Behavioral Learning of a Fuzzy Decision-Maker

Andrew Buck and James Keller
University of Missouri

4/24/2015

Modeling Decision-Makers

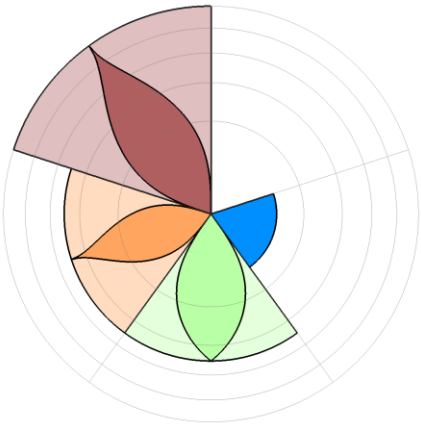
Humans are expert decision-makers, but it's not always clear what is being optimized.

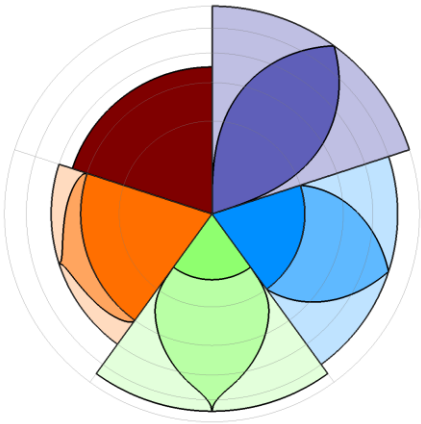
People generally make *rational* decisions, but they may not have perfect information, or enough time to fully evaluate.

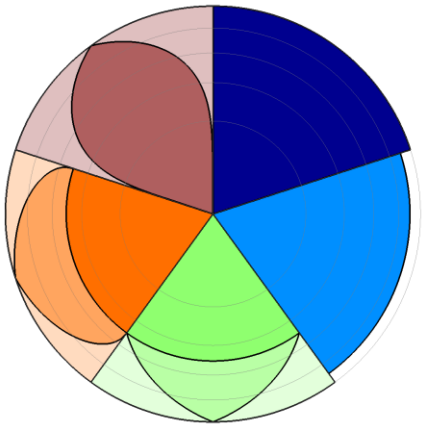
We use *fuzzy weighted graphs* to model the *mental map* of a decision-making agent and the principle of *bounded rationality* to model how to make choices.

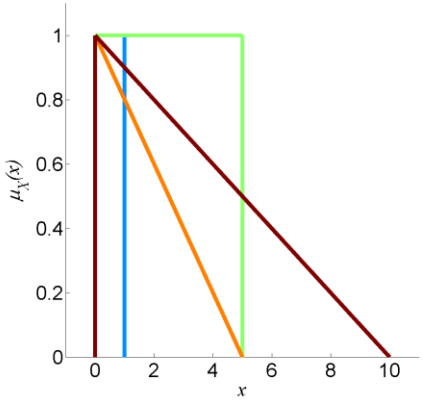
Fuzzy Rose Diagrams

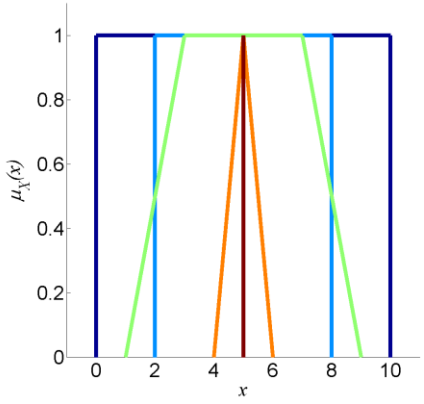
Fuzzy rose diagrams are a compact way to display vectors of fuzzy numbers.

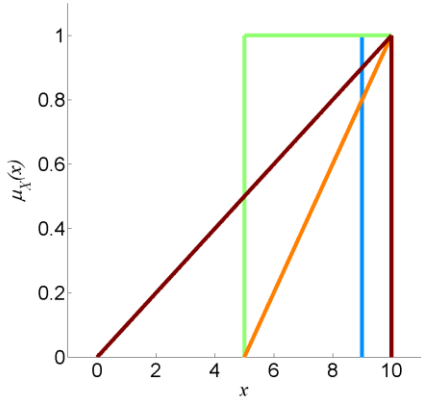




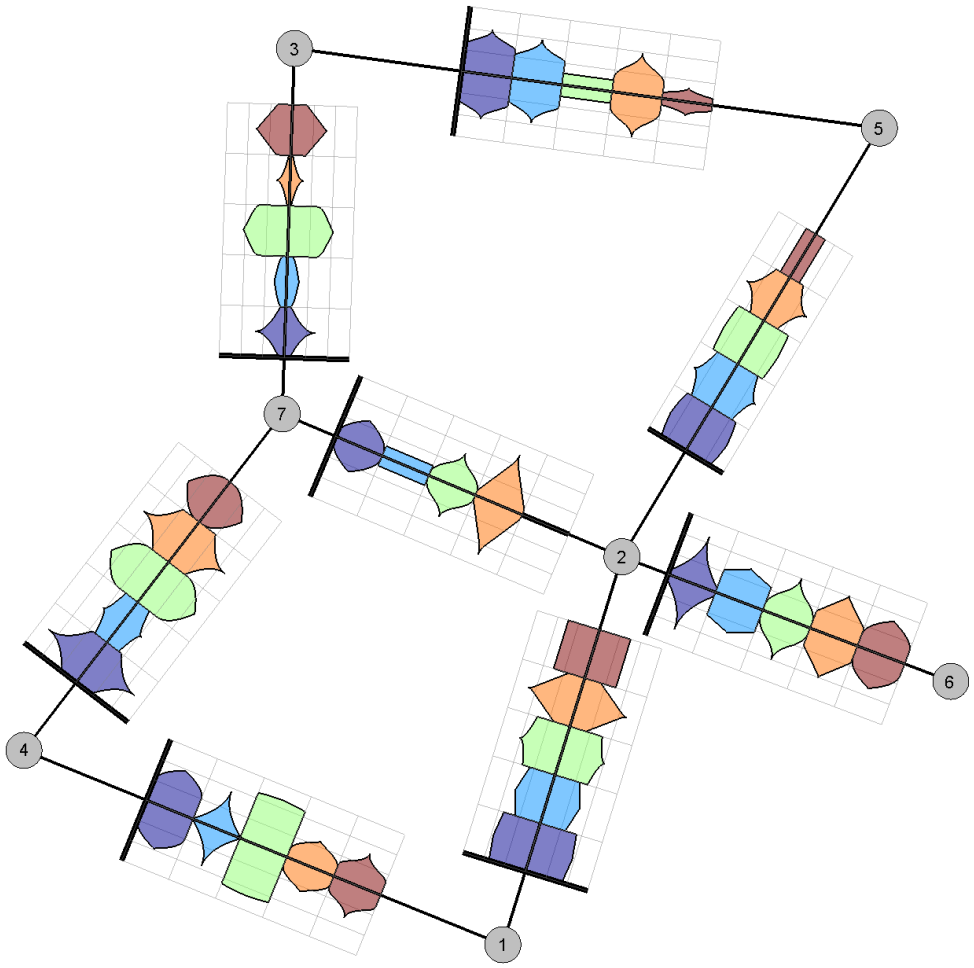








Fuzzy Weighted Graphs



Fuzzy weighted graphs have fuzzy weight vectors attributed to the edges (and possibly nodes).

