修士論文

UMLとNSLによる二人完全情報ゲームAIに関する研究

学籍番号　4BJNM017
氏名 玉城　良
指導教員　清水　尚彦　印

東海大学大学院
情報通信学研究科

目次

図 目 次

表 目 次

第1章　はじめに

1.1　研究背景

人工知能分野の一つである二人完全情報ゲームの AI（Artificial Intelligence）に関する研究は古くから行われており，その中でも探索問題について取り組まれてきた．二人完全情報ゲームは，プレイヤが自分の手番の時に相手の行った情報を全て知ることが出来る．さらにゲームにおいて盤面の状態が有限であるため，相手の手を予測することができる．総当たりで局面を探索するには，チェスでは 10^{123}，将棋が 10^{220}，囲碁が 10^{360} と，探索すべき局面の数が問題となる．1950 年には C.Shannon がチェスの探索問題に対して，図 1.1 のような評価関数に基づくミニマックス法，すなわちゲーム木探索を提唱した [1]．ゲーム木探索は，総当り探索でなく任意の深さや幅まで探索を行い，評価関数によって局面の価値を判断する．

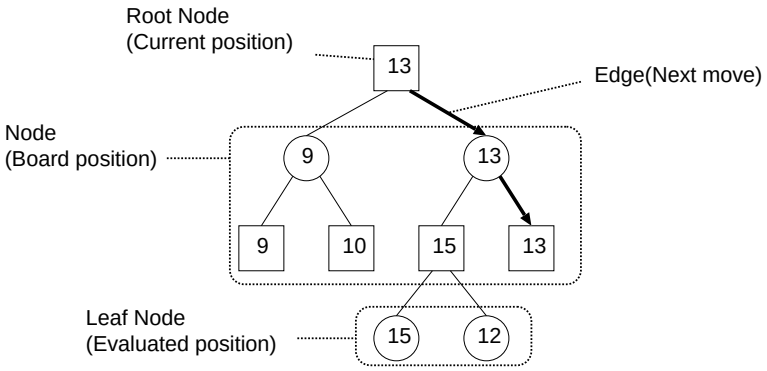

図 1.1: ゲーム木探索の例

1.2 関連研究

コンピュータゲーム研究の一つの目的は，強いゲーム AI（Artificial Intelligence）の開発であり，ゲーム AI はゲームの特徴を考慮した局面評価関数（局面を評価し数値化する関数）と，相手より多くの局面を先読みし評価することが重要視される．

1.2.1 評価関数

評価関数は，探索した局面が AI にとって，その局面までのゲーム展開の優劣を数値化するものである．評価した数値が大きければその局面は AI にとって有利であり，小さければ不利である．評価関数の作成において，局面の何を評価すべきで，どの評価項目の重み付けを行うかを正確に作成する手法が未だ確立されていない．

以下で評価関数のパラメータ調整に関連する研究について述べる．まず機械的な手法に頼らない方法として，人の手によって評価関数を作成するものが考えられる．人の手による評価関数の作成は，評価項目や評価の重み付けは人が調整することになる．評価関数の調整における課題は，評価項目（局面の特徴）が多量であった場合，膨大なパターンのパラメータを調整し，試行錯誤しなければならない点である．

人の手による調整では評価関数の強化に限界があるため，人がコンピュータにプログラムせずに，コンピュータ自身が学習するアルゴリズムが研究されてきた．そのような学習アルゴリズムは，機械学習と呼ばれ 1959 年にサミュエルによって定義されている []．機械学習は，教師あり学習（Supervised learining）や強化学習（Reinforcement learning）・遺伝的アルゴリズム（Genetic Algorithm：GA）などを分類している．

教師あり学習は，棋譜のデータを教師として，評価関数のパラメータを自動的に調整するものである．教師あり学習の代表的な応用例では，保木により開発されたコンピュータ将棋ソフトである Bonanza が挙げられる []．Bonanza は，プロ棋士等の棋譜約 6 万局を教師として学習を行っており，この評価関数のパラメータ自動学習はコンピュータゲームにおいて多くの功績をもたらしている．今日では，教師あり学習において，どのように教師データを用意するかが課題となっている．

遺伝的アルゴリズム（GA と呼ぶ）は，多量のランダムなパターンのパラメータの中から最も解に近似したパターンを進化させていくアルゴリズムである．目標の強さを GA の解として設定することで，その強さと

なるように評価関数のパラメータを調整している研究が行われている []. これは教師データ（人の棋譜）を GA の解に設定し，教師データに近づくようにパラメータを収束させている．本研究では，目標の強さを解として遺伝的アルゴリズム上で設定せずに，教師なしで，より強い評価関数のパラメータを自動で調整できるか実験する．

1.2.2 ゲーム木探索

ゲーム AI を強くするためには，評価関数による正確なゲーム展開の判断と，より深い先読みが重要となる．より深い先読みをするための手法としては，アルゴリズムの工夫やハードウェア性能の向上，並列化などの計算方法の工夫が挙げられる．

ゲーム木探索は探索した中で最大の損害を最小とするように着手選択を行う．このような戦略をミニマックス法と呼ぶ．ミニマックス法は，自分の手番の探索ノードでは最も評価値の高いエッジを選択し，相手の手番においては最も低い評価値のエッジを選択する．しかしながらミニマックス法は与えられた探索深さまでの全てのノードを探索するため，探索しなくてもよいノードまで探索している．ミニマックス法に木の枝刈りの要素を取り入れているのがアルファベータ法である []．アルファベータ法は探索する必要のないノードをカットすることで探索効率を向上させている．このように，高速な探索のためにアルゴリズムの改良や試行錯誤が従来行われている．

1997 年には，IBM のチェス AI である DeepBlue がチェスチャンピオン G.Kasparov に勝利した [2]．スーパーコンピュータである DeepBlue は，チップ設計概念であるマルチコアの先駆けとして，32 個のプロセッサにより 1 秒間に約 2 億手の探索を可能とした．DeepBlue はこの勝利により，高速演算によるゲーム木探索の有用性を示した．加えて，後継機である IBM Blue Gene®が科学分野の研究所などで稼働しており，ゲーム AI 研究で培った要素技術が他分野に応用できることを示した．

