

Ms. Pac-Man におけるモンテカルロ木探索

池畑望[†] 伊藤毅志[†]

2007 年より IEEE の CIG シンポジウムの中で Ms. Pac-Man の自動コントロールを競う大会が開かれている。この大会以来、Ms. Pac-Man は、デジタルゲーム AI の研究対象として注目を集めつつある。これまでの大会では、知識ベースを用いた古典的な手法による AI が最も良い成績を収めているが、その性能には限界が見え始めており、知識ベースに代わる新しいアプローチが求められている。そこで、本稿では囲碁で成功したモンテカルロ木探索による Ms. Pac-Man コントローラーを実現し、その有効性を検証した。モンテカルロ木探索は乱数によって生成された未来局面についてのシミュレーションを繰返すことで、専門的知識に頼らずに期待値の高い次の手を求めることができる。性能評価実験ではモンテカルロ木探索によるコントローラーは過去に Ms. Pac-Man Competition に参加した全てのプログラムよりも優秀な成績を示し、Ms. Pac-Man におけるコンピュータの世界記録を上回る結果を得た。

Montecarlo Tree Search In Ms. Pac-Man

Nozomu Ikehata[†] Takeshi Ito[†]

The competition of controller program for Ms. Pac-Man has been held in the CIG symposium of IEEE every year from 2007. Since this competition was held, Ms. Pac-Man has become to an attractive subject of research on digital game AI. In the competition by this year, the classical AI method by using the knowledge base has gotten the best result. But, since the improvement by this method is becoming a limit, a new approach is required. In this paper, we realized the Ms. Pac-Man controller by the using Monte-Carlo tree search which is effective on Go, and examined the effectiveness. By repeating the simulation about the future phase generated with the random number, the Monte Carlo tree search can select the next move with a high expected value, without depending on professional expertise. In an evaluation experiment, the controller by the Monte Carlo tree search showed results more excellent than all the programs which participated in Ms. Pac-Man Competition in the past.

1. はじめに

デジタルゲームプログラミングの国際学会である CIG (IEEE Symposium on Computational Intelligence and Games) では、毎年様々なデジタルゲーム AI を題材にした競技会が開催されている。その中の一つに Ms. Pac-Man を自動操作するプログラムの優秀さを競う Ms. Pac-Man Competition(以下本稿では MPC と呼ぶ)¹⁾がある。このコンペティションは、人間が持つ Ms. Pac-Man の世界記録(92,1360 点)をコンピュータが塗り替えることを最終目標として、プログラム同士に得点を競わせることで AI 技術全体の向上を図る。人間と条件を平等にするために、プログラムはゲームの内部情報を利用することを禁止されており、動作画面のスクリーンキャプチャを通して状況認識・判断を行う。

これまでに様々な AI 手法が Ms. Pac-Man プレイヤーに適用されてきたが、現在に至るまで全てのコンペティションにおいて知識ベースのシンプルなプログラムが最高得点を記録している。その理由として、リアルタイムアクションゲームでは一つの操作ミスが命

取りになり、プレイヤーは常に安定した入力を行う必要があるため、ロバスト性に不安がある強化学習やニューラルネットワーク等の機械学習手法が適さないことや、将棋や囲碁とのゲーム性の違いから二人完全情報ゲームで一般的に用いられる Minimax 探索の適用が難しいことなどがある。現状では、コンピュータの世界記録を持つ思考アルゴリズムは知識ベースを用いた手法によるものであるが、その得点は伸び悩んでおり、この手法の限界も見え始めている。そのため、既存手法よりも優れた新しいアプローチが求められている。

本研究ではそのアプローチの一つとして UCT (UCB applied to Trees)を Ms. Pac-Man に適用することを試みた。UCT はシミュレーションをベースにした木探索手法の一つで、ランダムシミュレーションによる平均報酬とノードの探索回数を考慮して、見込みのある手に対して多くの探索を行う手法である。Ms. Pac-Man のようなリアルタイムゲームでは全ての場合について探索を行うことが困難であるが、ランダムに生成した局面のシミュレーションを繰返すことで、それを補うことができると考えられる。また、UCT は予めゲームの知識や戦略モデルを与えなくても、条件に対応した適切な戦略を自動的に生成することができる。この性質は、プロフェッショナルが存在せず体系的な戦略が定まっていない Ms. Pac-Man のようなゲームにも有効

[†] 電気通信大学情報工学科

Department of Computer Science, University of
Electro-Communications

に働くことが期待される。

本稿では実際に構築した Ms. Pac-Man の自動操作システムの詳細について説明し、提案システムと MPC に出場した既存プログラムを比較した性能評価実験の結果を示す。

2. Ms. Pac-Man

Ms. Pac-Man は 1982 年にゼネラルコンピュータ社によって開発された Pac-Man の類似作品である。図 1 は、そのプレー画面のスクリーンショットである。

