

Group Game Project #2

Group24

李亭萱 0816016、林佳萱 0816047

1. Introduction to the problem

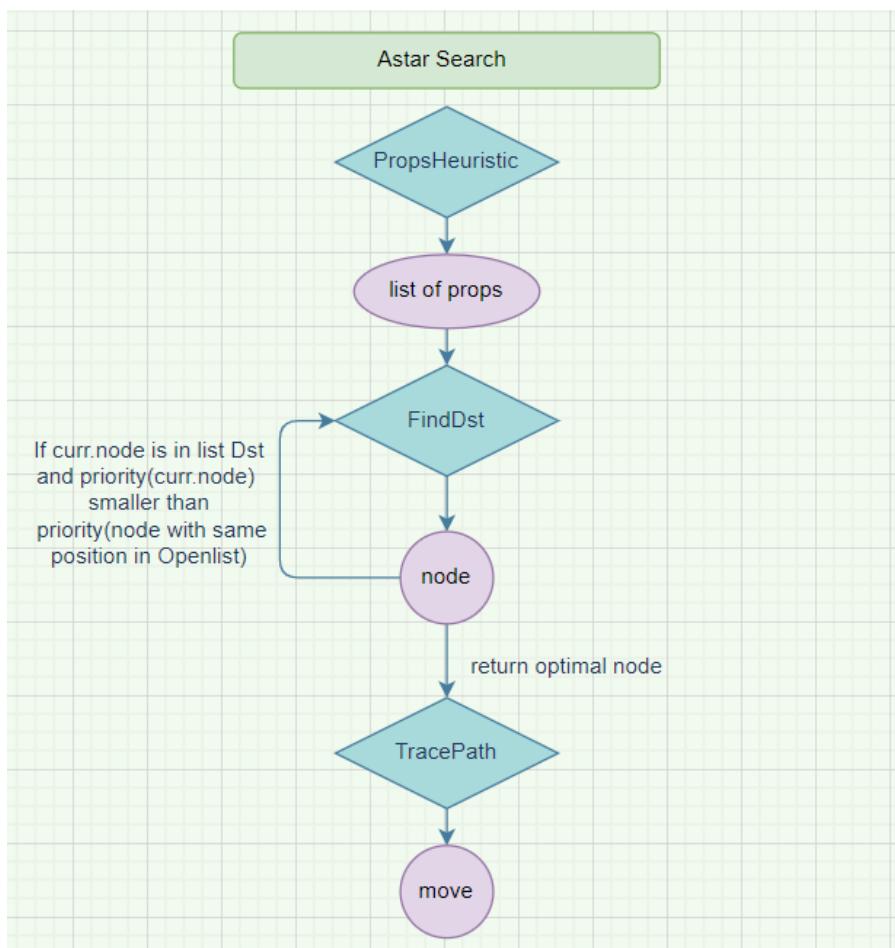
本次的小組作業是需要使用某種自訂的遊戲策略，在新版 Pacman 這個遊戲中取得勝利，也就是要在有限的時間內，與其他玩家競爭，想辦法吃到更多分數的同時也要閃避鬼避免被鬼吃掉。而我們這組採取的遊戲策略是利用 A* Search Algorithm 實作出 pacman agent，以下將逐一敘述 A* Search Algorithm 的實作細節以及實驗結果和心得。

