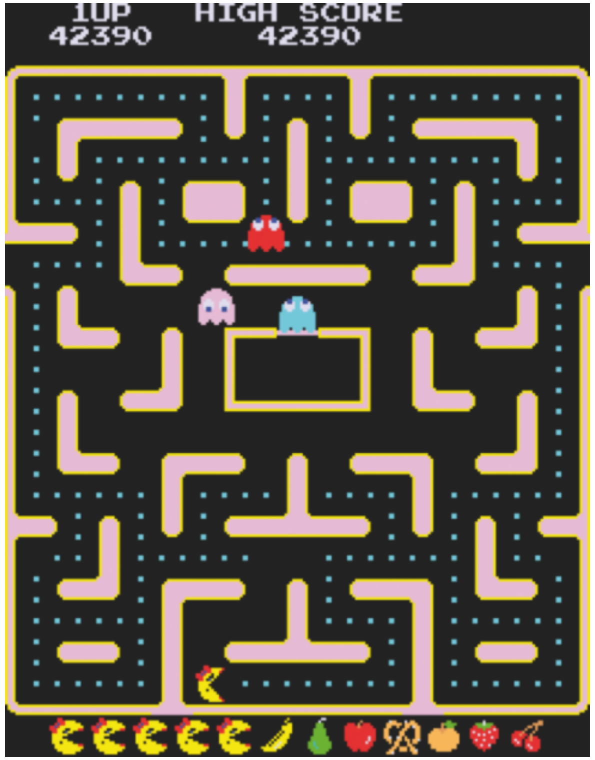
PATH PLANNING FOR TREASURE HUNTING WITH ADVERSARIES AND ITEMS

By Anand Patel
ENAE788V
Spring 2019

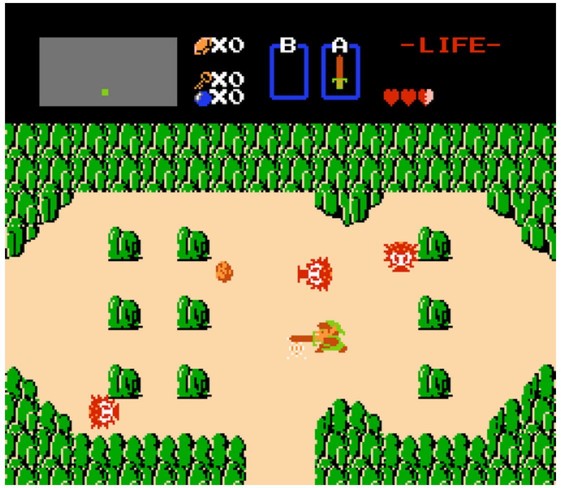
BACKGROUND

Design & solve a treasure hunting game with adversaries, items, and exit door goal.

- Games provide a complex, interesting testbed for motion planning
  - Unique rules
  - Actions alter environment
- Motivation:
  - Pursuit/Evasion Motion Planning Problems
  - Applications for entertainment
    - Human vs Robot: Can we beat the machines?
    - Robot vs Robot: Who designed better agent?