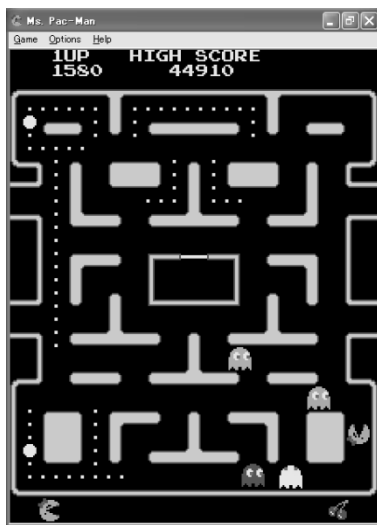


図 1 Ms. Pac-Man

プレイヤーは四方向レバーを使用し、壁に囲まれた迷路の中でパックマンを操作する。迷路の中に存在する四匹のゴーストを避けながら、迷路内に配置されたドットを全て食べるとステージクリアとなる。パックマンがゴーストに捕まるとミスとなり、パックマンの残数がなくなるとゲームオーバーになる。Ms. Pac-Man は Pac-Man と基本的なルールは全く同じであるが、マップの構成がステージごとに異なることやゴーストの動きが非決定的であることなど、幾つかの変更点加えられており、Pac-Man に比べてより難しい問題となっている。

プレイヤーの目的は高得点を獲得することであるが、これは単にステージをクリアすれば良いわけではなく、三つの方法を駆使して、ステージを攻略していく過程で効率的に得点を獲得していかなければならない。

その方法の一つは餌を獲得することである。ステージ内には二種類の餌が様々な場所に配置されており、餌に触れることで餌の種類に応じた得点を得ることができる(通常餌：10 点、パワー餌：50 点)。

また、パックマンは通常ゴーストから逃げる立場にあるが、パックマンがパワー餌を獲得した時にはゴーストは一定時間「いじけ状態」となりその立場が逆転

する。このとき、パックマンがゴーストに触れることでゴーストを倒すことができる。ゴーストを倒すことで得点を獲得し、さらに、一つのパワー餌の効果時間中に連続してゴーストを倒すことでボーナス点を獲得することができる(順に 200, 400, 800, 1600 点)。

三つ目の方法としてボーナスアイテムを獲得することがある。一定時間ごとにステージ内にボーナスアイテムが出現し、これにパックマンが触れることで得点を得ることができる。

3. 関連研究

3.1 Ms. Pac-Man の先行研究

ここでは、過去に行われてきた Ms. Pac-Man 研究の一部を紹介する。

Pac-Man AI における最も古い研究は Koza が行ったものである²⁾。彼のアプローチは「最も近い餌のある方向に最短経路で進め」「ゴーストが近づいたら逆方向に逃げろ」等の単純な行動の集合を予め定義し、状況に応じてそれらを使い分けるものである。彼のプログラムは Pac-Man を簡略化したシミュレータ上で実行され、最高で 5000 点を獲得した。彼の研究は現在の Pac-Man AI 研究の原点となった。

Wirth はパックマンの行動制御に影響マッピングを用いた³⁾。彼は「近くに多くの餌があるなら良い場所」「近くにゴーストがいるなら危険な場所」のようなゲームオブジェクトがマップに与える影響度を定義し、この影響度を元にマップ上の各地点に対する評価を行って、パックマンを評価値が最も高い地点に進ませた。彼はこの手法を用いて Ms. Pac-Man 上で最高で 7000 点を獲得するプログラムを作成した。

Robles は単純なゲーム木探索を用いてパックマンが移動可能な全ての経路を評価し、安全な経路と危険な経路を分類することを試みた⁴⁾。安全な経路の中で最高得点を獲得できる経路を選択することで、死亡率を下げながら高得点を獲得するような二つの視点からの柔軟な判断を実現した。彼の作成したプログラムは Ms. Pac-Man 上で最高で 15640 点を獲得した。(図 2 左)

松本はゲームのプレイ中に起こりうる様々な条件の組み合わせに対する行動のリスト(例えば、「付近に少なくとも一つの餌が存在し、かつ、最近傍のゴーストが一定距離離れているならば、最も近い餌に最短経路で移動する」)を予め定義することで、局面に応じた高速な判断を実現した⁵⁾。また、ゴーストをパワー餌の傍で接近するまで待ち伏せすることで、同時に多くのゴーストを撃退することに成功した。松本のプログラムは 2009 年度のコンペティションで優勝しており、最高で 30010 点を記録している。これは、コンピュータの持つ世界記録となっており、2010 年現在も彼の記録を超えるプログラムは現れていない。(図 2 右)

