Does human planning follow realistic goal-setting?

Joanne Yuan (G: joanneyu)

Amber Li (UG: amli)

Adam Snowdon (UG: asnowdon)

Abstract

Humans can make efficient navigation decisions based on their goals and corresponding time horizons. Algorithms can do so as well; however, their results do not always match human ones. How can we create algorithms that more closely match human behavior? In this paper, we investigate the exploration vs. exploitation strategy components of the search-and-rescue mission. In particular, we focus on how players change their strategies in response to varying time limit constraints and time-cost functions.

Keywords: planning; human behavior; navigation

Introduction

For humans to plan an agent in a search-and-rescue mission, the path they take depends greatly on the timeout factor as well as the expected value from exploring locations to rescue victims.

We are studying the maze orienteering problem in the context of a search-and-rescue mission, where an agent must execute sequential decisions in a spatial environment in order to rescue victims. This investigation will allow us to better understand how humans plan so that we can design more human-like planning algorithms, which would aid in inverse planning inference and the creation of virtual agents who can assist humans in planning tasks.

A theoretical signature of adjusting planning strategy to time-horizon is switching to shallower planning horizon as time-out approaches, in contrast to planning ahead while the time-out is still far in the future. We can test whether people respond to approaching time-out in line with this theoretical prediction. Game theoretic settings – such as sequential prisoners dilemma games demonstrate that humans indeed modify their strategy depending on time horizon, but this has not yet been tested in spatial planning contexts (Embrey,) (Bigoni,).

Our project mainly follows from two related works. In one study, the search-and-rescue mission was modeled as an orienteering problem on the room level and POMDP within the rooms (Yang,). Interestingly, simple heuristics explained human behavior better than more complex ones — directional models worked particularly well.

In the second related work, the authors applied decision-making principles from gambling tasks to the problem of sequential decision-making in a maze search task (Kryven,). They investigated expected utility maximization, discounted future utility, weighted probability utility, and the combination of the three as computational models to explain human planning behavior. In addition, they also evaluated a few short-sighted heuristics. While overall the combined model explained the human behavioral data the best, some individuals' actions were random or best explained by one of the heuristics, so many open questions remain.

In our paper, we compute discounted expected future utility in the search-and-rescue problem from the first paper in order to try to explain human behavior. We find that hyperbolic cost with larger timeouts seems to qualitatively fit with expected human behavior. Future work in combining our behavioral and model results will allow us to explore this more quantitatively. Our simulation code and figures produced by the model can be found here, and code for the behavioral experiment can be found here.

Methods

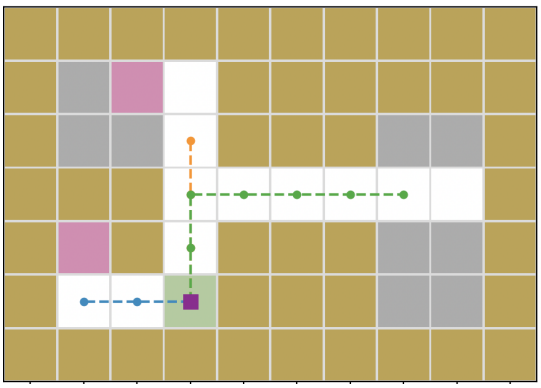


Figure 1: Example of search and rescue task maze. Each dotted line represents an initial trajectory the agent can take. The starting cell is highlighted in green, the hidden cells in gray, the victim cells in pink, and the walls in brown.

Search and Rescue Task

In the Search and Rescue task, an agent is placed within a grid world formatted as a maze, with cells that are 1) open cells (colored in white), 2) hidden cells (gray), 3) victim cells (pink), and 4) walls (brown). An example is shown in Figure 1. Each cluster of hidden cells represents a room, each of which can contain up to one victim. An agent can see and

move down paths consisting of open cells, but it cannot see or move through walls. In addition, it can take two actions — exploring a room and rescuing a victim. Exploring a room is a prerequisite to being able to rescue a victim. After exploring a room, all hidden cells in the room are revealed to become open cells.

Each agent is given a time limit to complete each task. The task terminates when either the time limit runs out or all victims have been picked up.

Computational Models

To plan a search trajectory, we create a tree with nodes storing the current position of the agent and state of the map so far. Each child of the node represents a new state where the agent has moved to a different position from the current position and made new observations.

