リアルタイムグリッド環境における
マルチエージェントの単一移動対象捕獲の探索法

Real-time grid-based Multi-Agent pursuit a moving target method

システム情報工学研究科知能機能システム専攻博士前期課程 1 年　唐 霄(XIAO TANG, 201620848)
指導教員：延原 肇

Abstract: Moving Target search in robotics and game area is been researched recent years. There are some of researches according to the problem of multi-agent pursuing a moving target, but few of them could be applied to real life. This research focus on this problem and proposed a speeding-up method for real-time grid environment basing on a Cover Heuristic method. Also, evaluation experiments are based on Benchmark maps and the result proved effectiveness of the proposed method.

Keywords: Multi-Agent, Heuristic Method, Real-Time

1. はじめに

ロボティクスおよびゲーム分野において、移動対象を追跡することは Moving Target Search (MTS)タスク[1]として定義されている。本研究は MTS の上、マルチエージェントの協調することを着目した。

Fig 1 に示すように、Pacman ゲームである。プレーヤが操作する Pacman が二つの Ghost AI を回避しながらもっと高いポイントを取得するように移動する。Pacman が追いかけられた場合、ゲームが終わりである。本研究は Pacman ゲームを簡単化し、マルチエージェントが単一対象を捕獲する研究となる。

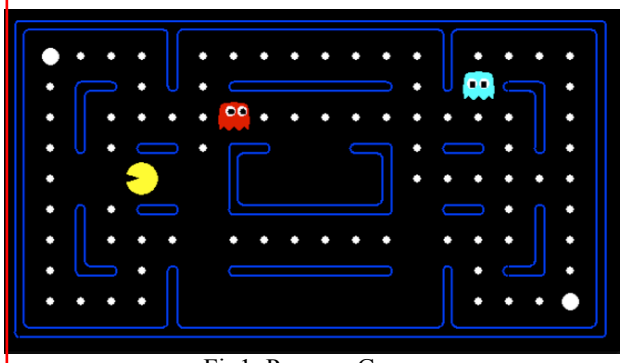

Fig1. Pacman Game

マルチエージェントの MTS 問題を効率的に解くことができれば、当該分野に大きなインパクトを与えることができる。MTS における代表的な研究として、Cover-heuristic 法（CH 法）[2]が挙げられる。しかし、CH 法が以下の問題がある。1. 計算量が多いこと、2. Tie-Breaking 問題の発生が改善の余地として残されている。

本研究では、1. CH 法の移動範囲探索過程を、ターゲット（Target, 捕獲対象を意味）とパーサ（Pursuer, 追手を意味）に分け、捕獲対象の移動範囲最小化問題として再定式化することで、計算量の削減を行う。さらに、2. CH 法における Tie-Breaking 問題の発生条件を定式化し、該当する場合に A star アルゴリズムを適用することで、この問題を解決する。

提案手法の有効性を、ベンチマーク地図[3]を利用して実験を行い、最大 8.054%計算量を削減することを示す。さらに、リアルタイム環境において、提案手法と現在代表的に利用されている A star アルゴリズムと比較し、高速に処理できることを示す。

2. Cover-heuristic 法

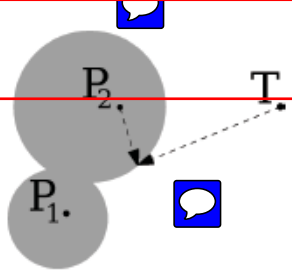

Fig2. Cover

Cover という概念を説明すると、Fig2 に示すように、ターゲット T が二つのパーサ P1 と P2 間のギャップを閉める前、そこからパスできるかを決定するタスクを考える。パーサ P1 と P2 がターゲット T より早くたどり着けるどのタイルを事前に知れたら、仮にターゲット T が示すように直線で移動するなら、パーサ P1 と P2 を交叉するタイルに移動させる。

CH 法では、グリッド地図上のパーサがターゲットより早く到着できるタイル数を求める。その数を pursuer-cover-set（PCS）と定義する。一方、ターゲットがパーサよりも早く到着できるタイルの数を、target-cover-set（TCS）と定義する。