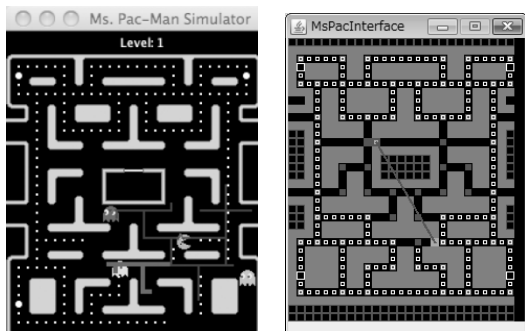


図2 RoblesのAI(左)と松本のAI(右)

3.2 UCT

UCTは多腕バンディット問題における効率的な方策であるUCB1をゲーム木探索に拡張したものであり、2006年にKocisらによって提案された⁶⁾。ゲーム木の各ノードにおいて(1)式で求められるUCB(Upper Confidence Bounds)値が最大になる子ノードが探索される。

$$\bar{X}_i + C \sqrt{\frac{\ln T}{T_i}} \quad (1)$$

(1)式において、 $\bar{X}_i$ は子ノード*i*の平均報酬、 T_i は子ノード*i*の探索回数、 T は親ノードの探索回数、 C はバランスパラメータである。従って、未探索の候補や将来有望な候補により多くのシミュレーションが割り当てられる。

UCTではシミュレーションの終局時の結果を評価値として用いることができるため評価関数を必要としない。そのため、戦略モデルや評価関数を作成する必要が無く、新しいゲームや研究対象としての歴史が浅く知識の蓄積が少ないゲームにも有効に作用する。

UCTは囲碁で成功して以来、多くの思考ゲームに適用され有効な成果を挙げており、ゲームを問わない汎用性の高さを示している。

3.3 RTSにおけるUCT

リアルタイムゲームの分野ではRTSにモンテカルロ法やUCTを応用した研究が報告されている。

Michalは無料で公開されているRTSゲームエンジンORTS(Open RTS)上に構築されたオリジナルゲームCapture the Flagにおいて、モンテカルロ・シミュレーションによる先読みを利用した戦略選択アルゴリズムを実装した⁷⁾。彼は予め用意しておいた幾つかの戦略をシミュレーションによって評価し、現在局面において最も効果的な戦略を適宜選択することで、既存のスクリプトベースのAIよりも優秀な性能を持つエージェントを作成することに成功した。

Michalの研究を受けて、Ballaは128×128のタイルで構成されるマップ上で二組の部隊が戦闘を行うRTS-Wargusにおける行動選択にUCTを適用した⁸⁾。

彼は敵兵士を倒すまでの時間や戦闘後の味方兵士の残り体力を報酬としたシミュレーションを行うことで、Michalのようにゲームの戦略の知識や評価関数を予め用意しなくとも自動的にRTSにおける戦略が生成されることを示した。

4. Ms. Pac-ManにおけるUCT

この章では実際に本研究で構築したMs. Pac-Manの自動操作システムの詳細について述べる。

手法の全体的な流れは次のようになる。

- (1) 画像認識により現在状況を把握する
- (2) 認識した情報を離散化空間に落としこむ
- (3) 離散化空間を対象にUCTアルゴリズムを行う
- (4) 方策を決定する
- (5) UCTの評価を元に入力を決定する

4.1 スクリーンキャプチャからの現在状況の抽出

MPCのルールではプログラムはゲーム画面のスクリーンキャプチャを通して状況認識を行う。現在状況を把握するためには、スクリーンキャプチャから「マップ情報」「キャラクタ情報」を抽出する。

図3はマップ情報を抽出する過程である。

- (1) SCから文字列画像を消去
- (2) 比較用餌画像をSCとマッチングさせて餌の種類や配置合計数を取得
- (3) SCから餌画像を消去
- (4) SCからパックマン画像を消去
- (5) 比較用壁画像をSCとマッチングさせて壁の形や配置を取得
- (6) マップ外の移動不可能な空間を設定
- (7) 巣をマップ中央に配置
- (8) ステージファイルを作成
- (9) ステージファイルを読み込みマップ情報を取得
- (10) テレポーターポイント(4.2節参照)の配置
- (11) ネットワークグラフ(4.2節参照)の作成
- (12) シミュレーション空間(4.2節参照)の作成
- (13) ポイント間の最短経路(4.2節参照)の計算
(SC: スクリーンキャプチャ)

図3 マップ情報の抽出

餌や壁の配置は比較用ビットマップ画像とSCのマッチングにより決定する。(1),(3),(4)で画像の一部を消去(黒で塗りつぶす)しているのは(5)で壁の認識を容易にするためである。(2),(5),(6)で認識した餌、壁、移動不可能空間の情報はステージファイルとして一度テキスト出力され、既に一度ステージファイルを生成したことがあるステージでは次回からはそれを読み込むこ

とで高速にマップを設定することができる。ステージファイルの読み込み以降の(10),(11),(12),(13)の処理は空間の離散化に関係するものであり、次節で説明する。