Each edge is an unwrapped linear fuzzy rose diagram.

A reference axis defines the order of the features.

Features are mirrored across the edges.

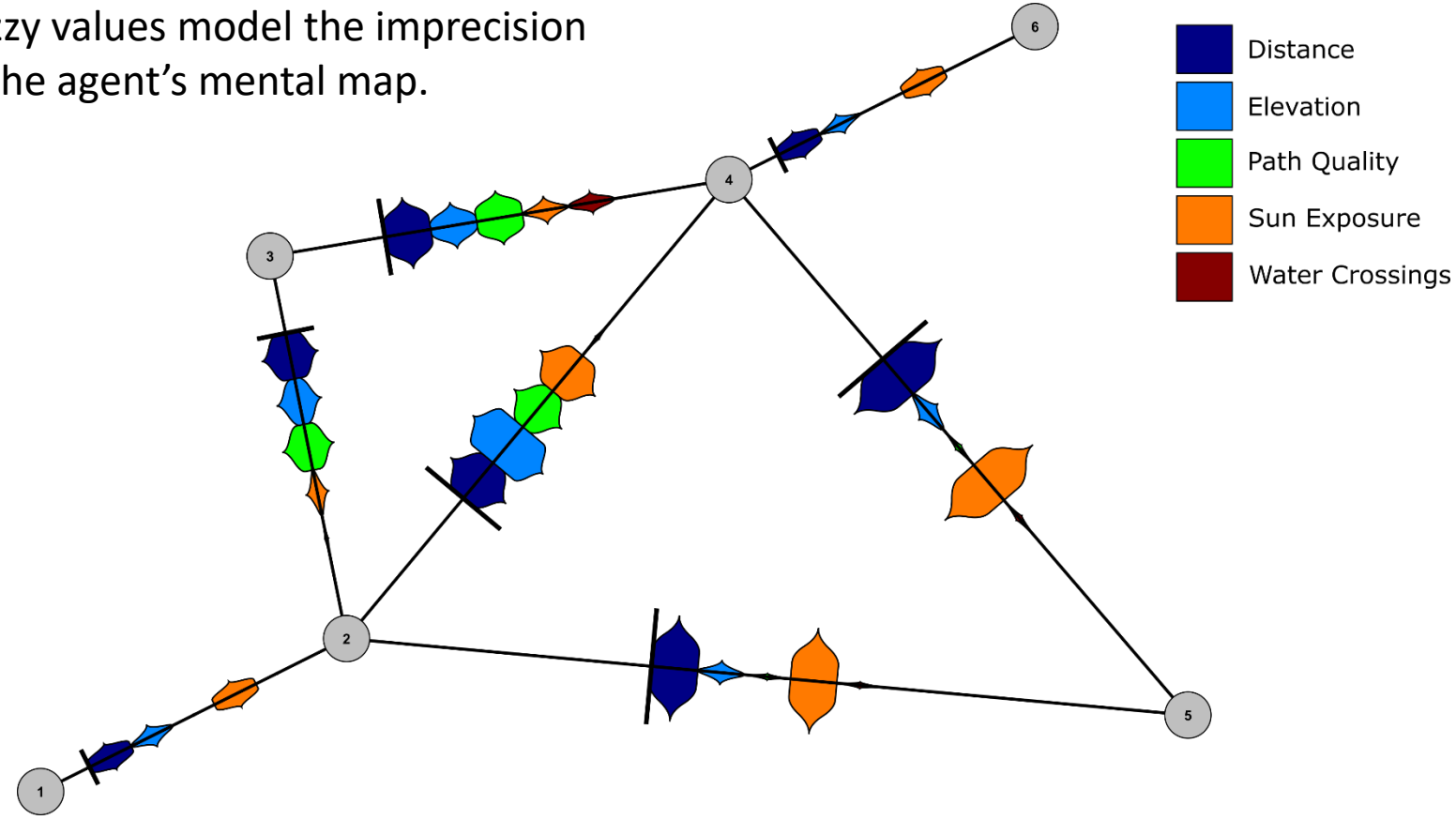
An Example Scenario

Consider a navigation problem with three possible choices:

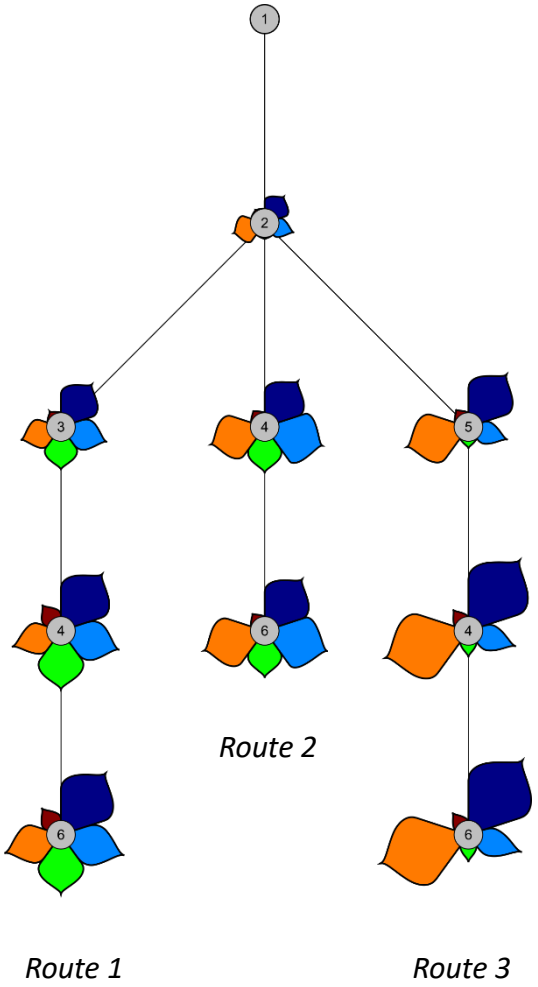
An Example Scenario

We can represent the environment as a fuzzy weighted graph.

Fuzzy values model the imprecision in the agent's mental map.

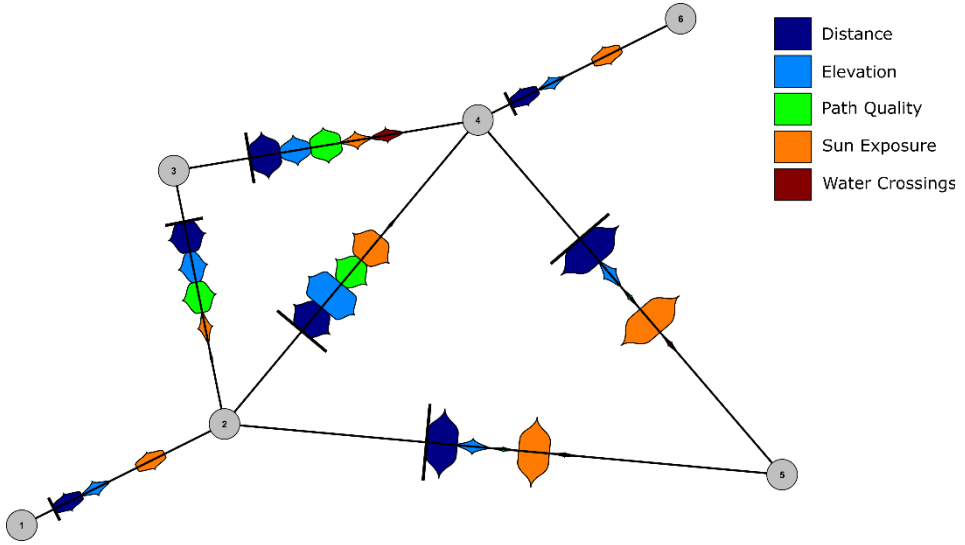


An Example Scenario



We can unfold the graph into a decision tree.