近年，チップ設計技術の新たな知見を得るために，ICFPT，HEART，RECONF などの各研究会において，二人完全情報ゲームである Blokus Duo や Trax などのゲーム AI を FPGA（Field Programmable Gate Array）に実装し，対戦させるコンテストが開催されている [3]．我々は，ゲーム木探索を並列化することによる高速な演算を目指し，二人完全情報ゲームである Blokus Duo と Trax のゲーム AI を専用回路として開発する．

1.3 Blokus Duo専用回路

我々の開発したBlokus Duo専用回路は，高位合成言語NSL（Next Synthesis Language）により設計し，3並列でゲーム木を探索し，ソフトウェア対比で約8倍の性能であった [4]．アーキテクチャとして，ステートマシンとスタックメモリによりゲーム木探索を実現した．回路規模は，CycloneIV EP4CE115 においてFPGAチップ全体の85%（LE数：96,411）を占め，ゲームAI専用回路が大規模かつ複雑であることを示した．

更なる高速化のためには，ゲームAI専用回路の並列度の向上が必要である．並列度向上には，回路規模を縮小させなければならない．本研究では回路規模を縮小させ，並列度を向上させることで，更なるゲーム木探索処理の高速化を目指している．

第2章　当研究の手法

新たに開発する Trax 専用回路の設計方針として，回路規模を抑えて並列度を向上させることで，更なる演算の高速化を目指す．大規模かつ複雑なゲーム AI 専用回路に対して，UML（Unified Modeling Language）を用いるモデルベースのハードウェア設計を導入し，設計複雑性を緩和させる．回路構造の設計においては，クラス図を用いる．クラス図の要素部に入出力信号やレジスタを記述し，操作部には制御信号やステートを記述して視覚的にハードウェアの構造を設計する．回路の動作やロジックの設計においては，NSL で記述する．図 2.1 に本開発方法の流れを示す．

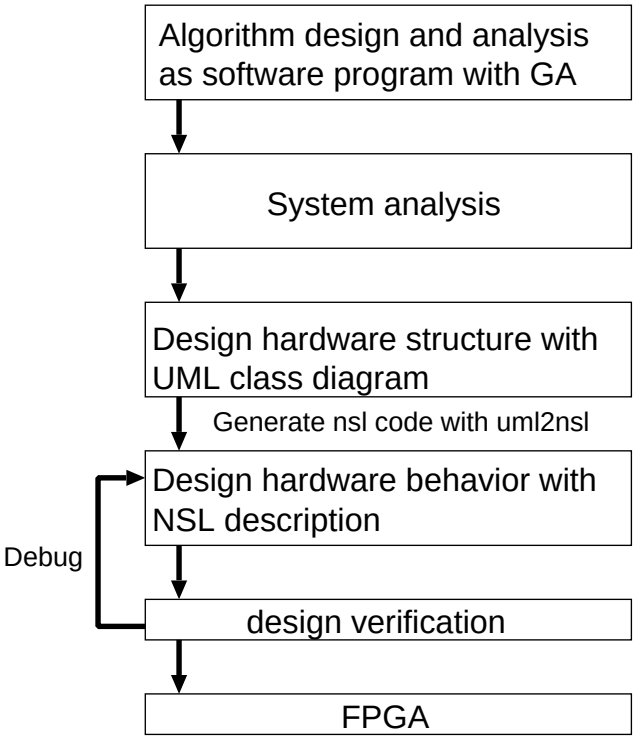

図 2.1: 本開発方法の流れ

第3章　Trax専用回路

3.1　システム概要

Trax 専用回路は，図 3.1 に示すように構成している．Terasic 社製 FPGA ボードの FPGA CycloneIV EP4CE115 上に回路を構成し動作させる．対戦は，シリアル通信を用いてホスト PC を介して行う．FPGA の内部構成として，大きく分けて，通信モジュール UART とゲーム木探索制御モジュール SearchController で構成している．通信モジュール UART は，データ受信モジュール Reciever と，復号化モジュール Decoder，符号化モジュール Encoder，データ送信モジュール Sender により構成する．

Reciever は，データ受信信号 RXD から受け取った 1 ビットデータを ASCII コードに変換し，バッファリングする．バッファリングは，16 進数データ 0x2a を受信したときに完了し，バッファに格納したデータを Decoder により復号化し，手データ Move として SearchController に信号を送り探索を開始する．探索が終了したとき，Move を Encoder に入力し，符号化を行った後，Sender によりデータ送信を行うことで，対戦を行う．

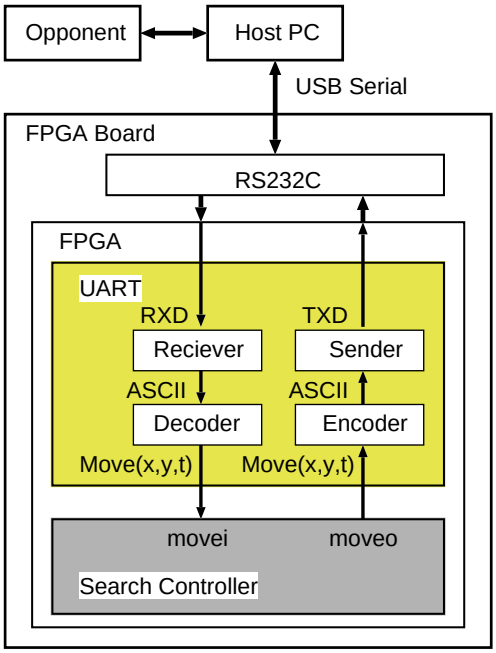

図 3.1: ゲーム AI 専用回路のシステムブロック図

図 3.2 にシステムのクラス図を示す．DE2_115 モジュールをトップモジュールとし，通信モジュール UART と探索制御モジュール SearchController を子モジュールとしてインスタンス化している．DE2_115 モジュールは，UART と SearchController の制御バス，データバスとして構成している．