We use Discounted Expected Utility to compute the utility at each node. At each node, the agent discovers and explores some number of hidden cells. Depending on the agent's observation at that point, there are two resulting outcomes — one where the room contains a victim and the other where it does not. The agent has full observability. If the room contains a victim, the agent can either pick up the victim or not pick up the victim. In the case of picking up the victim, the agent reaps an exploration reward and a victim reward, both of which are functions of the size of the room. We then subtract the time cost and add the maximum expected future utility from continuing on to child nodes of the tree. The equation is as follows:

$$\begin{aligned} Q(N_i) = & p_i \cdot \max\left((1+\alpha)n_{ir} - c(s_i+e_i+k) + \gamma \max_j Q(N_j),\right. \\ & \alpha n_{ir} - c(s_i) + \gamma \max_j Q(N_j)) + \\ & \left.(1-p_i)\cdot(\alpha n_{ir} - c(s_i) + \gamma \max_j Q(N_j))\right) \end{aligned} \tag{1}$$

where p_i is the probability of finding a victim in the room, s_i is the number of steps to get to the node, e_i is the expected number of steps to get to the victim, $c(t)$ is the cost function, n_i is the room size, and $\max_j Q(N_j)$ is the maximum expected value of the children nodes of N_i.

Note that p_i becomes either 0 or 1 upon the agent discovering a room and observing whether or not the room contains a victim. When calculating expected utility, we assume that each hidden cell has equal probability of hiding a victim, so p_i is the number of cells in the discovered room divided by the total number of hidden cells originally in the map. To calculate e_i, we assume that the victim is located at the centroid of the room, and calculate the Manhattan distance from the agent's position to that location.

In addition, we set $r = 1.0$ to be the reward for picking up each victim, $\alpha = 0.2$ to be the ratio of exploration reward to victim reward (so exploration reward is $0.2 \times 1.0 = 0.2$), and $k = 5$ to be the number of steps needed to pick up a victim (in order to model the fact that picking up a victim takes extra time). We assume that each step taken by the agent is equivalent to 400 ms, which is the average time it takes for a human participant to click on a grid space when doing this search-and-rescue task.

We track the number of steps to reach the node for each trajectory. If the number of steps taken is equivalent to a time exceeding the set timeout, the value of the node is zero.

We explore three different settings for the cost function, which penalizes the agent using a "subjective cost" that is calculated as function of true cost (the amount of time taken).

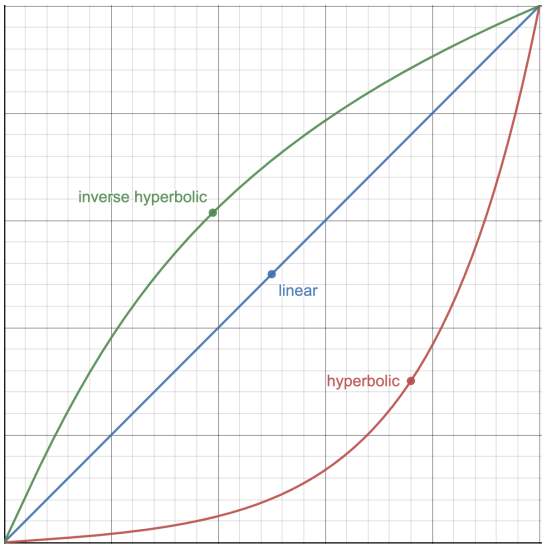


Figure 2: Different costs as a function of time. The green line represents the inverse hyperbolic cost function, the blue line the linear cost function, and the red line the hyperbolic cost function.

Linear

$$C(t) = t \tag{2}$$

With a linear cost function, true and subjective costs are equal.

Hyperbolic

$$C(t) = \frac{1}{2}(e^t - e^{-t}) \tag{3}$$

With a hyperbolic cost function, at relatively low values of true cost, the increase in subjective cost is smaller than actuality. In other words, the agent perceives less-than-actual cost near the beginning of the task. Here, we use sinh as our hyperbolic function.

Inverse Hyperbolic

$$C(t) = \ln t + \sqrt{t^2+1} \tag{4}$$

With an inverse hyperbolic cost function, at relatively low values of true cost, the increase in subjective cost is greater than actuality. So the agent perceives greater-than-actual cost near the beginning of the task. Here, we use inverse sinh as our hyperbolic function.