図4はキャラクタ情報の抽出の過程である。

- (1) Ms. Pac-Man から SC を取得
 - (2) 比較用パックマン画像を SC とマッチングさせて一致したらパックマンの位置や向きを取得
 - (3) 比較用ゴースト画像を SC とマッチングさせて一致したらゴーストの位置や向きを取得
 - (4) (3)をゴーストが4匹見つかるか、全ピクセルにおいてマッチングが成立しなくなるまで行う
 - (5) (4)でゴーストが4匹見つからなかったら、比較用いじけゴースト画像を SC とマッチングさせて一致したら位置や向きを取得
 - (6) (4)で見つからないゴーストに(5)のいじけゴーストの情報を割り当てる
- (SC : スクリーンキャプチャ)

図4 キャラクタ情報の抽出

キャラクタ情報も餌や壁の配置と同様に予め用意しておいたビットマップ画像とスクリーンキャプチャの比較によって決定される。注意しなければならないのは、キャラクタがワープトンネルを潜って画面外に行ってしまった時や、オブジェクト同士が完全に重なってしまった時である。この時、一定時間キャラクタを見失ってしまうが、一つ前のフレームの情報を参照することで現在状態を補完する。

4.2 空間の離散化

リアルタイムゲームの世界は将棋や囲碁のような離散的なゲームの世界に比べると情報量が多い。ある局面を完全に再現しようとした場合、例えば将棋では「盤上の駒の配置」「持ち駒の種類や数」「現在の手番」「持ち時間」などの幾つかの情報を考慮するだけで良いが、Ms. Pac-Man の世界は表1のような様々な情報を考慮してもまだ足りない。

表1 Ms. Pac-Man で扱う情報の一部

明示的な情報	マップ構成
	パックマンの位置・状態・向き
	ゴーストの位置・状態・向き
	残りの餌の数
	スコア
	パックマンの残数
	ボーナスアイテムの位置や種類
非明示的な情報	パックマンやゴーストの速度
	パワー餌の残り効果時間
	餌を食べた時の速度減衰

一般的に、情報量が多いリアルタイムゲームの空間では情報の全てを考慮するのではなく、環境の一部を切り取った環世界のモデル化を行う。環世界とは生物学の概念で種特有の知覚世界のことであり、ここではAIが認識することが可能である世界のことを指す。しかし、リアルタイムゲームの環境には明示的ではないルールや情報も多く、環境を単純にモデル化することができない。世界から何を切り取るか、それをどう解釈するのかを決めることは知性の本質的な問題となる。

・ ネットワークグラフ

Ms. Pac-Man のマップは囲碁や将棋のような盤の目ではなく、壁やワープトンネルを含むより複雑な空間である。そこで、空間を認識するためにマップ上に場所の指標となるポイントを配置する。リアルタイムゲームでは空間認識において、ポイントとポイントをつなぐリンクによって構成されるネットワークグラフが広く用いられる。本システムにおいてはパックマンやゴーストの位置を大雑把に把握する時や、任意地点から任意地点への最短経路を計算する際に使用される。

図5は提案システムのネットワークグラフである。

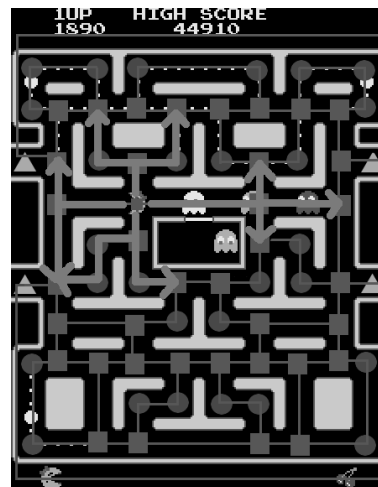


図5 提案システムのネットワークグラフ

コーナーポイント(図5丸)、クロスポイント(図5四角)、テレポーターポイント(図5三角)とそれぞれを繋いだリンクによって構成される。ポイントは図5のような配置・接続関係にある。ネットワークグラフの中で重要なものに経路がある。経路とはクロスポイントとクロスポイントを繋げたものである。経路の中でもパックマンの周辺のクロスポイントと接続しているものを特に「パックマンと接続している経路」と定義する。これは、図5の矢印に相当するものであり、UCTの評価対象となる。

・ シミュレーション空間

時間連続的な空間では更新回数に対する状況の進行速度が非常に遅い。例えば Ms. Pac-Man では、パックマンは一回の更新につき 1 ピクセル程度移動する。更新処理を 60 回行くと、パックマンは 60 ピクセル移動することになるが、パックマンの画像サイズが 15 ピクセルであることを考えると、パックマン 4 体分の距離しか移動していないことになる。これは、実際にはパックマンがほとんど移動していないことに等しい。このような環境でシミュレーションを行うことは時間的制約の面から見ても効率的ではない。