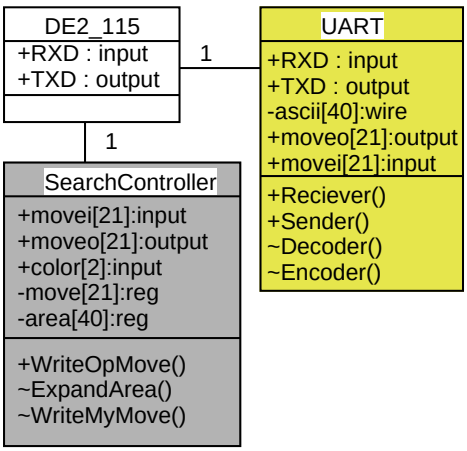

図 3.2: ゲーム AI 専用回路のシステムクラス図

3.2　ゲーム木探索制御モジュール

ゲーム木探索制御モジュール SearchController の内部構成と処理の流れを図 3.3 と図 3.4 に示す．また，表 3.1 に主なモジュールとレジスタの概要を示す．処理の流れとして，はじめに SearchController は相手の手 movei を盤面メモリに書き込む．次に，盤面サイズを手の配置された方向に拡張し，更新した盤面状態で探索をはじめる．

探索するための構成として，ノード処理モジュール NPU（Node Processing Unit）と評価関数 Eval を親ノード，子ノードの関係性をもたせるように制御信号で配線した．NPU1 から探索が開始され，NPU1 が子ノードを呼ぶとき，NPU0 が起動する．NPU0 が子ノードを呼ぶ場合は，Eval を起動させ局面の評価値を得る．NPU0 が探索を終了し親ノードを呼んだとき，NPU1 が起動する．NPU1 が親ノード呼び出しをした場合，探索が終了する．この探索構造において，NPU のユニット数を拡張することで，更に探索を深くすることが出来る．

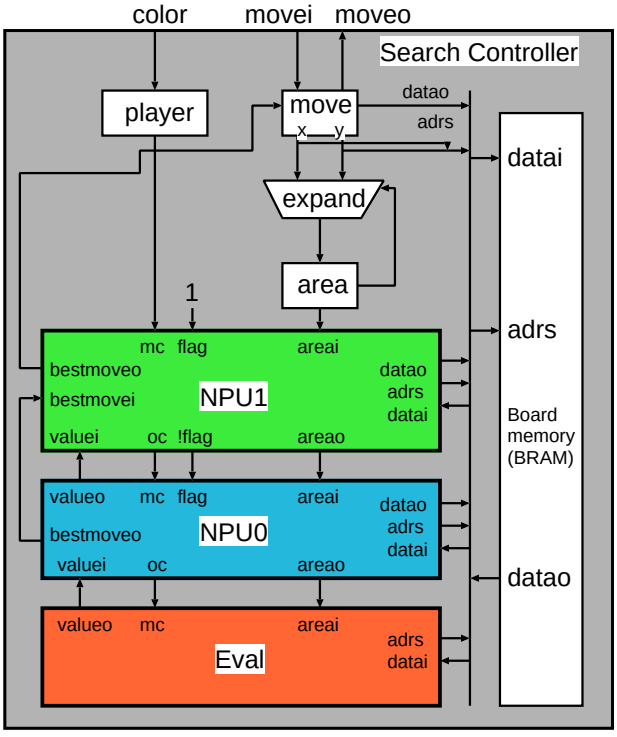

図 3.3: ゲーム木探索制御モジュールのブロック図

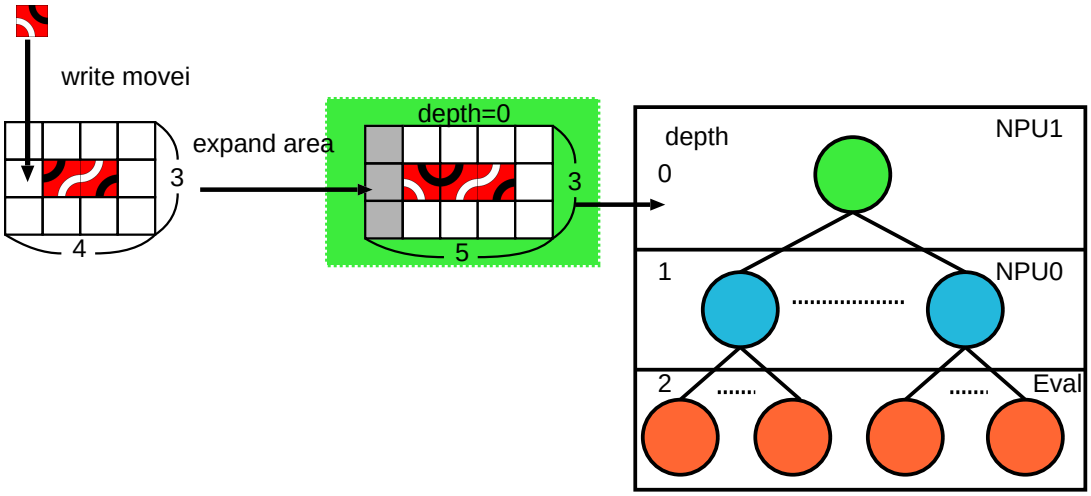

図 3.4: ゲーム木探索制御モジュールの処理の流れ

表 3.1: ゲーム木探索制御モジュールの内部構成概要

NPU1 モジュール	深さ 0 の幅探索
NPU0 モジュール	深さ 1 の幅探索
Eval モジュール	評価関数
player レジスタ	自分の線の色
move レジスタ	手データ（X 座標，Y 座標，タイル）
area レジスタ	盤面サイズ（上端，下端，左端，右端）

3.3 ノード処理モジュール

ノード処理モジュール NPU は，ノード内部において，盤面状態の更新，評価値の比較，親ノード呼び出し，子ノード呼び出しを行う．図 3.5 に，NPU のブロック図を示す．NPU は状態遷移機械により，各処理を順次実行する．