In order to make the three cost functions comparable, we scale the hyperbolic and inverse hyperbolic functions such that all three functions output the same subjective cost at timeout.

In order to compute the best path given a cost function, at each node, the agent greedily decides whether to rescue a victim (if one is found) based on the calculated node value, and it selects the child node with the best expected value to continue on its exploration path.

Mazes

For the Search and Rescue task we created mazes with blank open hallways, rooms that are clusters of black squares, and hidden red square victims under the black squares. We overall attempted to create maps where the participants of the experiment are forced to make decisions between rooms and routes to maximize their utility. The mazes are all between 60 and 100 total squares big because we wanted to fit the maze on one page and not make participants overwhelmed with a map too large. However, we also couldn't make the maze too small so that the users take different routes and provide us with more of an opportunity to learn more from their decision making. The mazes also each contain three or four rooms, clusters of black squares, and at least two victims, red squares. These features were to further promote variance in routes.

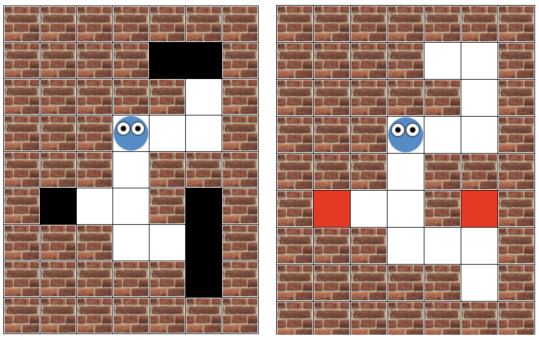


Figure 3: Example of maze from experiment. Blue circle with eyes is the agent, white squares are open halls to move through, black clusters represent rooms, and red squares are victims. On the left is the test maze initially given to a participant. On the right is how the same maze would look if participants explored all rooms

In Figure 3 above, our example maze is on the left and on the right is what the participant would see if they explored all of the rooms. With this maze, we tried to emphasize the user's decision to maximize utility under a time constraint by either going up and checking for a victim or skip the above one and check the two rooms below for victims. If the user goes up, they have to take 4 steps up to check and realize that there are no victims there, before taking 4 steps back down to the original starting spot. If the user goes down, they must take a total of 7 steps to check for victims in both rooms. The participant is forced to weigh the option of going up to explore a room of two squares in eight total steps against the option of going down first and potentially exploring four black squares in all and delaying the exploration of the top part of the maze, all while determining what is possible in the given time constraint. This maze layout and others were made to maximize trade off possibilities and give us insight into how participants may decision make under time constraints.

Limitations of the Models

Note that our model is simplified in two key ways. First of all, for each room that the agent stops by, the agent must explore the room. It cannot stop by a room and then move on to the next room. However, for the most part, this shouldn't overly limit our model's expressiveness — if it values going to another room, it should prioritize that room in its path selection. The only case in which this would be relevant is when the time cost for exploration is much larger than the potential victim reward for all rooms; however, this is an edge case that shouldn't occur for a well-designed maze.

Secondly, the agent cannot receive additional exploration reward for any hidden cells that it has revealed previously, but it can receive an exploration reward for just the victim cell. This reward structure was mainly due to the way our model was built, but led to interesting behavior that we will discuss more in the next section.

Behavioral Experiment

To study if human planning follows realistic goal-setting, we created a behavioral experiment task. Though we ran out of time to collect experimental data from human trials, we will outline our plans of experimenting.

We would first select a sample of about 15 participants of male and females across all age groups. For each participant we would give them the link to our experiment and ask that they complete it in a quiet, distraction free setting. The first page they will get when opening the link is a brief set of instructions that explain how our study will operate as well as an Informed Consent statement that states how participation is voluntary and anonymity is assured.

If they agree to these terms, the next page will provide more detailed instructions. A participant then clicks Continue and start the practice mazes. For each practice maze, it will look exactly the same as a test maze, except more detailed instructions at the top.