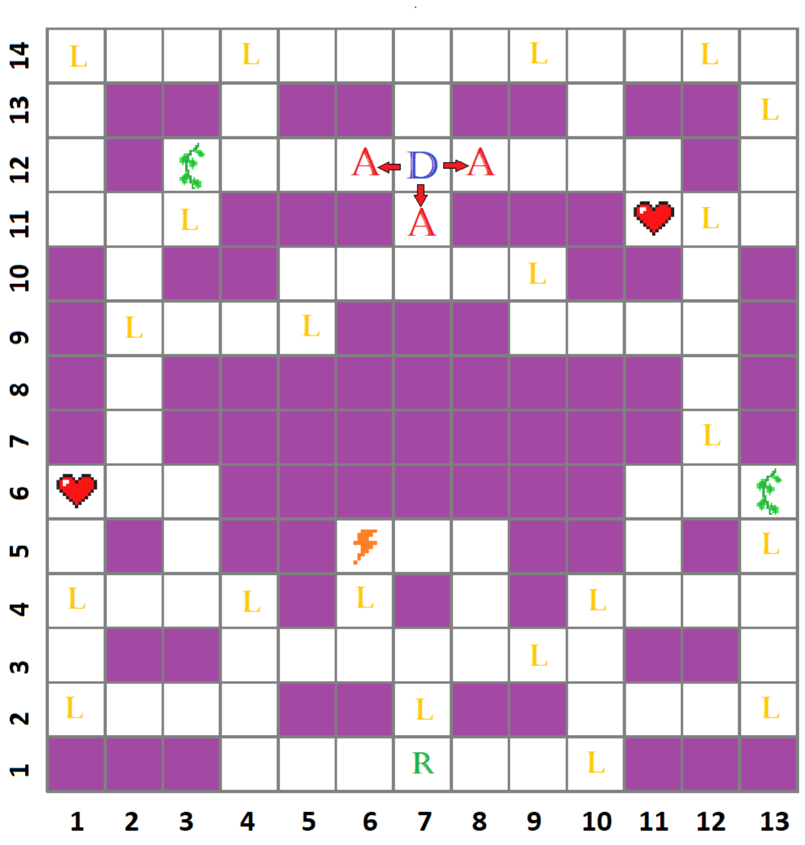
https://www.retrogamer.net/retro_games80/the-making-of-ms-pac-man/



https://www.usgamer.net/articles/long-time-coming-finishing-the-original-legend-of-zelda-in-2016

CONCEPT OVERVIEW

- Game Description:
  - 2D static grid ➔Undirected Graph: $G \in (V, E)$
    - Rooms ➔ Nodes: $v \in V = [v_1, v_2, \ldots, v_n]$
    - Connections between rooms ➔ Edges: $(v_i, v_j) \in E \Leftrightarrow (v_j, v_i) \in E$
    - 4 directions of movement, Turn-based
  - Explorer Agent *R*
    - *R* begins game at fixed start node v_{start}
  - Adversaries *A* move randomly
    - 3 adversaries (A_1, A_2, A_3) begin game at exit door *D*
  - <u>WIN</u>: *R* reaches fixed exit door *D*
  - <u>LOSE</u>: *R*, *A* occupy same node
  - Loot *L* and Items *I* placed at unique nodes
    - Collected by *R* by traveling onto node
      - Total Loot value: $L_T = 103$
      - Loot collected by explorer: L_C



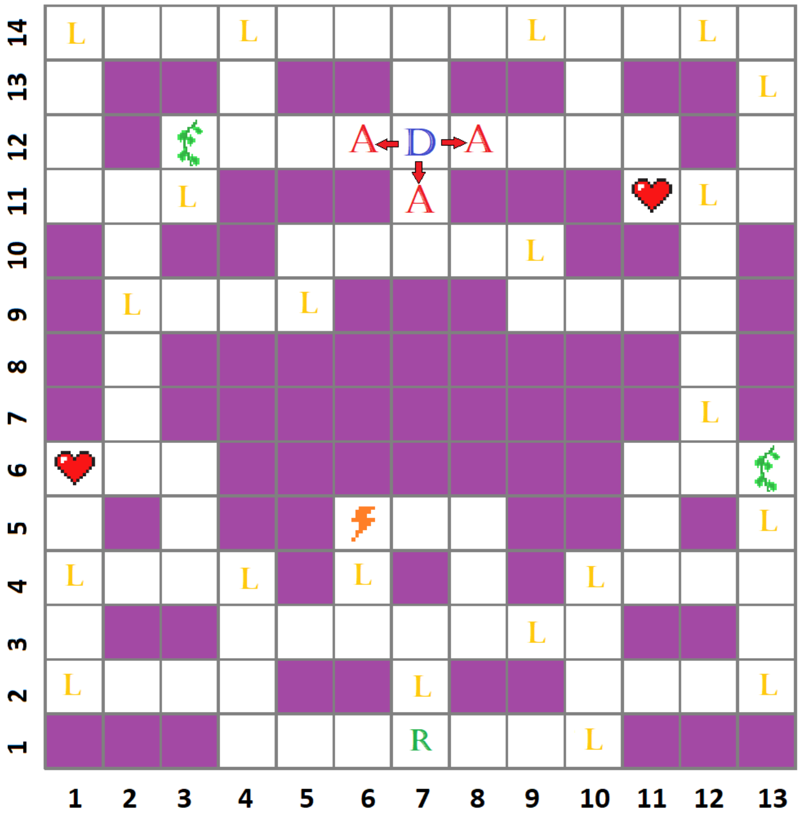
L-type	#	Value
L_1	1	13
L_2	2	11
L_3	3	7
L_4	4	5
L_5	5	3
L_6	6	2