そこで、ゲーム空間を離散化し、単純化することを考える。この「空間の離散化」フェーズは、盤面が既に離散化された状態である囲碁や将棋のようなゲームにはないリアルタイムゲームならではのものである。

図 6 は「マリオブラザーズ」における離散化の例である。ここでは「マリオ」「ブロック」「ファイヤーボール」「地形」を含む空間を一定区画ごとに区切ること、ゲームを単純化している。離散化された空間は、思考に必要な最低限の情報のみで再構成されている。

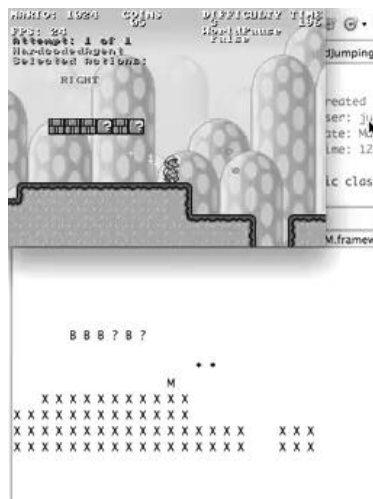


図 6 離散化されたマリオブラザーズ

表 1 の中で Ms. Pac-Man の自動操作をするために最低限必要な情報を考える。まずマップを移動するために「マップ構成」「パックマンの位置・状態・向き」が必要である。ゴーストをかわす行動を行うために「ゴーストの位置・状態・向き」の情報もある。餌を効率的に獲得することを考えるならば「餌の配置」「残りの餌の数」も必要である。

そして、これらの情報を元に現実空間を離散化した新しい空間を定義する(図 7 を参照)。この空間ではマップは 8 ピクセル×8 ピクセルの升目で区切られており、パックマンやゴーストはその升目のどれかに位置している。移動も升目を基準に行われ、一回の更新でキャラクターは一升移動する。これは現実空間上で 8 ピクセ

ル移動したことに等しい。同じ距離を移動させるにしても、離散化された空間は更新回数の面で非常に有利である。

一方で離散化による弊害もある。離散化は純粋に情報を切り捨てているので、切り捨てられた情報を全く考慮できなくなってしまう。切り捨てられた情報の重要性によっては、離散化されたことで高速な判断は行えても、正確な判断を行うことができなくなってしまう可能性がある。

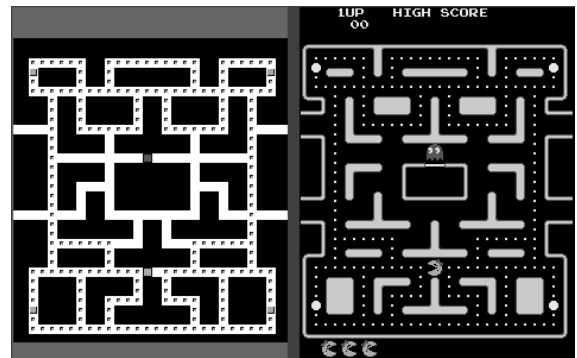


図 7 離散化されたゲーム空間(左)

4.3 UCT 探索

・ ゲーム木の構築

UCT は離散的なゲームを対象とするアルゴリズムであり、戦略を木構造で表せないものには適用できない。リアルタイムゲームに UCT を用いる際には、将棋における局面や合法手のようなノードや枝に相当する要素を明確に定義する必要がある。図 8 は本研究で用いた Ms. Pac-Man のゲーム木である。

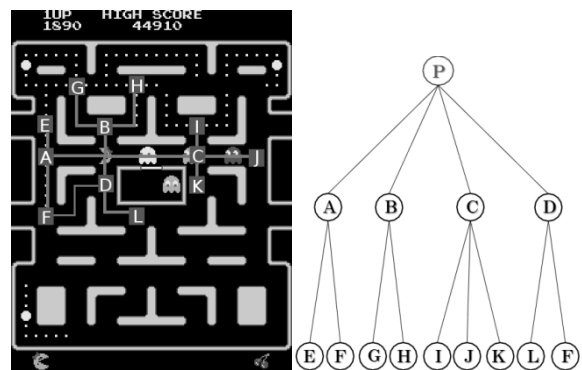


図 8 パックマンと接続している経路(左)と
それに対応するゲーム木の初期状態(右)

4.2 節で述べたとおり、UCT はパックマンと接続している経路を評価する。ノードはクロスポイント、枝は経路に相当する。ルートノードはパックマンの現在位置、子ノードと孫ノードはパックマンと接続してい