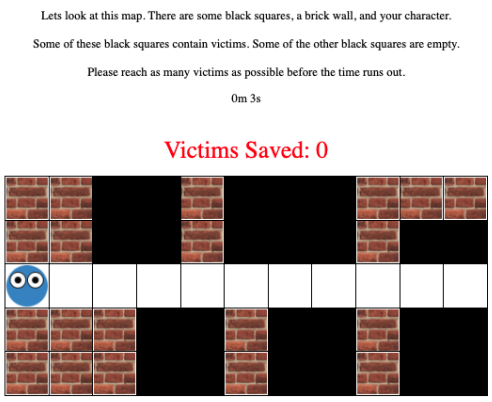


Figure 4: One of the three Practice Mazes shown for participants to learn the task.

After the three practice mazes, there is a two question quiz to make sure the user understands the task. The study will not continue until the correct answers are chosen.

Figure 5: Brief Quiz that is shown after the three practice mazes. A participant must answer questions correctly for the study to continue.

There are then ten test mazes. For a maze, the participant will be able to see a counter for how many victims have been saved, a timer for how much time is left, and fraction such as 7/10 to indicate they are on the seventh of ten mazes. For each maze, the time limit will be set to two times the maze's width of squares or move to the next page if all victims are found before time's up. The path and performance data of participants is automatically saved to a database for later analysis.

Figure 6: Test maze example. Includes a counter for how many victims saved, a timer for how much time is left, and 3 of 10 indicates being the third of ten test mazes.

Afterwards there is a page that says Thank you! and asks the participant "How did you make your decisions about which way to go?" with an answer box provided. This helps us qualitatively understand the participant's decision making. Next there is a final page with three questions to again take qualitative input to be used during analysis.

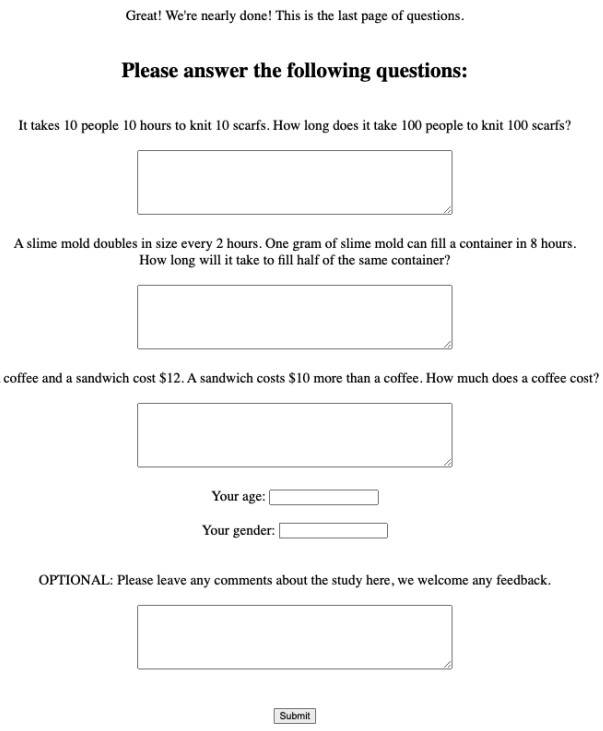


Figure 7: final page with three questions to again take qualitative input to be used during analysis

and the experiment is done!

Model-based Analysis

Because we haven't had the opportunity to conduct trials, we will outline what are model based analysis would look like. We would use both Maximum Likelihood Estimation and 4-fold Cross Validation to best fit the model's parameters to each each individual's decisions. Using this methods, we would get parameter estimates and analyze them by measuring bootstrapped correlations between all of the predicted choice probabilities, the given median parameters fitted to the entire subject population, and empirical choice frequencies aggregated over subjects. For the 4-fold Cross Validation, we would first split each of the participants' decisions into and 80 percent train and 20 percent test split, then with four cross-validation folds, fit the models to the training set data and use the held out test data to validate the fit. Based on these results, we would be able to tell what strategy or combination of strategies was most popular. We hypothesize that of the cost functions, the hyperbolic cost function most accurately describes human behavior. Based on initial qualitative observations, we thought participants would rush noticeably more as time runs out and so it seems time becomes more valuable as timeout approaches.

Results & Discussion

In our experiment, we varied two main parameters: the cost function and the timeout. An increased timeout should allow the agent to explore more rooms to potentially rescue more victims. The cost function, on the other hand, determines the penalty incurred per unit of time. We consider three main cost functions: linear, hyperbolic, and inverse hyperbolic. For linear cost, the true cost is the same as the subjective cost, but for the hyperbolic, the agent is penalized a less-than-actual cost at the beginning of the task whereas for the inverse hyperbolic, the agent is penalized a greater-than-actual cost at the end of the task. We would expect the agent under hyperbolic cost to explore more at the beginning end of the task and the agent under inverse hyperbolic cost to explore more at the end of the task.