PROBLEM STATEMENT

Problem: *Path Planning for Treasure Hunting with Adversaries and Items*

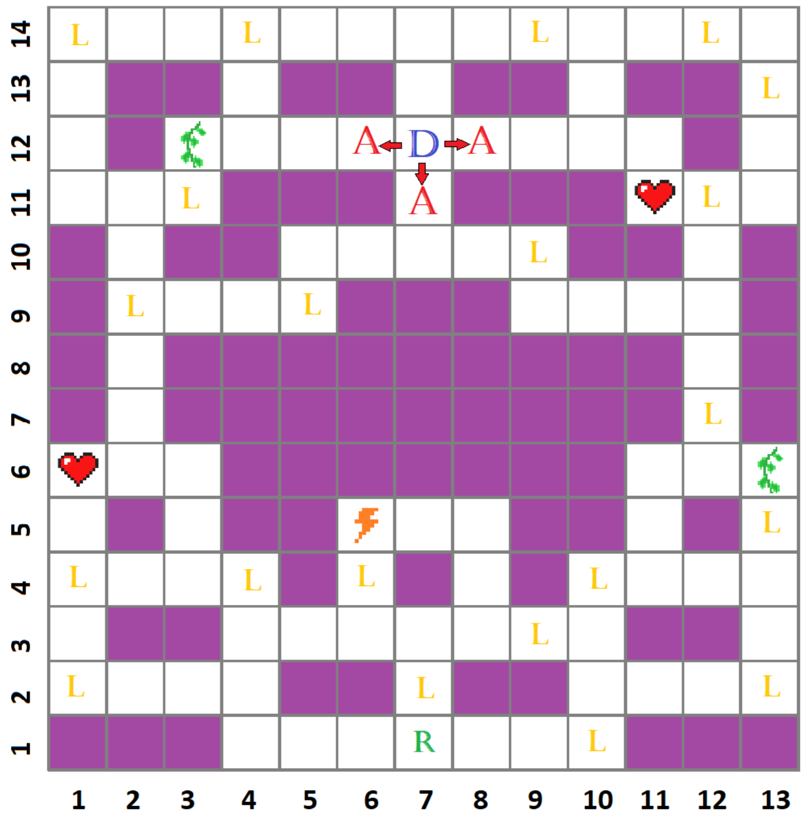
For a given labyrinth represented by G, containing explorer R, loot L, adversaries A, and items I, the explorer shall determine the path from the initial location of R to door D that maximizes the percentage of loot value collected $\frac{L_C}{L_T}$ while avoiding collision with any adversaries A.

➔ Algorithm Chosen: <u>Q-Learning Explorer Agent</u>



ITEMS

- Items are <u>used upon collection</u> by explorer
- Item 1: I_1= Charm, 9 turns
  - Nearest adversary: harmless to explorer
- Item 2: I_2= Root, 5 turns
  - All adversaries: Stationary, but harmful
- Item 3: I_3= Lightning, permanent
  - Kills nearest adversary: harmless & stationary



