A6 – CS4300 Assignment A6 Lab Report

Path Consistency

By Haochen Zhang & Tim Wei
11/15/2017

1. Introduction

Two agents are written for this assignment. The first agent, *CS4300_agent_Astar*, takes random action to find the cell with the gold just like in A1. After reaching the gold cell alive, it will then uses A* algorithm to find the shortest safe way back to the start cell. The second agent, *CS4300_agent_Astar_PC*, will use PC1 to mark the unvisited cell as safe based on the percept it received and A* algorithm to visit the unvisited safe cell; then find its way back to strat cell like agent *CS4300_agent_Astar*. We want to analyse and compare their behaviour on several different boards to answer the following questions:

- Under different boards (from simple to complex which based on how many random decision *CS4300_agent_Astar_PC* have to make), does the use of an path consistency algorithm improves the survival rate of the agents?
- If the agents survive, does the use of an path consistency algorithm improves the total steps taken?

2. Method

We implemented 2 agents, one, *CS4300_agent_Astar*, chooses random actions to find the gold, one, *CS4300_agent_Astar_PC*, uses PC1 to do so, and both uses an A* algorithm to find the way back to [1,1]. The A* algorithm uses a marked board, where visited cells are marked as clear and all other cells are marked as pit, and Manhattan distance as heuristic function to search for a lower cost from an initial cell to a goal cell.

CS4300_agent_Astar behave as such:

- Randomly chooses an action and keep track of the current state.
- Marks the cells visited as clear.
- If it moves onto the gold cell, GRAB, and uses the board and A* to prepare the return sequence.
- Once it gets back to the start cell, CLIMB.

CS4300_agent_Astar_PC uses PC1 as such:

- Cells are represented by a number from 1 to 16. Cells are numbered by this function:

$$Cell\,\# = x + (y-1) * 4$$

  Where x and y are the indexes of the cell.
- *G* is the connectivity graph for Wumpus World cells. *G* is a 16x16 boolean array, where each rows and columns represent a cell in the Wampus World, and 1 means that the cells neighbour each other, and 0 means that they are not.
- *D* is the set of labels of the cells. D is a 16x3 boolean array, where the row number represents a cell and the column number represent possible labels, {*C, P, B*}, where *C* means the cell is clear, *P* means there is a pit, and *B* means there is a breeze.
- *P* is the predicate function which implements Wumpus World rules which takes the following arguments:
  - i (int): start node index
  - a (int): start node domain value
  - j (int): end node index
  - b (int): end node domain value

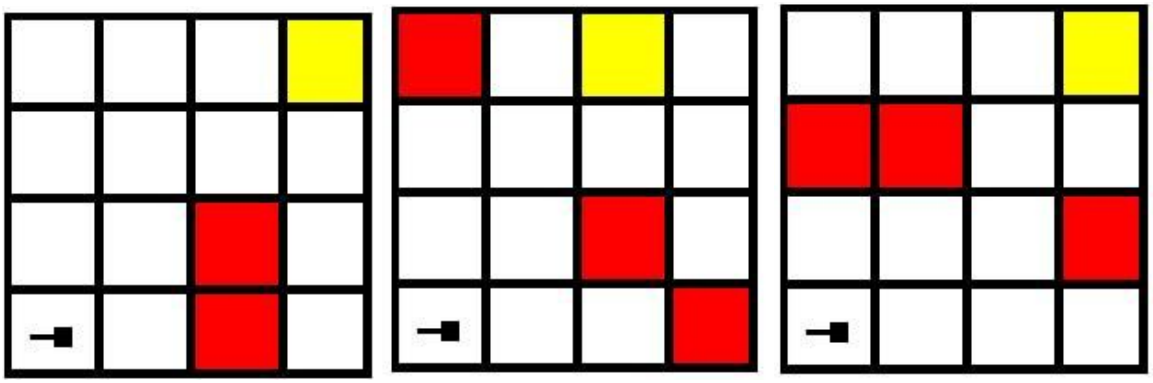
Where a and b are values between 1 to 3 representing {*C, P, B*}. *CS4300_P_no_attack* is used as *P*, which only return 0 if i and j are neighbours, a = 2, and b ≠ 3.

CS4300_agent_Astar_PC's behaviour differ from *CS4300_agent_Astar* before it moves onto the gold:

- Uses percepts to delete labels from *D* appropriately; e.g., if in cell [1,1] and percept(2) is 0 the agent is alive, then delete *B* and *P* label at cells [1,1].
- Uses PC to update *D*.
- If it is on a visited cell, skip the steps above.
- Determines if a safe cell exists that has not been visited, and if so uses A* to plan a travel action sequence to get there, otherwise randomly chooses from {Forward, Right, Left}.