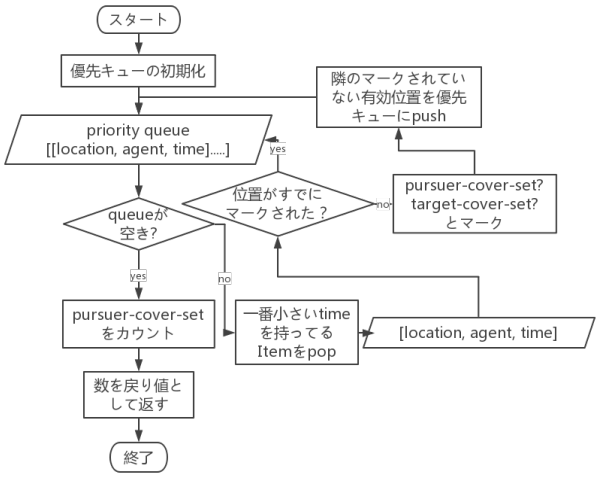

Fig 3. Cover 計算のフローチャート

一つのタイル位置に対する、位置、エージェント情報、タイムが含まれるタプル[location, agent, time]がある。Location は(x, y)の形式、x と y がエージェントの横軸と縦軸の数値である。Agent は該当エージェントの種類、”Pursuer”か”Target”となる。Time はエージェントがその位置にたどり着けるまでの移動時間、本研究はすべてのエージェントが一単位時間でタイル一つを移動するスピードト仮定する。優先キューは以上三つのデータを含めるタプル情報をストレージデータ構造である。タイル位置ごとに”Target-cover”と”Pursuer-cover”をマークする。

PCS の計算が Fig1 に示すような四つのステップである。

1、パーサとターゲットの位置を用いて time=0 の情報で優先キューを初期値する。例えば、二つのパーサの場合、優先キューが [[location = (x, y), agent = “Target”, time = 0], [location = (x1, y1), agent = “Pursuer”, time = 0], [location = (x2, y2), agent = “Pursuer”, time = 0]] である。
2、この優先キューが空きの終了条件が満ちたどうかを判定する。
3、空きの場合は、優先キューから一番小さい time を持つ item[location, agent, time]を pop する。該当 item が複数の場があるが、一回のループで一つだけを pop する。この位置がすでに”Target-cover”か”Pursuer-cover”とマークされた場合、2 に戻る。マークされていない場合、このエージェントの上下左右の 4 方向の有効位置（障害物と壁ではない）を用いて現在の位置、当該エージェントの種類、time+1 の[location, agent, time]を優先キューに push する。
4、優先キューが空き、ループが終了する。”Pursuer-cover”とマークされたタイルをカウントした結果が PCS の値である。

ターゲットおよびパーサは 1 単位時間に上下左右の 4 方向に移動可能とし、その位置を到達可能領域と定義する。あるパーサに関しては、上下左右のうち障害物が存在する場合を含め、最大 4 方向の PCS を Fig 3 に示すフローチャートにより計算する。その後、当該パーサは、最大の PCS を持つ方向に移動する。

この手法はエージェントから幅優先探索すると考えられ、Pursuer-cover と Target-cover の前部が会うことができたら、終了である。CH 法の時間の複雑さは線形的で、O(N)である。（N は地図上障害物と壁以外のタイル数）

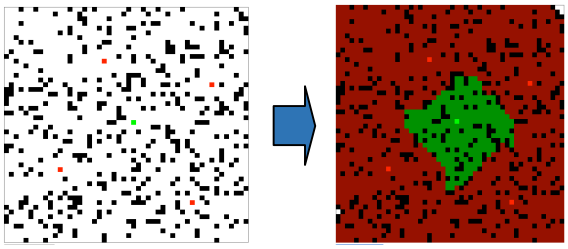

Fig 4. CH 法の計算例（左：初期状態、右：PCS と TCS の計算が完了した状態）

Fig 4 左のような初期地図（赤=パーサ、緑=ターゲット）を、CH 法により計算した結果を、Fig 4 右に示す。ここで、グリッドの赤い部分が PCS のタイルであり、緑の部分が TCS のタイルである。CH 法の目的は、PCS を最大化することによって、ターゲットの移動範囲をできるだけ抑えることである。

しかし、CH 法には 2 つの問題がある。1 つめは、PCS の計算に地図全体の探索を行うため、計算量が多くなってしまうことである。2 つめは、複数のパーサの中で、到達可能領域の PCS がすべて等しいパーサが存在する場合に Tie-Breaking 問題（どの方向に移動しても PCS が同じ状態）が発生してしまうことである。本研究では、これら 2 つの問題を解決するための手法を提案する。