Costs

In Figure 8 above, the best path trajectories under different cost functions are shown. The pink squares represent victims that are not picked up whereas the green squares represent victims that were picked up by the agent. The line segments in the figure are labeled in the legend by the step number.

The step by step trajectory for the linear cost path is shown in Figure 3 — the agent moves from its starting point to the closest victim, picking it up. Because it is still able to gain exploration reward from the victim cell and has extra time, it goes back and forth before exploring the two rooms to its right. However, it runs out of time before it is able to pick up its next victim, leaving one on the map.

Similarly, in Figure 9a, the trajectory for the hyperbolic cost path is shown. The agent takes the longer path to explore

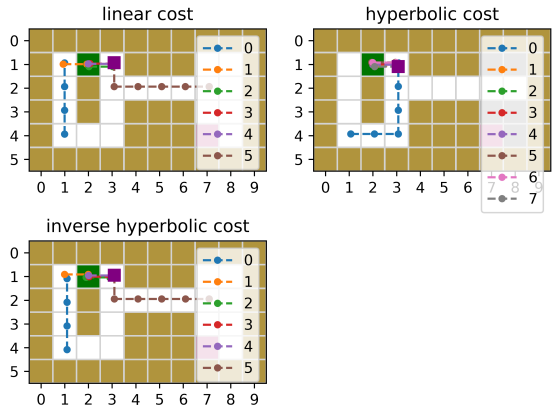


Figure 8: Best path trajectories. This figure shows the trajectories of the agent under different time costs with a 5s timeout. The legend shows the time ordering of the trajectories.

the first room, which positions it more closely to the other two rooms. However, after reaching that room, the agent picks up the victim, and chooses to take the guaranteed exploration reward over exploring the two rooms to the right. This follows our intuition that as time progresses, the agent becomes less willing to explore and more willing to exploit a guaranteed reward.

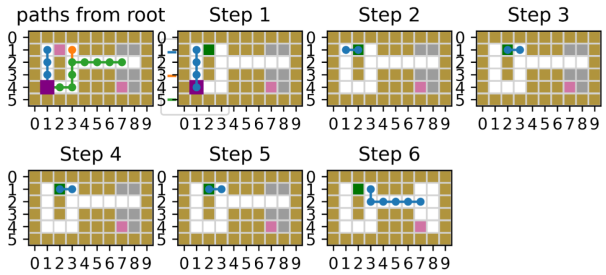
Interestingly, the inverse hyperbolic cost path (Figure **??**) takes the same path as the linear cost path.

As a whole, the trajectories computed with the inverse hyperbolic and linear cost functions are very similar — this makes sense because the differences in those two cost functions, as shown in Figure 8, are much smaller than the differences between them and the hyperbolic cost function. Linear and inverse hyperbolic tend to achieve a higher value compared to hyperbolic. However, we hypothesize that human behavior more closely follows the trajectory with hyperbolic cost — as the timeout approaches, time becomes more valuable.

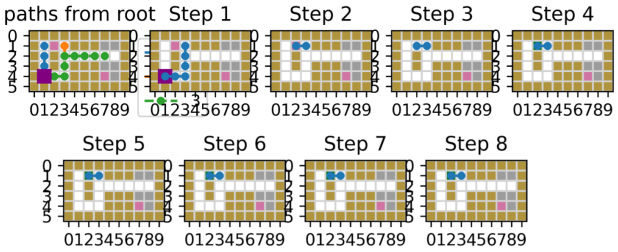
Timeout

We also observed how the agent's path through the mazes changed when given a 5 second timeout vs. a 10 second timeout. Below are some specific examples that demonstrate trade-offs between exploration and victim reward that the agent makes when faced with a shorter or longer time horizon.