- Agent starts at the root node.
- Each leaf node represents a different path.
- Features are aggregated along the paths.



Bounded Rationality as OWA

The agent must now choose from among these alternatives.

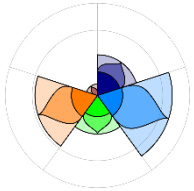
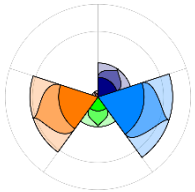
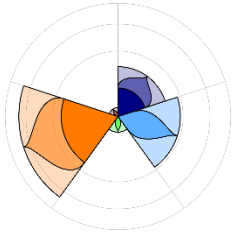
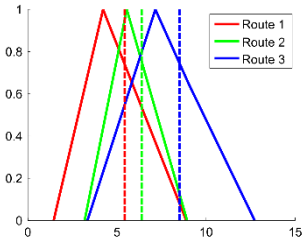
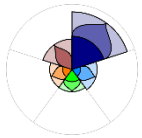
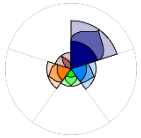
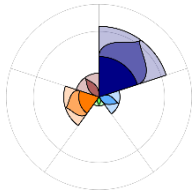
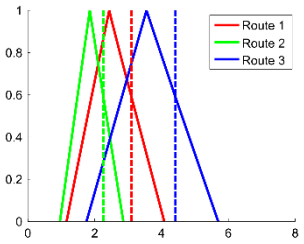
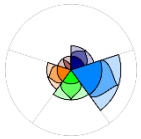
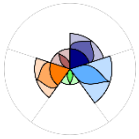
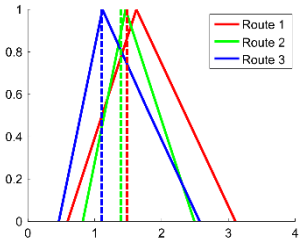
The agent is defined by:

- A vector of feature biases $\boldsymbol{\beta} = [\beta_1, \dots, \beta_N]$
- A vector of OWA weights $\boldsymbol{\omega} = [\omega_1, \dots, \omega_N]$
- An optimism/pessimism parameter λ used for defuzzification
  (The Liou and Wang index is used, which gives a linear weighting of the left and right integrals of the fuzzy number.)

We define the agent's cost interpretation of an aggregated fuzzy feature vector $\boldsymbol{F} = [f_1, \dots, f_N]$ as

$$C = \omega_1 a_{\sigma(1)} + \cdots + \omega_N a_{\sigma(N)}, \qquad \text{where}$$
$$a_i = \beta_i f_i$$
$$\sigma: [1, N] \to [1, N] \quad s.t. \quad a_{\sigma(i)} \geq a_{\sigma(i+1)}$$

Bounded Rationality as OWA

Agent Profile	Route 1	Route 2	Route 3	OWA Comparison
$\beta = [0.2, 0.9, 0.3, 0.8, 0.1]$ $\omega = [0.9, 0.8, 0.6, 0.2, 0]$ $\lambda = 0.7$				
$\beta = [0.4, 0.1, 0.1, 0.1, 0.7]$ $\omega = [1, 0.8, 0.5, 0.2, 0.1]$ $\lambda = 0.9$				
$\beta = [0.1, 0.4, 0.3, 0.1, 0.5]$ $\omega = [1, 1, 0.9, 0.7, 0.5]$ $\lambda = 0.3$				

Learning the Agent Profile

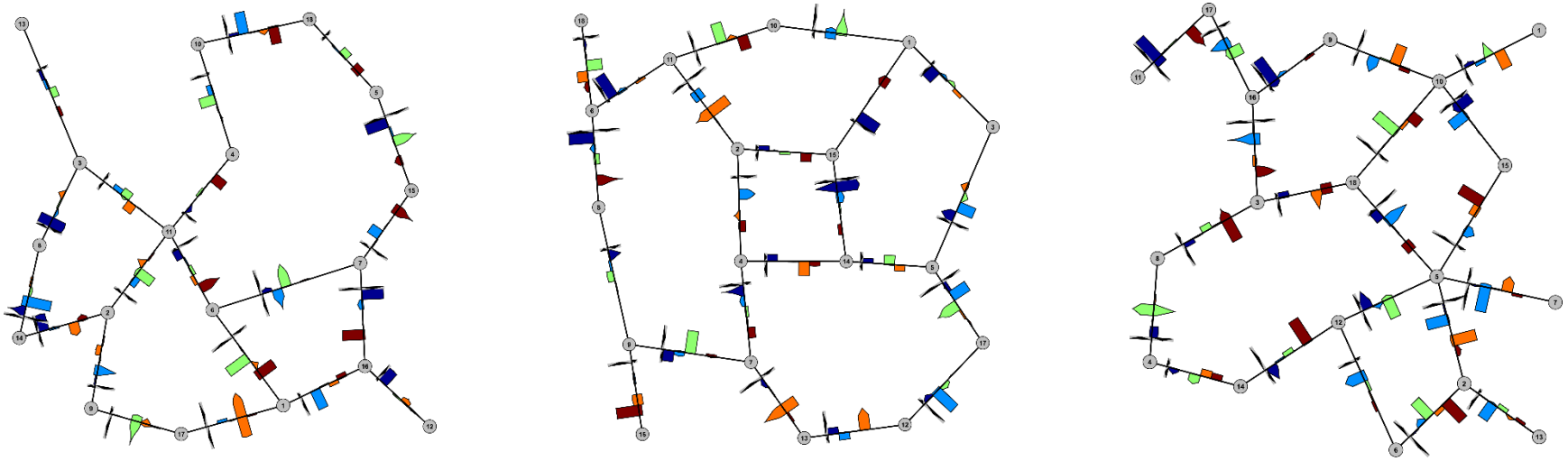
Can we learn the agent parameters by observing its choices?

Experiment:

- Generate a dataset of many different decision scenarios for 100 randomly sampled agent parameter vectors $\boldsymbol{x} = [\beta_1, \ldots, \beta_N, \omega_1, \ldots, \omega_N, \lambda]$ where each $x_i \in [0, 1]$.
  - $N = 5$
  - OWA weights sorted such that $\omega_i \geq \omega_{i+1}$
- Record the choice each agent makes for every scenario.
- Build a genetic algorithm to learn the agent profile.