ノード処理を行うための NPU の各処理と制御信号を表 3.2 に示す．NPU は，親ノードからの制御入力である Call により探索を開始する．はじめに，Init によるデータ初期化を行う．初期化するデータは，GenerateMove 内の手を生成するためのカウンタと評価値 value レジスタを初期化する．GenerateMove 処理は，X 座標，Y 座標，タイルデータのカウンタにより，手データ move の生成を行う．CheckMove 処理では，生成した move がその局面において，合法かどうかチェックする．合法･非合法の判断は，Trax ルールに準拠して行う．move が非合法である場合，再度 GenerateMove 処理を行う．合法である場合は，PlaceMove 処理により盤面を更新する．JudgeWin 処理は，更新した盤面が勝利条件を満たしているか判定する．盤面が勝利条件を満たしていない場合は，ToChild 制御出力により，子ノード NPU の探索開始を親モジュールに依頼する．子ノードが探索を終了したとき，親モジュールから制御入力 Return によりノード処理を再帰する．CompEval 処理では，子ノードから返った評価値と，前の探索で得た評価値を比較し，価値を評価する．UndoMove 処理では，次のノードを評価するために，一手分盤面状態を戻し，再び GenerateMove 処理に遷移する．GenerateMove 処理がカウントを終えたとき，制御出力 ToParent により探索終了の信号を送る．

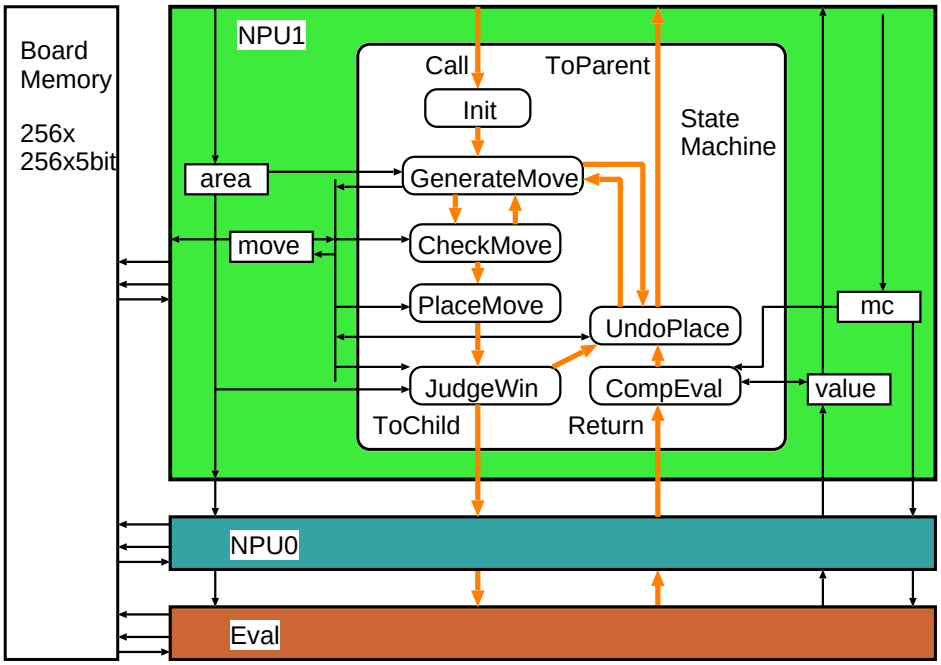

図 3.5: ノード処理モジュール NPU のブロック図

表 3.2: NPU の制御信号と手続き処理

制御入力信号	Call	探索開始入力
制御入力信号	Return	再帰入力
制御出力信号	ToChild	子ノード呼び出し
制御出力信号	ToParent	親ノード呼び出し
手続き	Init	探索データの初期化
手続き	GenerateMove	手データ move の生成
手続き	CheckMove	move が合法かチェック
手続き	PlaceMove	move で盤面状態を更新
手続き	JudgeWin	終局かチェック
手続き	CompEval	評価値による局面比較
手続き	UndoPlace	盤面状態を元に戻す

3.3.1 NPU 内部構成

図 3.6 に NPU のクラス図を示す．ノード処理を行うための手続きにおいて，1 クロックで処理が終わらないものは，可読性のためにモジュール分けを行った．NPU は，手生成モジュール Gen と，合法チェックモジュール Check，終局判定モジュール Judge を子モジュールとして持つ．Gen は，GenerateMove 処理を行った際に起動し，内部で手データ move を持ち，起動毎に move をカウントして出力する．Check は，move を入力とし，判定結果を返す．Check は，move の周辺座標の盤面データを読み出すことで合法チェックを行なっている．Judge は，盤面上に勝利条件であるループとヴィクトリーラインがあるか判定している．