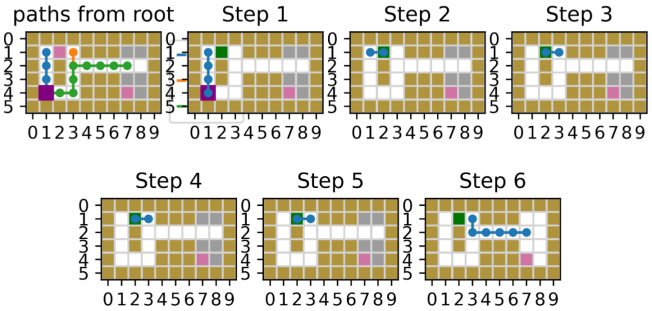
In map 1, there is a hidden cell with a victim located around the corner at the lower left of the maze. In the case of the linear cost function, the longer timeout of 10s allows the agent to discover this hidden victim, while it does not discover the victim when the timeout is 5s, generally showing that the agent is able to explore and discover more with a longer time horizon. See Figures 10a and 10b for the comparison.



(a) Linear cost path.

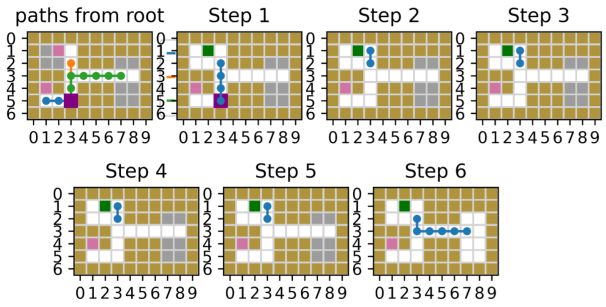


(b) Hyperbolic cost path.

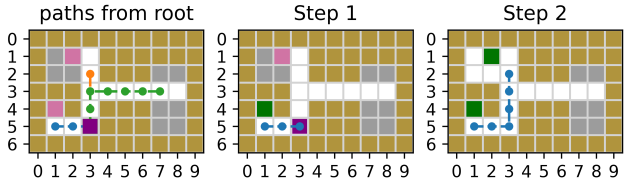


(c) Inverse hyperbolic cost path.

Figure 9: Step-by-step path trajectory under different cost functions. In a), the linear cost path is shown. The agent takes the shortest path to the nearest victim and stays there for several steps to get the exploration reward before moving to the other room to get the exploration reward. The agent runs out of time before picking up the next victim. In b), the hyperbolic cost path is shown. The agent takes a slightly longer path to the closest victim compared to the linear cost path, which positions it closer to the second room. However, the agent chooses not to explore it, remaining in the first room. In c), the inverse hyperbolic cost is shown, with the same trajectory as the linear cost path.



(a) 5s timeout



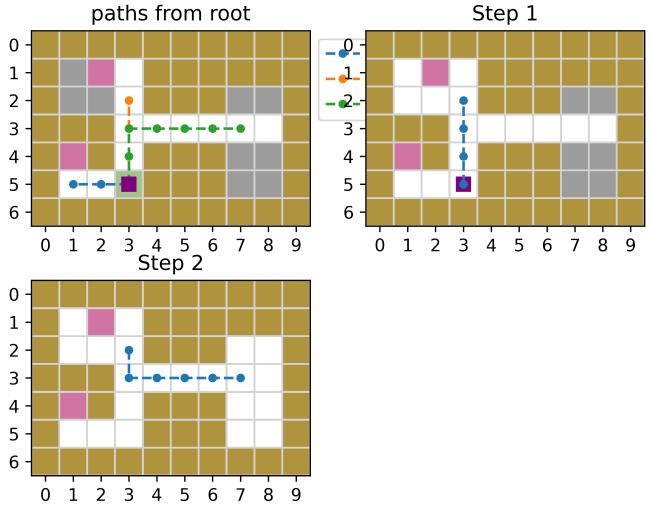
(b) 10s timeout

Figure 10: In a), the linear cost path for map 1, with a 5s timeout is shown. The agent explores two rooms and picks up one victim. In b), the first two steps of the linear cost path with a 10s timeout is shown. In this case, the agent explores the closest room that was not previously explored, picking up a victim there before also exploring the next room and picking up a victim there as well.

