C-008

$\alpha\beta$ 法を用いた Trax ゲームソルバに関する研究
Study for Trax game solver with alpha-beta pruning

坂本 洋平 † 松田 紘作 † 大久保 慎也 ‡ 長名 保範 †
Yousuke Sakamoto Kousaku Matsuda Shinya Okubo Yasunori Osana

1 はじめに

近年、人工知能の研究は活発に行われておりその性能は著しく向上している。人工知能研究の一分野であるゲーム情報学においても 1997 年にコンピュータチェスの Deep blue が世界チャンピオンに勝利したことを機に研究が盛んに行われている。ゲーム情報学では主に明確化されたルールのもとで、コンピュータが人間とどこまで対等に戦うことができるかという研究が行われている。

本論文ではボードゲームの一種である Trax の強力なゲームソルバの作成を目標としている。このゲームソルバは幾つかの定石を検出するとともに完全情報ゲームにおける探索アルゴリズムの一種である $\alpha\beta$ 法を用いて設計している。また、FPGA アクセラレーションで実装することを最終的な目標としている。Trax だけでなく、リバーシやブロッカス デュオなどのゲームソルバ開発の研究が現在も各国で行われている [1] [2]。

2 プログラム構成とアルゴリズム

Trax は 1980 年に考案されたストラテジーゲームである。二人のプレイヤーは赤と白の線が描かれたタイル（図 1）を用い、線をつなげていきラインと呼ばれるパターン、あるいはループを形成したプレイヤーが勝者となる。ただしラインで勝利することは稀であるため、ここでの説明はループに絞って行う。

このゲームでは 1 ターンに置かれるタイルの数が一定ではない。また、防がなければ次のターンで敗北する状態をループアタック、ループアタックが二つ以上できる状態をマルチループアタック、あと一手でループアタックを作ることができる状態をコーナーをという [3]。

2.1 プログラム構成

ゲームソルバのクラス構成を図 2 に示す。まず基底クラスである trax_board は盤面情報を 2 次元配列で保持し、与えられた手が反則でないかの判断および勝敗の判定を行う。

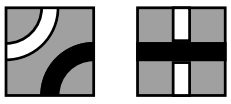

図 1 Trax で用いられるタイル

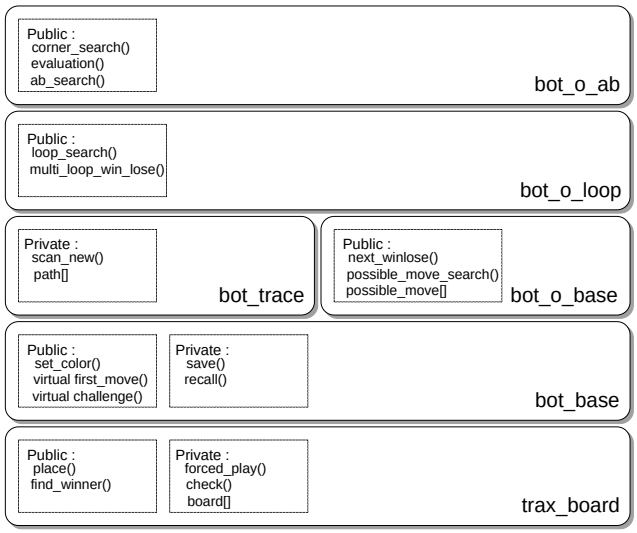

図 2 ゲームソルバのクラス構造

trax_board の上に bot_base があり、このクラスでは相手の入力に対しソルバ自体が手を返すための関数が実装されている。またソルバではあるタイルを盤面に仮想的に置き、それが合法手であるかの判別し、合法であった場合盤面の評価を行うという処理を繰り返す。よって盤面の状況を保存、復元する機能が必須である。それらの機能もこのクラスで実装した。

bot_o_base では合法手を探索、保持し、以降の探索はその保持した合法手を用い行う。またこのターンで勝利する場合その手を選択し、次のターンで敗北する場合それを防ぐ。

盤面の評価を行うためには自分または相手の線がどこで始まりどこで終わっているかというパスをリストとして保持しておく必要がある。そのための関数を bot_trace に実装した。

bot_o_loop は bot_o_base と bot_trace を多重継承し、双方のクラス内の関数を利用することができる。このクラスでは bot_trace のパスリストを用い、ループアタックの検出を行っている。このターン自分のマルチループアタックを行うことができる場合その手を選択する。また次のターン相手にマルチループアタックが行われる場合それを防ぐ。

bot_o_ab ではまずコーナーを探索する。探索した自分と相手のループアタックとコーナーの数から盤面の状態に点数をつける。そのための計算式を評価関数、点数を評価と呼ぶ。その評価を用いた $\alpha\beta$ 法により最適な手を選択する。

