Multicriteria Pathfinding in Uncertain Simulated Environments

Presented by
Andrew Buck

In Partial Fulfillment of the Requirements for the Degree
Doctor of Philosophy

Advisor: Dr. James Keller
Committee: Dr. Alina Zare, Dr. Marjorie Skubic, and Dr. Mihail Popescu

University of Missouri

May 9, 2018

Motivation

Suppose you had to plan a route to some goal, and were faced with multiple routes.

Each route has different qualities that make it more or less appealing.

Which route would you take?

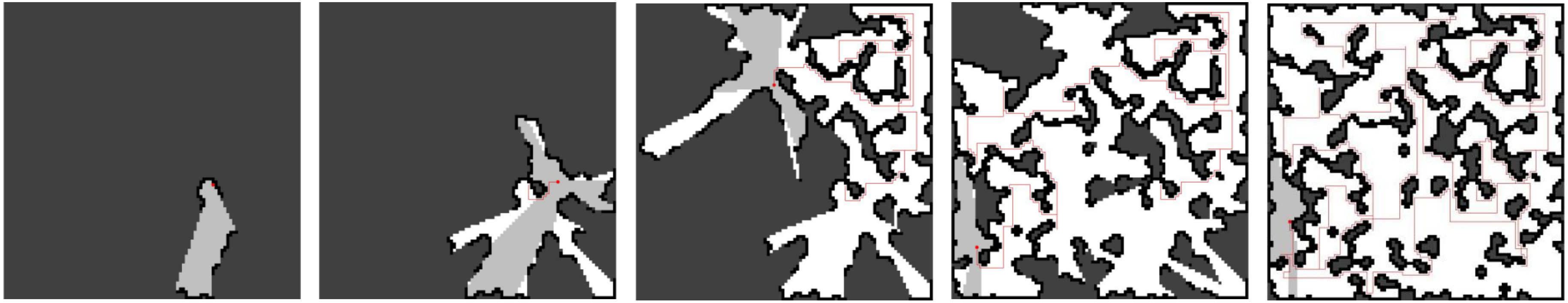
How might you design an autonomous agent to act in your place?

Motivation

What happens if the environment is only partially observable?

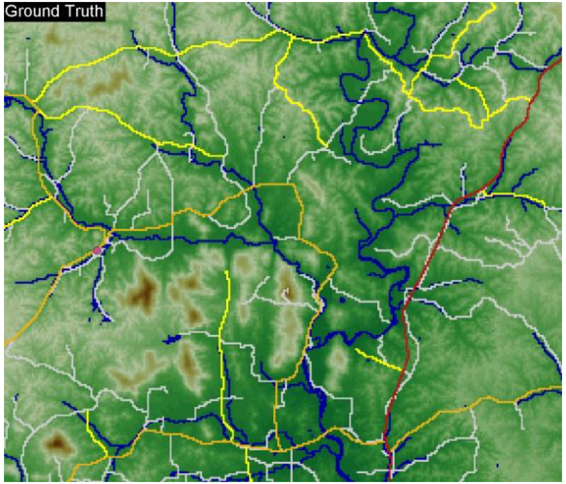
How should the agent explore the environment?

How does the agent manage the uncertainty?

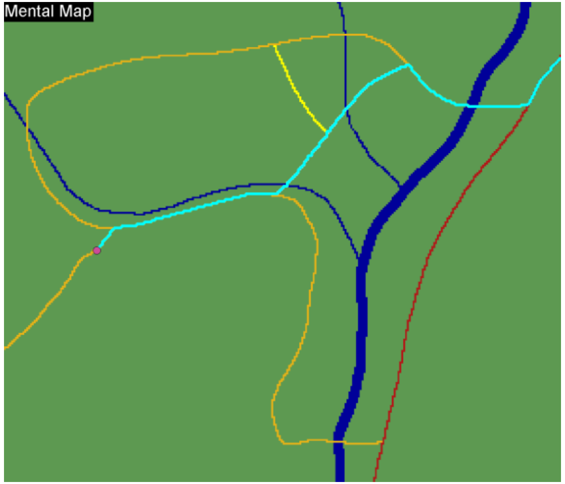


Motivation

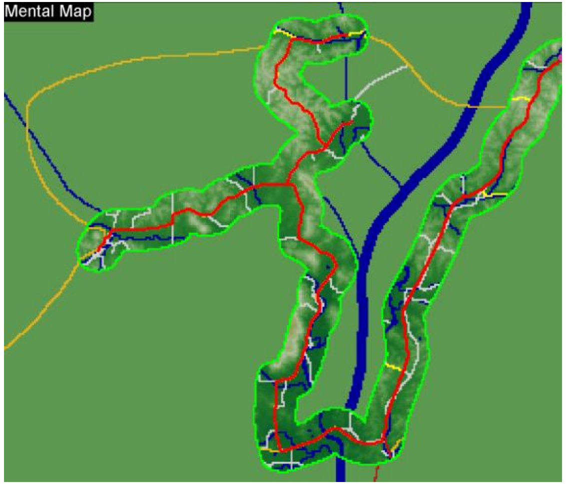
What does the agent’s mental map look like?



A real-world environment has various features representing the ground-truth.



The agent has a simplified version of the environment used for planning.



As the agent explores, it discovers new information and updates its mental map.

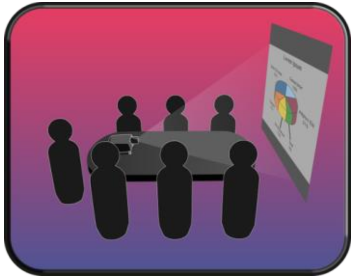
Applications

In general, these are

Sequential Multicriteria Decision-making Problems with Uncertainty

Some more examples:
- Navigating through physical environments
- Optimal packet routing on computer networks with uncertain loads
- Making long-term business decisions based on variable market factors
- Designing optimal strategies for games with hidden information





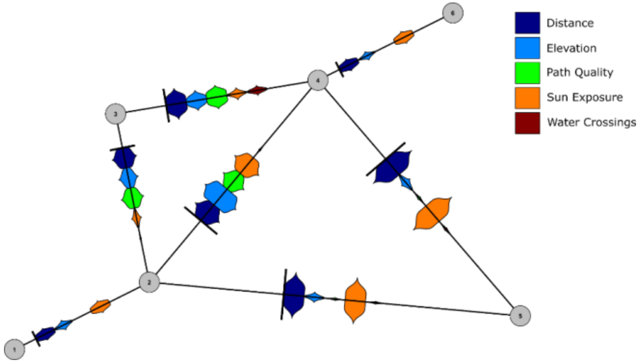

Why Pathfinding?

We can represent these as pathfinding problems:

- Represent the problem space as a fuzzy weighted graph
- Choose a sequence of actions that leads to the “best” outcome

The pathfinding domain is ideal to study these types of problems.

- Simple to visualize and interpret
- Proxy problems for other domains



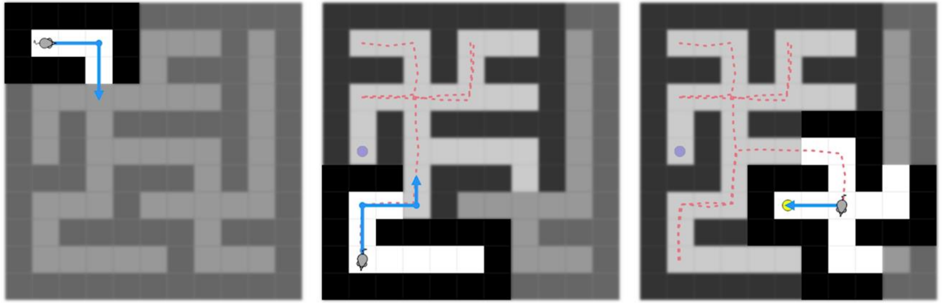
Why Simulated?

"Real" problems can be difficult to study.

- Example: Movement history with GPS tracker
  - Data may be incomplete
  - Don't know agent's goals or preferences
  - Limited availability

Simulations give greater control.

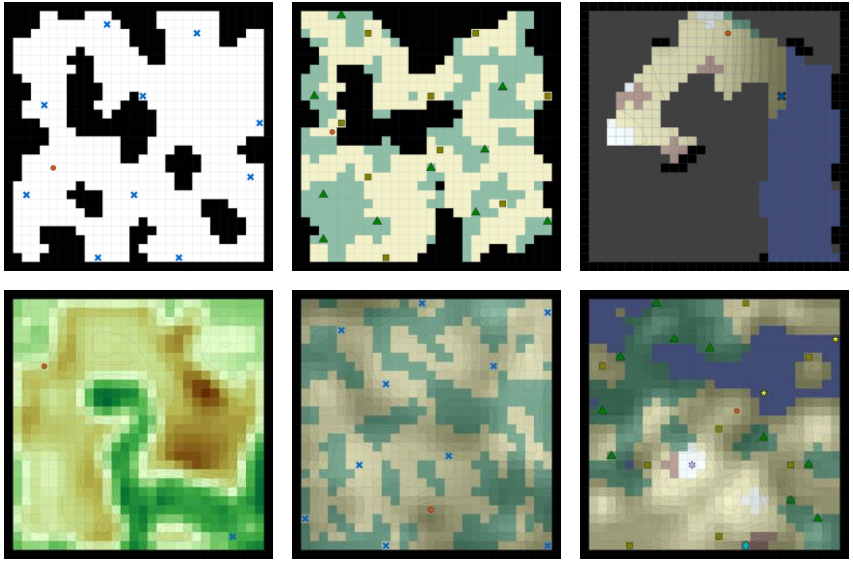
- We can create problems that investigate specific questions
- Easier to create a mental map of what the agent knows
- Potential to create an unlimited number of scenarios



The CMM Framework

The Computational Mental Map (CMM) Framework was developed to study these types of problems.

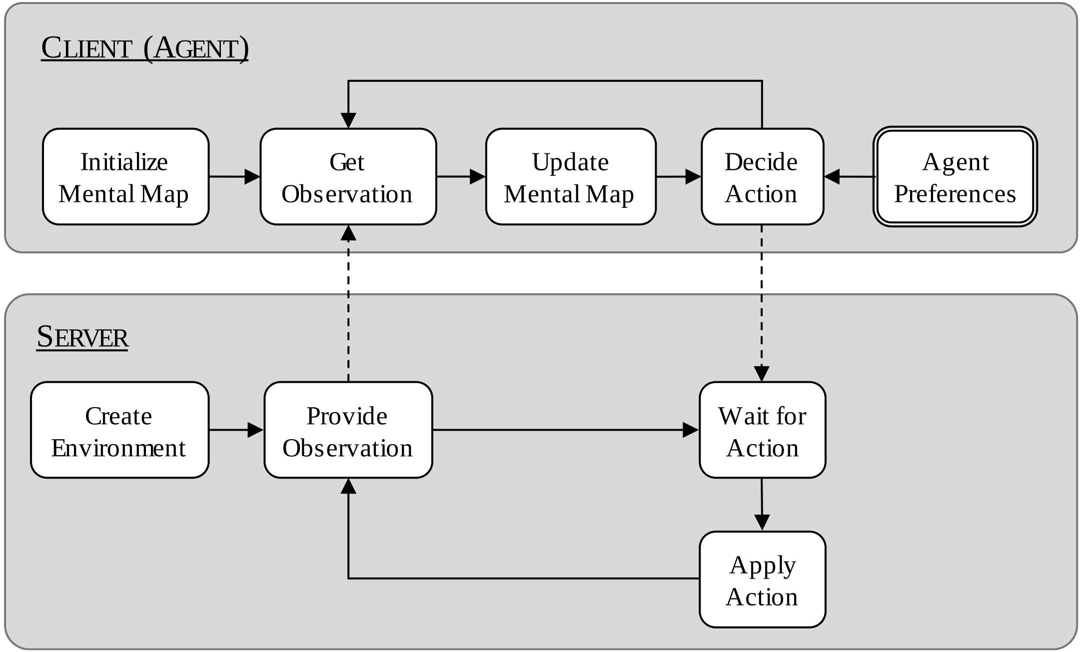
- Procedurally generated grid worlds
- Multiple attributes
  - Terrain (categorical)
  - Elevation (real-valued)
- Limited visibility
- Various problems represented as a resource gathering game



CMM Framework Architecture

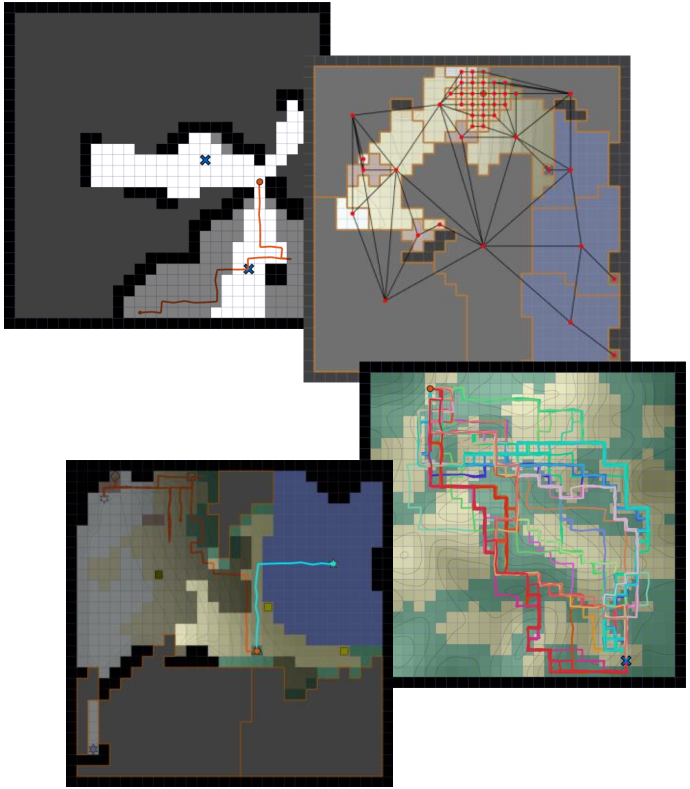
Two main components:

- Server
  - Defines the problem
  - Provides observations
  - Implements actions
- Client (agent)
  - Maintains a mental map of the environment
  - Decides where to go
  - Provides actions to the server



The Big Picture…

- Creating problem scenarios
- Managing the mental map
  - Getting new observations
  - Constructing the action graph
    - Computing single-step features
  - Clustering similar regions
  - Building the region graph
    - Aggregating fuzzy features
- Multiobjective Fuzzy Least-Cost Path Problems
  - Pre-scalarized decomposition approach
  - MOEA/D approach
  - A greedy agent strategy
- Future work

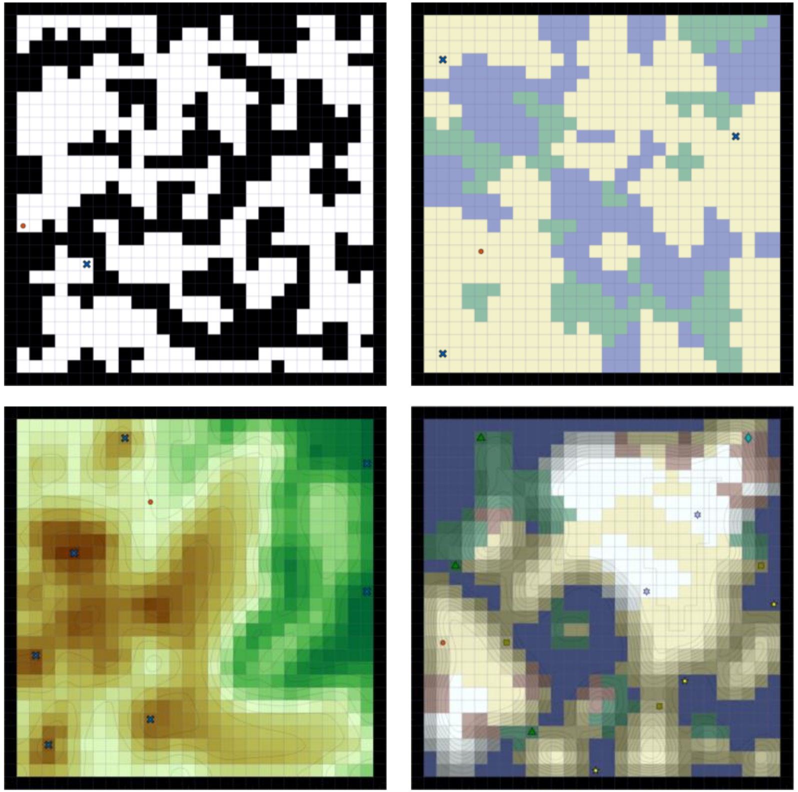


Creating Problem Scenarios

The CMM framework uses grid worlds to provide a finite action space.