Generating the Dataset

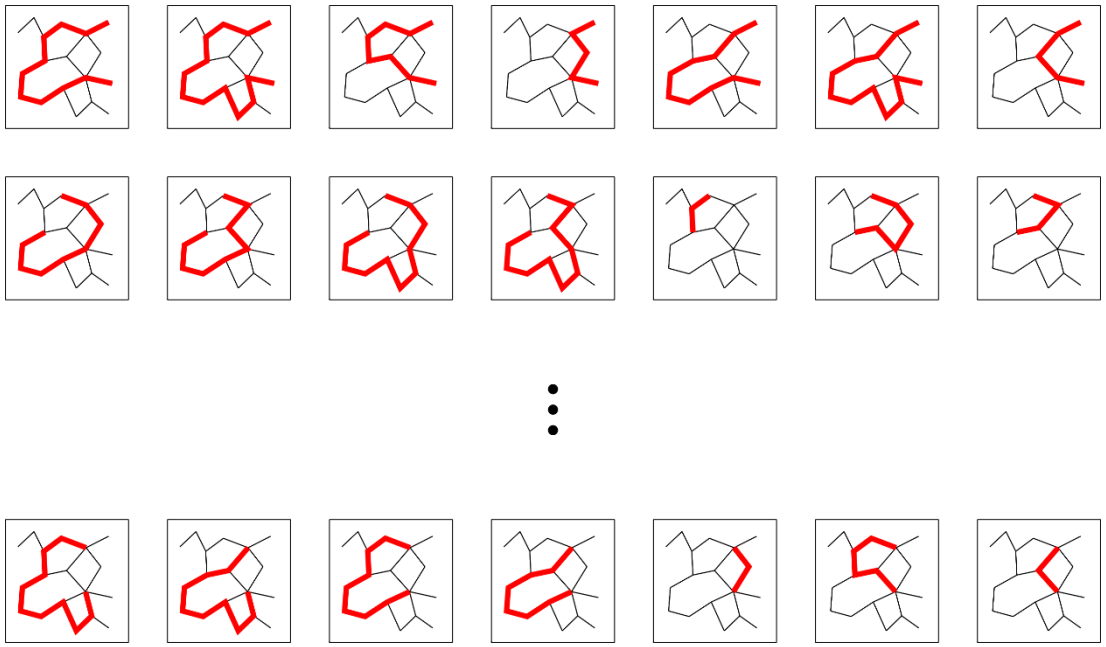
For each agent, we create three random fuzzy weighted graphs.



Each graph is a directed Urquhart graph computed from a set of uniformly sampled vertices. The Urquhart graph is obtained by removing the longest edge from each triangle in the Delaunay triangulation of the vertices.

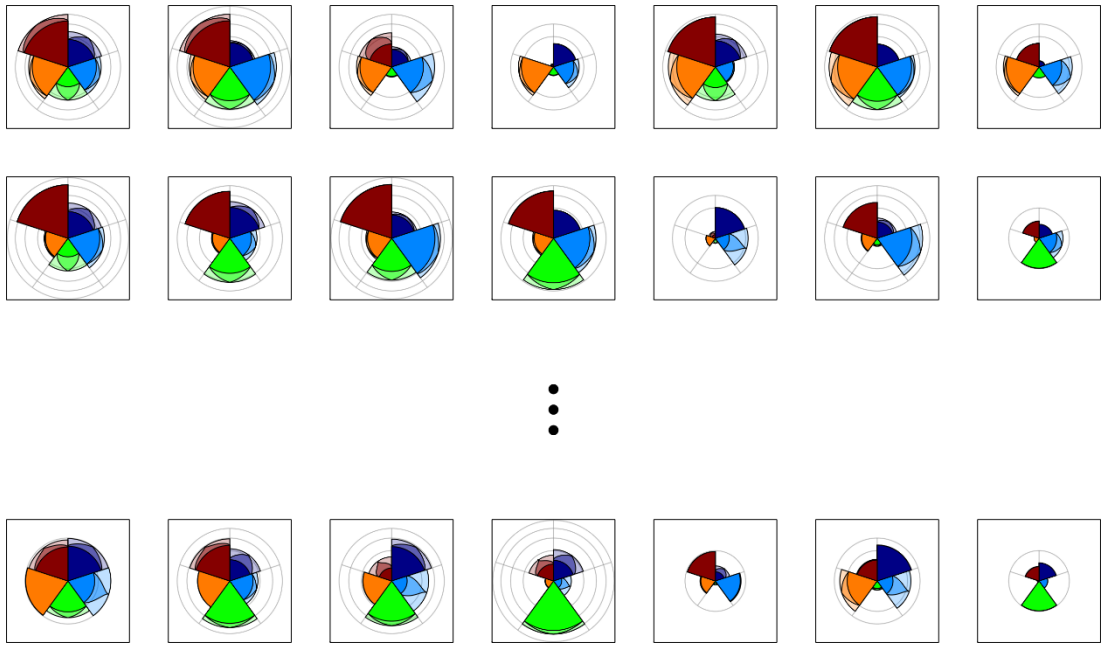
Generating the Dataset

For each pair of points in a graph, we find the set of all non-looping paths between them. Each set represents the decision-making scenario of finding a route between these two points.



Generating the Dataset

We then evaluate each path according to the agent's profile. The results are ranked and the best (least cost) path is chosen.



Designing the GA

Individual chromosomes represent the agent parameter vector $\boldsymbol{x} = [\beta_1, \ldots, \beta_N, \omega_1, \ldots, \omega_N, \lambda]$ where each $x_i \in [0, 1]$.

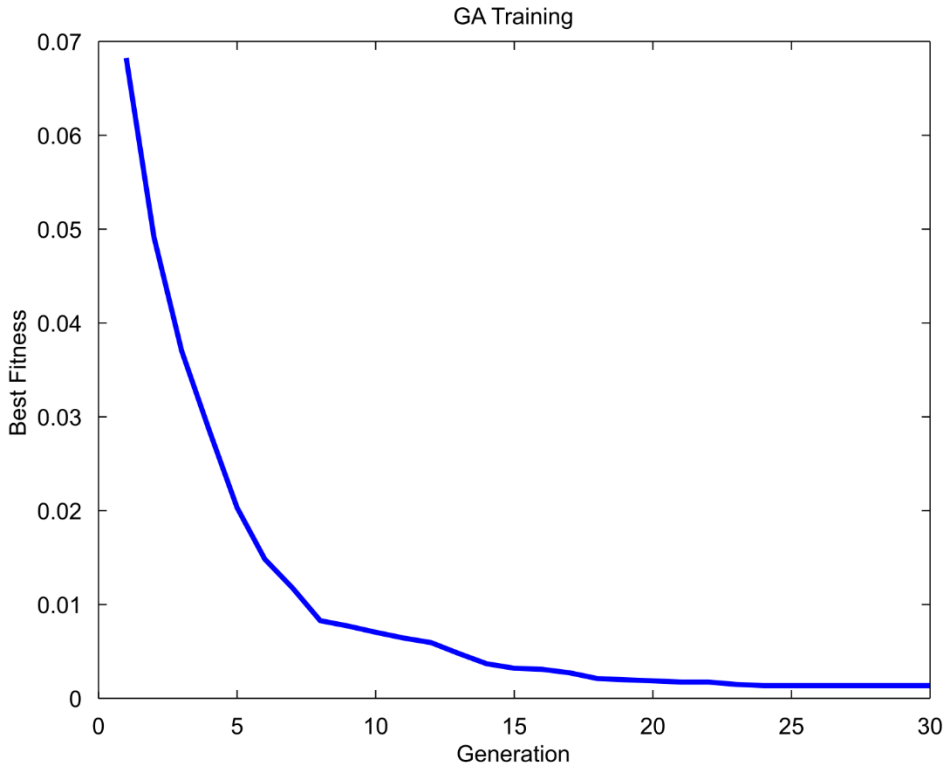
To compute the fitness, we evaluate each decision scenario according to the chromosome profile and measure the percentage of scenarios for which the incorrect choice is made.

We perform 5-fold cross validation using a population size of 200, uniform crossover rate of 0.8, Gaussian mutation rate of 0.2, and an elite size of 10.

The algorithm runs until a solution is found with perfect prediction accuracy or there are 10 stall generations.

