Project 1: Your Friend is in Danger and You are a Brave Bot 16:198:520

You are the space roomba on a deep space salvage vessel, and your best friend is the captain. It's just the two of you out here, against the world / the infinite void of space. Unfortunately your ship has been boarded by aliens. The captain is hiding under the floor panels, but it's up to you to get to them and teleport you both to safety - avoiding the aliens as you do.

1 The Ship

The layout of the ship (walls, hallways, etc) is on a square grid, generated in the following way:

- Start with a square grid, $D \times D$, of 'blocked' cells. Define the neighbors of cell as the adjacent cells in the up/down/left/right direction. Diagonal cells are not considered neighbors.
- Choose a square in the interior to 'open' at random.
- Iteratively do the following:
  - Identify all currently blocked cells that have exactly one open neighbor.
  - Of these currently blocked cells with exactly one open neighbor, pick one at random.
  - Open the selected cell.
  - Repeat until you can no longer do so.
- Identify all cells that are 'dead ends' - open cells with one open neighbor.
- For approximately half these cells, pick one of their closed neighbors at random and open it.

Note: How big an environment D you can manage is going to depend on your hardware and implementation, but you should aim to generate data on as large an environment as is feasible.

2 The Bot

The bot occupies an open cell somewhere in the ship (to be determined shortly). The bot can move to one adjacent open cell every time step (up/down/left/right). If the bot enters a cell occupied by an alien, or an alien enters the bot's cell, the bot is grabbed, and fails its mission.

3 The Aliens

K many aliens are initially randomly placed at open cells throughout the ship. At every time step, the aliens each can choose to move, up/down/left/right. The aliens can be advanced according to the following steps:

- Order the aliens at random.
- In the selected order, for each alien, choose an action at random that doesn't move that alien into a blocked or alien-occupied cell. If no action is feasible, that alien may stay in place. This is the only instance where an alien may stay in place.

- Move the selected alien according to that action.
- Process each alien in this way, in the selected order.

Note, the order should be re-selected at random every time the aliens are advanced. This simulates the fact that the aliens are choosing independently how to move each time.

4 The Captain

The Captain will be hiding at a random open cell. It does not matter if the cell is occupied by aliens, the Captain is effectively hiding in the floor or ceiling. The bot's goal is to make it to this cell, at a time when the cell does not have an alien in it. If the bot successfully enters this cell on its turn, the bot and the Captain are teleported to safety and the simulation ends.

5 The Task

At time $t = 0$, first place the bot, then the K aliens at random, distinct open cells (chosen uniformly from the available open cells). Additionally, place the Captain at a non-bot-occupied open cell. At each time step $t = 1, 2, 3, 4, \ldots$, the following happens in sequence:

- The bot decides which open neighbor to move to.
- The bot moves to that neighbor.
- If the bot enters the Captain's cell, they teleport away and the simulation ends.
- If not, the aliens then advance.
- Then the aliens advance.
- If at any point, the bot enters an alien-occupied cell, or an alien enters a bot occupied cell, the task fails and the simulation ends.

Note that there are two goals we can potentially use to evaluate the quality of a bot:

- Is the bot able to save the Captain?
- Is the bot able to avoid the aliens?

These goals can potentially be orthogonal - the bot could survive and avoid the aliens for a very long time by just staying in place or moving away from the nearest aliens, but this strategy would not lead to saving the Captain in a reasonable amount of time. For a given bot then, we can evaluate the quality of the bot on two metrics:

- As a function of the number of aliens K, how often (on average) is the bot able to save the Captain within 1000 time steps, for randomly generated ships and initial positions.
- As a function of the number of aliens K, in instances where the bot does not save the Captain in 1000 timesteps, how often (on average) does the bot survive for the duration?

Note that the second metric is inherently conditional.

6 The Bots

At any time, the bot can choose from the following actions: move to a neighboring open cell, or stay in place. The bot must make a choice, based on some assessment of the current arrangement of the ship (walls, bots, and Captain), and knowledge of how the future *might* unfold. The thing that differentiates the strategies is how the bot decides what to do.

For this assignment, you will implement and compare the performance of the following strategies.