We use several methods to create the environments:

- Binary cellular automata
- Fashion-based cellular automata
- Fractal terrain
  - Based on region partitioning
- Additional rules



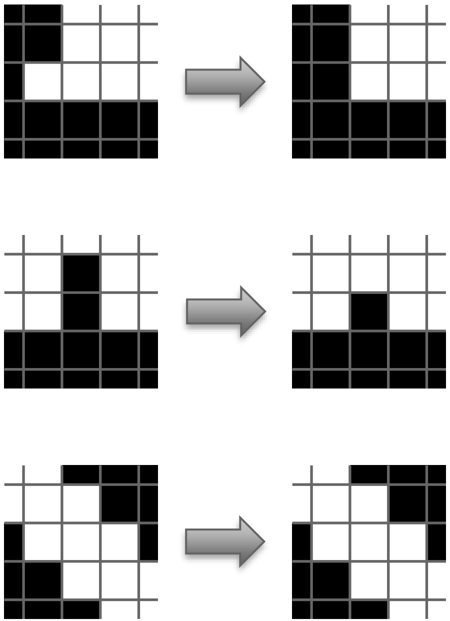
Cellular Automata

Cave-like environments are represented as a binary occupancy grid.

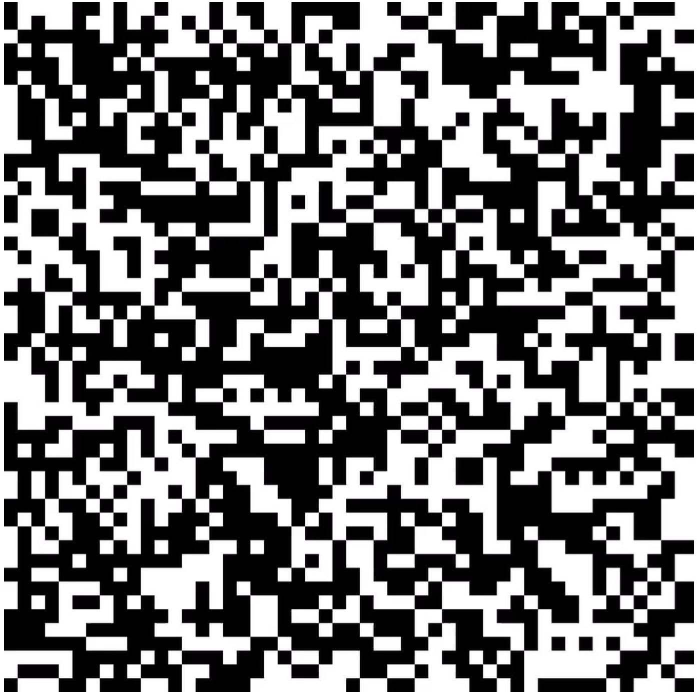
- Created using cellular automation (CA) rules
- Similar to Conway's Game of Life

Step 1) Initialize a random occupancy grid with probability p_0
Step 2) For k generations:
 Step 2.1) Count the number of open and closed neighbors of each grid cell
 Step 2.2) If an open cell has $<r_d$ open neighbors, it becomes closed
 Step 2.3) If a closed cell has $>r_b$ open neighbors, it becomes open
 Step 2.4) Clean up boarders
Step 3) Until open regions are connected:
 Step 3.1) Find the smallest open region
 Step 3.2) *Either*, dilate this region or expand it by a random grid cell
 Step 3.3) Clean up boarders and diagonal artifacts

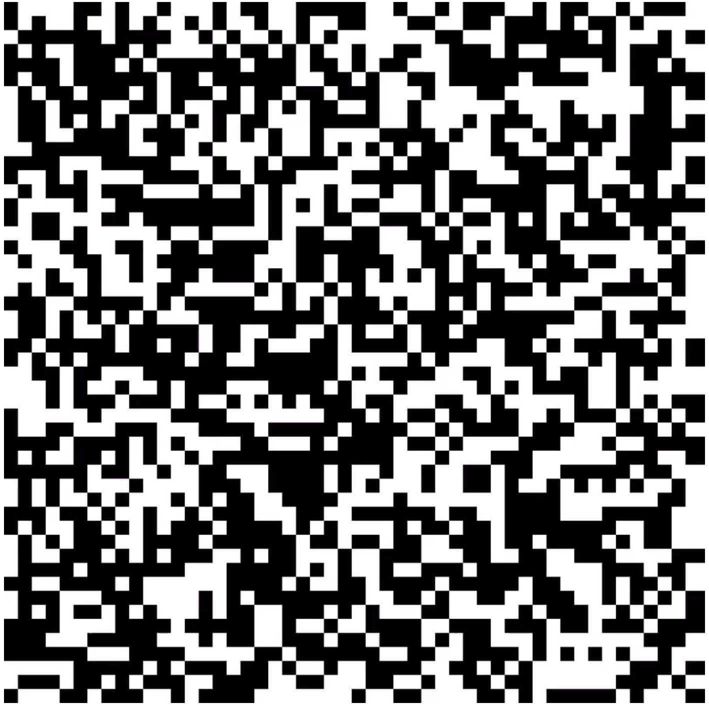
Example with r_d=3 and r_b=5



Cellular Automata Examples

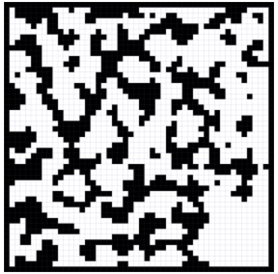


Example using dilation

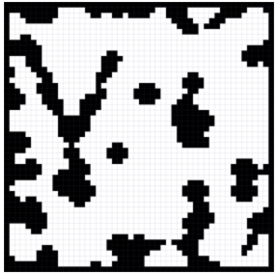


Example using random expansion

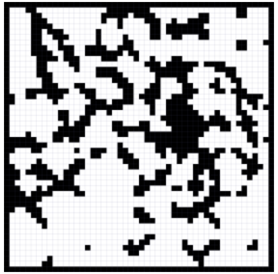
Cavern Map Examples



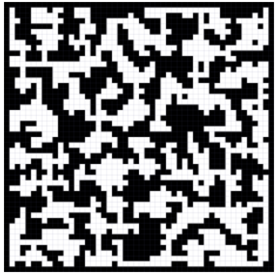
$p_0 = 0.5$, $r_b = 4$, $r_d = 3$, $k = 1$, dilate



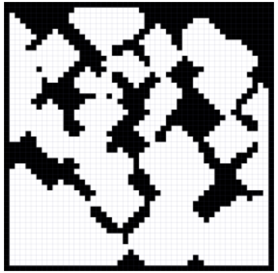
$p_0 = 0.5$, $r_b = 4$, $r_d = 3$, $k = 10$, dilate



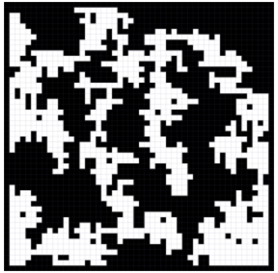
$p_0 = 0.5$, $r_b = 6$, $r_d = 4$, $k = 1$, dilate



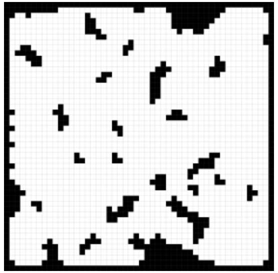
$p_0 = 0.5$, $r_b = 6$, $r_d = 4$, $k = 1$, random



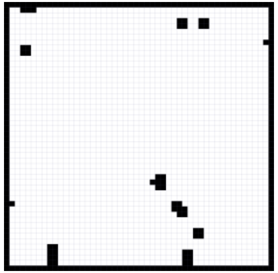
$p_0 = 0.2$, $r_b = 4$, $r_d = 2$, $k = 10$, dilate



$p_0 = 0.2$, $r_b = 4$, $r_d = 2$, $k = 10$, random



$p_0 = 0.8$, $r_b = 6$, $r_d = 4$, $k = 30$, dilate



$p_0 = 0.8$, $r_b = 5$, $r_d = 3$, $k = 30$, dilate

Binary Terrain Environments

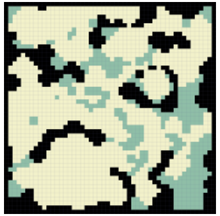
The CA algorithm can be used to create binary terrain environments.

Consider two types of terrain:

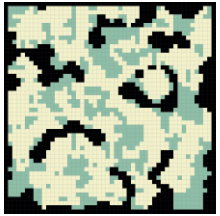
- Meadow
- Forest

Additional options:

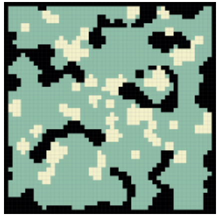
- Use cave walls
- Make meadow region unconnected



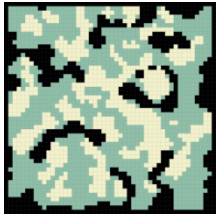
$p_0 = 0.5$, $r_b = 5$, $r_d = 3$, connected, dilate



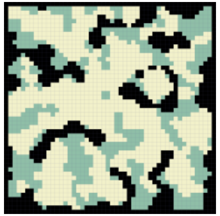
$p_0 = 0.5$, $r_b = 5$, $r_d = 3$, connected, random



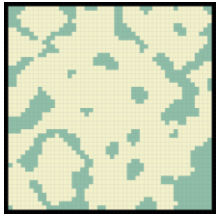
$p_0 = 0.4$, $r_b = 5$, $r_d = 3$, not connected



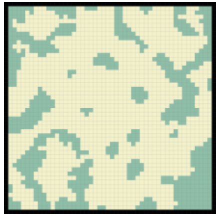
$p_0 = 0.5$, $r_b = 5$, $r_d = 3$, not connected



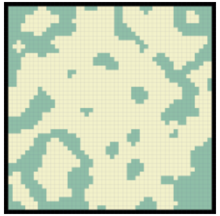
$p_0 = 0.6$, $r_b = 5$, $r_d = 3$, not connected



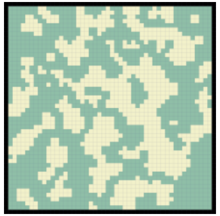
$p_0 = 0.5$, $r_b = 4$, $r_d = 3$, connected, dilate



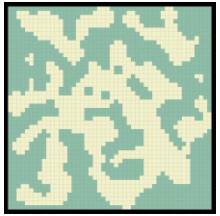
$p_0 = 0.5$, $r_b = 4$, $r_d = 3$, connected, random



$p_0 = 0.5$, $r_b = 4$, $r_d = 3$, not connected



$p_0 = 0.5$, $r_b = 5$, $r_d = 3$, not connected



$p_0 = 0.6$, $r_b = 5$, $r_d = 4$, not connected

Trinary Terrain Environments

Multiple terrain types can be defined using fashion-based cellular automata.

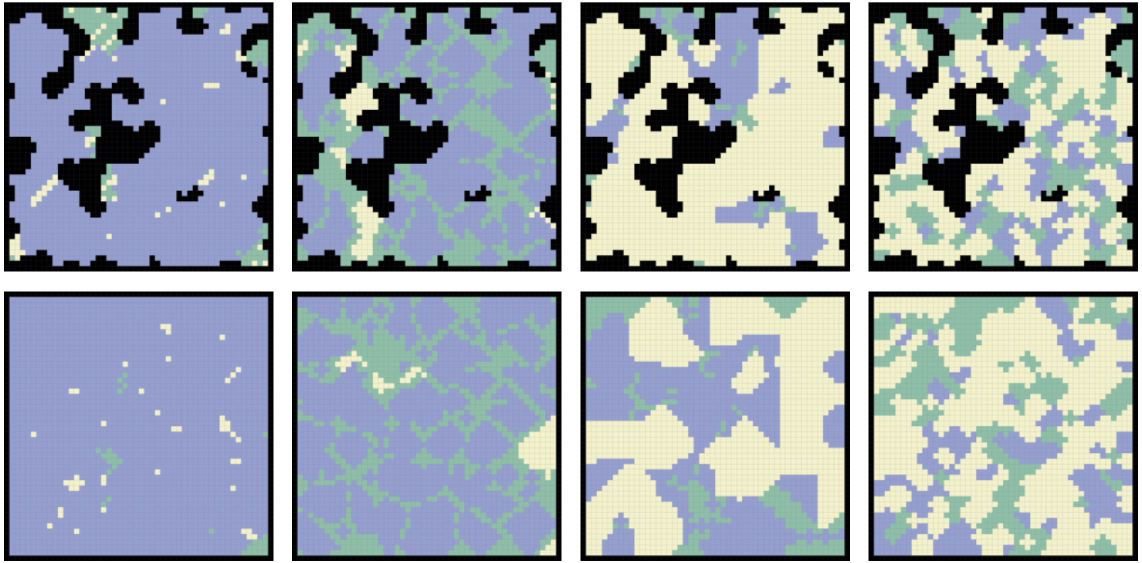
- Meadow, forest, & water

Let P_0 be the starting probability of each terrain type.

Define a rule R to score the compatibility of adjacent terrain types.

Score all cells and assign each cell the terrain label of the highest scoring neighbor.

Repeat for k iterations.