† 琉球大学工学部
‡ 琉球大学大学院理工学研究科

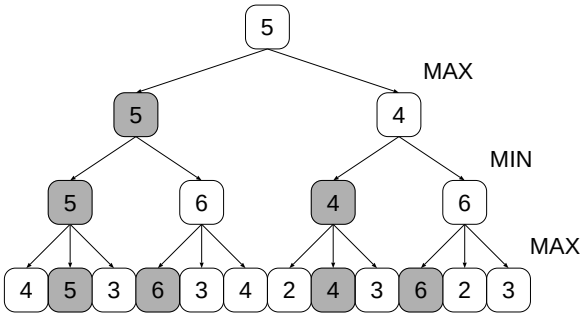

図 3　minmax 法における探索木

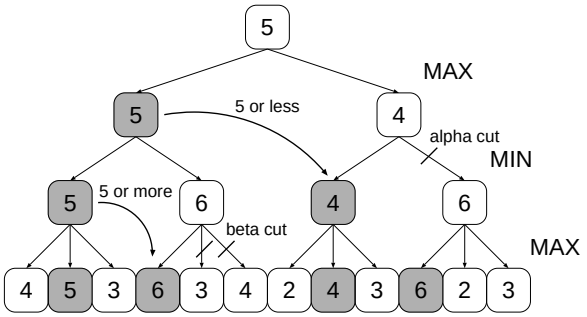

図 4　$\alpha\beta$ 法における探索木

2.2　アルゴリズム

まずゲームの探索において終盤まで全ての手を探索するにはとてつもなく膨大な量の計算を行う必要があり、現実的に不可能である。よって実際には現在の手からある程度先までの探索を行う。今回は 3 手まで探索する場合について説明する。$\alpha\beta$ 法は minmax 法を改良した探索アルゴリズムであるためまず minmax 法について解説する [4]。

minmax 法は自分が最善の手を打ちその次に相手が自分にとって最も不利な手を打つことを想定した評価値探索法である。図 3 に minmax 法によるの探索木を示す。minmax 法はまず 3 手置き、その時の盤面の評価を行う。それを何度も繰り返し全ての通りの盤面で評価を計算する。この探索法により 3 手先において自分が最も有利な状況となるようなプレイングを行うことができる。しかし minmax 法では探索木の全ての末端を探索するため計算量が膨大なものとなる。そのため探索量を削減するために改良した手法が $\alpha\beta$ 法である。

$\alpha\beta$ 法では minimax 法の探索木において枝刈りをすることで計算量を減らしている。最終的な結果は minmax 法と同一なものとなり探索時間のみが異なる。$\alpha\beta$ 法の探索木を図 4 に示す。図 4 における 4 段目のにおいて 1 通りの 4 段目の探索が終了しその評価の最大値である 5 が 3 段目に保持された時、以降の探索で 5 以上の評価が算出された場合その後の計算を行わず、そのままその評価を選択する。よって 6 が算出された場合その後の探索は行わず 6 を 3 段目で保持する。同様に、1 通りの 2 段目の評価が 5 と決定し、それ以降に 3 段目が計算された時その評価が 5 以下ならばそれ以降の計算を行わない。このようにして 2 種類の枝刈りを行うことで探索時間を大幅に短縮することができる。

表 1　実行時間プロファイルの結果と通りの合計数

ソルバ	実行時間 [ms]	通りの合計数
minmax 法	2498	136574
$\alpha\beta$ 法	223	11104
チューニング後	32	6687

3　実装と評価

現在ゲームソルバの実装は全てソフトウェアで行っている。プログラミング言語は C++ を用い、2.1 で述べたようにクラス継承を多用し複雑なアルゴリズムを作成している。探索アルゴリズムとしては $\alpha\beta$ 法を使用している。実装に伴いアルゴリズムの性能向上及び探索時間の削減に成功した。また $\alpha\beta$ 法の実装後にプロファイルを行い、その結果を参考にチューニングを行うことでさらなる性能向上を実現した。

評価としてある 7 手の棋譜において minmax 法を実装したソルバ、$\alpha\beta$ 法を実装したソルバ、チューニングを行ったソルバの 3 つのソルバにおけるプロファイルを行いその実行時間を調べた。また計算したゲーム展開の通りの合計数を算出した。その結果を表 1 に示す。このように実行時間の大きな短縮に成功している。実行時間を短縮することでさらに多くの探索を行うことが出来るためゲームソルバの性能においても向上していると言える。

4　おわりに

本研究ではボードゲームの一種である Trax におけるゲームソルバを作成した。基底となる Trax のプレイを可能とするクラスに様々な機能を追加することで性能を向上させるとともに計算量が膨大となるアルゴリズムを改良を繰り返し探索時間の削減に成功した。

またプロファイルを行い各関数毎の実行時間の割合を知ることが出来たためハードウェア化する際の検討に用いることが出来るのではないかと考えている。

参考文献

[1] Javier Resano Javier Olivito, Carlos González. FPGA implementation of a strong reversi player. In *Proceedings of the 2010 International Conference on Field-Programmable Technology (FPT)*, 2010.

[2] Erik Altman Joshua S. Auerbach David F. Bacon Ioana Baldini Perry Cheng Stephen J. Fink Rodric M. Rabbah. The Liquid Metal Blokus Duo design. In *Proceedings of the 2013 International Conference on Field-Programmable Technology (FPT)*, 2013.

[3] Donald Bailey. *Trax Strategy For Beginners*. Donald Bailey, 1997.

[4] 小谷善行, 岸本章宏, 柴原一友, 鈴木豪. ゲーム計算メカニズム. コロナ社, 2010.