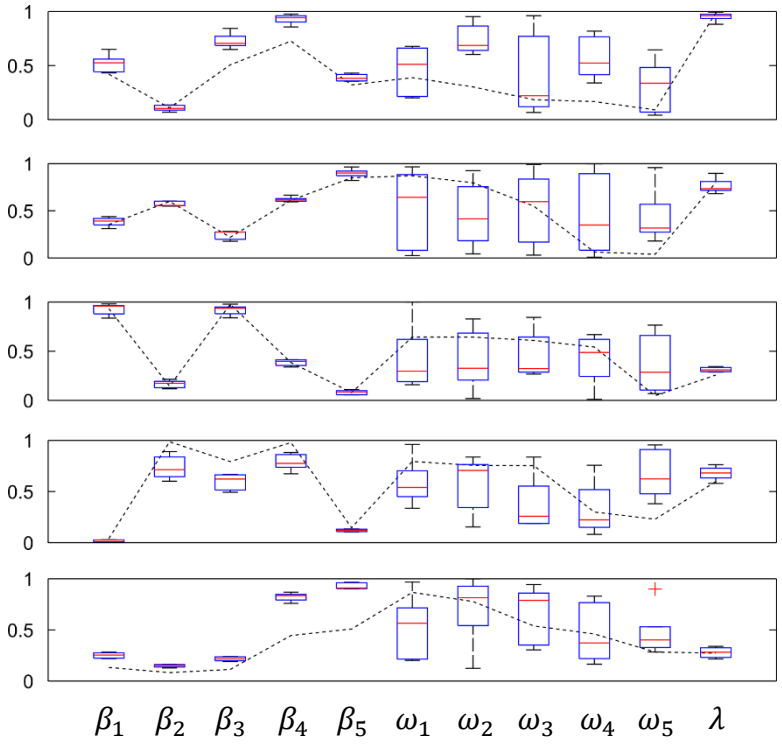
Results

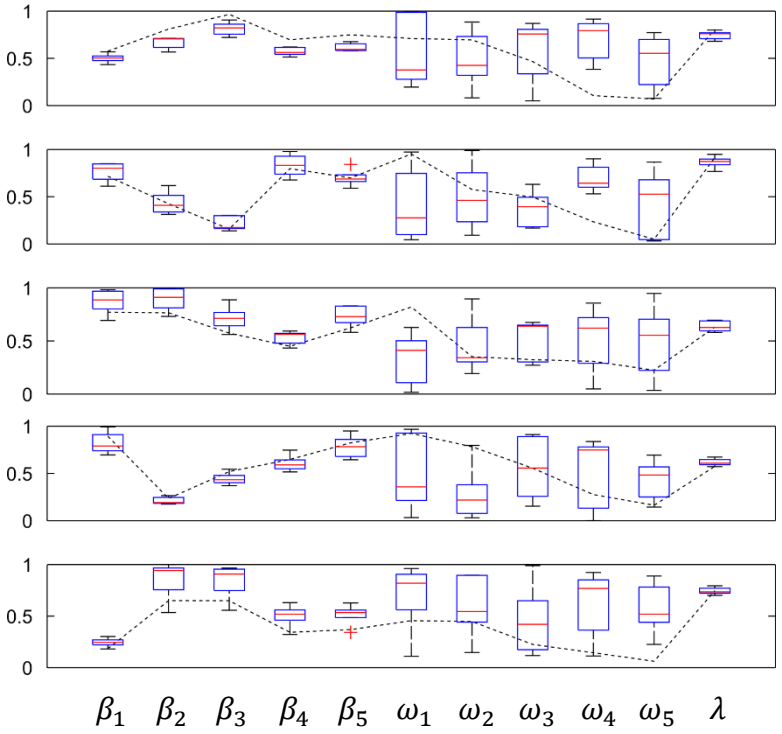
Overall average prediction accuracy rate: 99.34%



Results

Example parameter prediction distributions:





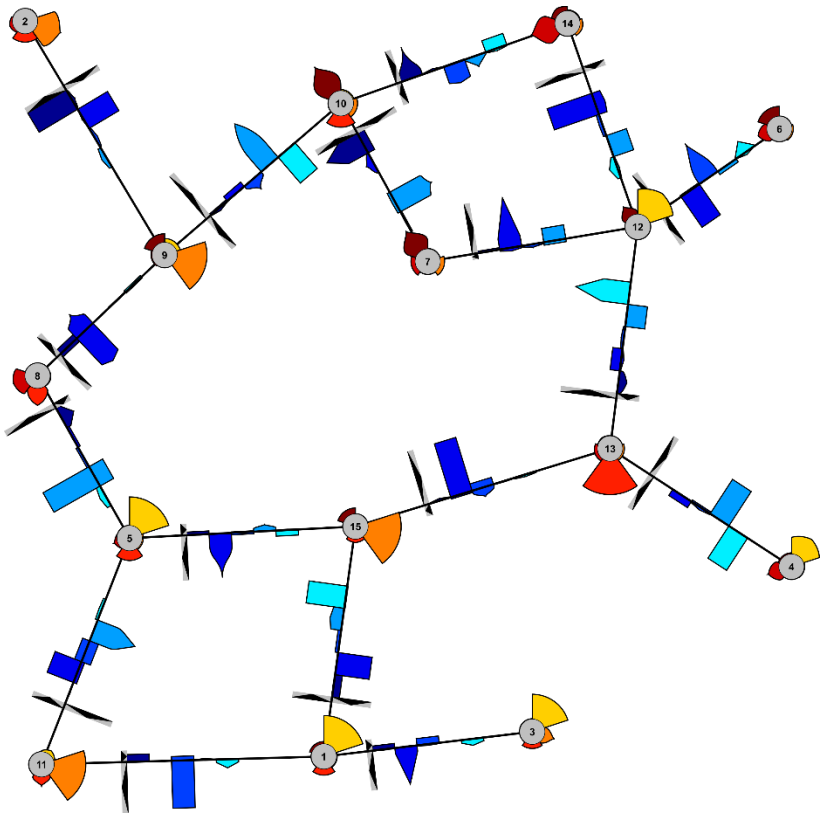
Next Steps

Our results show that the agent bias terms can be recovered, but the OWA weights may be more difficult.

Future work:

- Stochastic models to add uncertainty to the policy
- Modeling spatial uncertainty and conflating with ground truth
- Understanding agent desires and beliefs with a Bayesian Theory of Mind
- Learning the reward function with Inverse Reinforcement Learning