る経路の両端のクロスポイントである。ルートノードから孫ノードまでゲーム木を降りると、パックマンと接続している経路の一つをパックマンが移動したことになる。木を降りるときには、パックマンの動きは通過ノードに従った決定的なものであるが、ゴーストの動きはゲーム木に束縛されず非決定的である。すなわち、ルートノードから同じ枝やノードを経由して同じ末端ノードに到達した場合、パックマンの位置は同じであるがゴーストの位置はランダムに設定される。同様の理由で、ポイント A を経由してポイント F に到達した場合とポイント D を経由してポイント F に到達した場合でもパックマンの位置は等しいがゴーストの位置はランダムである。末端ノードの到達回数が閾値を超えるとその末端ノードが展開される。このとき新たに生成される子ノードは、末端ノードから到達できる親ノードを除いた全てのクロスポイントである。

・ UCB

UCT はゲーム木の降下中に各深さにおいて UCB1 値が最も高い子ノードを選択する。この時の(1)式における報酬 $\bar{X}_i$ は現在の方策によって用いられる値が異なる(方策についての詳細は 4.4 節で説明する)。例えば、方策として生存を重視する場合には、 $\bar{X}_i$ には生存率が与えられる。この場合、木は生存率を重視して成長し、UCT は生存率に関してより正確な値を計算することができるようになる。報酬を切り替えることで木の成長方法を自由に設定できるのは UCT の大きな利点である。

・ シミュレーションと報酬

UCT が末端ノードに到達するとシミュレーションが開始される。シミュレーション中は 4.2 節の離散化されたゲーム空間上で以下のルールに従ってパックマンとゴーストが交互に移動を行う。

- ・ P・G は経路を逆送しない
- ・ P・G はクロスポイントでのみ移動方向を決定する
- ・ P は餌を N 個獲得するたびに行動を 1 回休む
- ・ P はコーナーを N 回曲がるたびに行動を 1 回増やす
- ・ G はトンネルを N 歩進むたびに行動を 1 回休む
- ・ いじけ状態の G は N 歩進むたびに行動を 1 回休む

P : パックマン G : ゴースト

N : 経験的に定められた定数

図 9 シミュレーションのルール

図 9 は、シミュレーションの高速化や Ms. Pac-Man におけるパックマンやゴーストの挙動をシミュレータ上で模倣することを目的に設定したものである。これらのルールの中には表 1 の非明示的な情報が一部利用されている。しかし、Ms. Pac-Man の内部仕様は非公開であり、そもそも内部情報を利用することは MCP

の理念に反するため正確な値は考慮しない。上記の値 N は経験的手法により設定される。

また、ゴーストには隠れパラメータとして「攻撃性」があり、これは餌の残数に反比例してゴーストがパックマンに近づきやすくなるものである。この攻撃性を再現するため、シミュレーション中に一定の確率でゴーストをパックマンに近づけることとする。この確率 P はゴーストの種類ごとに次の式で定義される。

$$P(G) = A(G) \times \frac{(AP - RP)}{AP} \quad (2)$$

A(G) はゴースト G の攻撃性、AP は全ての餌の数、RP は残された餌の数を表す。A(G) はゴーストの種類に応じて表 2 の値が代入される。

ゴーストがパックマンに近づく時、ゴーストはパックマンとの距離をなるべく縮めようとするが、このときゴーストが目指す目標ポイントはゴーストの種類に応じて、「パックマンの移動方向にあるパックマンの最近傍クロスポイント」「パックマンの逆移動方向にあるパックマンの最近傍クロスポイント」のいずれかが設定される。

表 2 ゴーストの攻撃性と接近時の目標ポイント

ゴーストの種類	攻撃性	目標ポイント
Blinky(赤)	0.9	パックマンの移動方向
Pinky(ピンク)	0.8	パックマンの逆移動方向
Inky(水色)	0.7	パックマンの移動方向
Sue(オレンジ)	0.6	パックマンの逆移動方向

種類によってゴーストが目指すべきポイントが異なるのは、二匹以上のゴーストによるパックマンの挟み撃ちが行われる確率を高くするためである。シミュレータは意図的にパックマンが不利になる状況が多くなるように設計している。これは図 1 のようないわゆる「詰み」の状況を UCT で正確に把握し、回避するためである。

シミュレーションはパックマンがゴーストに触れて死亡した時、シミュレーションを開始してから経過サイクル数が一定数を超えた時、全ての餌を獲得してステージをクリアした時の三つの条件で終了する。この時、報酬として、パックマンが生存したどうかの二値、餌の獲得数、ゴーストの撃退数が与えられる。餌の獲得数は現在ステージに残っている餌の数、ゴーストの撃退数は画面に存在するいじけゴーストの数で割ることで 0 から 1 の範囲で正規化される。

MPC では 1 秒間に大体 15 回の頻度で Ms. Pac-Man からスクリーンキャプチャを受け取り、思考を行って入力を返すことをルールに定めている。このことを考慮して、一サイクルの UCT の合計シミュレーション回数は 60ms の間に行うことが可能な最大数とする。