I-type	#	Type
I_1	2	Charm
I_2	2	Root
I_3	1	Lightning

ADVERSARY BEHAVIOR

Each turn…

<u>Random Walk:</u>

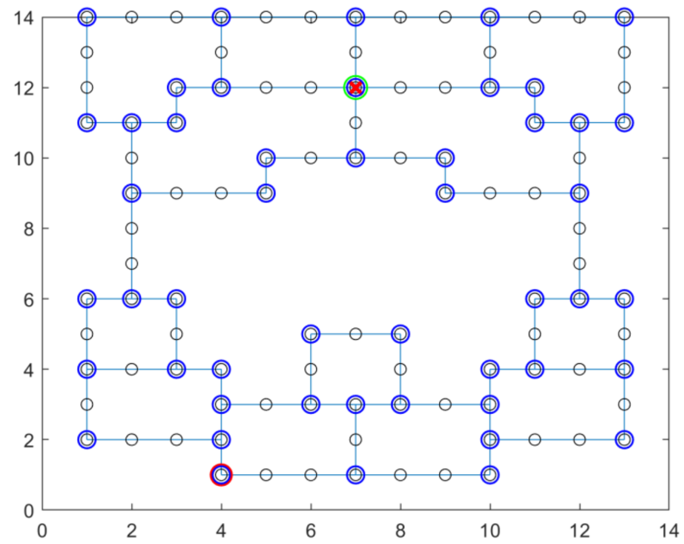
If in hallway:

Continue in same direction of movement

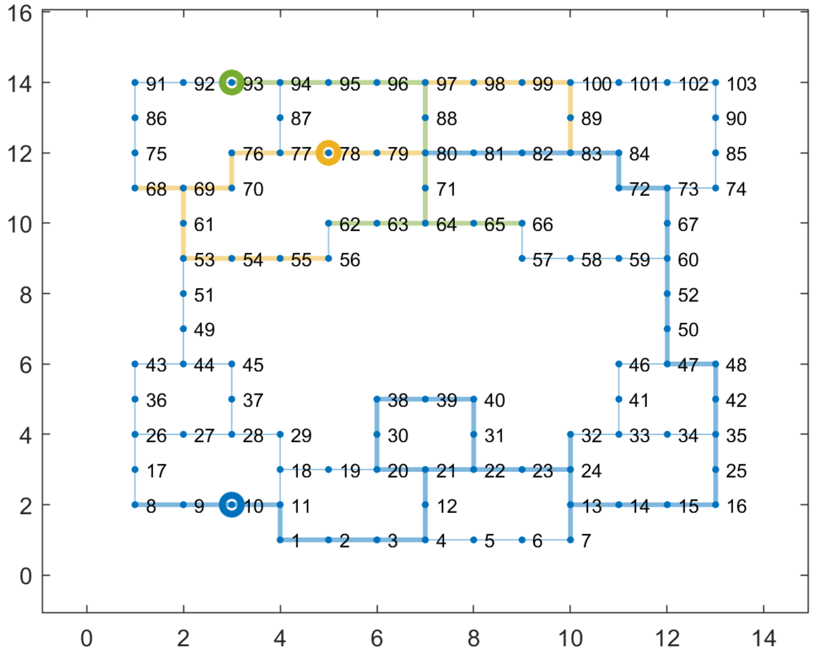
If @ intersection:

Randomly pick from possible directions at current node

- *Non-Deterministic*



Blue circles = intersection nodes



Q-LEARNING

Q(s,A) = Q-Table (multi-dimensional array), initialized @ 0

$$Q^{new}(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a_t) - Q(s_t, a_t))$$

Q-learning: An off-policy TD control algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
    $S \leftarrow S'$
  until S is terminal

https://medium.com/deep-math-machine-learning-ai/ch-12-1-model-free-reinforcement-learning-algorithms-monte-carlo-sarsa-q-learning-65267cb8d1b4

<u>Epsilon-greedy annealing:</u>

For current observation *S*, select random action *A* with probability *epsilon*. Otherwise choose action based on greatest Q-value (greedy):

$$A = \max_A Q(S, A)$$

PARAMETERS & REWARDS USED

- Learning Rate *alpha* = 0.0003 [Segarro2017mspacman]
- Discount Factor *gamma* = 0.945 [Segarro2017mspacman]
  - *alpha* & *gamma* need trial and error for fine tuning
- Training = 7000 episodes, Testing = 500 episodes
- *Epsilon* decays per episode
  - 0.9 ➔0.1
  - Decay rate chosen based on 7000 training episodes