3. 提案手法

まずは、CH 法の高速化手法を提案する。Fig 5 2 つの場合に示すように、PCS が黄色の線で囲まれた範囲の時点で、TCS の範囲が増大しないことがわかる。

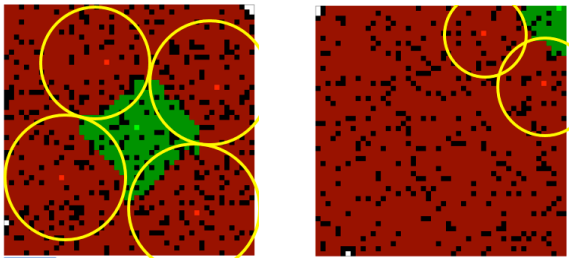

Fig 5. CH 法の計算が冗長な場合

特に、Fig 5 右に示す例の場合、CH 法では、PCS が地図上の大半を占めるので、計算の必要がない部分も探索することになってしまう。これは Fig 3 の終了条件が優先キューの空き状態になっているためである。

これを改善するため、CH 法の優先キューに対して、提案手法は target-queue と pursuer-queue をそれぞれ定義する。

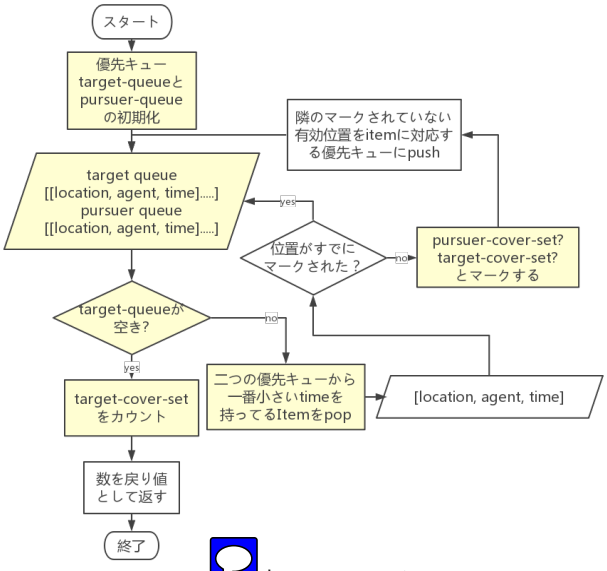

Fig 6. CH 高速のフローチャート
(黄色部分は CH 法と違う部分である)

Fig 6 のフローチャートのように、終了条件を target-queue の空き状態にする。

CH 法と違って、提案手法は TCS を計算する。計算過程は以下のようになる。

1、パーサとターゲットの位置を用いて、time=0 で target queue と pursuer queue それぞれ優先キューを初期値する。
2、Target queue 優先キューが空きの終了条件が満ちたどうかを判定する。
3、空きの場合は、target queue と pursuer queue 優先キューから一番小さい time を持つ item[location, agent, time]を pop する。この位置がすでに”Target-cover”か”Pursuer-cover”とマークされた場合、2 に戻る。マークされていない場合、このエージェントの上下左右の 4 方向の有効位置（障害物と壁ではない）を用いて現在の位置、当該エージェントの種類、time+1 の[location, agent, time]を対応する優先キューに push する。
4、Target queue 優先キューが空き、ループが終了する。”Target-cover”とマークされたタイルをカウントした結果は TCS の値である。

あるパーサに関しては、上下左右のうち障害物が存在する場合を含め、最大 4 方向の TCS を Fig 4 に示すフローチャートにより計算する。その後、当該パーサは、最小の TCS を持つ方向に移動する。

この手法も幅優先探索と考えられ、時間の複雑さは O(N)であるが、計算冗長な場合があるので、O(N)より短いである。（N は地図上障害物と壁以外のタイル数）

2 つめの問題である Tie Breaking 問題は、複数のパーサの中で、到達可能領域の TCS がすべて等しいパーサが存在する場合に発生する。

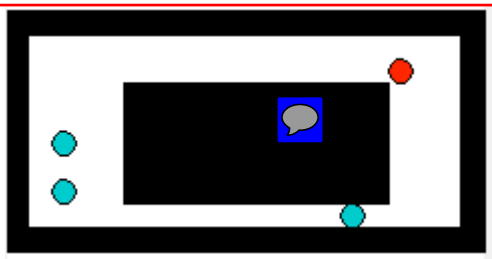

Fig 7. Tie-Breaking 例

Fig 7 に示す例では、左下のパーサが上下左右 4 方向の TCS がすべて等しいとなり、PCS の増加に役に立たない。この場合、Tie-Breaking のパーサがターゲットを囲むより、ターゲットに近づける A star アルゴリズムを適用して、この問題を解消する。

4. 評価実験

提案手法の有効性を確認するため、3 種類のベンチマーク地図[3]等に基づき、従来の CH 法と、平均探索時間の観点で比較した。平均探索時間は二つのパーサと一つのターゲットの 10 回ランダムの初期配置の PCS や TCS を実行する時間の平均値である。