4.4 方策

囲碁は対戦相手に勝利することを目的としており、UCT では「勝率」を最大化することが非常に良いと言われている。Ms. Pac-Man の目的は高得点を獲得することであるが、囲碁とは異なり、平均得点を UCT で最大化すれば良いわけではなく、実際には、「生存」「ステージクリア」「ゴーストの撃退」など、複数の目的を状況に応じて切り替える必要がある。

現在状況に対して、パックマンが何を考慮して行動すべきかを定めたのが「方策」である。提案システムは「生存」「食事」「追跡」の 3 つの方策を設定し、これらを状況に合わせて切り替える。「生存」は生き延びることを重視し、「食事」は餌の獲得を優先する。また、「追跡」はより多くのゴーストを撃退することを目的とする。

方策は UCT の評価や画面状態に応じて変更される。例えば、現在の方策が「生存」である時、UCT で経路を評価した結果、全ての経路の生存率が一定閾値以上であれば、パックマンは現在安全な状態であると言える。このとき、方策を「食事」に変更し、餌をより多く獲得することを目指す。また、画面上にいじけ状態のゴーストが存在しているならば、ゴーストを一匹でも多く撃退するために方策を「追跡」に変更する。

4.5 入力

UCT の評価と現在の方策に従って、パックマンが次に移動する経路をパックマンと接続している経路の中から表 3 のように選択する。

表 3 移動経路の決定方法

生存	パックマンと接続している経路の中で最も生存率が高い経路を選択する
食事	パックマンと接続している経路の中で生存率が閾値以上である経路のうち、最も餌の獲得率が高い経路を選択する
追跡	パックマンと接続している経路の中で生存率が閾値以上である経路のうち、最もゴーストの撃退率が高い経路を選択する

最終的に Ms. Pac-Man に送られる入力は選択された経路への最短移動方向となる。入力は前述したとおり 1 秒間に 15 回の頻度で Ms. Pac-Man に送られる。

5. 性能評価実験

本章では、実際に構築したシステムを実行した性能評価実験について述べる。

5.1 実験環境

実験は CPU Core2 Duo 3GHz, メモリ 2GB のマシン環境で行った。実装は C++ 言語を用いた。

5.2 提案システムの性能評価

提案システムの性能を評価するために、提案システムに Ms. Pac-Man を 60 回プレイさせた。その結果を表 4 と図 10 に示す。

表 4 提案手法が獲得した最小得点・最大得点・平均得点

Min	Max	Mean
6470	44910	22230

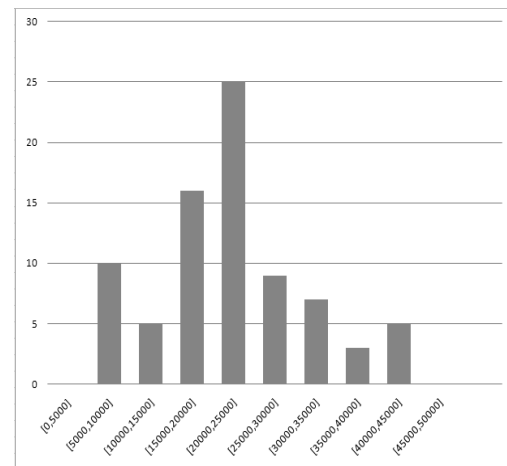


図 10 提案手法が獲得した得点の分布

実験の結果、20000 点から 25000 点の間に得点が最も集中しており、平均値もその範囲に収まっていることが判った。しかしながら、最小得点は 6470、最大得点で 44910 と分布には大きなばらつきが見られた。パックマンの死亡要因として最も多かったのが二体以上のゴーストに挟み撃ちされることであった。この状況が発生しやすかったのが、パワー餌を全て消費してしまったステージ後半において、ステージ角や長い直線などのゴーストに挟み撃ちをされやすい場所に餌が残ってしまった場合である。ゲームの性質上、一度このような状況に陥ってしまうと、残された餌を獲得することが非常に難しくなる。このような状況は序盤のステージにおいても頻繁に発生するため、まだ得点を殆ど獲得していない状態でこの状況に嵌ってしまった場合には、全くスコアを獲得できないままゲームオーバーになってしまう。提案手法では餌の獲得順序やパワー餌の獲得タイミングは殆ど考慮されておらず、そのためにこのような不安定な結果が生じてしまったと考察できる。効率的な通常餌・パワー餌の獲得方法の考案は今後の大きな課題の一つである。

一方で、運良く餌を効率的に獲得できたプレイではスコアは大きく伸びていた。ゴーストに挟まれないようになるべく逃げ道を確保しつつ移動を行い、多くのゴーストを効率的に撃退することができていた。この結果は、UCT を用いた先読みや方策の切り替えによる