CS4300_PC is the implement PC1 function. It uses *G* and *P* to create a basic *R* structure and delete values from *R* according to *D*. Then it performs PC1 on this structure. When PC1 is done, it look at all possible values for each cell and turn them back into a matrix the same size as *D*.

We then run both agents 500 times on 3 different boards and record the chance of the agents surviving and the steps taken until CLIMB for the surviving runs.

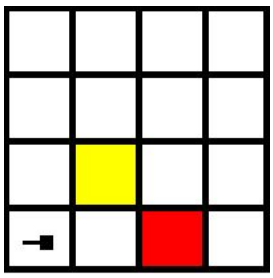


(a) Board 1 (b) Board 2 (c) Board 3

Figure 1. Boards Used in the Report.

3. Verification of Program

CS4300_Wumpus_A_star was verified in A2. We have optimized our agent to run *CS4300_PC* only when it is on a unvisited cell. In this case, *CS4300_PC* will only be called the moment it reaches a safe cell.

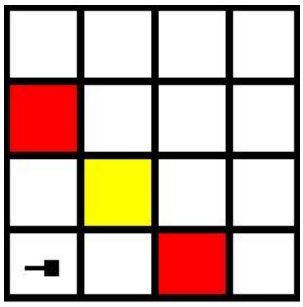


For board 1, agent will go to cell(2,1). After feeling a breeze, it will back up and go to cell(1,2). Keep exploring will lead it to cell(2,2). And that is the end of *CS4300_PC* call. The table below shows *D* returned by *CS4300_PC* on each unvisited cell.

Board 1

D	Cell 1 (1,1)	Cell 2 (2,1)	Cell 3 (1,2)
Cell 1,1 (1)	1,0,0	1,0,0	1,0,0
Cell 2,1 (2)	1,0,1	1,0,1	1,0,1
Cell 3,1 (3)	1,1,1	1,1,1	1,1,1
Cell 4,1 (4)	1,1,1	1,1,1	1,1,1
Cell 1,2(5)	1,0,1	1,0,1	1,0,0
Cell 2,2(6)	1,1,1	1,1,1	1,0,1
Cell 3,2 (7)	1,1,1	1,1,1	1,1,1
Cell 4,2 (8)	1,1,1	1,1,1	1,1,1
Cell 1,3 (9)	1,1,1	1,1,1	1,0,1
Cell 2,3 (10)	1,1,1	1,1,1	1,1,1
Cell 3,3 (11)	1,1,1	1,1,1	1,1,1
Cell 4,3 (12)	1,1,1	1,1,1	1,1,1
Cell 1,4 (13)	1,1,1	1,1,1	1,1,1
Cell 2,4 (14)	1,1,1	1,1,1	1,1,1
Cell 3,4 (15)	1,1,1	1,1,1	1,1,1
Cell 4,4 (16)	1,1,1	1,1,1	1,1,1

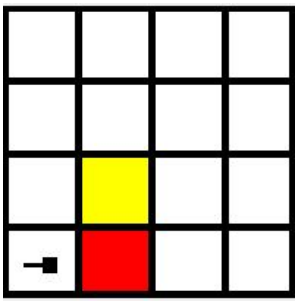
In board2, agent will go to cell(1,2). After feeling a breeze, it will back up and go to cell(1,2). At this point, agent will receive breeze again which leads to no safe cells. Our agent will then start to randomly pick action and no long call function *CS4300_PC* .



Board 2

D	Cell 1 (1,1)	Cell 2 (2,1)	Cell 3 (1,2)
Cell 1,1 (1)	1,0,0	1,0,0	1,0,0
Cell 2,1 (2)	1,0,1	1,0,1	1,0,1
Cell 3,1 (3)	1,1,1	1,1,1	1,1,1
Cell 4,1 (4)	1,1,1	1,1,1	1,1,1
Cell 1,2(5)	1,0,1	1,0,1	1,0,1
Cell 2,2(6)	1,1,1	1,1,1	1,1,1
Cell 3,2 (7)	1,1,1	1,1,1	1,1,1
Cell 4,2 (8)	1,1,1	1,1,1	1,1,1
Cell 1,3 (9)	1,1,1	1,1,1	1,1,1
Cell 2,3 (10)	1,1,1	1,1,1	1,1,1
Cell 3,3 (11)	1,1,1	1,1,1	1,1,1
Cell 4,3 (12)	1,1,1	1,1,1	1,1,1
Cell 1,4 (13)	1,1,1	1,1,1	1,1,1
Cell 2,4 (14)	1,1,1	1,1,1	1,1,1
Cell 3,4 (15)	1,1,1	1,1,1	1,1,1
Cell 4,4 (16)	1,1,1	1,1,1	1,1,1