- **Bot 1** - This bot plans the shortest path to the Captain, avoiding the current aliens, and then executes that plan. The movement of the aliens is ignored as the bot executes this plan. *Note: This is a terrible plan, and should only be effective when there are few aliens, and they are very far away. This bot serves primarily as a baseline for comparison for the other bots.*
- **Bot 2** - At every time step, the bot re-plans the shortest path to the Captain, avoiding the current alien positions, and then executes the next step in that plan. Because it repeatedly replans, it constantly adapts to the movement of the aliens.
- **Bot 3** - At every time step, the bot re-plans the shortest path to the Captain, avoiding the current alien positions *and any cells adjacent to current alien positions, if possible*, then executes the next step in that plan. If there is no such path, it plans the shortest path based only on current alien positions, then executes the next step in that plan. *Note: Bot 3 resorts to Bot 2 behavior in this case.*
- **Bot 4** - A bot of your own design.

Note: The only restriction I am putting on Bot 4 is that it cannot be a version of Bot 3 that just puts a wider buffer on the aliens. Do something more interesting. Be creative.

Each bot will be evaluated on a number of ships, at different numbers of aliens.

7 Data, Analysis, and Writeup

You need to write code to implement and analyze Bots 1 through 4, based on the following:

1) Implement Bot 1, Bot 2, Bot 3. Collect data on repeated runs (randomly generating ships and positions), for as many runs as is feasible to do, over a range of alien counts K. For each Bot, plot the two indicated graphs as a function of K (success or rescue rates, and survival rates among unsuccessful runs).
   - You should ideally collect enough data that the trends are obvious, and easy to distinguish between bots (i.e., enough data for each bot that the plots are relatively smooth).
   - Note that once K is sufficiently large, all bots should always die and or fail to be successful. You don't really need to collect data beyond this K.
   - Make clear note of what that K is for each bot.
   - In your discussion, you should note what representation and algorithms you used in general, and for each bot. Note that you'll want to collect a lot of data (as much as possible), and so be clear about whatever decisions you made in your implementations to get things running as efficiently and quickly as possible.
   - Your code should be commented to the point where the grader believes you wrote it and believes you understand what it is doing.

- Your writeup should be clear to the point where an informed grad student not in this class could implement your ideas and replicate your results, without looking at your code.
- When presenting your data, your graphs should be clearly labeled and easy to read and parse.
- You should analyze the results of your data, and identify and justify any trends and distinctions you see between the bots.
- The grade breakdown for this section:
  - 20 points for implementation.
  - 10 points for discussion.
  - 10 points for data collection.
  - 20 points for analysis.

2) Explain the design and algorithm for your Bot 4, being as specific as possible about a) how you are representing things for your bot, b) how your bot is algorithmically processing that representation, and c) how to understand what your bot is doing in the context of systematically searching through the space of possible plans of action. How does your bot factor in the available information? Implement your Bot 4, collect the same data as for Bot 1, Bot 2, Bot 3, and compare your bot to the previous.

   - Note again you will be collecting a lot of data, so efficiency is a good goal to have - be clear in your report about any choices you make as to how to speed up or improve your implementation.
   - Same recommendations re quality and quantity of data, and presentation of data.
   - Same recommendations as to thoroughness and quality of discussion and report.
   - In your analysis, be clear about in what ways your bot performs better than Bot 1, Bot 2, Bot 3, and in what ways (if any) it performs worse. If it performs worse - how, under what circumstances (i.e., specific arrangements of walls/aliens/Captain), and why? And could it be fixed?
   - The grade breakdown for this section.
     - 10 points for implementation.
     - 10 points for discussion.
     - 10 points for data collection.
     - 10 points for analysis and comparison.

Bonus 1 - 15 points: *In the above, we considered randomly generated ship layouts. If you were to design the ship layout deliberately, maybe the bots would be more effective. The only restriction is that all open cells must be reachable from each other. Write a program to design/discover the best ship layout you can. Explain your logic and algorithm design choices, as well as giving the final ship layout. Justify, if you can, why it is as effective as it is. Is this layout realizable in terms of our ship generation process? Why or why not?*

Bonus 2 - 20 points: *Consider restricting bots so that they have to make a decision within a certain amount of time - limiting the amount of computation they can do. Note that Bot 1, for instance, requires no intermediate computation as it executes the rescue, as it never changes its plan. How could you implement such a constraint? How would it change the structure of your bots? Implement Bot 1, Bot 2, Bot 3 under such a constraint (I leave to you what a reasonable compute threshold is, but note clearly in your analysis a) what you decided, and b) how often the bots run up against this cutoff). How should bots behave if they reach the computational cutoff without a definite decision? Design a Bot 5 with this constraint in mind, and repeat all appropriate analysis.*