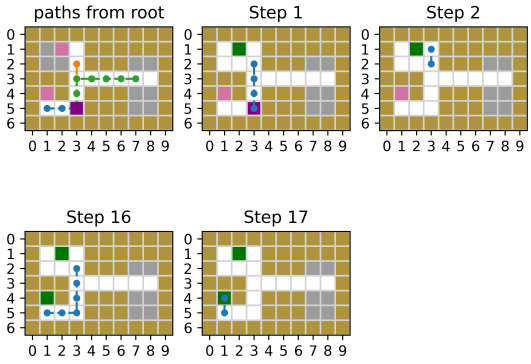
Also in map 1, with the hyperbolic cost function, we see that the agent opts to only collect exploration reward in the 5s timeout setting, presumably because the victim reward is not worth the time cost of picking up the victim (see Figure 11a). Compare this to the 10s timeout setting, where the agent does rescue victims (see Figure 11b).

Map 3 presents an interesting setting since there are two hidden rooms that are both 6 steps away from the agent's starting position. However, the rooms are different in shape with the lower room being long and skinny, so the expected number of steps to reach the victim is much greater in that room. In the case of linear cost, we compare the agent's path with 5s timeout to the path with 10s timeout (see Figures 12a and 12b). When faced with the shorter horizon, the agent chooses to explore the upper room first, but with the longer horizon, it first goes to the lower room. This makes sense because the cost of each step increases as time goes on, so it is more efficient for the agent to take more steps at the start of the task, which it is able to do with a longer time horizon.

For map 5, when faced with the hyperbolic cost function, the agent's strategy changes when it has a longer time horizon. In the 5s timeout case, it opts to first explore the closer room, and then the farther hidden room, which offers a greater exploration reward (see Figure 13a). In the 10s timeout case, though, it decides to expend more steps to go for the greater exploration reward first (see Figure 13b). This effect doesn't

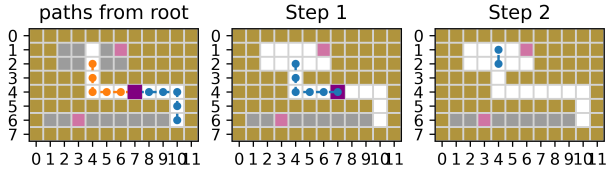


(a) 5s timeout

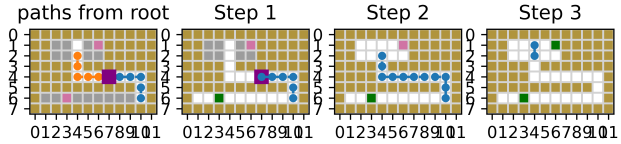


(b) 10s timeout

Figure 11: In a), the hyperbolic path for map 1 with a 5s timeout is shown. The agent explores all the rooms but does not pick up any victims. In b), selected steps for the hyperbolic path with 10s timeout is shown. The agent explores only two rooms but picks up victims it sees.



(a) 5s timeout



(b) 10s timeout

Figure 12: Map 3 is interesting since the two hidden rooms present differing numbers of expected steps to reach a victim if there is one hidden in the room. In a), the first two steps of the linear path for map 3 with 5s timeout is shown. The agent chooses to explore the upper room first. In b), the first three steps of the linear path for map 3 with 10s timeout is shown. The agent explores the lower room first.

occur with the linear and inverse hyperbolic cost functions, which makes sense since the hyperbolic function offers a much lower time cost at the beginning of the task.

Exploration vs Exploitation

After discovering a victim, the agent has two strategies it can take. The first is exploiting the exploration cost for remaining within line of sight for the victim, a reward that it can guarantee. The second is exploring other rooms to potentially find other victims. In the maps that we observed, we found the agent using a combination of both strategies.

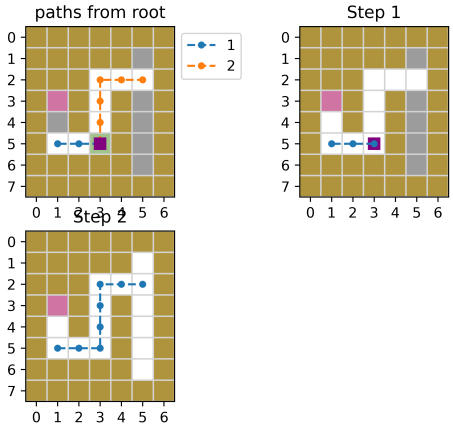
In Figure 14 below, we can see that after finding the first victim, the agent chooses to exploitation over exploration, remaining within line-of-sight of the victim to achieve the guaranteed reward for a time instead of branching out to explore more rooms.

We can see the opposite behavior in 10b in the previous section. The increased time horizon allows the agent to explore all the rooms and pick up all the victims in the map instead of just choosing to stay in one place.