In board 3, the agent will pick a random action from the start. In the case we choose, the agent pick turn left and go forward. This will case agent to call *CS4300_PC* because it opens up new path to go to. After choosing to go to cell(2,2), it will find the gold.



Board 3

D	Cell 1 (1,2)
Cell 1,1 (1)	1,0,1
Cell 2,1 (2)	1,1,1
Cell 3,1 (3)	1,1,1
Cell 4,1 (4)	1,1,1
Cell 1,2(5)	1,1,1
Cell 2,2(6)	1,1,1
Cell 3,2 (7)	1,1,1
Cell 4,2 (8)	1,1,1
Cell 1,3 (9)	1,1,1
Cell 2,3 (10)	1,1,1
Cell 3,3 (11)	1,1,1
Cell 4,3 (12)	1,1,1
Cell 1,4 (13)	1,1,1
Cell 2,4 (14)	1,1,1
Cell 3,4 (15)	1,1,1
Cell 4,4 (16)	1,1,1

All those result produced by our agent match our hand developed solutions.

4. Data and Analysis

The result shows increases of survival and decreases on both average total steps taken and its variance on all boards using PC1 (show in the table below).

	Survival rate (%)	Mean steps taken (±95% CI)	Variance of steps taken
Board 1 (random)	14.4	64.7083 (±6.3228)	749.2799
Board 1 (PC1)	100	38 (±0)	0
Board 2 (random)	6.40	50.9063 (±6.7961)	384.7329
Board 2 (PC1)	15.0	48.1467 (±3.8023)	282.2620
Board 3 (random)	4.60	47.5217 (±8.0444)	387.4427
Board 3 (PC1)	13.8	45.1304 (±5.4085)	525.4092

5. Interpretation

In this assignment, we posted the questions listed below. We are able to answer them with the results in this section:

- Under different boards (from simple to complex which based on how many random decision *CS4300_agent_Astar_PC* have to make), does the use of an path consistency algorithm improves the survival rate of the agents?

Because the behavior of our *CS4300_agent_Astar_PC* is to check all safe cell with the lowest Y value and if the Y value is the same, it will choose the cell with the lowest X value. So we design our first board based on that and the performance of *CS4300_agent_Astar_PC* is much better than *CS4300_agent_Astar*. Because *CS4300_agent_Astar_PC* will always choose the same solution on board 1, its variance is 0.

- If the agents survive, does the use of an path consistency algorithm improves the total steps taken?

Based on our board layout, board 2 will cause *CS4300_agent_Astar_PC* to make random choices in two separated section while board 3 only cause one. It should be the main cause that its performance on board 2 and 3 are worse than on board 1. This also makes contribute to board 2 having higher average total steps and variance than board three.

For future work, after Wumpus is added into the board, PC1 algorithm will show more advantages and even more with more complex board. We will be able to have more statistic around PC1 algorithm since there is only breeze and pit that need to be checked.

6. Critique

A huge problem during debugging was that there was no way we could step through PC1 and check for every step to be correct with Wumpus World. It was only possible for us to recognize that the output was wrong, while we have no idea what part of the algorithm went wrong. It may be helpful to come up with a smaller problem for PC to solve so that stepping through it becomes possible.

After debugging, we did not time how long *CS4300_agent_Astar_PC* would take for each board. We let it ran for several hours until we realize we had no time. We added an optimization we used for A4 and A5 (only update when the agent step on a new cell) and lower the trial number from 2000 to 500. We could have done this in the first hour and lessen the time used.

In future work, we can look into how the performance differs from AC3 to PC1.

7. Log

Tim Wei (Section 2, 4, 6)
A total of 6 hours was spent performing the experiment in Matlab.
A total of 5 hours was spent performing the experiment in writing the report.

Haochen Zhang (Section 1, 3, 5)
A total of 5 hours was spent performing the experiment in Matlab.
A total of 4 hours was spent performing the experiment in writing the report.