$P_0 = [0.5, 0.3, 0.2]$
$$R = \begin{bmatrix} 0.5 & 0.6 & 0.4 \\ 0.9 & 0.4 & 0 \\ 0 & 0.9 & 0.5 \end{bmatrix}$$

$P_0 = [0.3, 0.4, 0.3]$
$$R = \begin{bmatrix} 0.6 & 0.3 & 0.2 \\ 0.7 & 0.1 & 0.9 \\ 0.8 & 0.1 & 0.8 \end{bmatrix}$$

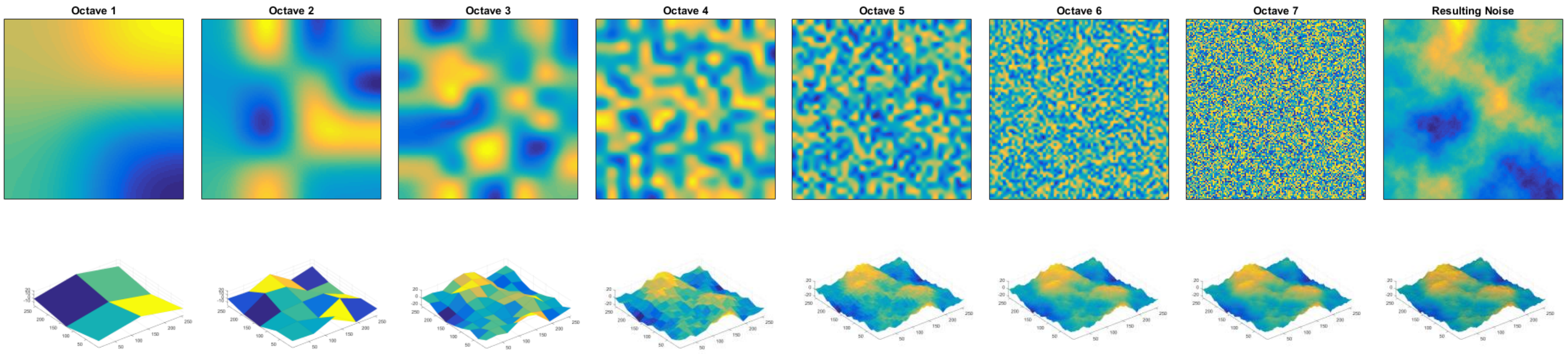
$P_0 = [0.1, 0.8, 0.1]$
$$R = \begin{bmatrix} 0.9 & 0.2 & 0.1 \\ 0.5 & 0.2 & 0.8 \\ 0.7 & 0.2 & 0.8 \end{bmatrix}$$

$P_0 = [0.5, 0.3, 0.2]$
$$R = \begin{bmatrix} 1 & 0.2 & 0.8 \\ 0.4 & 1 & 0.8 \\ 0.9 & 0.4 & 1 \end{bmatrix}$$

Fractal Terrain

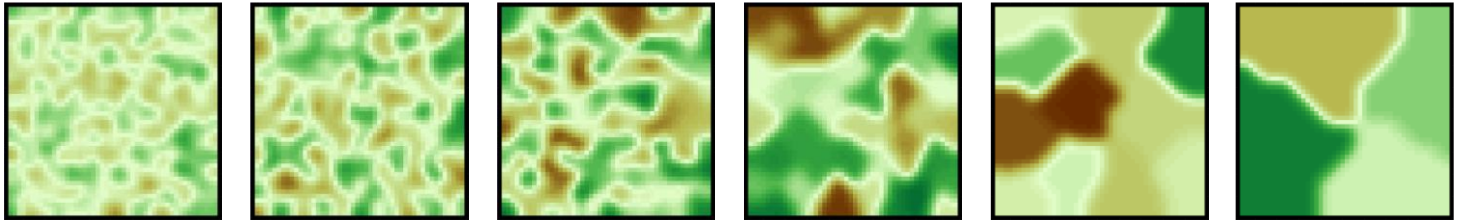
We use fractal noise to represent the heightmap of the elevation feature.

- Random noise at multiple scales is added together
- Larger scales get more weight, providing broad terrain features
- Smaller scales get less weight, adding fine details and texture

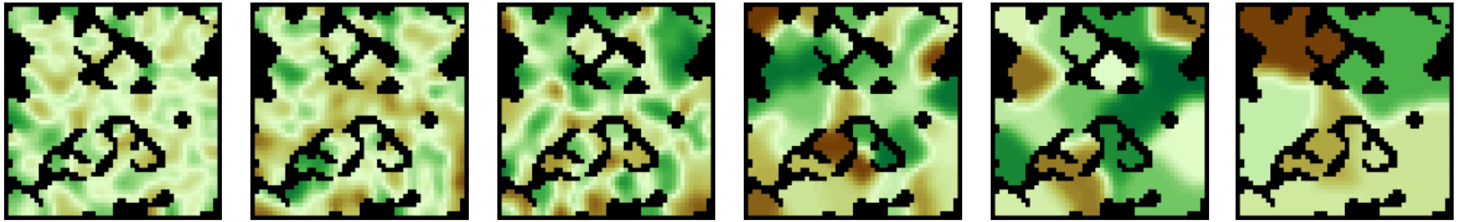


Region-Based Noise

The noise functions at each scale are created by partitioning the environment into distinct regions and assigning a random elevation value to each region.



$s = 1$ $s = 2$ $s = 4$ $s = 8$ $s = 16$ $s = 32$



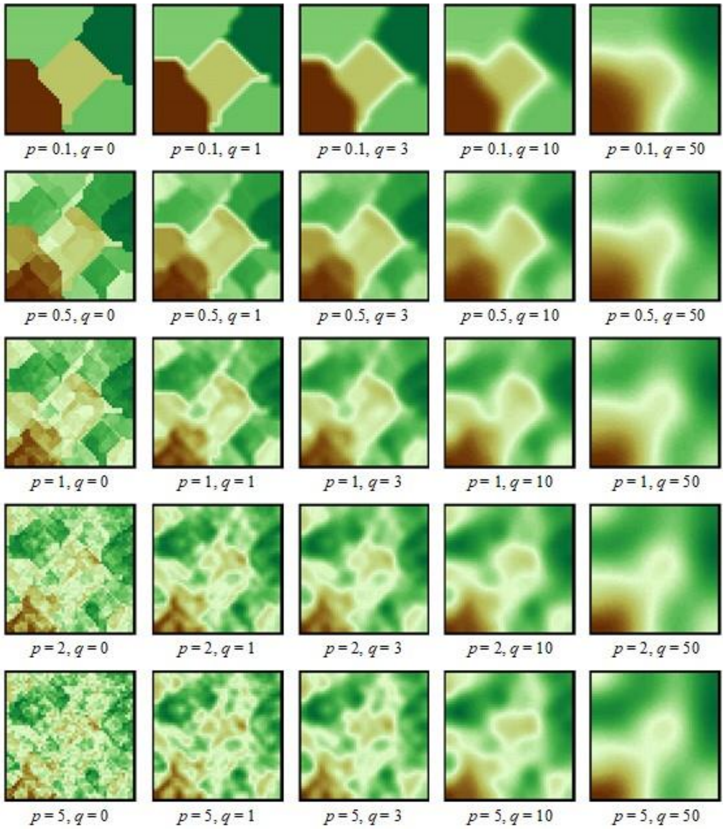
$s = 1$ $s = 2$ $s = 4$ $s = 8$ $s = 16$ $s = 32$

Smoothing and Noise

Two parameters control the blending of multiple noise scales:

- p controls the weight
  - Each noise scale s is weighed by $s^{1/p}$
  - Higher values result in a rougher terrain

- q controls the smoothing
  - A mean filter is applied to each noise image for q iterations
  - Larger values give smoother edges



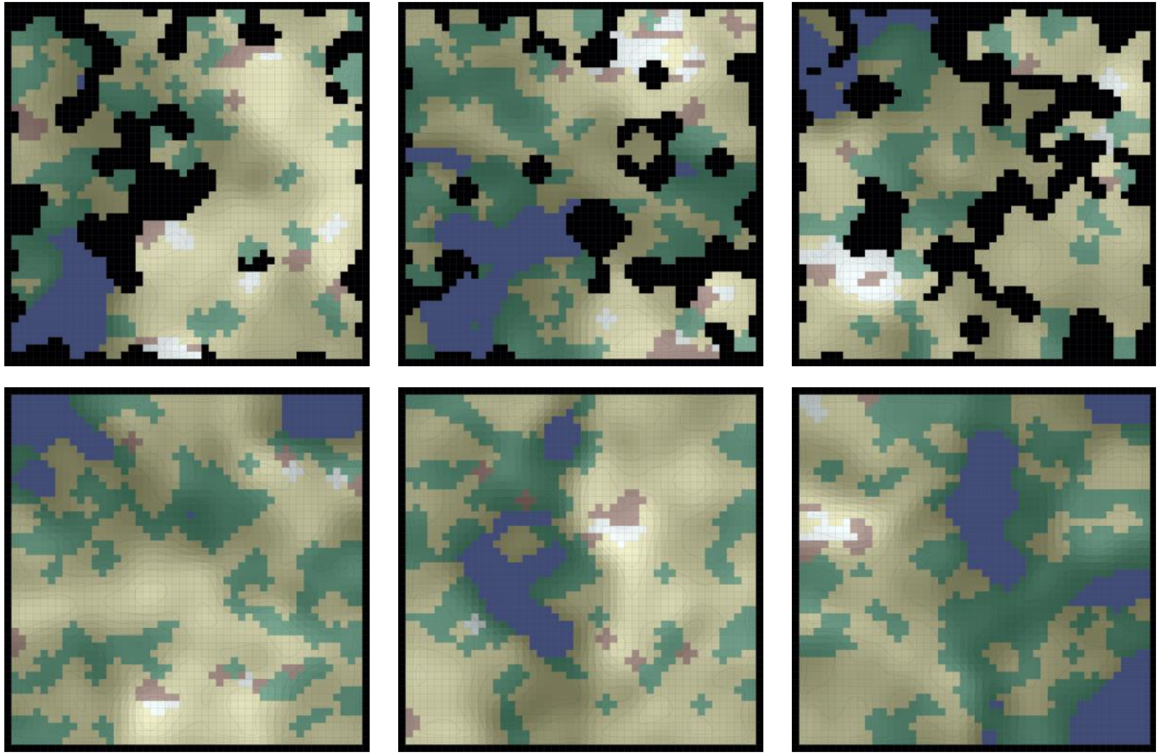
Full World Environments

We create full world environments by combining fashion-based cellular automata with elevation.

Five terrain types:

- Meadow
- Forest
- Water
- Rock
- Snow

Water is placed at the lowest elevations.



Resource Placement

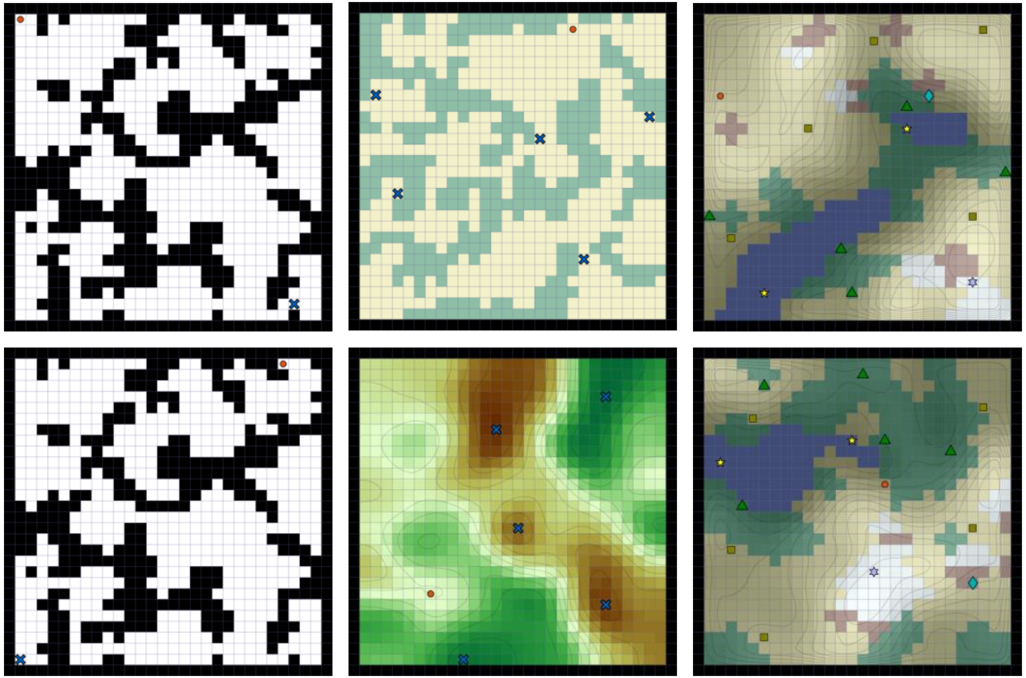
The agent is randomly placed somewhere in the environment.

Different problem types can be created based on how the resources are placed.

Shortest Path Problem
- One goal
  - Placed randomly
  - Maximum distance from the agent

Traveling Salesman Problem
- Multiple goals
  - Placed in open areas
  - Elevation extrema

Traveling Purchaser Problem
- Multiple resource types
  - Different for each type of terrain
  - Different distributions

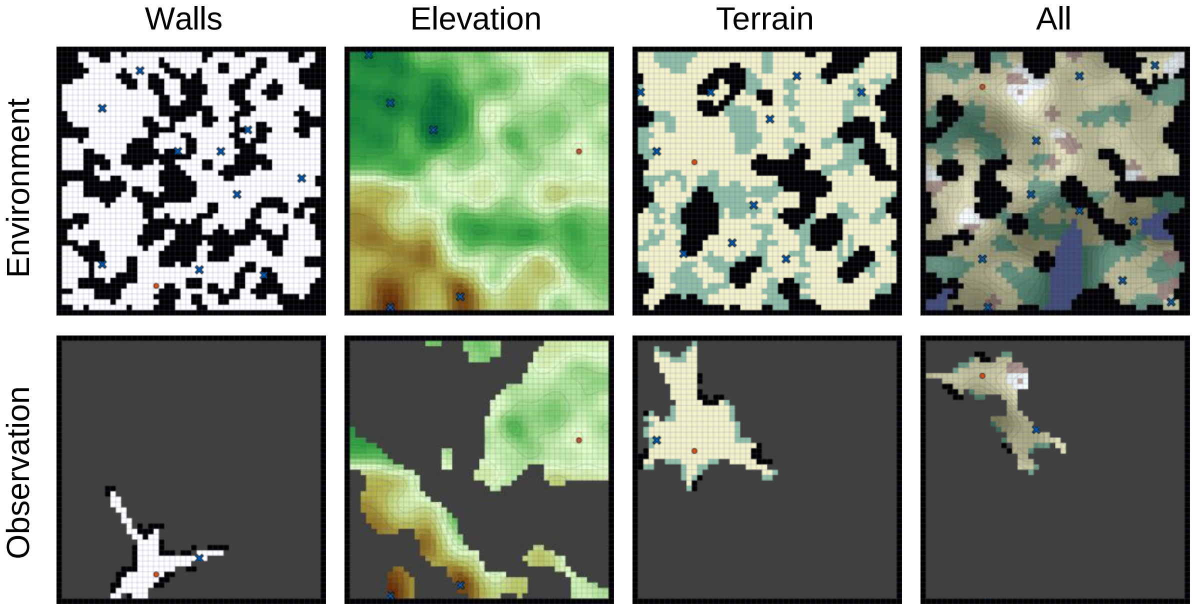


Observing the Environment

In some scenarios, we restrict what the agent can observe.

The agent computes a viewshed region from its current location.

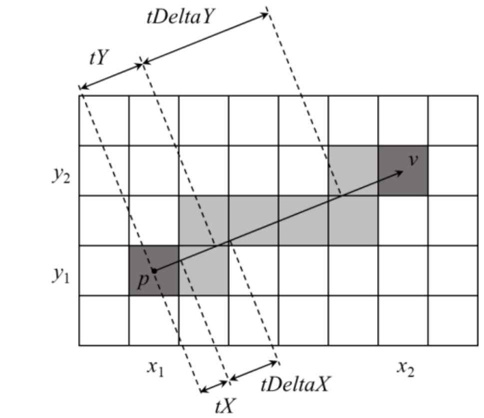
Visibility can be obstructed by
- Walls
- Elevation (hills)
- Terrain (forests)

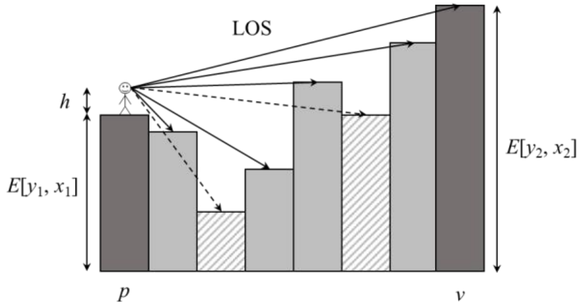


Checking Visibility

To check if a grid cell v is visible from cell p,
- Draw a vector $\overrightarrow{pv}$ from p to v
- Find all cells that intersect $\overrightarrow{pv}$
- Compute the elevation angle from p to each of these cells
- If the elevation angle from p to v is greater than all other cells, then the cell is visible

To compute the viewshed mask,
- Compute the visibility of each cell, working outward from the agent's current location
- Continue processing in each direction until encountering a wall or obstacle





Initializing the Mental Map

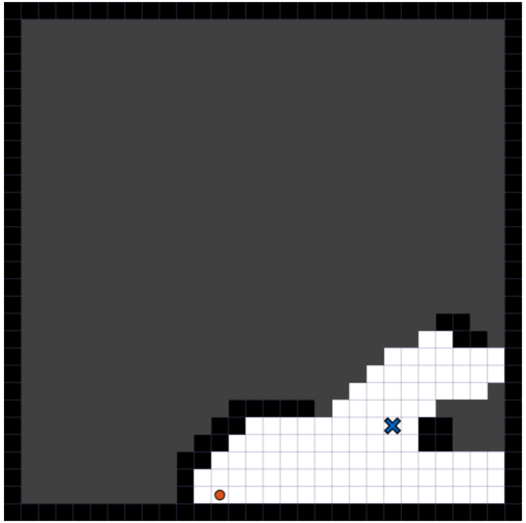
The agent starts with an empty mental map.