Stochastic Models

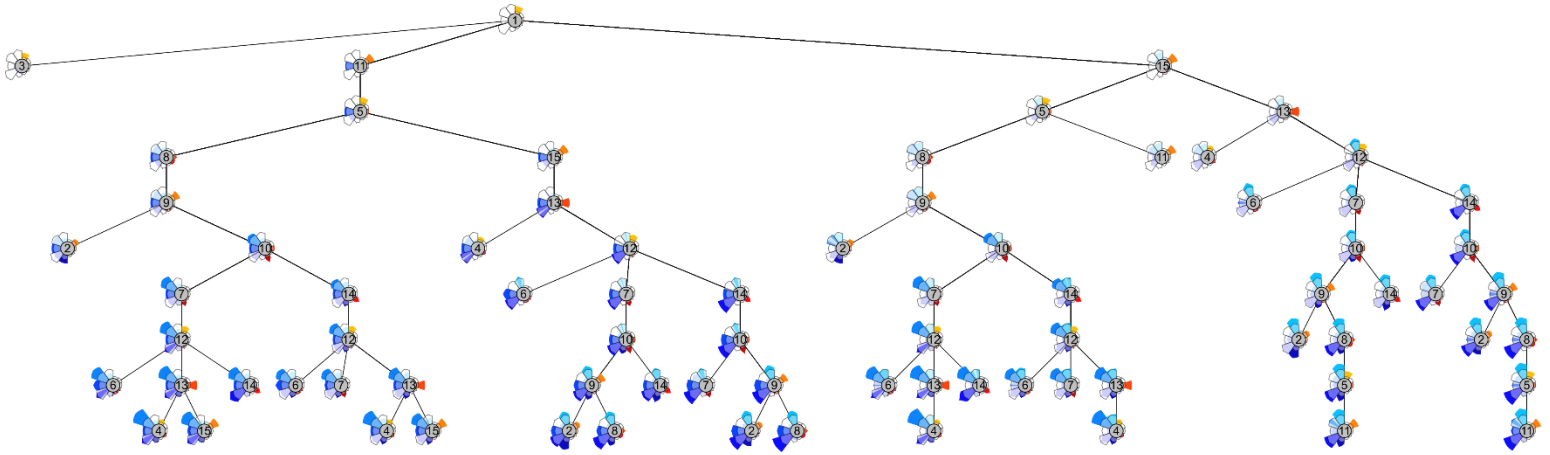


Mental map graph has a reward vector for each destination node and a cost vector for each edge.

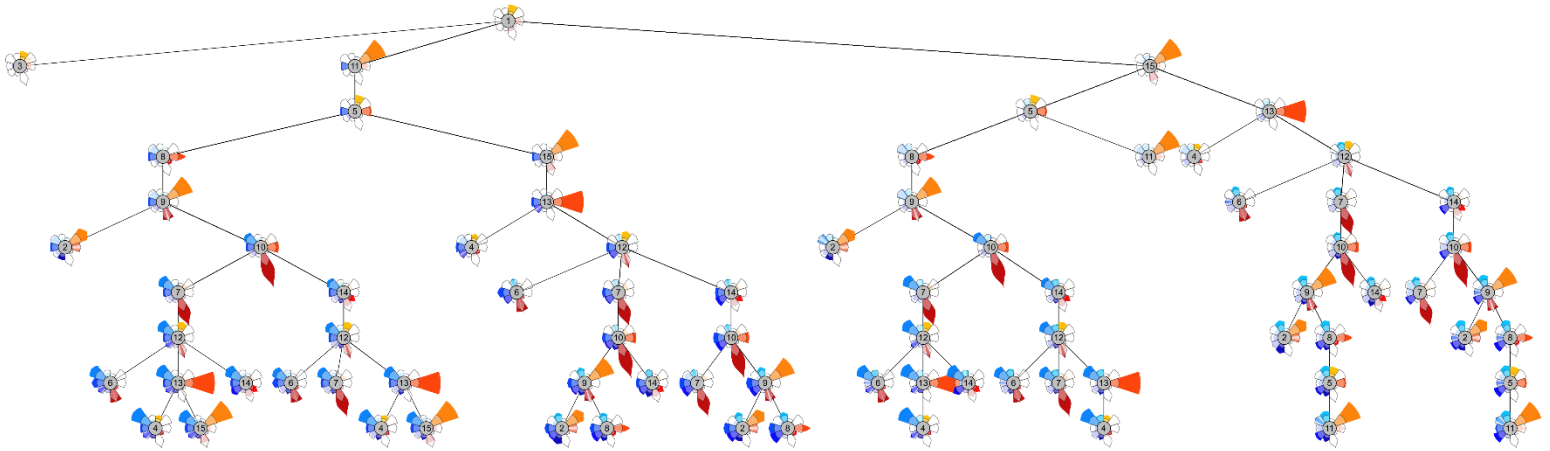
All possible paths to all possible destinations are considered when planning where to go.