EVENT	REWARD	DESCRIPTION
Loot	3*LootValue	Explorer enters loot room
Item	6	Explorer enters item room
Door	400	Explorer enters exit door room
Enemy	-1000	Explorer is in same room as hostile enemy
Step	-1	Explorer performs a move

```
% number of episodes for training on or testing on.
numEpisodes = 7000;

if(Training == 1)
   % Learning Rate
   alpha = 0.0003;
   % Discount Factor
   gamma = 0.945;
   % Annealing Exploration
   epsilon = 0.9;   % starting epsilon value
   epsilon_decay_rate = 0.0005;
   epsilon_min = 0.1;   % minimum epsilon value
elseif(Testing == 1)
    alpha = 0;
    gamma = 0;
    % No exploration during testing
    epsilon = 0;   % starting epsilon value
    epsilon_decay_rate = 0;
    epsilon_min = 0.1;   % minimum epsilon value
end
```

REPRESENTING THE GAME ENVIRONMENT

- Avoid over-fitted exploring agent by representing states S with info that translates between G (intersection nodes, relative distances, etc.)

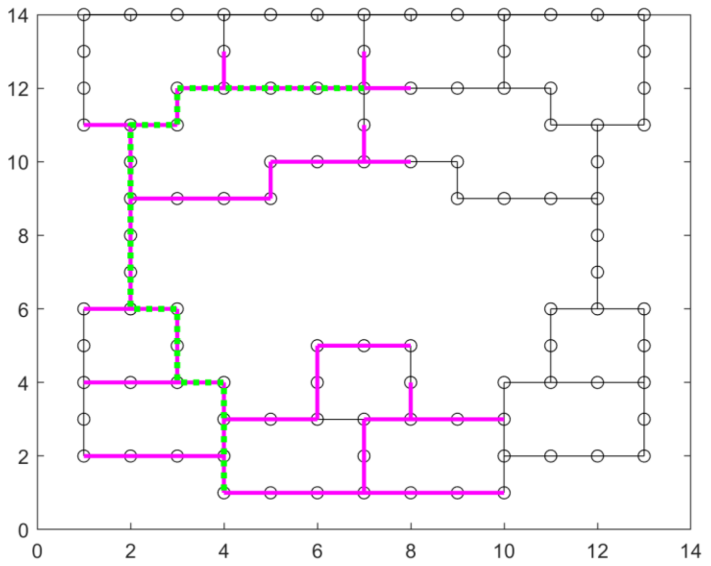
$$Q(S,A) = Q(A, a(A), b, c, d, e(A), f(A), g(A), h(A), i(A))$$

- Actions + 9 Higher-order state/action inputs = 10-D Q-Table
  - Observe S in Q-Learning ➔ <u>Calculate (S,A) for all actions available to the Explorer @ current node</u>

STATE/ACTION INPUTS

Q(S,A) = Q(A, a(A), b, c, d, e(A), f(A), g(A), h(A), i(A))

- A: [down, left, right, up] = [1,2,3,4]
- a(A): DoorInput(A) = distance to door [0,24]
  - Manhattan Distance used for all distances
  - 24 is max distance between nodes in this G
- b: LightningInput = if lightning has been used [0 or 1]
- c: CharmInput = current duration of charm [0,9]
- d: RootInput = current duration of root [0,5]



STATE/ACTION INPUTS

$$Q(S,A) = Q(A, a(A), b, c, d, e(A), f(A), g(A), h(A), i(A))$$

e(A): EnemyInput(A) = $(D_{max} - D_{i,explorer} + D_{i,adversary})$ [0,24]