- Complete uncertainty

The mental map is initialized by the first observation.

Environment

Mental Map



Updating the Mental Map

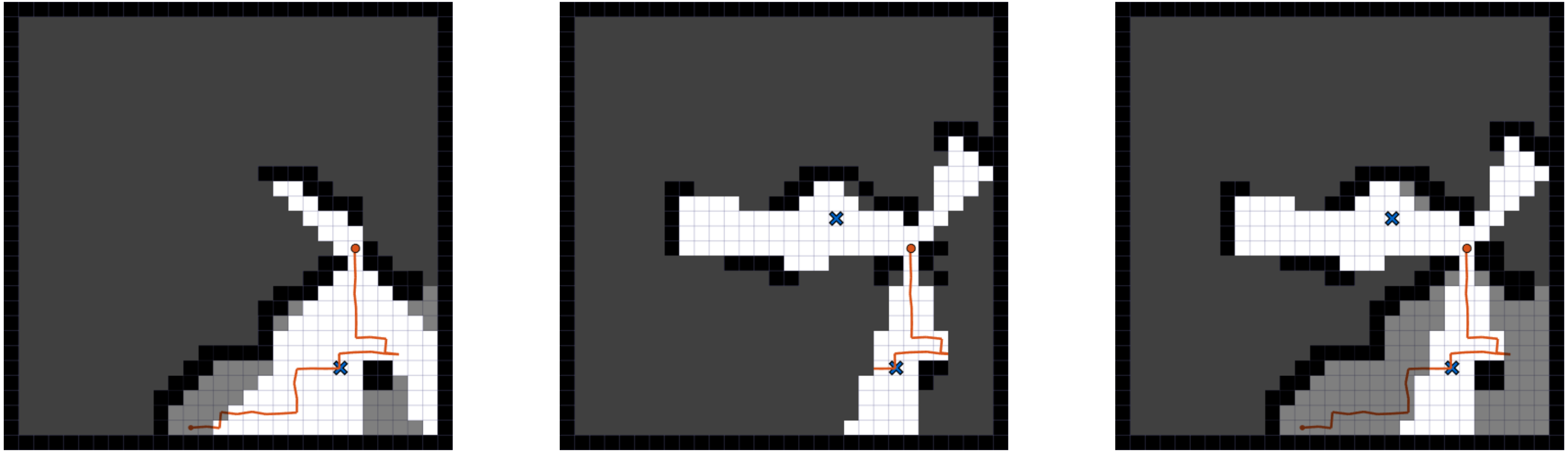
As the agent explores, new observations are integrated into the mental map.

All observed attributes are updated, including terrain and elevation.

Mental map before update

New observation

Mental map after update



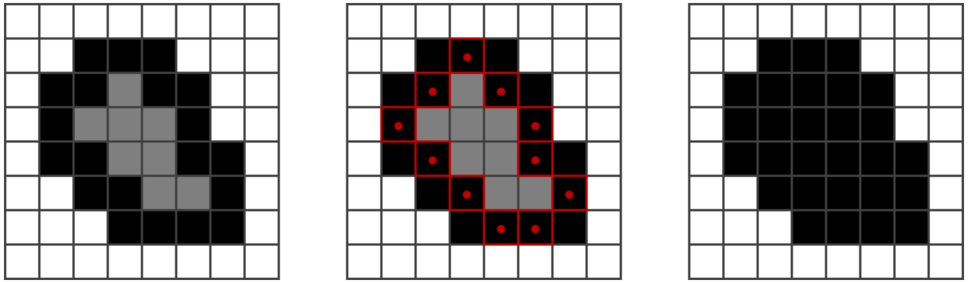
Mental Map Heuristics

The agent can use additional heuristics to improve the mental map.

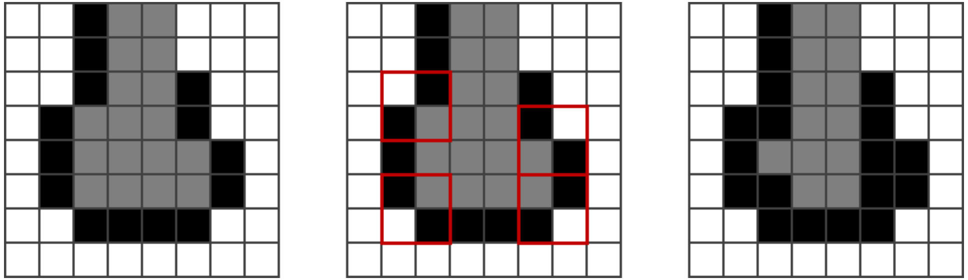
Fill-in unreachable areas
- Unobserved regions that are surrounded by walls can't be reached
- Replace with walls

Fix diagonal artifacts
- Diagonal passageways are forbidden when creating environments
- This can resolve certain ambiguities

Filling-In Unreachable Areas



Fixing Diagonal Artifacts

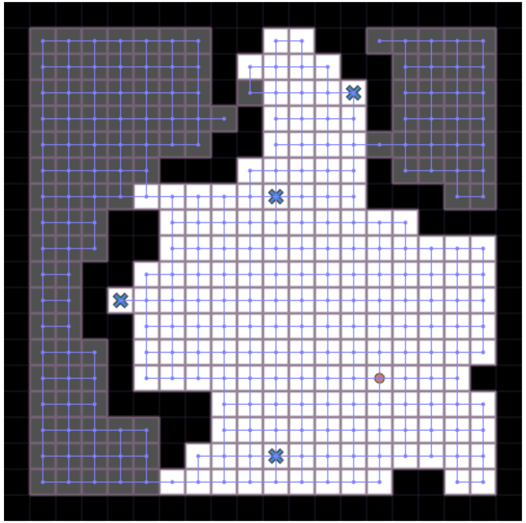


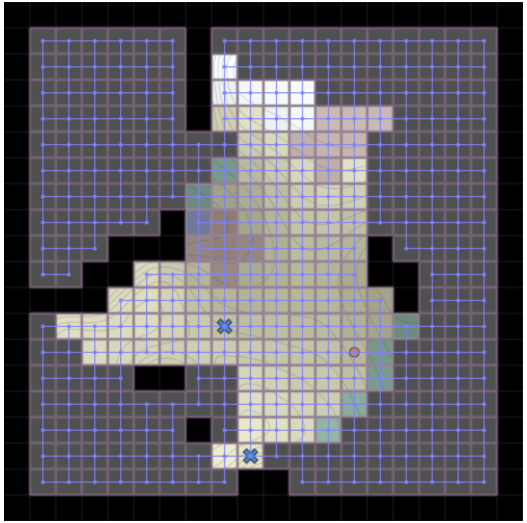
The Action Graph

The agent can move up, down, left, and right.

The action graph covers all the grid cells that are traversable.

Each edge in the action graph connects two adjacent cells and is attributed with multiple features.





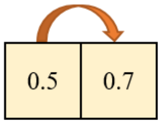
Observable Features

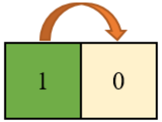
Consider two adjacent cells, c_1 and c_2 with terrain and height properties:

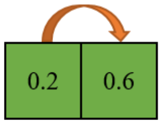
- Terrain types t_1 and t_2
- Heights h_1 and h_2

Features:

- f_d Distance
- $f_{t(i)}$ Terrain type
- $f_{t\{i,j\}}$ Terrain transition (symmetric)
- $f_{t\langle i,j\rangle}$ Terrain transition (directional)
- f_h Elevation difference (absolute)
- $f_{h\uparrow}$, $f_{h\downarrow}$ Elevation difference (directional)

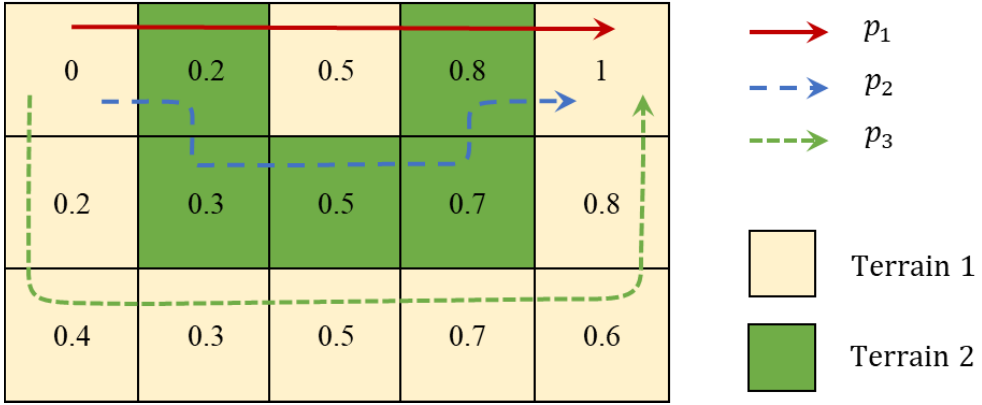






$t_1 = 1$	$t_1 = 1$	$t_1 = 2$	$t_1 = 2$
$t_2 = 1$	$t_2 = 2$	$t_2 = 1$	$t_2 = 2$
$h_1 = 0.5$	$h_1 = 0.4$	$h_1 = 1$	$h_1 = 0.2$
$h_2 = 0.7$	$h_2 = 0.1$	$h_2 = 0$	$h_2 = 0.6$
$f_d = 1$	$f_d = 1$	$f_d = 1$	$f_d = 1$
$f_{t(1)} = 1$	$f_{t(1)} = 0.5$	$f_{t(1)} = 0.5$	$f_{t(1)} = 0$
$f_{t(2)} = 0$	$f_{t(2)} = 0.5$	$f_{t(2)} = 0.5$	$f_{t(2)} = 1$
$f_{t\{1,1\}} = 1$	$f_{t\{1,1\}} = 0$	$f_{t\{1,1\}} = 0$	$f_{t\{1,1\}} = 0$
$f_{t\{1,2\}} = 0$	$f_{t\{1,2\}} = 1$	$f_{t\{1,2\}} = 1$	$f_{t\{1,2\}} = 0$
$f_{t\{2,2\}} = 0$	$f_{t\{2,2\}} = 0$	$f_{t\{2,2\}} = 0$	$f_{t\{2,2\}} = 1$
$f_{t\langle 1,1\rangle} = 1$	$f_{t\langle 1,1\rangle} = 0$	$f_{t\langle 1,1\rangle} = 0$	$f_{t\langle 1,1\rangle} = 0$
$f_{t\langle 1,2\rangle} = 0$	$f_{t\langle 1,2\rangle} = 1$	$f_{t\langle 1,2\rangle} = 0$	$f_{t\langle 1,2\rangle} = 0$
$f_{t\langle 2,1\rangle} = 0$	$f_{t\langle 2,1\rangle} = 0$	$f_{t\langle 2,1\rangle} = 1$	$f_{t\langle 2,1\rangle} = 0$
$f_{t\langle 2,2\rangle} = 0$	$f_{t\langle 2,2\rangle} = 0$	$f_{t\langle 2,2\rangle} = 0$	$f_{t\langle 2,2\rangle} = 1$
$f_h = 0.2$	$f_h = 0.3$	$f_h = 1$	$f_h = 0.4$
$f_{h\uparrow} = 0.2$	$f_{h\uparrow} = 0$	$f_{h\uparrow} = 0$	$f_{h\uparrow} = 0.4$
$f_{h\downarrow} = 0$	$f_{h\downarrow} = 0.3$	$f_{h\downarrow} = 1$	$f_{h\downarrow} = 0$

Aggregating Path Features



	Distance	Terrain Type		Terrain Transition (symmetric)			Terrain Transition (directional)				Total Slope (absolute)	Total Slope (directional)		Max Slope (absolute)	Max Slope (directional)	
	f_d^{sum}	$f_{t(1)}^{\text{sum}}$	$f_{t(2)}^{\text{sum}}$	$f_{t\{1,1\}}^{\text{sum}}$	$f_{t\{1,2\}}^{\text{sum}}$	$f_{t\{2,2\}}^{\text{sum}}$	$f_{t\langle 1,1\rangle}^{\text{sum}}$	$f_{t\langle 1,2\rangle}^{\text{sum}}$	$f_{t\langle 2,1\rangle}^{\text{sum}}$	$f_{t\langle 2,2\rangle}^{\text{sum}}$	f_h^{sum}	$f_{h\uparrow}^{\text{sum}}$	$f_{h\downarrow}^{\text{sum}}$	f_h^{max}	$f_{h\uparrow}^{\text{max}}$	$f_{h\downarrow}^{\text{max}}$
p_1	4	2	2	0	4	0	0	2	2	0	1	1	0	0.3	0.3	0
p_2	6	1	5	0	2	4	0	1	1	4	1	1	0	0.2	0.2	0
p_3	8	8	0	8	0	0	8	0	0	0	1.4	1.2	0.2	0.2	0.2	0.1

Fuzzy Terrain Features

When one or both cells are unobserved, we represent the feature as a fuzzy number.

A triangular fuzzy number is defined by three values:

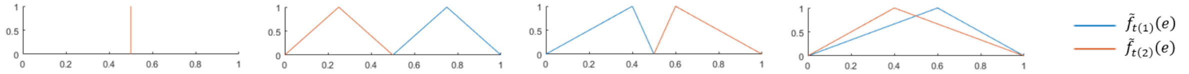
- Minimum
- Mean (peak)
- Maximum

Assume a prior likelihood $p(i)$ for each terrain type.

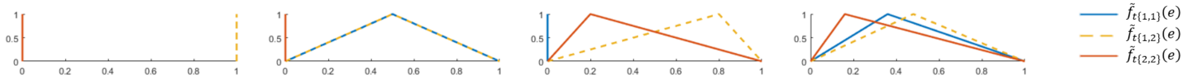
The fuzzy feature represents the range of possible crisp feature values [min, max] and the expected value (peak).

