手の後に駒の種類を指定する.

5. 実験

表1に示す動作環境で, 前項までに述べた探索アルゴリズム及び改良案をある局面に対して適用したときにかかる探索時間を測定した. クライアント機には動作環境①のコンピュータを使用した. 図4に動作環境①と動作環境②においてαβ法を適用したときの探索時間を示す. 図5, 図6は動作環境②においてのそれぞれの探索時間を示している.

表1 サーバ機の動作環境

	動作環境 ①	動作環境 ②
OS	Windows2000	WindowsXP
CPU	PentiumIII 500MHz	Pentium4 2.6GHz
メモリ	192MB	512MB

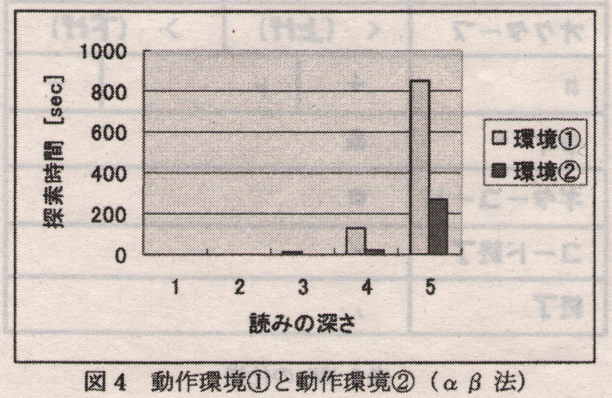

図4 動作環境①と動作環境②（αβ法）

これらの図より, MinMax 法を用いた探索と比べ相当数のノード探索が省略されて計算量が減少したことがわかる. また, クライアント機の性能に関わらずサーバ機の性能が良ければ高速な探索が実現できることがわかる.

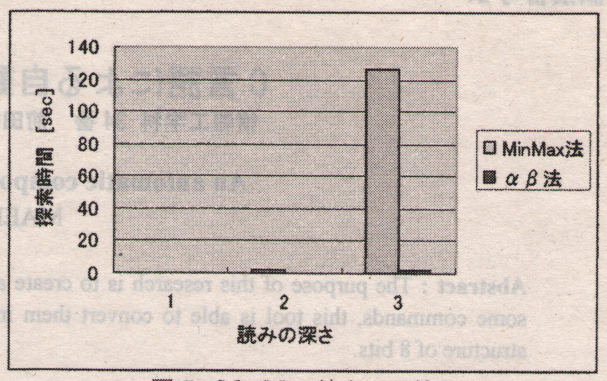

図5 MinMax 法とαβ法

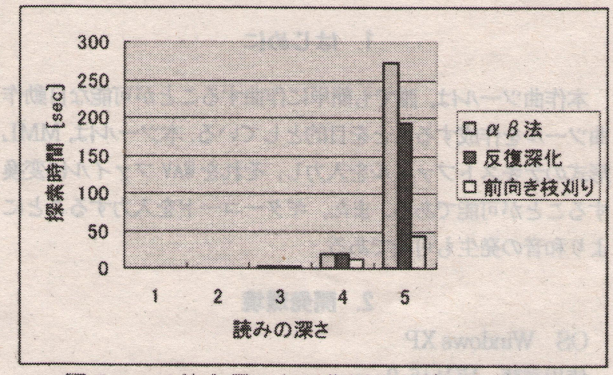

図6 αβ法を用いた反復深化と前向き枝刈り

6. まとめ

コンピュータチェスのアプリケーションを作成して思考ルーチンに使用される代表的なアルゴリズムについて比較した. また, αβ法の特徴を活かし思考ルーチンの高速化を行った. その結果, 計算量を削減することができ探索アルゴリズムの改良は成功した.

また, 動作環境①のような性能の低いコンピュータでも, サーバ・クライアント方式を導入し処理を分割することで高速な思考ルーチンを実現することが出来た.

7. 今後の課題

ゲーム木の探索時間はハードウェアの性能に大きく左右されるため複数台のコンピュータを用いて処理を分散させることで高速化を目指す. また, 評価関数の改善や定跡の導入などで無駄な着手を省く. 更に, GUI によるインターフェースを作成する. なお, 本ソフトウェアはオープンソースにて公開予定である.

参 考 文 献

[1] 池 泰弘 著: *コンピュータ将棋のアルゴリズム*(2005).
[2] 粂井 康孝 著: *猫でもわかるネットワークプログラミング*(2003).
[3] http://www.lcv.ne.jp/~yoshito/
[4] http://wisdom.sakura.ne.jp/system/winapi/winsock/winSock1.html