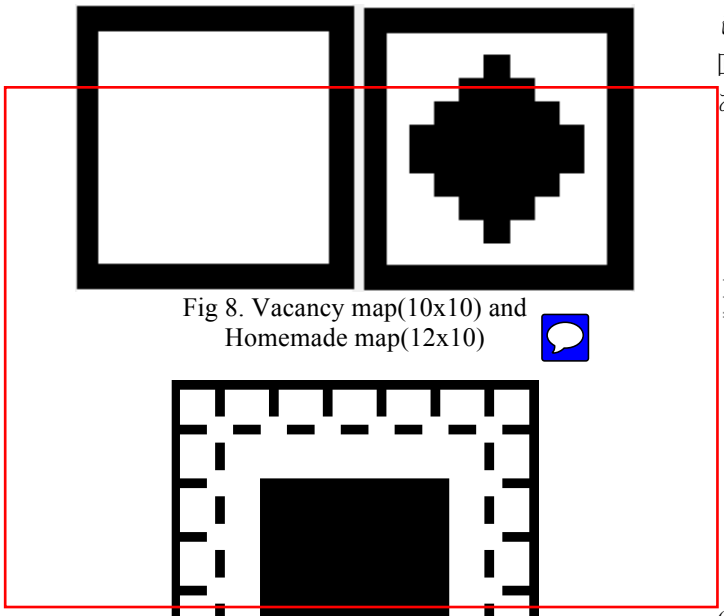

Fig 8. Vacancy map(10x10) and Homemade map(12x10)

Fig 10. Maze map(40x40)

Table 1. CH 法と提案手法の比較実験

	CH 法(s)	提案手(s)	高速化の割合(%)
Vacancy map (10x10)	0.000513	0.000471	8.054%
Homemade map (12x10)	0.000331	0.000324	2.264%
Maze map (40x40)	0.004877	0.004751	2.586%

Table 1 に示すよう、最大 8%の高速化の割合の結果が得られた。先行研究と比べて、高速化の有効性を示す。

リアルタイムとは人の目にスムースに見えるようなフレームレート制限である。本研究は Pacman ゲームを参考し、1s で 10 タイル移動するという制限を決めた。つまり、一つのターンの移動のため、すべてのエージェントの計算時間の和を 0.1s に抑えることである。

リアルタイムの有効性を検証するため、本研究の提案手法とゲーム業界で代表的に使われている A star アルゴリズムと比較する。Homemade map および Maze map を用いて、ターゲットとパーサをそれぞれランダムに初期配置した状態から捕獲する実験を行った。Homemade map で 20 回の実験、maze で 50 回の実験。パーサが二つである。

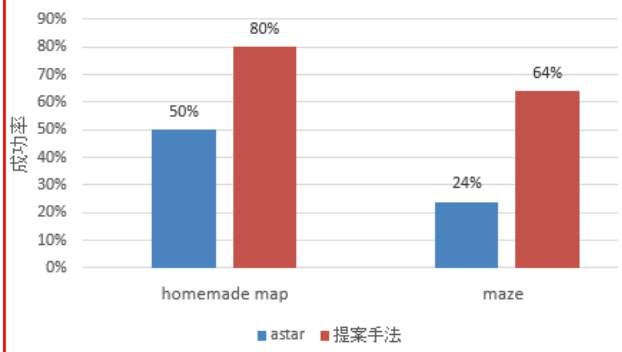

Fig 9. A star と提案手法の比較実験

Fig 10 に示した結果から、小さいマップ(homemade map)で、提案手法の 80%成功率は A star の 50%より高く、大きなマップ(maze map)で提案手法は A star の 24%より大部上がって、64%であった。

以上の評価実験を通して、提案手法の有効性を示すことができた。

5. まとめ

本研究では、先行研究の CH 法の高い計算量と Tie-Breaking 問題解決するための手法を提案した。提案手法をプログラムに適用し、高速化を達成することと高い捕獲成功率を示した。経路探索において、マップの抽象化(Abstraction)と詳細化(Refinement)を用いてもっと高速することができる。今後は、より高速できる手法を研究する予定である。

参考文献

[1] T Ishida, RE Korf, “Moving Target Search”, in *IJCAI*, 1991.

[2] A Isaza, J Lu, V Bulitko, R Greiner, “A Cover-Based Approach to Multi-Agent Moving Target Pursuit”, in *AIIDE*, 2008.

[3] NR Sturtevant, “Benchmarks for grid-based pathfinding”, in *IEEE Transactions on Computational Intelligence and AI in Games*, 2012.