$e.t_1 = 1$	$e.t_1 = 1$	$e.t_1 = \text{NIL}$	$e.t_1 = \text{NIL}$
$e.t_2 = 2$	$e.t_2 = \text{NIL}$	$e.t_2 = 2$	$e.t_2 = \text{NIL}$
$e.o_1 = 1$	$e.o_1 = 1$	$e.o_1 = 0$	$e.o_1 = 0$
$e.o_2 = 1$	$e.o_2 = 0$	$e.o_2 = 1$	$e.o_2 = 0$
$p(1) = 0.5$	$p(1) = 0.5$	$p(1) = 0.8$	$p(1) = 0.6$
$p(2) = 0.5$	$p(2) = 0.5$	$p(2) = 0.2$	$p(2) = 0.4$

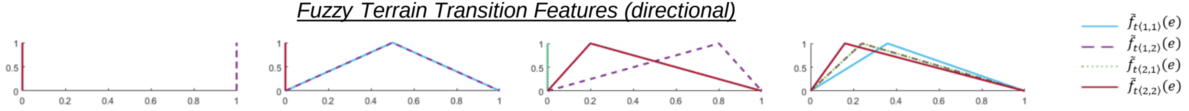
Fuzzy Terrain Type Features



Fuzzy Terrain Transition Features (symmetric)

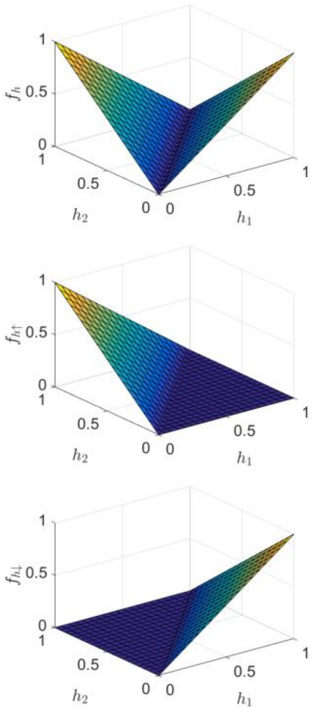


Fuzzy Terrain Transition Features (directional)

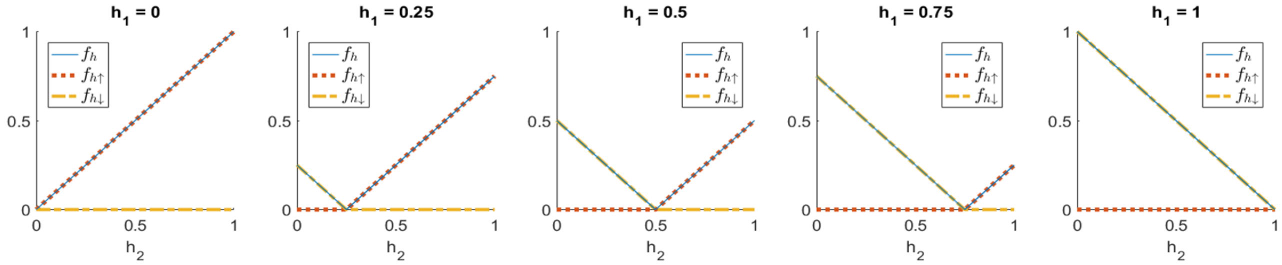


Uncertain Elevation Features

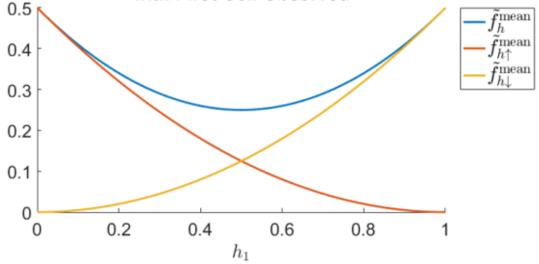
Elevation Difference Features with Both Cells Observed



Elevation Difference Features with First Cell Observed



Expected Elevation Difference with First Cell Observed



Assumes all elevations are equally likely

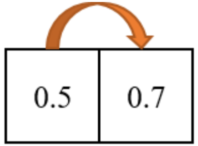
Expected Elevation Difference with Both Cells Unobserved

$$\tilde{f}_h^{\text{mean}}(e) = \int_0^1 \int_0^1 |x - y| dx\, dy = \frac{1}{3}$$

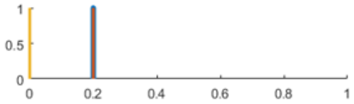
$$\tilde{f}_{h\uparrow}^{\text{mean}}(e) = \int_0^1 \int_0^y (y - x) dx\, dy = \frac{1}{6}$$

$$\tilde{f}_{h\downarrow}^{\text{mean}}(e) = \int_0^1 \int_y^1 (x - y) dx\, dy = \frac{1}{6}$$

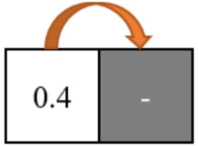
Fuzzy Elevation Features



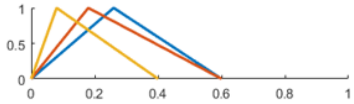
$e.h_1 = 0.5$
$e.h_2 = 0.7$
$e.o_1 = 1$
$e.o_2 = 1$



Features are crisp when both cells are observed

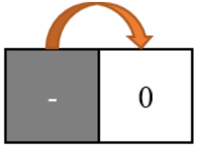


$e.h_1 = 0.4$
$e.h_2 = \mathrm{NIL}$
$e.o_1 = 1$
$e.o_2 = 0$

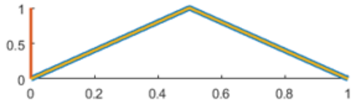


$\tilde{f}_{h\uparrow}$ range is [0, 0.6]
$\tilde{f}_{h\downarrow}$ range is [0, 0.4]

The mean of $\tilde{f}_h$ is higher because it includes both $\uparrow$ and $\downarrow$ slopes

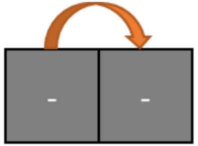


$e.h_1 = \mathrm{NIL}$
$e.h_2 = 0$
$e.o_1 = 0$
$e.o_2 = 1$

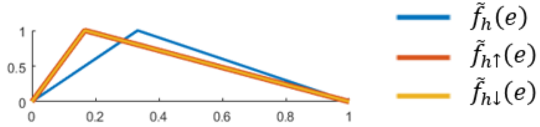


$\tilde{f}_{h\downarrow}$ is a crisp 0

All values of $\tilde{f}_{h\uparrow}$ and $\tilde{f}_h$ are equally likely



$e.h_1 = \mathrm{NIL}$
$e.h_2 = \mathrm{NIL}$
$e.o_1 = 0$
$e.o_2 = 0$



Range of all features is [0, 1]

Mean of $\tilde{f}_h$ is 1/3
Mean of $\tilde{f}_{h\uparrow}$ and $\tilde{f}_{h\downarrow}$ is 1/6

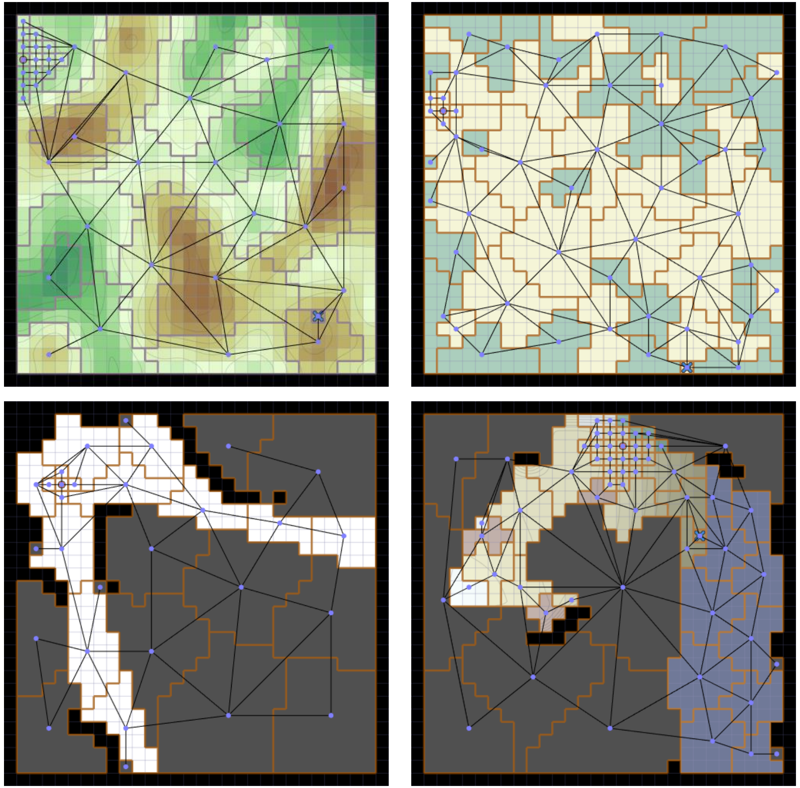
The Region Graph

The action graph represents individual movement actions.

We can summarize the action graph by clustering similar regions and constructing a new graph over the regions.

The region graph
- Reduces the size of the search space
- Helps with high-level planning
- Less precise than the action graph

Typically, we keep a small part of the action graph around the agent to facilitate the next immediate action.



Region Partitioning

Each observed terrain type and the unobserved regions are partitioned separately.

The SLIC algorithm clusters nearby cells into regions.

<u>SLIC Algorithm:</u>

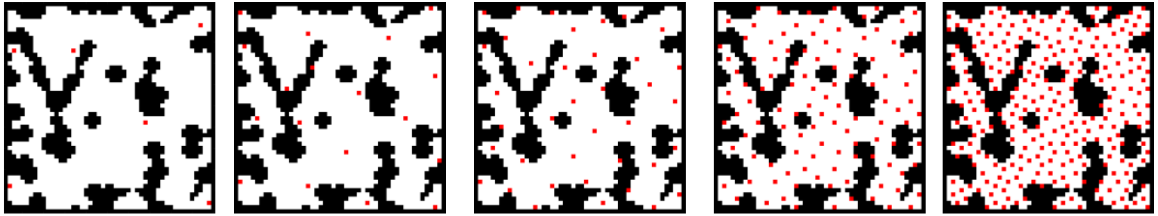
Step 1) Sample cluster centers with separation distance r

Step 2) Adjust cluster centers to the neighboring cell with the minimum gradient

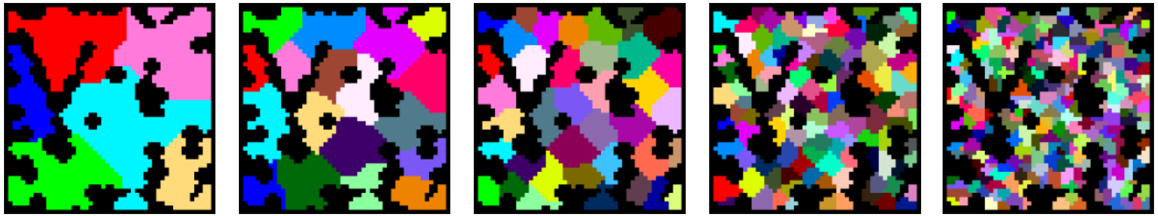
Step 3) For n iterations:

- **Step 3.1)** Compute the elevation-weighted distances between cells and cluster centers
- **Step 3.2)** Assign cells to the closest clusters
- **Step 3.3)** Update cluster centers using the region centroids

Step 4) Fix any orphan cells



$r = 32$ $r = 16$ $r = 8$ $r = 4$ $r = 2$



$r = 32$ $r = 16$ $r = 8$ $r = 4$ $r = 2$

Fuzzy Region Features

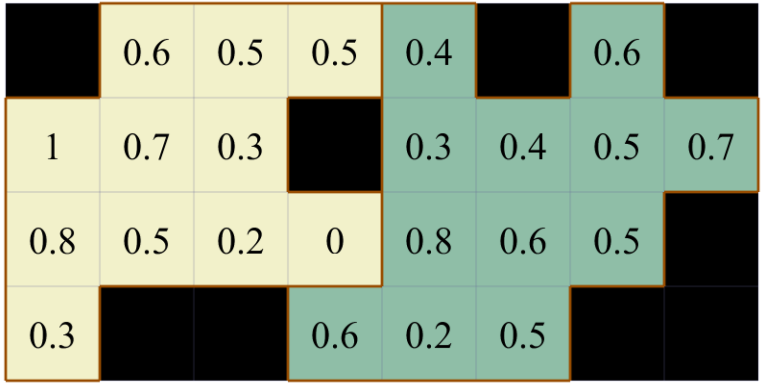
Region features are computed between each pair of adjacent regions, R_1 and R_2.

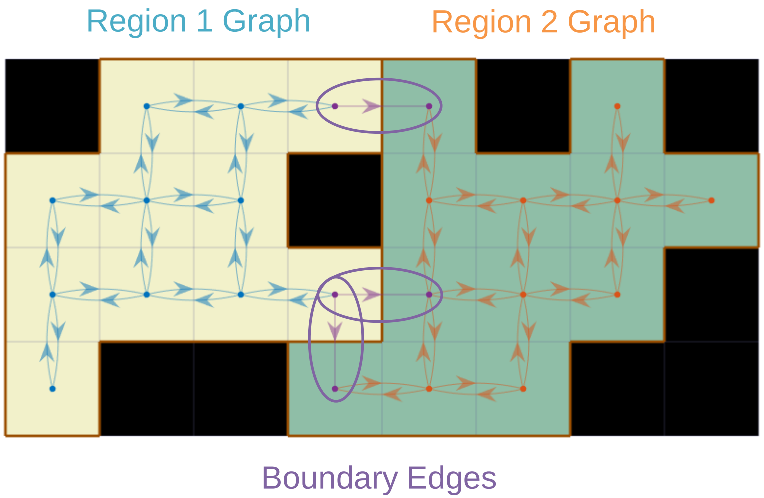
- Fuzzy values represent the min, mean, and max cost of moving between regions.

Two computation methods:

- Boundary edge distance aggregation
- Opposing centroid distance approximation

Region 1 (meadow) to Region 2 (forest)
Showing Elevation Values





Boundary Edge Distances

Region cost matrices U^{d1} and U^{d2} give the distances from each cell to each boundary edge k.

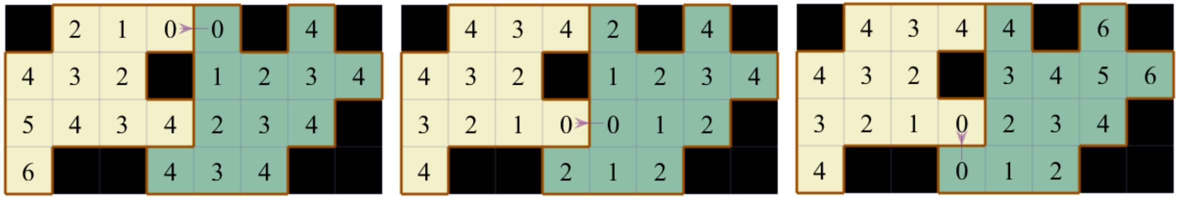
The overall cost matrix C^d gives the minimum cost required to go from any cell $i \in R_1$ to any cell $j \in R_2$.

$$C_{ij}^{d} = \min_{k}\left\{U_{ik}^{d1} + 1 + U_{kj}^{d2}\right\}$$

The fuzzy region distance feature is defined as the min, mean, and max of the values in C^d.