2. How game AI works

- 在遊戲進行的過程中，所有的角色都是在盤面上同時移動，而我方在對戰中需要控制的指令有控制的方向和是否釋放地雷。
- pacman agent 每次都是經由呼叫 getStep 來產生當下的指令，因為 agent 必須知道上一次的移動方向，所以每次都會記錄當下的位置以及前進的方向，由於 pacman 接收到的位置資訊可能是校正過的(有時候會校準到中心位置)，因此不一定只有移動方向上的座標會產生差值，因此需要利用前一次紀錄的位置和當前的位置來判斷前一次的移動方向。
- 因為 pacman 沒有辦法 reverse 方向，所以不會在放下地雷後，突然轉反方向而把自己炸死。因此我們釋放地雷的部分，考慮的是要陷害其他玩家和鬼，在 power mode 的狀態下，因為我們首要條件是吃鬼得分，因此炸鬼不是我們的首要目的。在這個狀態下，當我們會把地雷放在其他玩家附近，反之當我們並非在 power mode，我們就要把炸彈放在鬼和其他玩家的附近。
- 另外，釋放地雷也要避免一次將兩個炸彈放的距離靠太近，因為沒辦法一次連續炸同一個玩家兩次，這樣就浪費炸到其他人的機會了，所以 pacman agent 會自己計算次數來判斷跟玩家的距離，才不會在靠很近的地方連續放兩個地雷。
- 方向控制的部分，則是由 game agent 進行 Astar search，首先會先設定好目標點，再尋找出最短路徑到達目標位置。其估算距離的方法是利用 $f(n) = g(n) + h(n)$ 表示 node 目前的總代價，在選擇下一個 node 的時候，其 $f(n)$ 值越低，優先級就越高(因為代價小)。其中 $g(n)$ 視為從起始點到 node 的實際距離，而 $h(n)$ 則是 node 到目標位置的估算距離。
- 至於目標的選擇，首先考慮是否處於 power mode 狀態下，在不同狀態下「食物種類」的優先順序也會有所不同，其中會考量到 props 種類、和「食物」的相差距離、其附近的「食物」密度等權重。

3. Our Algorithm

- AStarSearch 需要遍歷周圍 4 個方向(上下左右)的 4 個 node, 並將其有效的 node(可安全通過的 node)計算其 $f(n)$ 後, 放入一個 OpenList 中, 以便下次循環可以直接取出預估距離最短的 node 繼續遍歷。
- 另外 AStarSearch 中也需要一個 CloseList 用來儲存已經遍歷過的 node, 以防止重複遍歷。
- 我們的 A* Search 主要分成 propsHeuristic、AStarSearch、findDST、danger、tracePath 五個主要部分。



fig(1). Astar Search Procedure

> PropsHeuristic

- 這個 function 是為了讓我們在進行 AstarSearch 之前, 可以先選定好最佳的目標位置, 也就是選擇我們想要吃的目標「食物」。因為 Pacman 是 real-time 的遊戲, 所以盤面上除了 wall 以外的東西都有可能隨時變動, 因此 PropsHeuristic 每次都會去紀錄盤面上剩下的 item(pellets、power pellets、landmines)有哪些, 至於在 power mode 的狀態下, ghost 也會加入「食物」的選擇中。

- 紀錄完所有的可能選擇後，就要去計算各自的權重來進行優先排序，考量的條件有 props 種類、和「食物」的相差距離、其附近的「食物」密度等因素。

> Danger

- 在利用 A* search 尋找最佳路徑的同時，也要避免炸彈攻擊和被鬼吃掉，因此我們需要先利用 danger 產生出所有危險的位置，移動時就可以避開這些 dangerous position，走在安全的路徑，才不會被關到 hub 浪費遊戲時間。
- 考量到的 dangerous position 包含炸彈的位置，以及為了以防萬一包含炸彈的鄰接四格區，目的就是為了不要靠近危險。由於炸彈的位置不一定是在 cell 的中心點，而且 pacman 只要距離炸彈12個points 的 Manhattan distance 以內就會被炸死，所以必須用炸彈的實際距離加上12個points 去分別計算炸彈的鄰接四格區。
- 在非power mode的狀態下，danger position 要加上鬼的位置，在判斷對我方可能造成危險的鬼時，考量的因素有鬼是否被牆擋住以及鬼是否被關在 hub 中，還有鬼與我方的距離。與鬼距離的大小和設定範圍的大小是成正比的，因為與鬼距離太近的時候，如果危險範圍設得太大，pacman 有可能會沒有位置可以走。距離較遠時，就可以把範圍設大一點，遠離危險區。

> AStarSearch & FindDST & TracePath

- 每次 agent 呼叫 AStarSearch 時，這個 function 就會先利用 PropsHeuristic 去找出所有可以考慮的目標選項。
- 接著 function 會呼叫 FindDST 從目標列表中條選優先序最高的食物設為目標位置，然後尋找最佳路徑，若在找尋路徑的過程中遇到一個新的 node，而這個點也在我們的目標列表中(也是我們想要吃得東西)，FindDST 就會將這個 node 設為新的目標位置然後去尋找其最佳路徑。
- 所以 FindDST 會進行遞迴直到找出目前的 optimal node 和最佳路徑，接著利用 TracePath 就可以找出下個移動位置，流程可參考 fig(1). Astar Search Procedure。