D_{max}: 24

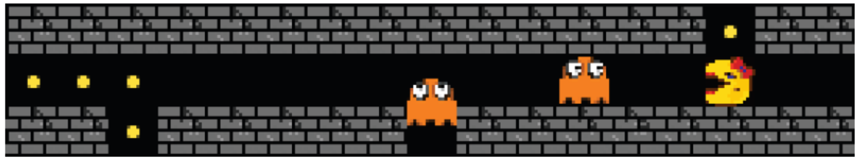
$D_{i,explorer}$: distance between explorer and closest intersection along action A

$D_{i,adversary}$: distance between closest intersection along A and the nearest adversary

➔Input is capped at 24; equidistant is worst case threshold

Gives adversary threat for a given action A

[bom2013]



Why nearest enemy only considered.
For the "left" action, even though 2 ghosts approach, only the closest one to Ms. Pacman determines danger since they share the worst input value.

[bom2013]

How input changes.
For the "right" action, the ghost being closer to the intersection than Ms. Pac-Man makes this the highest threat level (input = 24). If the ghost was further away from the intersection than Ms. Pacman, the input would be lower.

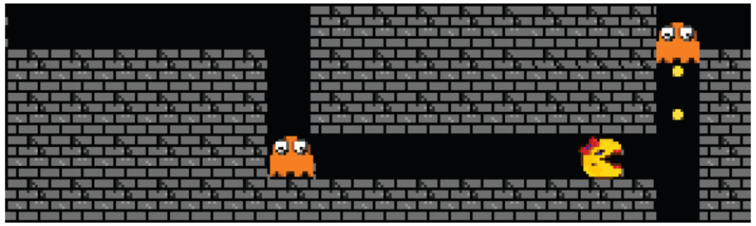
STATE/ACTION INPUTS

$$Q(S,A) = Q(A, a(A), b, c, d, e(A), f(A), g(A), h(A), i(A))$$

- f(A): LootProxInput(A) = distance to nearest loot [0,24]
  - Tells which action gets explorer closer to loot
- g(A): ItemProxInput(A) = distance to nearest item [0,24]
  - Tells which action gets explorer closer to item

STATE/ACTION INPUTS

[bom2013]



Shows entrapped Ms.Pacman approaching SafeRoute 1 intersection away as best chance for survival

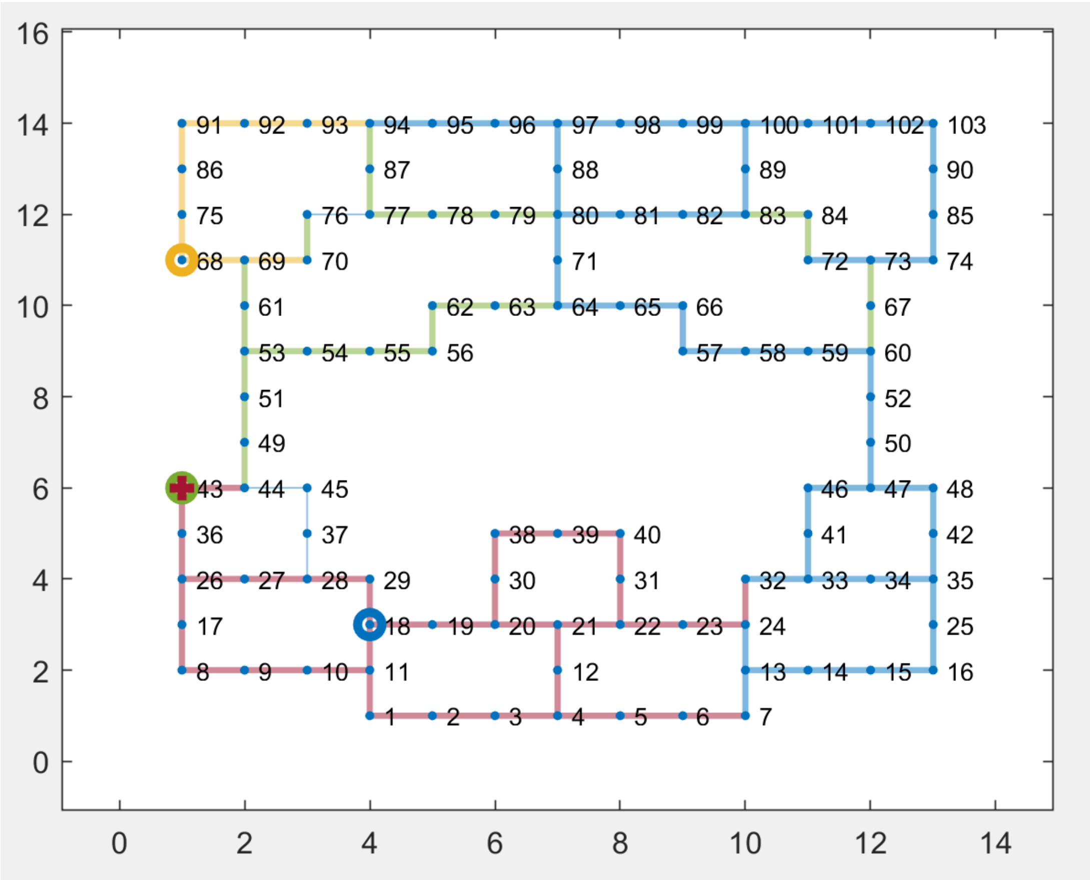
Q(S,A) = Q(A, a(A), b, c, d, e(A), f(A), g(A), h(A), i(A))