Boundary Edge Distances

k = 1 k = 2 k = 3



$$U^{d1} = \begin{bmatrix} 4 & 4 & 4 \\ 5 & 3 & 3 \\ 6 & 4 & 4 \\ 2 & 4 & 4 \\ 3 & 3 & 3 \\ 4 & 2 & 2 \\ 1 & 3 & 3 \\ 2 & 2 & 2 \\ 3 & 1 & 1 \\ 0 & 4 & 4 \\ 4 & 0 & 0 \end{bmatrix} \quad U^{d2} = \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 4 \\ 1 & 1 & 3 \\ 2 & 0 & 2 \\ 3 & 1 & 1 \\ 2 & 2 & 4 \\ 3 & 1 & 3 \\ 4 & 2 & 2 \\ 4 & 4 & 6 \\ 3 & 3 & 5 \\ 4 & 2 & 4 \\ 4 & 4 & 6 \end{bmatrix}$$

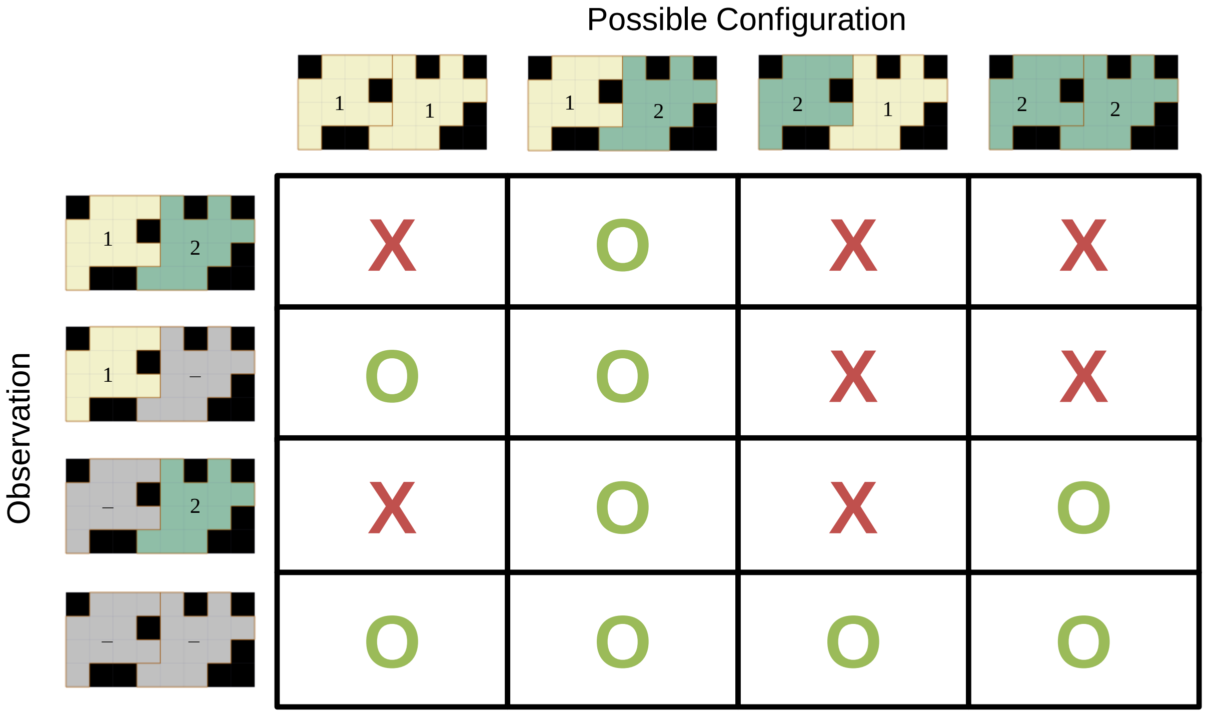
$$C^{d} = \begin{bmatrix} 5 & 5 & 6 & 5 & 6 & 7 & 6 & 7 & 9 & 8 & 7 & 9 \\ 4 & 6 & 5 & 4 & 5 & 6 & 5 & 6 & 8 & 7 & 6 & 8 \\ 5 & 7 & 6 & 5 & 6 & 7 & 6 & 7 & 9 & 8 & 7 & 9 \\ 5 & 3 & 4 & 5 & 6 & 5 & 6 & 7 & 7 & 6 & 7 & 7 \\ 4 & 4 & 5 & 4 & 5 & 6 & 5 & 6 & 8 & 7 & 6 & 8 \\ 3 & 5 & 4 & 3 & 4 & 5 & 4 & 5 & 7 & 6 & 5 & 7 \\ 4 & 2 & 3 & 4 & 5 & 4 & 5 & 6 & 6 & 5 & 6 & 6 \\ 3 & 3 & 4 & 3 & 4 & 5 & 4 & 5 & 7 & 6 & 5 & 7 \\ 2 & 4 & 3 & 2 & 3 & 4 & 3 & 4 & 6 & 5 & 4 & 6 \\ 5 & 1 & 2 & 3 & 4 & 3 & 4 & 5 & 5 & 4 & 5 & 5 \\ 1 & 3 & 2 & 1 & 2 & 3 & 2 & 3 & 5 & 4 & 3 & 5 \end{bmatrix}$$

Region Terrain Features

Terrain type and transition features can be computed from U^{d1}, U^{d2}, and C^{d}.

If one or both regions are unobserved, then use terrain priors to evaluate the likelihood of each possible configuration.

The region distance feature does not depend on observability, only the spatial layout.



Region Elevation Features

To compute the region elevation features, distance is replaced by cost.

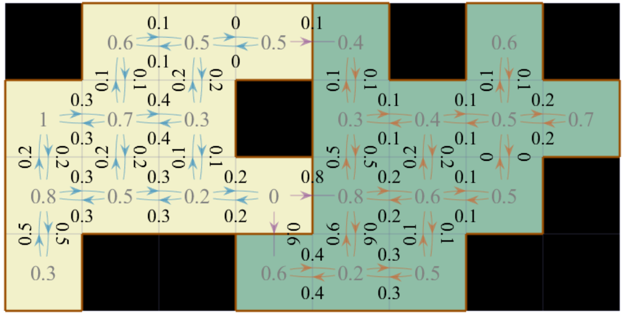
We use a grid-optimized version of the Bellman-Ford algorithm to compute the total and maximum elevation feature costs to each boundary edge.

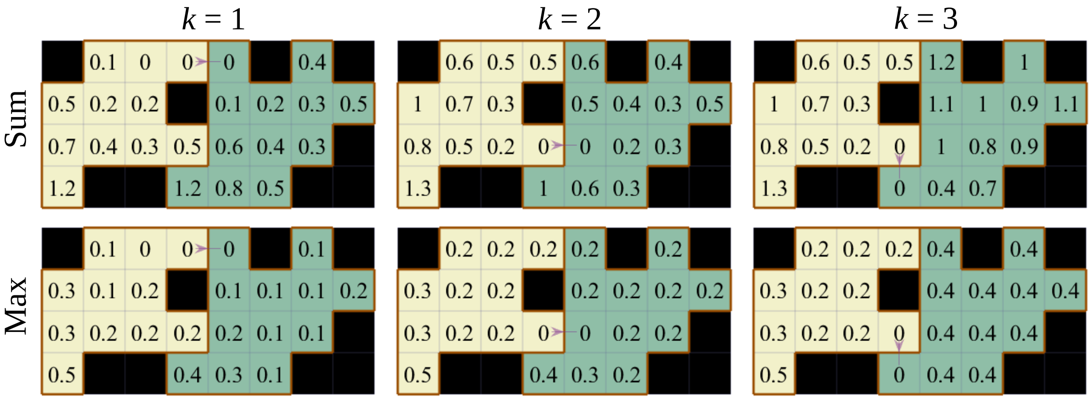
The overall cost matrix is computed as either

$$C_{ij}^{\text{sum}} = \min_k \{ U_{ik}^1 + u_k^{\text{bnd}} + U_{kj}^2 \}$$
$$C_{ij}^{\text{max}} = \min_k \{ \max (U_{ik}^1, u_k^{\text{bnd}}, U_{kj}^2) \}$$

where u_k^{bnd} is the cost of boundary edge k.

Absolute Elevation Difference Costs





Unobserved Elevation Features

If a region is unobserved, the elevation costs are unknown.

We assume all height values are independent* and are uniformly distributed between 0 and 1.

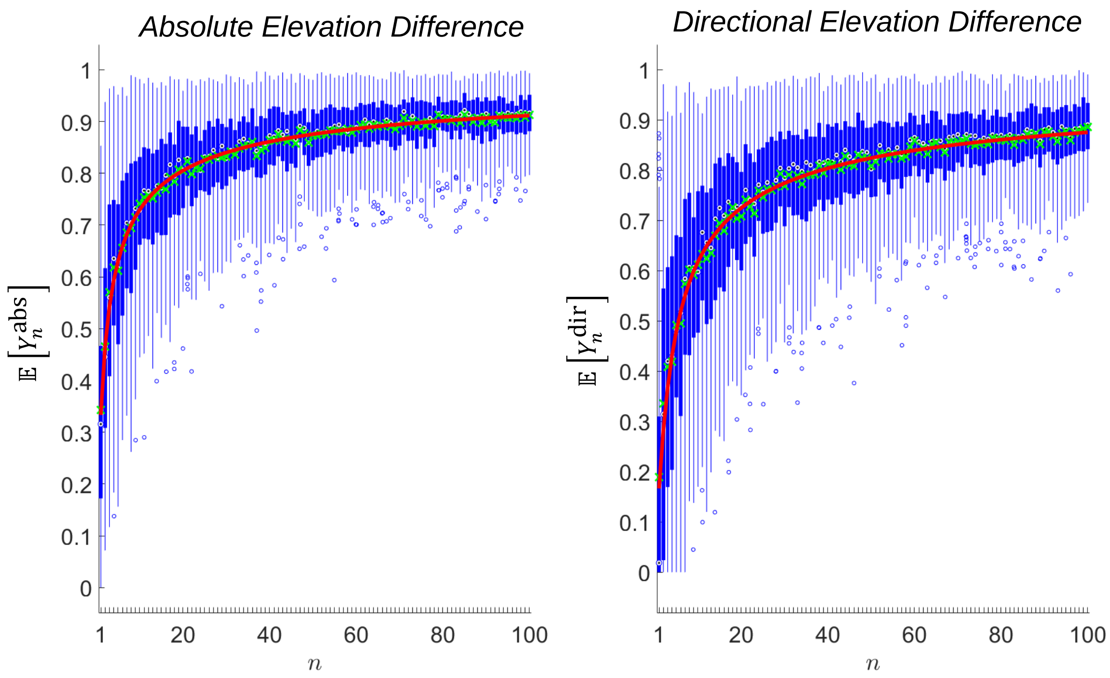
Minimum cost is the minimum of the single-step boundary edge costs.

Maximum cost is derived from the max distance cost.

Expected cost for total elevation change is U^d times unobserved single-step average $\left(\frac{1}{3} \text{ or } \frac{1}{6}\right)$.

Expected cost for max elevation change is the expected max of U^d randomly sampled feature costs.

Expected Maximum Value of n Elevation Feature Costs



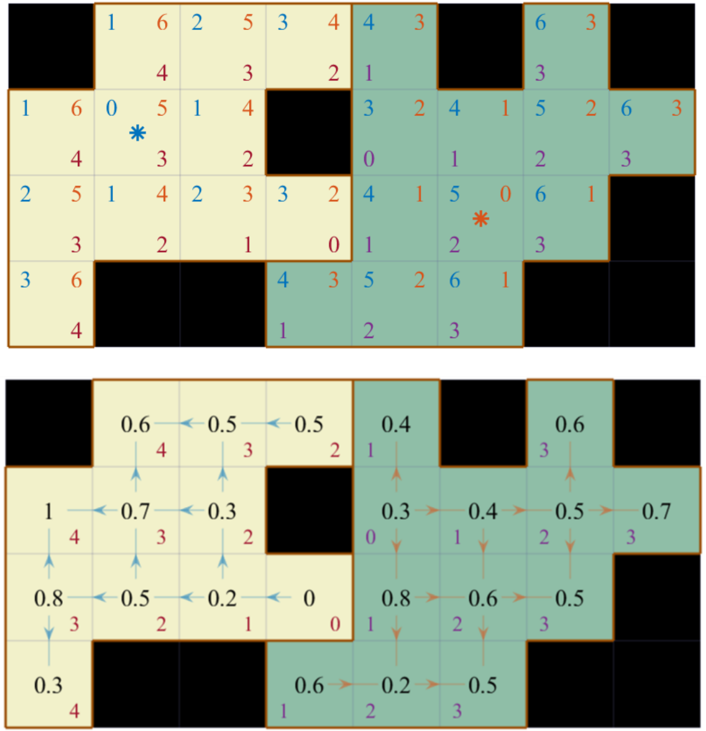
Centroid Distance Approximation

The Bellman-Ford algorithm can be computationally expensive if applied multiple times to each boundary edge.

The distance to the region boundary can be approximated from the distance to the centroid of the other region.

Single-step features can be grouped into sets based on distance from the region boundary.

Approximate features based on these sets can be much faster to compute.



$$S_1^1 = \{f(0, 0.2)\}$$

$$S_2^1 = \begin{Bmatrix} f(0.2, 0.3) \\ f(0.2, 0.5) \end{Bmatrix}$$

$$S_3^1 = \begin{Bmatrix} f(0.5, 0.8) \\ f(0.5\ 0.7) \\ f(0.3, 0.7) \\ f(0.3, 0.5) \\ f(0.5, 0.5) \end{Bmatrix}$$

$$S_4^1 = \begin{Bmatrix} f(0.8, 1) \\ f(0.8\ 0.3) \\ f(0.7, 1) \\ f(0.7, 0.6) \\ f(0.5, 0.6) \end{Bmatrix}$$

$$S_1^2 = \begin{Bmatrix} f(0.3, 0.4) \\ f(0.3, 0.8) \\ f(0.3, 0.4) \end{Bmatrix}$$

$$S_2^2 = \begin{Bmatrix} f(0.6, 0.2) \\ f(0.8, 0.2) \\ f(0.8, 0.6) \\ f(0.4, 0.6) \\ f(0.4, 0.5) \end{Bmatrix}$$

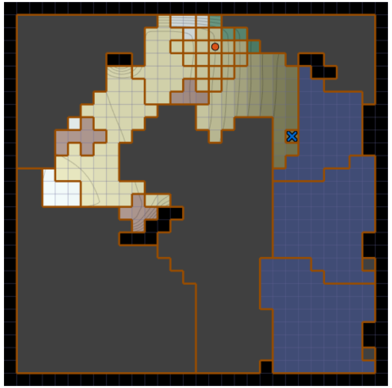
$$S_3^2 = \begin{Bmatrix} f(0.2, 0.5) \\ f(0.6, 0.5) \\ f(0.6, 0.5) \\ f(0.5, 0.6) \\ f(0.5, 0.5) \\ f(0.5, 0.7) \end{Bmatrix}$$

Updating the Region Graph

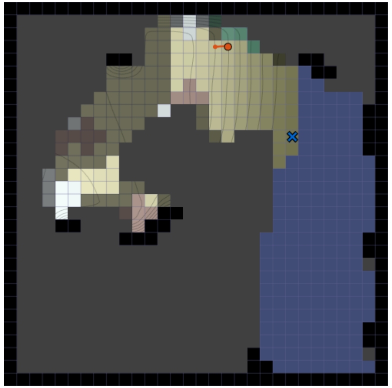
The region graph is updated when the agent moves.

The local region and region boundaries may change.

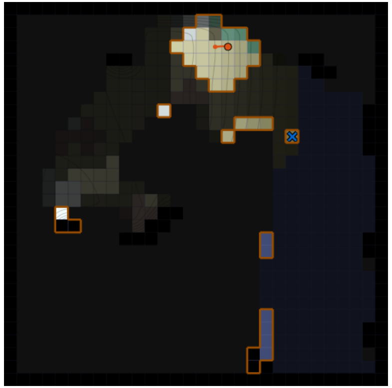
Features only need to be recomputed for regions that have changed.