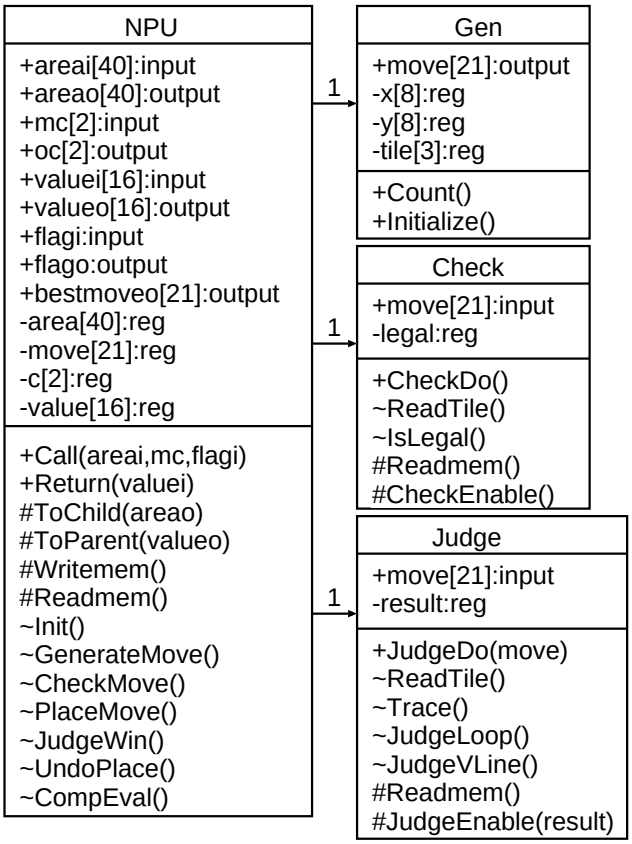

図 3.6: ノード処理モジュールのクラス図

3.4 評価関数

評価関数モジュール Eval は，盤面状態を入力として，局面の価値を数値化する．評価関数は，重み付き線形多項式として評価値を算出する．各項は Trax 盤面状態の特徴を数値化（評価パラメータ）したものである．また，我々は機械学習に分類される遺伝的アルゴリズム（GA：Genetic Algorithm）を用いて，最適な評価パラメータの算出を試みた [6]．GA によって算出した評価パラメータ AI と，ランダムパラメータ AI，人間を対戦相手とした結果を Table.3.3 に示す．結果として，人間に勝利出来る強さのパラメータを算出したことがわかる．

表 3.3: Trax における GA 算出パラメータ AI の対戦結果

相手	対戦数	勝	負	引分
ランダムパラメータ AI	50	24	4	22
人間	5	5	0	0

3.5 実装結果

Blokus Duo と Trax の専用回路の FPGA 実装の結果を Table.3.4 に示す．速度と対戦検証環境として，Blokus Duo と Trax の GUI プラットフォームをそれぞれ開発した．Trax はゲーム展開により盤サイズが異なるため，Trax の計算時間測定は Blokus Duo の盤サイズである 14x14 盤にゲームを展開して測定した．ソフトウェアゲーム AI の計算時間は，Intel Core i3，4GB メモリで動作する LinuxOS で測定した．

Trax 専用回路は並列度が 1 であるにも関わらず並列度 3 の Blokus Duo 専用回路より計算時間が短かった．Blokus Duo は，お互いが盤面に手を置けなくなったら終局となる．一方 Trax は，終局パターンを最短で 3 手番で形成でき，終局分だけ Trax ゲーム木のノード数が少なくなることから，Trax の計算時間が Blokus Duo より短いことが考察できる．

ゲーム AI 専用回路はシステム内で共有しなければならないデータが多く，機能分けがしづらい．Blokus Duo の設計では，一つのモジュールに機能が依存しており，並列化した際に，複製する必要のない回路まで複製した結果，回路規模が増大した．また，Blokus Duo の 14x14 の盤面データを，FPGA のブロック RAM としてでなく，レジスタとして用意したことも，回路増大の原因として見れる．

表 3.4: CycloneIV EP4CE115 への実装結果

システム	Blokus Duo	Trax
回路規模（TLE 数）	85%(96,411)	6%(6,320)
14x14 盤の計算時間 [S]	8.24[S]	0.59[S]
探索並列度	3	1
SW 対比性能	約 8 倍	約 2 倍
動作周波数	50MHz	50MHz

第4章　おわりに

大規模かつ複雑な回路設計に対して，UML によるモデルベースのハードウェア開発を導入した．視覚的な見通しの良い設計手法により，設計の複雑性を回避し，冗長性を無くすようにした結果，既存の Blokus Duo 専用回路と比較して，Trax 専用回路は回路規模を削減することが出来た．

今後は，ノード処理回路 NPU を並列で動作させ，ゲーム木探索専用回路の性能向上を目指す．

付 録A　Trax評価パラメータの準最適化

付録 A.1　遺伝的アルゴリズム

Trax はチェスや囲碁，将棋と比較して研究が進んでおらず，定石や棋譜が豊富でない．人力による評価パラメータの設定は，ヒューリスティックや，設計者の棋力が必要である．Trax のように，研究段階のゲームにおいて，ゲーム初心者による評価関数設計は困難である．

我々は，そのような問題に対し，遺伝的アルゴリズムを用いて，Trax ゲーム AI の評価関数のパラメータの準最適化を行うことにした．

遺伝的アルゴリズム（GA：Genetic Algorithm）は，Holland により提案された生物進化の過程を工学モデルとしたメタヒューリスティックアルゴリズムである [7].

GA は，乱数として生成した数列を一つの個体とし，任意の個体数を用意することで集団としている．GA で解きたい問題に対して適応度関数を作成し，個体の持つパラメータを適応度関数に入力することで，個体の優秀度を算出する．集団の中で，適応度の高い個体を優先して生存させていき，個体同士でパラメータを交叉させたり，突然変異により新たな個体を生み出し，それを何世代も繰り返すことにより，優秀な個体を算出させていく．また，GA では個体数を任意の数としているため，GA の解探索は必ずしも最適とは限らない．

付録 A.2 Trax 評価パラメータ

ゲーム木探索において，局面の評価は葉ノードで行う．TRAX プレイヤの評価関数を線形多項式とし，以下に列挙する評価項目を式の項として定義する．また，$_{my}$ は自分の手の情報，$_{op}$ は相手の手の情報を表す．

- LO_{my}, LO_{op}: ループを形成したか．
- LI_{my}, LI_{op}: ヴィクトリーラインを形成したか．
- NFP: 連鎖の数．
- AZ_{my}, AZ_{op}: 図 A.1 上段左に示す，始端と終端の方向の角度が 0 度になる線の数．始端と終端が隣り合って配置されている場合，次の一手によりループになり得る．
- AN_{my}, AN_{op}: 図 A.1 上段右に示す，角度が 90 度になる線の数．この角度からのループやヴィクトリーラインの形成は，連鎖を使用する場合を除いた場合 2 手以上かかるため，相手への妨害手として有効である．
- AH_{my}, AH_{op}: 図 A.1 下段左に示す，角度が 180 度になる線の数．始端から終端までの長さが 7 ある場合，次の一手によりヴィクトリーラインになり得る．
- AT_{my}, AT_{op}:図 A.1 下段右に示す，角度が 270 度になる線の数．効果は，角度が 90 度の線と同様である．