- h(A): EntrapmentInput(A) = (SafeRoutes –SafeRoutes(A))/SafeRoutes [0,10]
- Safe route = Shortest path from explorer to 3 intersections away that can be reached before any adversary
  - SafeRoutes = number of safe routes from current node
  - SafeRoutes(A) = number of safe routes beginning with action *A*
  - If length(SafeRoutes) = 0, use 2 intersections away, 1 intersection away etc.
    - If 1 intersection away = 0, set input to worst case ➔ 10 (totally entrapped)
- % of safe routes NOT along an action A

- i(A): SameDirectionInput(A) = if explorer is traveling in same direction [0 or 1]

TRAINING (IN PROGRESS)

- Training on 7000 episodes [bom2013], [segarro2017]
  - Started Monday night, still training
  - Fixed loot/item locations
  - No graphics ➔ visualization done post-training/testing



<u>Training trial for 10 episodes:</u>
Explorer (red) uses 1xLightning, 1xCharm.
Killed adversary (yellow) @ node 68.
LOSS: caught by adversary (green) @ node 43.

TESTING (NEXT STEPS)

When training completes…

- Testing on 500 episodes [segarro2017]
  - On same G (fixed loot/items)
  - On G with random loot/item locations per episode 🡐assess policy transfer
- Metrics to analyze:
  - Win %
  - % Loot Value collected per win
  - Items used per win 🡐 impact of items on success

REFERENCES

[1] A. D. Tijsma, M. M. Drugan, and M. A. Wiering, "Comparing exploration strategies for q-learning in random stochastic mazes," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016, pp. 1–8.

[2] R. A. Bianchi, C. H. Ribeiro, and A. H. Costa, "Heuristically accelerated q–learning: a new approach to speed up reinforcement learning," in *Brazilian Symposium on Artificial Intelligence*. Springer, 2004, pp. 245–254.

[3] L. L. DeLooze and W. R. Viner, "Fuzzy q-learning in a nondeterministic environment: developing an intelligent ms. pac-man agent," in *2009 IEEE Symposium on Computational Intelligence and Games*. IEEE, 2009, pp. 162–169.

[4] M. Emilio, M. Moises, R. Gustavo, and S. Yago, "Pac-mant: Optimization based on ant colonies applied to developing an agent for ms. pac-man," in *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. IEEE, 2010, pp. 458–464.

[5] L. Bom, R. Henken, and M. Wiering, "Reinforcement learning to train ms. pac-man using higher-order action-relative inputs," in *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 2013, pp. 156–163.

[6] A. Segarra, "q-mspacman," https://github.com/albertsgrc/q-mspacman, 2017.

QUESTIONS?