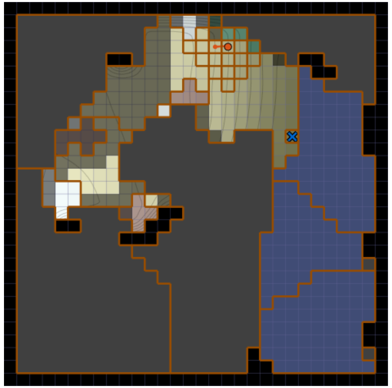
Original regions



Agent moves to the right and gets a new observation



Identify cells that need to be reclustered

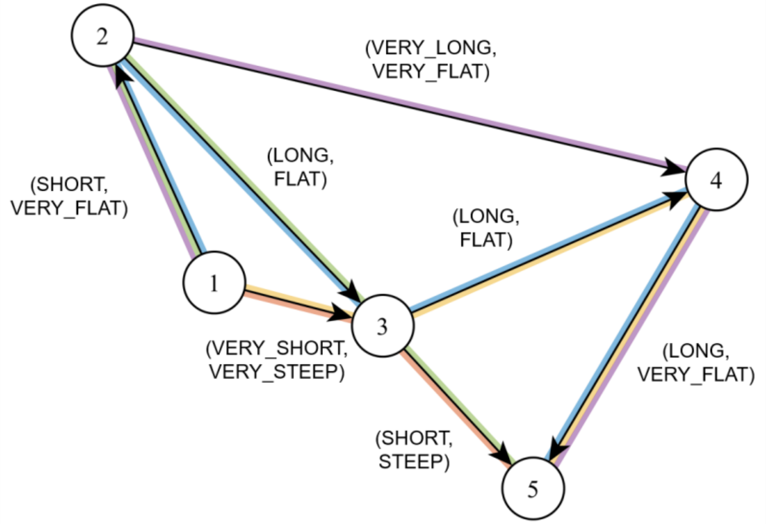


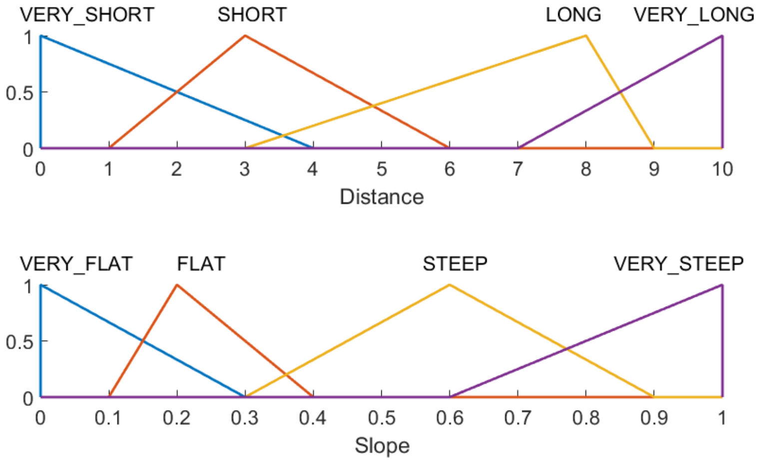
Compute new region boundaries. Update features for regions that have changed.

The MO-FLCPP

The multiobjective fuzzy least-cost path problem (MO-FLCPP) finds an optimal path between two vertices of a fuzzy weighted graph.

Consider this example:





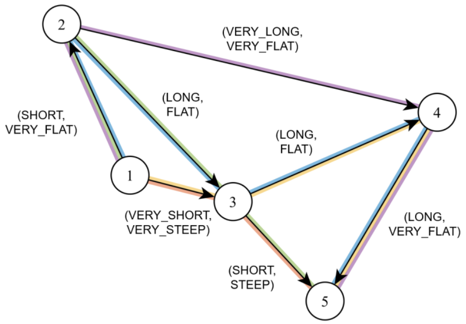
Aggregated Path Cost

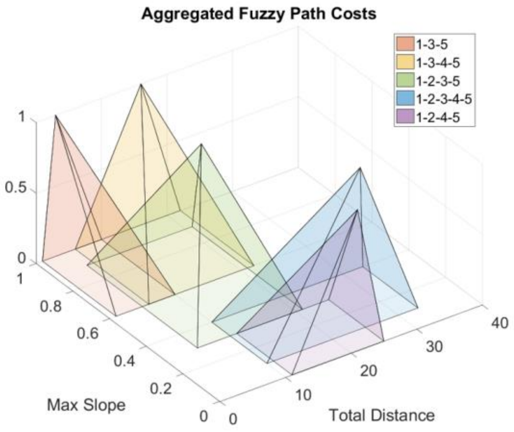
There are five paths through the graph from vertices 1 to 5.

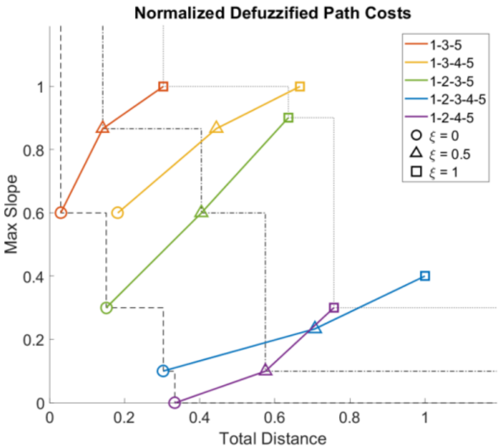
By plotting the aggregated path costs, we can determine which solutions are Pareto optimal.

A defuzzification parameter ξ indicates the degree of optimism/pessimism.

Path	Color	Total Distance	Max Slope
1-3-5	Red	Tri(1, 3, 10)	Tri(0.6, 1, 1)
1-3-4-5	Yellow	Tri(6, 16, 22)	Tri(0.6, 1, 1)
1-2-3-5	Green	Tri(5, 14, 21)	Tri(0.3, 0.6, 0.9)
1-2-3-4-5	Blue	Tri(10, 27, 33)	Tri(0.1, 0.2, 0.4)
1-2-4-5	Purple	Tri(11, 21, 25)	Tri(0, 0, 0.3)







Choosing a Path

A scalarization function $g(\boldsymbol{X}|\boldsymbol{\lambda})$ reduces the multidimensional solution vector $\boldsymbol{X}$ to a real value using the weight vector $\boldsymbol{\lambda}$.

We consider three scalarization methods:

- Weighted sum

$$g^{\mathrm{ws}}(\boldsymbol{X}|\boldsymbol{\lambda}) = \sum_{i=1}^{m} \lambda_i X_i$$

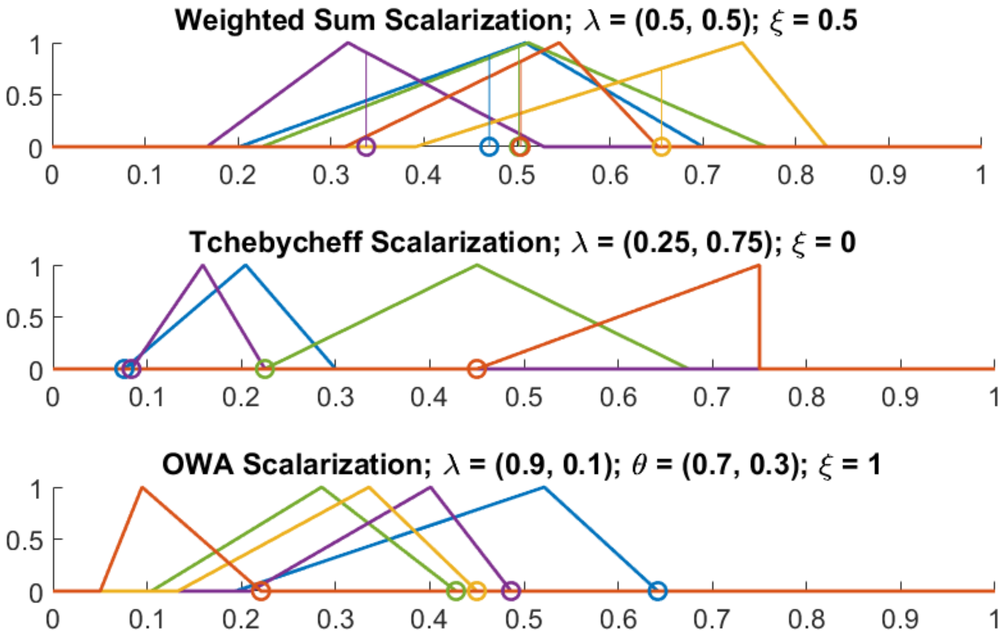
- Tchebycheff

$$g^{\mathrm{te}}(\boldsymbol{X}|\boldsymbol{\lambda}) = \max_{i=1,\ldots,m} \lambda_i X_i$$

- Ordered Weighted Average (OWA)

$$g^{\mathrm{OWA}}(\boldsymbol{X}|\boldsymbol{\lambda},\boldsymbol{\theta}) = \sum_{i=1}^{m} \theta_i B_{(i)}$$

where $B_{(i)}$ is the i^{th} largest $\lambda_i X_i$.
$\boldsymbol{\theta}$ is used to define different operators.



Finding a Path

Feature normalization:

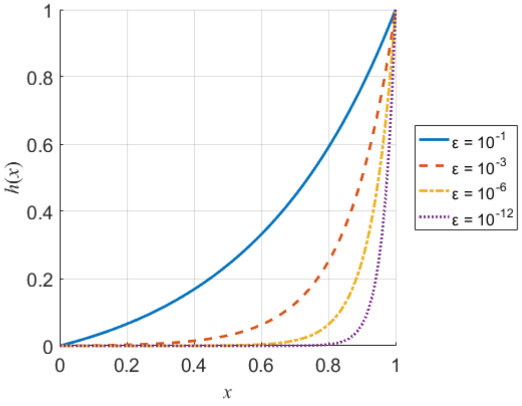
- Initially features are normalized by observed edge values
- As complete solutions are discovered, features are normalized by the range of the Pareto front

Exponential scaling:

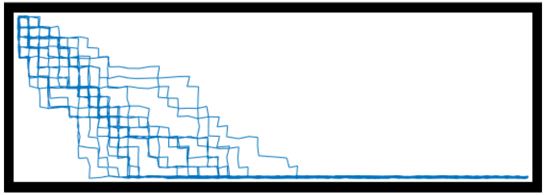
- To combine summation and maximization objectives, scale the maximization features logarithmically
- This approximates a minimax path as a shortest path

Selection bias:

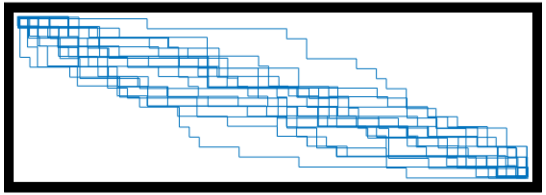
- Add a small amount of random noise to features to distinguish equivalent paths
- Find a shortest path using Dijkstra's algorithm



Uniform Transition Probability



Dijkstra's Algorithm with Noise



Pre-Scalarized Decomposition

A fast way to get a solution for the MO-FLCPP is to reduce the multidimensional fuzzy edge features to crisp scalar values and use standard Dijkstra's algorithm.

This method depends on
- The scalarization function g^{ws}, g^{te}, or g^{OWA}
- The objective weight vector $\boldsymbol{\lambda}$
  - and OWA weights $\boldsymbol{\theta}$ if using OWA
- The defuzzification parameter ξ
- How the features are normalized
  - Reference point $\boldsymbol{z}^{\text{me}}$, or $\boldsymbol{z}^{*}$

Shortest paths for the example problem:
- p^{D}: Pre-scalarized using max edge features, $\boldsymbol{z}^{\text{me}}$
- p^{M}: Scalarized after aggregating path costs using $\boldsymbol{z}^{\text{me}}$
- p^{*}: Scalarized after aggregating using Pareto optimal normalization, $\boldsymbol{z}^{*}$

		Weighted Sum			Tchebycheff		
ξ	$\boldsymbol{\lambda}$	p^{D}	p^{M}	p^{*}	p^{D}	p^{M}	p^{*}
0	(0, 1)	P	P	P	P	P	P
	(0.25, 0.75)	B	P	P	B	G	B
	(0.5, 0.5)	G	R	P	G	G	G
	(0.75, 0.25)	R	R	R	R	R	G
	(1, 0)	R	R	R	R	R	R
0.5	(0, 1)	P	P	P	P	P	P
	(0.25, 0.75)	P	P	P	P	G	P
	(0.5, 0.5)	P	R	P	P	R	P
	(0.75, 0.25)	P	R	R	P	R	R
	(1, 0)	R	R	R	R	R	R
1	(0, 1)	P	P	P	P	P	P
	(0.25, 0.75)	P	P	P	P	P	P
	(0.5, 0.5)	P	R	P	P	R	P
	(0.75, 0.25)	P	R	R	P	R	R
	(1, 0)	R	R	R	R	R	R

MOEA/D for the MO-FLCPP

Better solutions can be found by normalizing to the set of Pareto optimal solutions.

MOEA/D is a multiobjective evolutionary algorithm based on decomposing a problem into single-objective subproblems.

<u>MOEA/D Algorithm:</u>

Step 1) Initialization

- **Step 1.1)** Initialize a population of N pre-scalarized solutions using different objective weight vectors $\boldsymbol{\lambda}$
- **Step 1.2)** Initialize the external population EP containing the non-dominated solutions

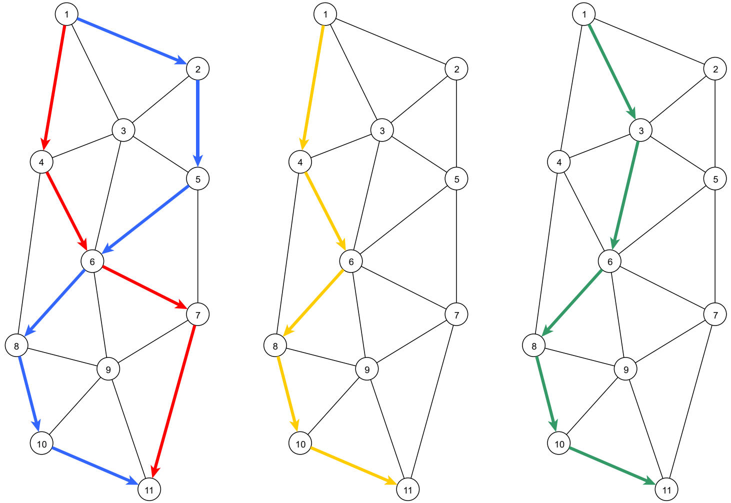
Step 2) Until stopping criteria is met:

For each weight vector i:

- **Step 2.1)** Create a new solution with a neighboring path using crossover and mutation
- **Step 2.1)** Replace outperformed solutions with the new path
- **Step 2.3)** Update EP

Step 3) Return EP as the set of Pareto optimal solutions. The agent can scalarize these and choose the solution that best satisfies the agent's original preferences.

Two neighboring paths | Crossover | Mutation



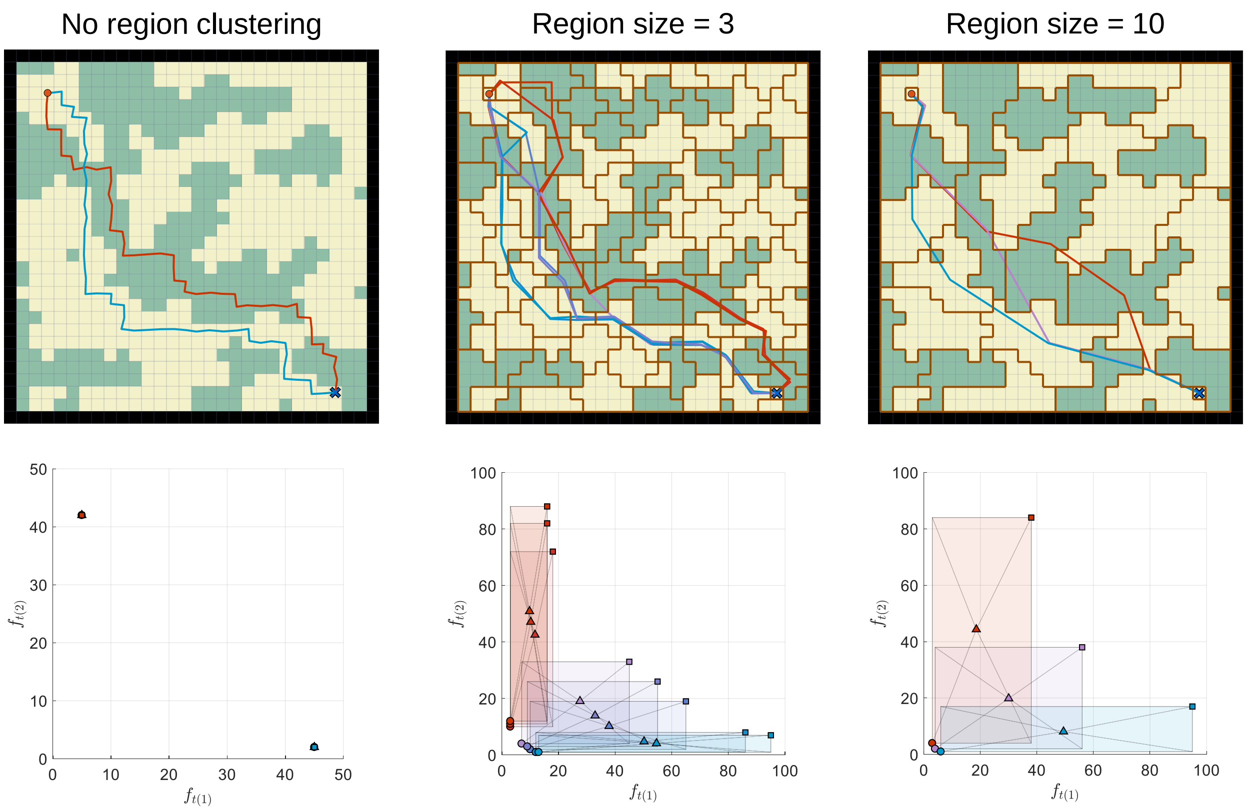
Binary Shortest Paths (WS)

We use the MOEA/D algorithm to find all Pareto optimal shortest paths in a binary terrain environment.