TRAX プレイヤの評価関数は，式 (A.1) に示すように，上記で定義した評価項目とその係数 A-M から成る線形多項式である．

$$\begin{aligned} v \;=\; & LO_{my}A + LO_{op}B + LI_{my}C + LI_{op}D + NFPE + AZ_{my}F \\ & + AZ_{op}G + AZ_{my}H + AN_{op}I + AH_{my}J + AH_{op}K \\ & + AT_{my}L + AT_{op}M \end{aligned} \tag{A.1}$$

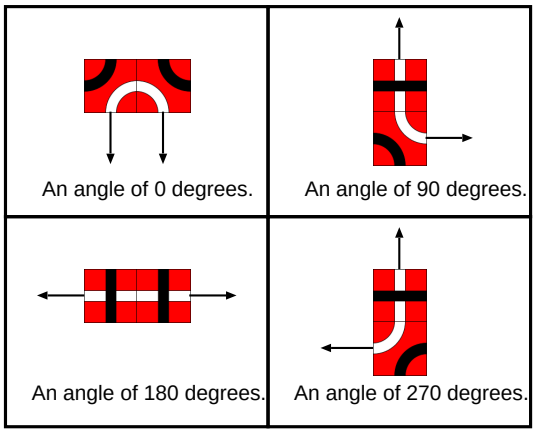

図 A.1: 上段左：始端と終端の方向の角度が 0 度になる白線．上段右：角度が 90 度になる白線．下段左：角度が 180 度になる白線．下段右：角度が 270 度になる白線．

付録 A.3 適応度関数

我々は，Trax ゲーム AI を準最適化するために，適応度関数として Trax ゲームのホストプログラムを用意した．ホストプログラムには，引数として 2 つの個体を入力し，対戦を行わせた結果を，個体 1 の勝利（1），個体 2 の勝利（2），引分（0）として出力する．適応度は，総当たりで全ての個体同士を対戦させ，その勝率を求める．

付録 A.4 実行パラメータ

GA の実行フローを図 A.2 に示す．実行フローにおいて，世代数が 20 以上になった場合，GA の実行を終了する．以下は，GA の実行フローに対するパラメータを示す：

Initialize Population 200 体の乱数列個体を生成.

Evaluation Fitness Trax ホストプログラムで総当たり戦，勝率算出.

Reproduction 勝率上位 50%の個体を下位 50%の個体へ上書き.

Crossover ランダムで選出した 2 つの個体のパラメータの一部を交換.

Mutation ランダムで選出した 1 つの個体のパラメータの一部を反転.

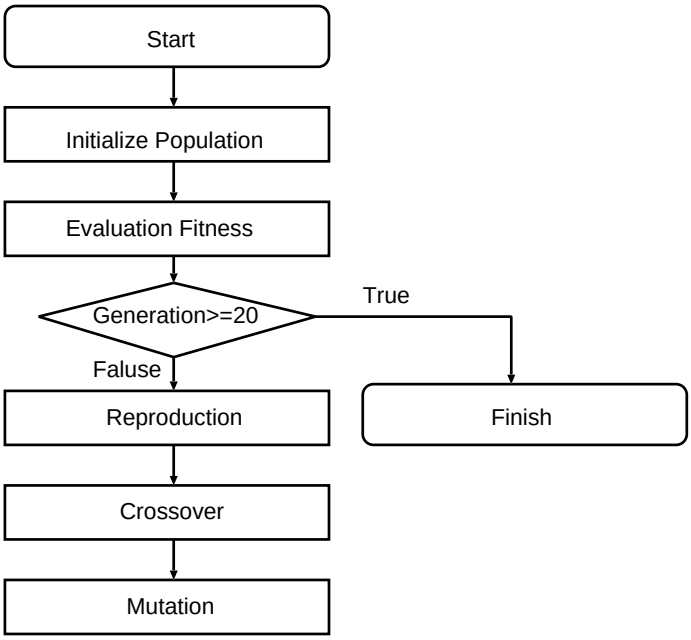

図 A.2: 遺伝的アルゴリズムの実行フロー

付録 A.5　　Reproduction

図 A.3 は，GA フローにおける Reproduction の実行を示す．200 個体の内，勝率上位 50%（1st-100th）に順位付けられた個体を勝率下位 50%（101th-200th）の個体へ上書きすることにより，より優秀な個体が生存できるようにする．

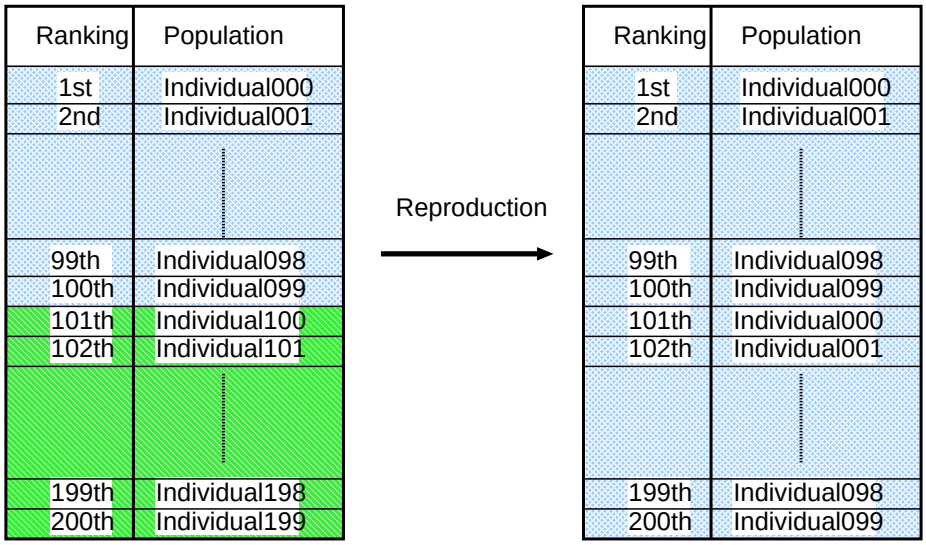

図 A.3: 勝率上位 50%再生

付録 A.6　　Crossover

図 A.4 は，GA フローにおける Crossover の実行を示す．200 個体の中から，ランダムに選択された 2 つの個体の間で，ランダムに選択されたパラメータを交叉させる．この実行は，優秀な個体同士のパラメータを交換することで，更に優秀な個体の生成を出来るようにする．