Stochastic Models

Unbiased decision tree

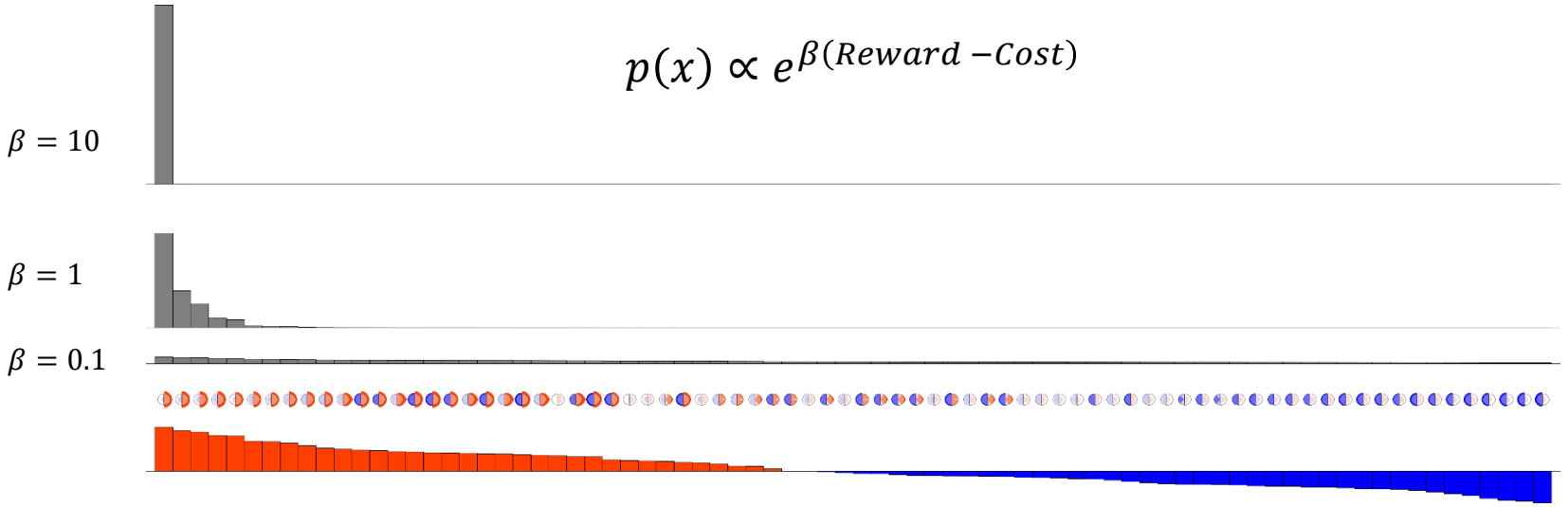


Decision tree with agent biases

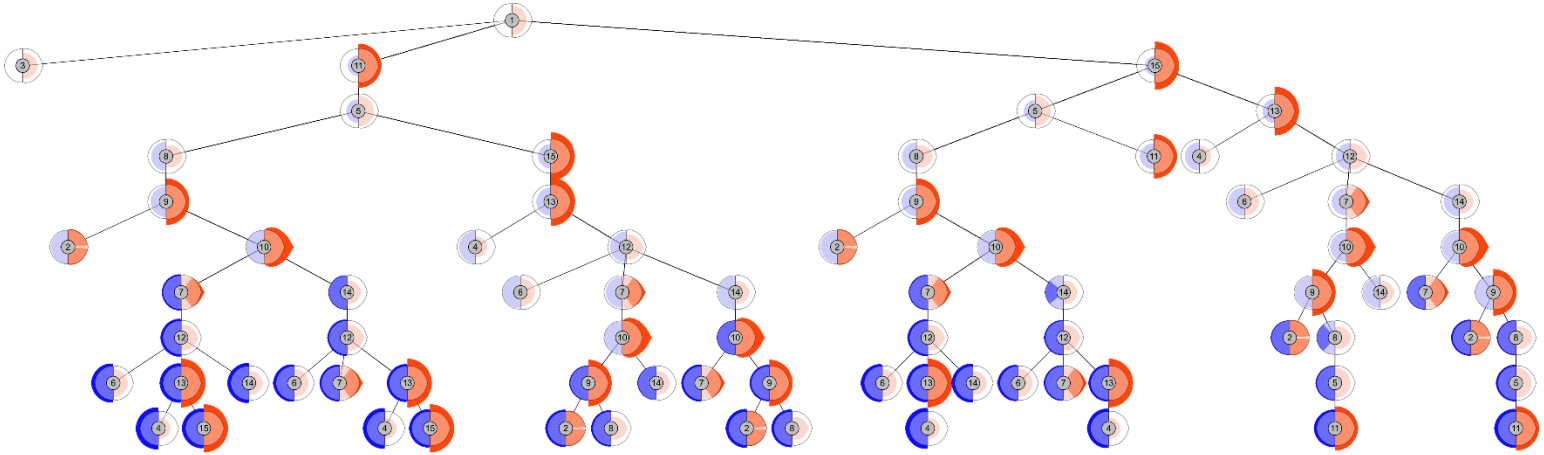


Stochastic Models

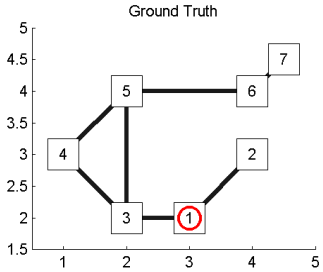
$$p(x) \propto e^{\beta(Reward - Cost)}$$

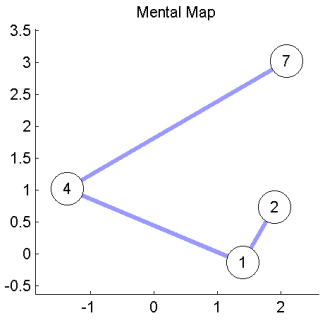


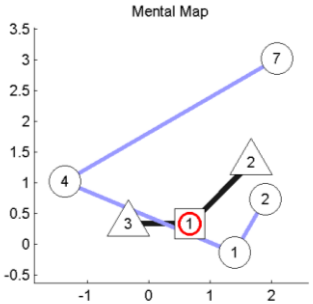
Decision tree with OWA costs (blue) and rewards (red)

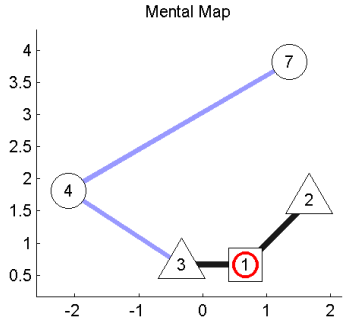


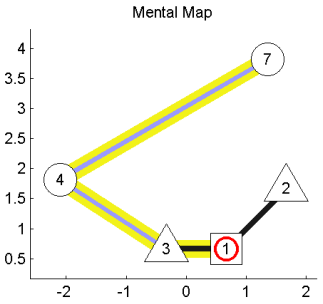
Conflating with Ground Truth

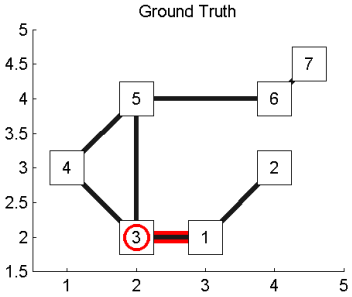


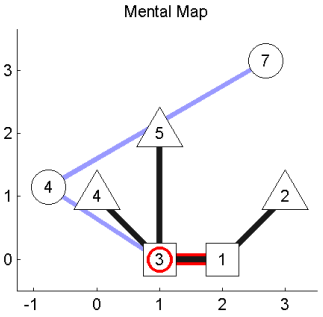


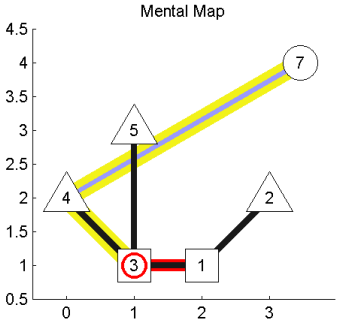


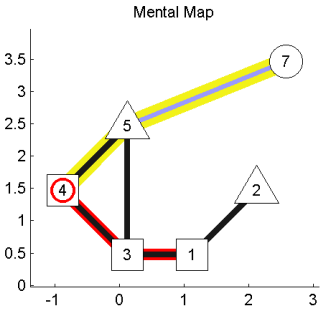


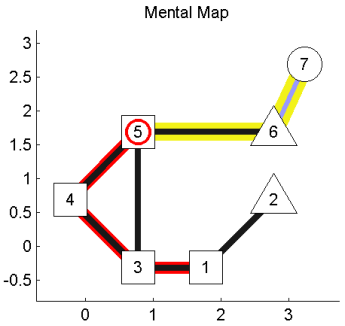


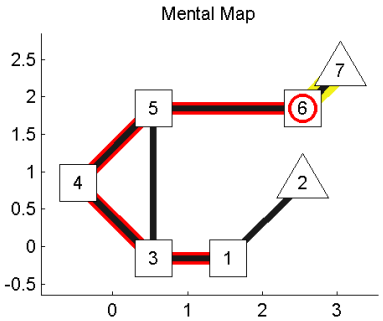


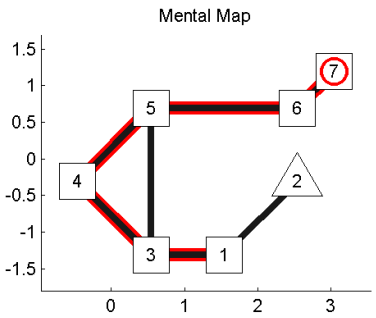






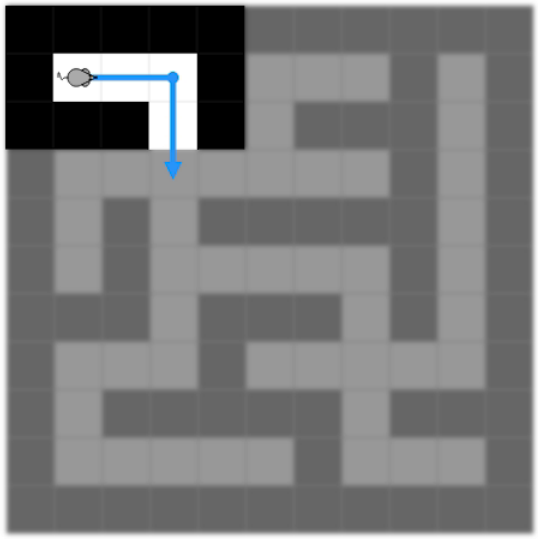




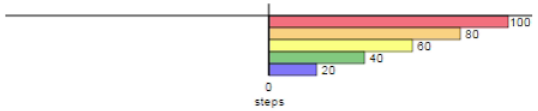


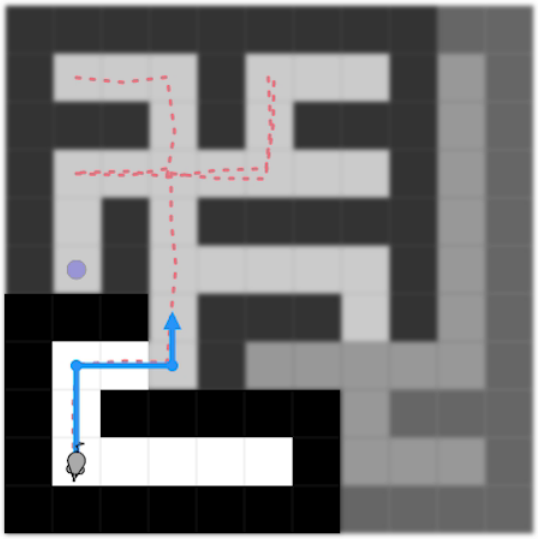
Bayesian Theory of Mind

Can we infer the agent's reward structure and mental map state just by observing their actions in the environment?