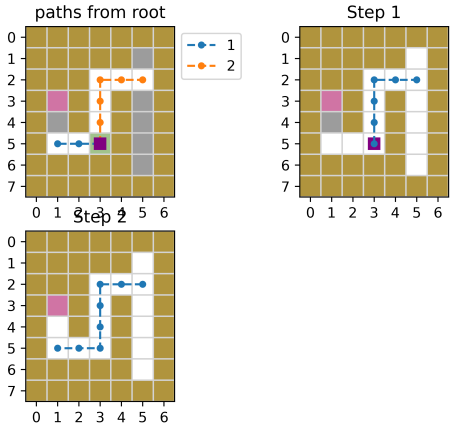
Future Work

Model

In the future, we'd like to extend our model in a couple of ways. First, in our results above, linear and inverse hyperbolic cost functions produced much of the same behavior, potentially because the structures of the functions were very similar (see Figure 2). We'd like to experiment with some other forms of the inverse hyperbolic function to see if we can observe a greater difference in the trajectories. In addition, with our current scaling, the cost at the timeout is consistent across



(a) Hyperbolic cost path for map 5 with 5s timeout



(b) Hyperbolic cost path for map 5 with 10s timeout

Figure 13: In a), the hyperbolic path for map 3 with 5s timeout is shown. The agent chooses to explore the closer room first. In b), the hyperbolic path for map 3 with 10s timeout is shown. The agent first explores the farther room, which offers a greater exploration reward.

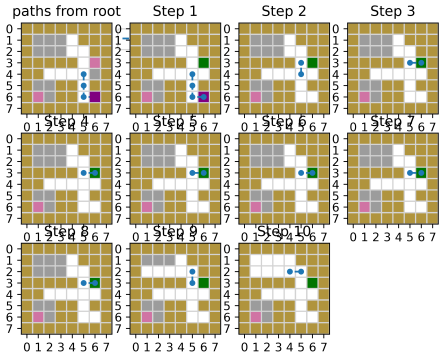


Figure 14: Linear cost path for map 6 with 5s timeout. The agent chooses exploitation over more exploration after finding the victim on the right side of the map.

all cost functions. However, for any given time t that is not 0 or timeout, the inverse hyperbolic function always has value greater than the linear function, which is always greater than the hyperbolic function. We'd like to try scaling the functions such that area under the curve is the same instead.

Secondly, our currently reward structure gives an exploration reward for the victim when the agent is in sight of it, even when the victim has already been picked up. While in the real-world, this could be useful for scenarios in which victims are injured and require some monitoring and provides some exploration vs exploitation trade-offs, we'd also like to test trajectories in which the exploration reward is provided only once for the victim, the first time it is seen.

In this paper, we primarily explore different settings of the cost function and timeout. There is also potential in exploring variations in other parameters that we treated as constants, such as the ratio of exploration reward to victim pickup reward and various settings for gamma, the discount factor. Our code also currently only runs on smaller maps with shorter timeout periods, but we'd like to optimize it to explore a greater variety of maps in the long-run.

Finally, we'd like to create some random agents or heuristic-driven agents to compare a greater variety of possible trajectories.

Acknowledgments

We would like to thank Professor Josh Tenenbaum for a wonderful semester of Computational Cognitive Science lectures, as well as Suhyoun Yu and Marta Kryven for their guidance, expertise, and invaluable feedback throughout this project.

In terms of contribution, Joanne and Amber worked jointly on the simulation model, including expanding and adapting an existing script to fit this search-and-rescue problem, experimenting with different parameter settings, and producing best path trajectories and figures; they also wrote the sections of the paper relevant to modeling. Adam was primarily responsible for the behavioral experiment, including setup,

code, and the relevant paper sections.

References

Bigoni, C. M. S.-A. S. G., M. (2015). Time horizon and cooperation in continuous time..

Embrey, F. G. R.-Y. S., M. (2017). Cooperation in the finitely repeated prisoner's dilemma..

Kryven, Y. S. K.-W. M. T. J., M. (2021). Adventures of human planners in maze search task..

Yang, K. M. S.-H. T. J., Z. (2021). Modeling human planning in a life-like search-and-rescue mission. In *Proceedings of the annual meeting of the cognitive science society* (p. 43).