生存率・撃退率を重視したプレイが効果的に働いた結果と言え、予め知識を殆ど与えていないにも関わらず自動的に効果的な戦略が形成されたことを示している。これは UCT の適用が成功した他のゲームにも見られた成果であり、Ms. Pac-Man においても UCT が有効である可能性を示唆している。

5.3 既存プログラムとの性能比較

Ms. Pac-Man Competition のホームページ上で公開されている大会結果を元に、提案手法と既存プログラムの性能を比較する。比較対象となるプログラムは 2009 年と 2010 年の Ms. Pac-Man Competition に参加した全 11 プログラムである。表 5 は Ms. Pac-Man Competition に参加したプログラムの成績である。これは Ms. Pac-Man を大会出場プログラムに 10 回ずつプレイさせた時の得点である。

表 5 Ms. Pac-Man Competition 参加プログラムの成績

プログラム名	Min	Max	Mean
Ms. Pac-Man Competition 2009 出場プログラム			
StrathPac	2880	8740	5676
Robles	2820	15640	7665
Pambush3	2290	30010	17102
NTNU	3180	9000	5974
Ms. Pac-Man Competition 2010 出場プログラム			
Bruce	2640	10820	5720
CoboPac	3310	5790	4478
Emilio	1970	21250	9343
Pambush4	6370	20580	15353
Java	8710	14660	10812
Kim	3940	18690	8545
Sojoodi	3080	9990	5933

11 プログラムの中で最大得点、平均得点共に最高得点を獲得しているのが 2009 年度のコンペティションで優勝した「Pambush3」である。2010 年度のコンペティションで優勝した「Emilio」の最高得点は「Pambush3」次いで二位であるが、平均得点では「Pambush4」や「Java」に劣り、安定した結果を残せてはいない。最低得点が最も高かった「Java」は平均得点でもある程度の結果を残しているが、「最高得点」では「Pambush」や「Java」「Kim」に劣っている。

提案手法は最高得点・平均得点を見れば、全 11 プログラム中最も優秀である。特に、提案手法の出した最高得点 44910 点は「Pambush3」の 30010 点を大きく上回り、事実上の世界記録となっている。最低得点が「Java」より劣ってしまったことは、前述した餌の獲得順序やパワー餌の獲得タイミングの最適化を行うことで改良できると考える。

6. おわりに

本研究では UCT 探索を用いてリアルタイムゲームである Ms. Pac-Man における最適移動方向の決定手法について示した。性能評価実験においては、予め Ms. Pac-Man の知識や戦略を殆ど与えていないにも関わらず、挟み撃ちの回避やゴーストの効率的な撃退などの効果的な戦略が自動的に形成された。また、提案手法は過去の Ms. Pac-Man Competition に出場した全ての自動操作プログラムよりも高得点を獲得することができ、コンピュータが持つ世界記録を大きく上回ることができた。この結果から、UCT は Ms. Pac-Man においても有効である可能性を示した。

今後の課題としてはアルゴリズムの改良などが挙げられる。特に餌を効率的に獲得する方法を考案することは、ステージの後半に危険な場所に餌を残してしまうことを防ぎ、生存率を高めることに非常に効果的であると考えられる。

Ms. Pac-Man Competition の発起人である Simon M. Lucas は 2011 年度の Ms. Pac-Man Competition において 50000 点を超える得点を獲得することができるプログラムの出現を期待しており、提案プログラムがその記念すべき最初の到達者となることを大いに期待する。

参 考 文 献

- 1) Simon M. Lucas, "Ms. Pac-Man Competition," in <http://cswww.essex.ac.uk/staff/sml/pacman/PacManContest.html>.
- 2) J. R. Koza, "Genetic Programming on the Programming of Computers by Means of Natural Selection", MIT Press, 1992.
- 3) N. Wirth and M. Gallagher, "An influence map model for playing ms. pac-man," IEEE Symposium on Computational Intelligence and Games, 2008, pp. 228-233.
- 4) D. Robles and S. Lucas, "A Simple Tree Search Method for Playing Ms. Pac-Man," in IEEE Symposium on Computational Intelligence and Games, 2009, pp. 249-255.
- 5) R. T. Hiroshi Matsumoto, Takashi Ashida, "Ice pambush 3," a 2009 IEEE Symposium on Computational Intelligence and Games competition entry.
- 6) L. Kocsis and C. Szepesvari, "Bandit based monte carlo planning," European Conference on Machine Learning, 2006, pp. 282-293.
- 7) Michael Chung, Michael Buro, Jonathan Schaeffer. "Monte Carlo Planning in RTS Games," IEEE Symposium on Computational Intelligence and Games, 2005.
- 8) R. Balla and A. Fern. "UCT for tactical assault planning in real-time strategy games," In Proceedings of International Joint Conference on Artificial Intelligence, 2009, pp. 40-45.