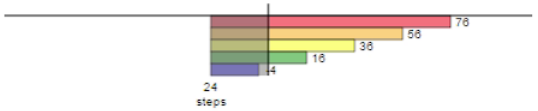


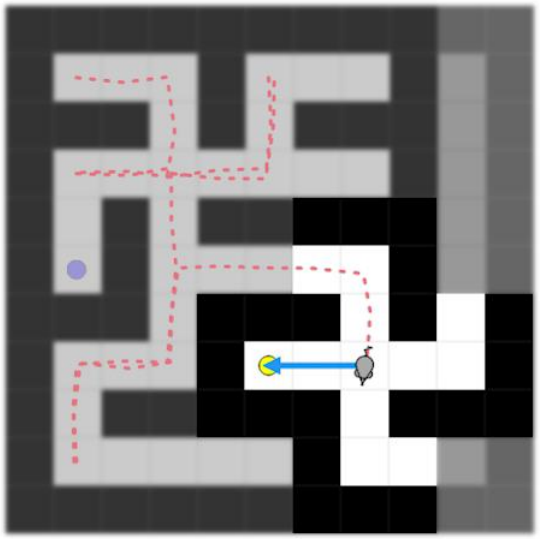
This mouse is searhing for one of five possible rewards in the maze.



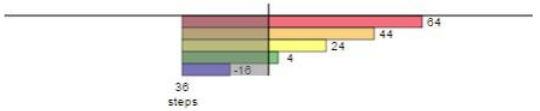


As the mouse moves, it expends energy, decreasing the total reward it can receive.





Here, the mouse has settled for the yellow reward.



Inverse Reinforcement Learning

Reinforcement Learning is the problem of determining an optimal policy for a given environment in order to maximize a given reward.

Inverse Reinforcement Learning is defined as follows:

Given

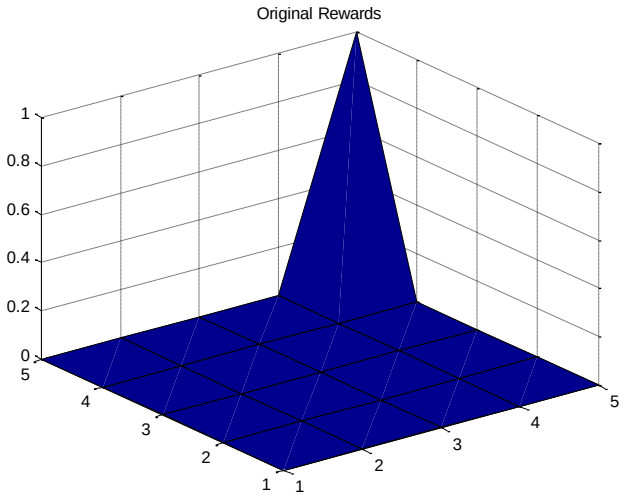
1. Measurements of an agent's behavior over time, in a variety of circumstances
2. If needed, measurements of the sensory inputs to that agent
3. If available, a model of the environment

Determine

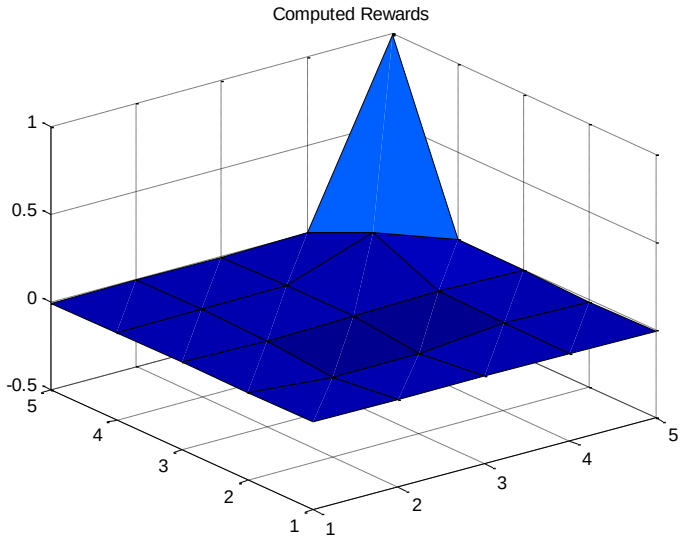
The reward function being optimized

Inverse Reinforcement Learning

1. Generate a gridworld environment and place rewards.
2. Compute an optimal policy.
3. Given only the policy and the transition function (stochastic), determine the reward being optimized.



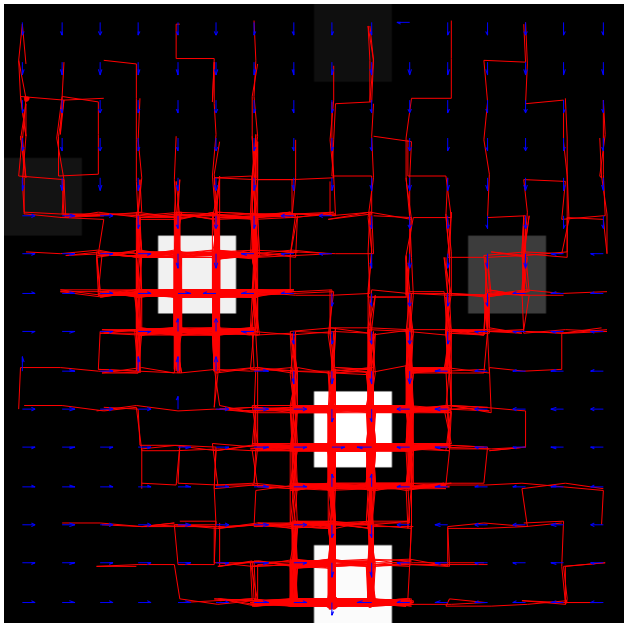
5	0.244 ▶	0.338 ▶	0.484 ▶	0.694 ▶	1.000
4	0.192 ▶	0.260 ▶	0.362 ▶	0.502 ▶	0.694 ▲
3	0.142 ▲	0.190 ▶	0.262 ▶	0.362 ▲	0.484 ▲
2	0.106 ▲	0.138 ▲	0.190 ▲	0.260 ▲	0.338 ▲
1	0.080 ▶	0.106 ▶	0.142 ▶	0.192 ▲	0.244 ▲
	1	2	3	4	5



Inverse Reinforcement Learning

1. Generate trajectories through an environment with a known reward function.
2. Compute a feature vector for each trajectory.
3. Estimate the policy from the trajectories and learn the feature weights.
4. Infer the original reward function from the learned weights.

Original Rewards



Computed Rewards