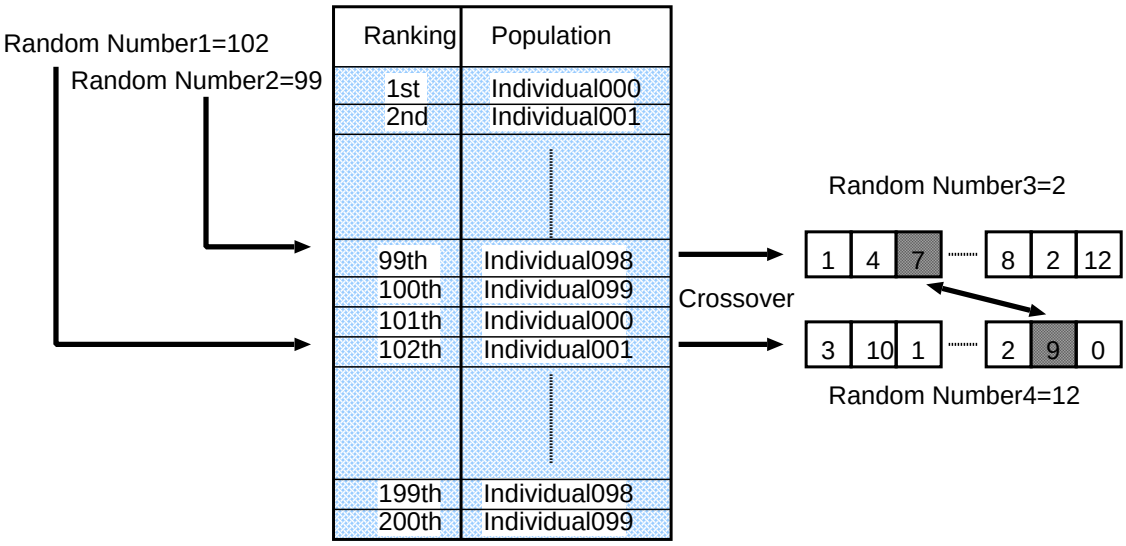

図 A.4: 2 個体間での一点交叉

付録 A.7　　Mutation

図 A.4 は，GA フローにおける Mutation の実行を示す．200 個体の中から，ランダムに選択された 1 つの個体の内，ランダムに選択されたパラメータの数値の何れかを反転させる．この実行は，個体を突然変異させることで，個体の収束を避ける．

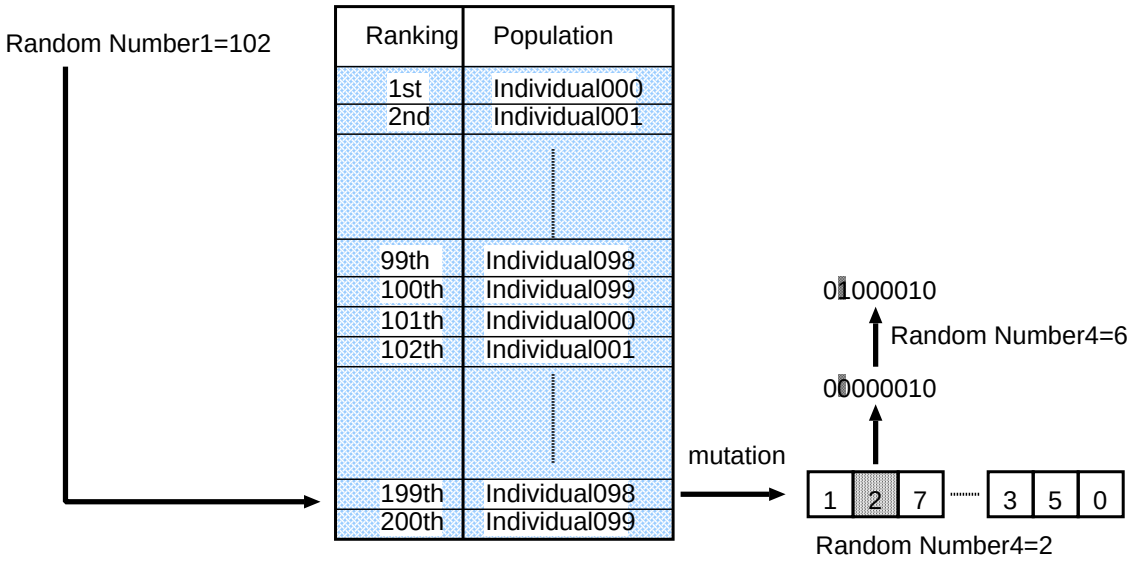

図 A.5: 突然変異

表 A.1: GA の実行結果

評価項目	準最適化パラメータ
LO_{my}	122
LO_{op}	-125
LI_{my}	36
LI_{op}	-107
NFP	-56
AZ_{my}	102
AZ_{op}	-37
AN_{my}	-4
AN_{op}	97
AH_{my}	-26
AH_{op}	-23
AT_{my}	-81
AT_{op}	58

付録 A.8　実験結果

GA では 200 通りのパラメータの TRAX プレイヤを対戦させ，その勝率を元に集団の中で適応度の高いプレイヤの発見，および進化的計算を行った．表 A.1 は，TRAX プレイ経験者の人間が設定したパラメータと，GA によって準最適化されたパラメータの数値を示す．準最適化パラメータの TRAX プレイヤは，自らのループ形成を基本とした戦略を取りつつ，相手のヴィクトリーラインやループ形成の妨害をするといった振る舞いを対戦の中で確認できた．