4. Experiments and results

> Experiment 1: h(n)

goal: find best heuristic function with the distance

- Exact Heuristics: 用 node 到目標位置的實際距離，但是這個方法無法達到預期效果，而且會花很多時間。
- Approximation Heuristic: 嘗試了兩種估算距離的方法，
 1. Euclidean Distance: $h = \sqrt{((n.x - goal.x)^2 + (n.y - goal.y)^2)}$
 2. Manhattan Distance: $h(n) = abs(nl.x - goal.x) + abs(n.y - goal.y)$第二種的估算法，表現會比較好，因為如果用 Euclidean 的話，算的是 diagonal distance，但我們並不能隨意走動，而是有限制四個方向。

> Experiment 2: 選定目標位置的優先排序

goal: find best “food” heuristic function with the distance

- 考量的條件有 props 種類(landmine, powerpellet, pellet, ghost-power mode)、和「食物」的相差距離、其附近的「食物」密度。

- 一開始我們都是選擇距離最近的食物去當目標位置，但是這樣沒辦法吃到比較高的分數，因此除了距離之外也要考慮盤面上食物較高的區域以及食物本身的分數高低。

- 後來找出最佳的食物權重計算方式是=

食物相鄰的食物數量 * 食物的分數權重 / 食物與我方的距離的1.5次方

- 距離的考量，我們有實驗了[1, 2]之間不同次方係數，因為若權重分配太大，可能會沒辦法到達食物密度高的地區，反之太小的話可能會有捨近求遠的結果，因此最終選擇係數為1.5。

- 以下是食物的分數權重在不同 power mode 的情況下的分配，因為在遊戲規則吃到 ghost 的話可以獲得 200 分，因此我方的策略就是在要盡量吃到越多 ghost，所以一開始要先想辦法吃到 power pellet，然後抓鬼。

mode	landmine	power p.	pellet	ghost
normal	5	3000	10	x
power	5	200	10	400

- 至於 landmine 因為只有釋放炸彈的功能，而非我們所優先考量的。

> Experiment 3: safety distance

goal: determine the optimal steps to keep away from ghost

- 與鬼距離太近的時候，如果危險範圍設得太大，pacman 有可能會沒有位置可以走。

- 還有鬼與我方的距離，其中距離的判斷我們又將其細分為 distance 小於 40 以及 distance 大於 40。距離較小的鬼在設定其危險範圍時會比較小，反之與鬼之間距離較大時就可以將危險範圍設大，盡可能的遠離危險地區。

- 我們有嘗試不一樣的 safety 範圍，後來選定 distance 為 40 以達到 optimal performance。

5. Conclusion and experiences

- 完成這個 pacman agent 可以說是路途曲折，首先遇到的問題是 time limit 很短，所以不能用 MCTS 或是其他的 tree search，效能會很差。

- 後來我們也嘗試設計了rule-based agent，但是因為條件數太多且複雜，也沒有符合我們預期的結果。

- 因此，最後決定使用 A* search，因為它可以同時具備 optimal 和 complete 的特點，再加上 heuristic function(node 到終點的“權重推測值”)，使其能夠有效減少無謂的計算，不會去計算到一些距離終點很遠(aka 沒用)的頂點，有效降低運算時間。

- 為了要設計良好的 heuristic function 外，在 A* 中目標頂點的選擇也很重要，在不同狀態下要選擇的「食物種類」以及其優先排序也會影響效能。
- 在本次作業中，我們還遇到的其中一個問題是，因為一開始不知道有 reverse 方向的問題，導致 agent 執行到一半經常卡住，所以如果我們發現 pacman 停住不動，就有可能是其當前的 move 和先前移動方向相反，所以只要把方向倒過來就可以解決了。
- 另外，pacman 只有在經過中心點時，接收到方向的指令才可以轉向，所以我們前一次 send move 的時候不一定會成功轉變方向，所以實際的移動方向可能會跟我們 send 的方向不同，因此我們需要自己利用前一次的位置和當前的位置去計算出正確的方向。
- 雖然遇到很多問題，但學到很多東西，而且看到設計出來的 pacman 越來越聰明，會覺得很開心，謝謝教授和助教！

6. Contributions of individual team members

- code: 林佳萱 0816047
- report: 李亭萱 0816016