付 録B Trax

付録 B.9 Trax 基本ルール

本章は，二人完全情報ゲーム Trax の基本的なルールについて解説する．Trax は図 B.1 に示す 2 種類のタイルを用いて，相手と交代に図 B.2 のように線を繋げながら盤上に置いていく．線が繋がっていない場合や，違う色の線と接続した場合，違法手となる．

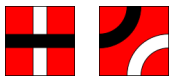

図 B.1: 使用するタイル

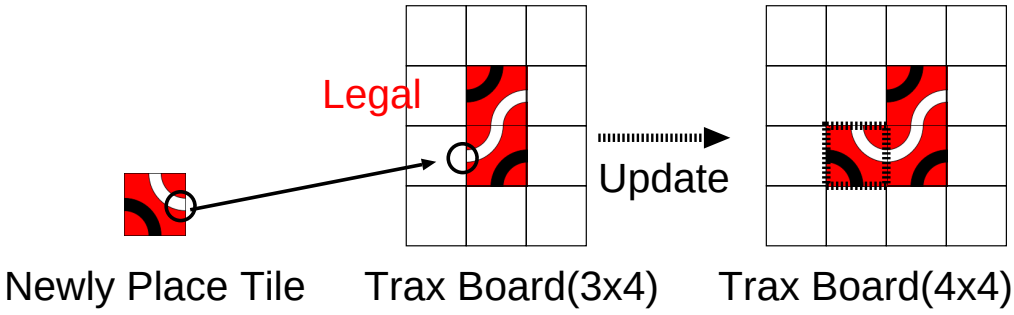

図 B.2: タイルの配置方法

付録 B.10 Trax 勝利条件

Trax における 2 つの勝利条件を図 B.3 に示す．図 B.3 上のループ局面は，白線が円を成形したため，白線プレイヤの勝利となる．ループを成形する線の長さは制限されていない．図 B.3 下のヴィクトリーライン局面は，白線が長さ 8 以上の線を成形したため，白線プレイヤの勝利となる．ヴィクトリーラインの成形条件は，長さ 8 以上かつ，線の両端が左右か上下を向いていた場合，条件が満たされる．

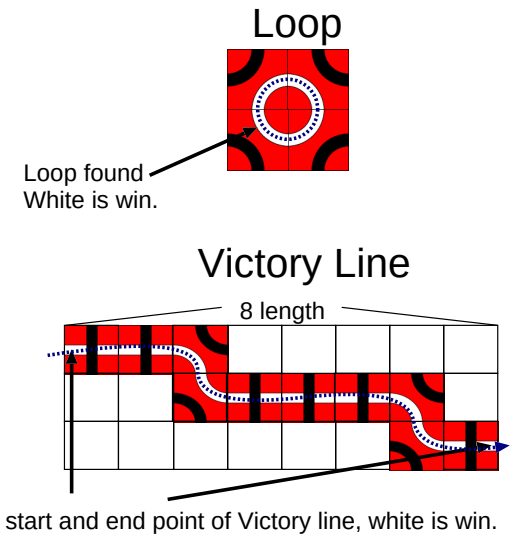

図 B.3: 勝利条件を満たす局面例：ループ（上図），ヴィクトリーライン（下図）

参考文献

[1] Claude Shannon, ”Programming a Computer for Playing Chess”, Philosophical Magazine, ser.7, vol.41, no.314, 1950.

[2] Feng-hsiung Hsu, ”IBM’s Deep Blue Chess Grandmaster Chips”, IEEE Computer Society Press, vol.19, pp.70-81, 1999.

[3] Trax/FPGA JP, `http://lut.eee.u-ryukyu.ac.jp/traxjp/index.html`, 2016-2.

[4] Ryo Tamaki, Naohiko Shimizu, ”Implementation of Game Tree Search Method by using NSL”, CSCEET2014, Malaysia, pp.99-104, 2014.

[5] Daiki KANO, Ryota YAMZAKI, Naohiko SHIMIZU, ”The Method For Hardware Design to Generate NSL from UML Diagram and The Experiments with FPGA”, ICCEETS2012, pp.594-601, 2012.

[6] 玉城　良，清水　尚彦，”TRAX プレイヤにおける遺伝的アルゴリズムを用いた評価関数の発見と UML モデルベース開発”，リコンフィギャラブルシステム研究会，信学技報，Vol.115，No.398，pp.237-242，2016-1.

[7] John H. Holland, ”Genetic Algorithm”, `http://www2.econ.iastate.edu/tesfatsi/holland.GAIntro.htm`,2015.

業績リスト

1. 梶原　太一, 野元　一馬, 何　スウショウ, 青山　卓也, 玉城　良, 廣瀬　翔太, 津端　祐亮, 秋山　大樹, 大塚　祐史, 清水　健太, 清水 尚彦, ”研究室導入教育の開発事例”, 第 38 回パルテノン研究会, Vol.38, pp.43-50, 2012.

2. 青山 卓也,玉城 良,廣瀬 翔太,清水 尚彦, ”NSL による Blokus Duo のハードウェアシステム・GUI 実装” 第 39 回パルテノン研究会, Vol.39, pp.43-46, 2013.

3. Ryo TAMAKI, Naohiko SHIMIZU, ”Implementation of BlokusDuo Hardware with high level behavioral language NSL”, ITC-CSCC2014, pp.602-605, 2014.

4. Ryo Tamaki, Naohiko Shimizu, ”Implementation of Game Tree Search Method by using NSL”, CSCEET2014, Malaysia, pp.99-104, 2014.

5. 玉城　良，清水　尚彦，”NSL による BlokusDuo Multi Game AI システムの実装”，東海大学紀要 情報通信学部，Vol.7, No.2, pp.9-16, 2015.

6. 玉城　良，清水　尚彦，”UML モデルベース開発を用いた TRAX プレイヤの設計事例”，第 41 回パルテノン研究会，vol.41, pp.45-50, 2015.

7. 玉城　良，清水　尚彦，”TRAX プレイヤにおける遺伝的アルゴリズムを用いた評価関数の発見と UML モデルベース開発”，リコンフィギャラブルシステム研究会，信学技報，Vol.115，No.398，pp.237-242，2016-1.