Features:

- $f_{t(1)}$: Distance in meadow
- $f_{t(2)}$: Distance in forest

Using weighted sum scalarization.



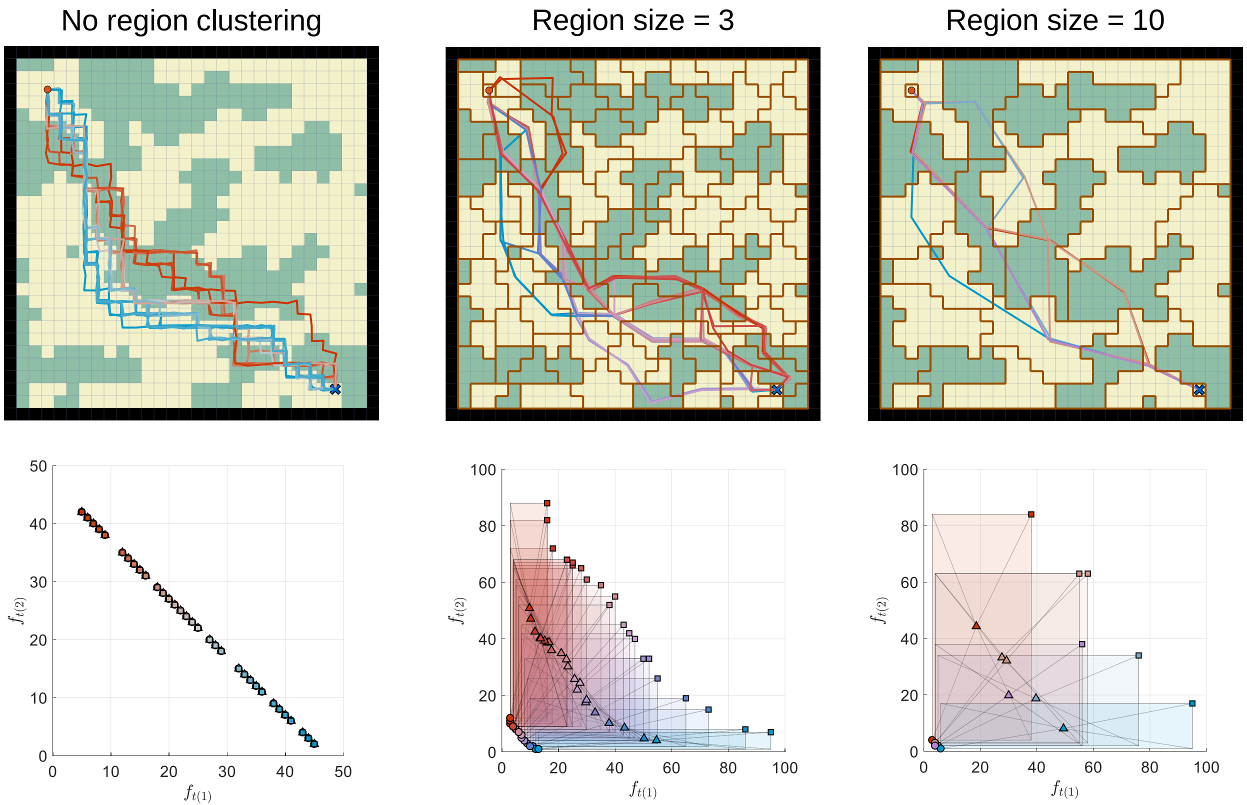
Binary Shortest Paths (TE)

We use the MOEA/D algorithm to find all Pareto optimal shortest paths in a binary terrain environment.

Features:

- $f_{t(1)}$: Distance in meadow
- $f_{t(2)}$: Distance in forest

Using Tchebycheff scalarization.



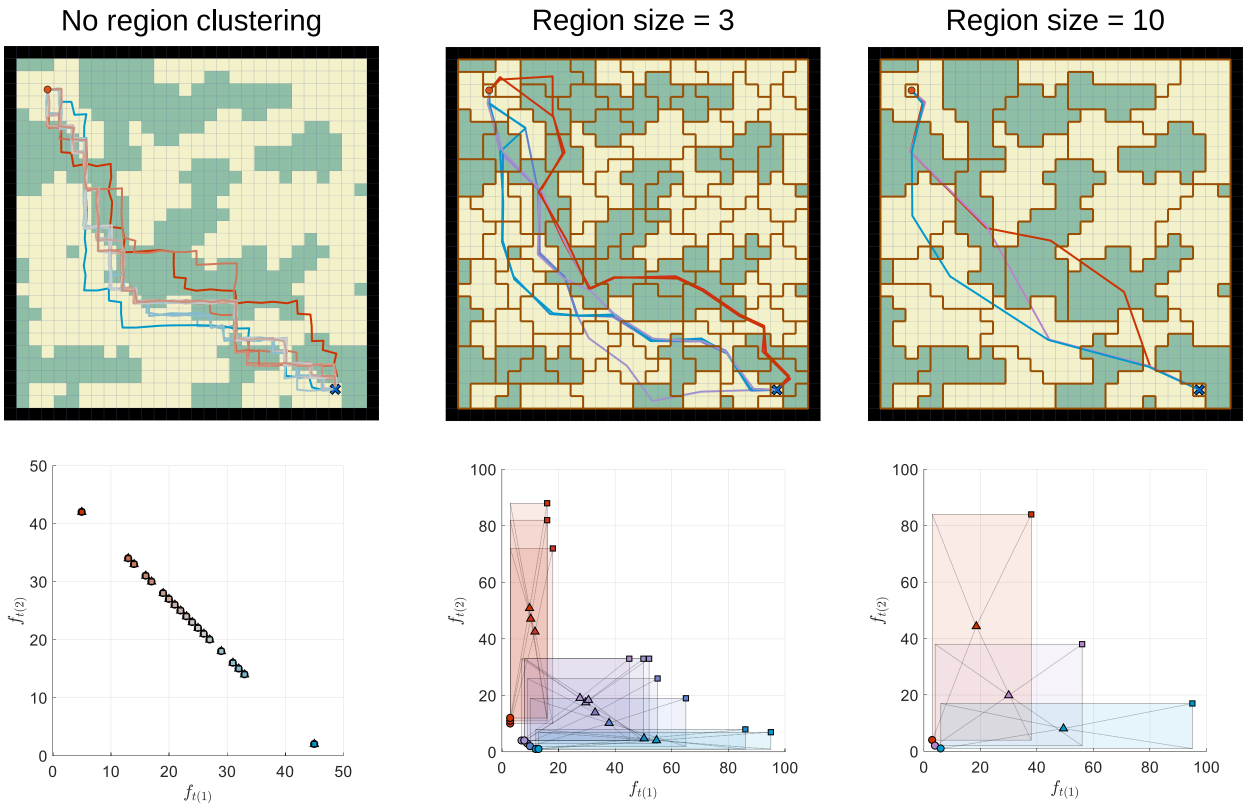
Binary Shortest Paths (OWA)

We use the MOEA/D algorithm to find all Pareto optimal shortest paths in a binary terrain environment.

Features:

- $f_{t(1)}$: Distance in meadow
- $f_{t(2)}$: Distance in forest

Using Ordered Weighted Average scalarization.



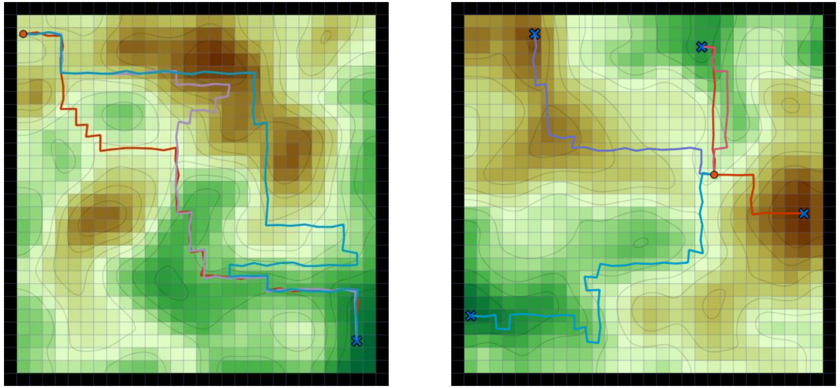
Distance and Elevation

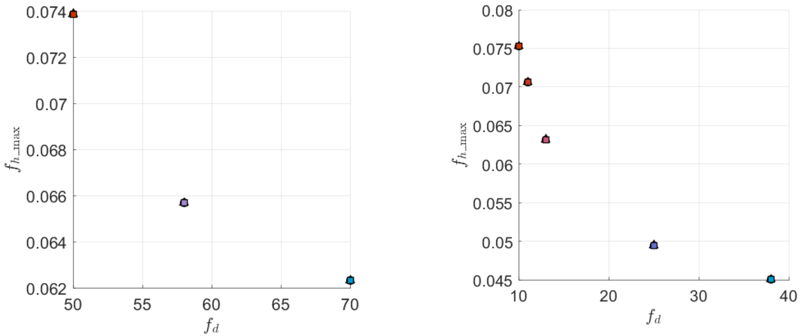
These examples show how summation and maximization features can be combined.

Features:

- f_d: Total distance
- f_{h_max}: Maximum slope

Using Ordered Weighted Average scalarization.





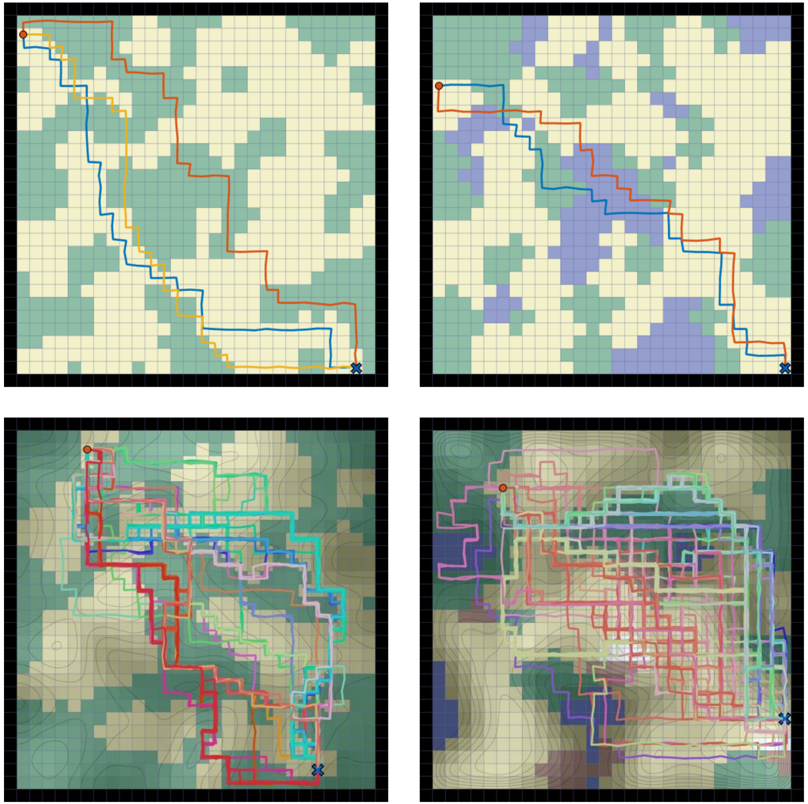
Many Objectives

Terrain transition features allow for additional agent behaviors.

- Binary terrain
  - Red agent favors forest
  - Blue agent favors meadow
  - Yellow agent favors the edge
- Trinary terrain
  - Red agent prefers the order meadow, water, forest
  - Blue agent prefers the order meadow, forest, water

Visualizing the objective space with more than 2 or 3 objectives is challenging.

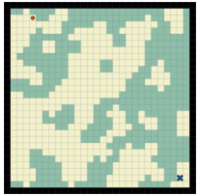
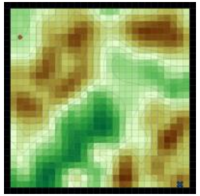
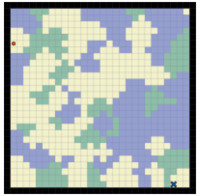
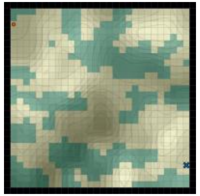
Paths are colored based on objective weight similarity.



MOEA/D Improvement

We designed an experiment to compare solutions found using the pre-scalarized decomposition method and MOEA/D.

- 10 problem types
- 30 test environments of each type
- Define $10 \times N$ weight vectors for each problem, where N is the number of objectives
- Find solutions for each weight vector using both approaches
- Measure the percent improvement of MOEA/D over pre-scalarization


Average percent improvement of MOEA/D over pre-scalarization

Prob. #	# of Sum Obj.	# of Max Obj.	Avg. Nodes	Avg. Edges	$\xi = 0$			$\xi = 0.5$			$\xi = 1$		
					WS	OWA	TE	WS	OWA	TE	WS	OWA	TE
1	2	0	103	507	0.00	1.39	8.49	0.00	1.67	9.30	0.00	1.78	9.94
2	1	1	66	318	5.35	7.10	13.12	5.27	5.08	5.44	5.57	5.75	7.02
3	0	2	64	300	12.43	12.01	12.94	5.71	5.87	6.55	0.93	0.79	1.17
4	3	0	65	311	-0.02	0.83	5.03	0.00	0.14	0.67	0.00	0.09	0.56
5	2	1	92	447	9.64	13.73	20.53	7.93	7.08	8.51	9.80	8.55	9.27
6	5	1	93	454	17.02	21.75	30.27	5.32	5.93	11.00	5.46	5.17	9.91
7	6	0	109	545	-0.47	2.32	8.87	-0.15	2.39	9.82	-0.15	3.06	12.35
8	15	0	93	454	-0.08	1.64	8.02	-0.03	1.53	8.00	-0.04	2.02	11.73
9	15	2	91	445	26.31	31.75	42.06	7.14	9.14	16.35	4.36	5.38	12.22
10	26	3	93	450	15.68	21.66	29.27	6.34	8.01	13.46	4.55	5.85	11.46

Greedy Agent Example

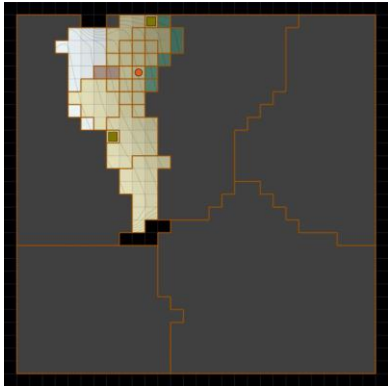
We define a greedy agent strategy that can solve generic problems in the CMM framework.

<u>Greedy Agent Strategy:</u>

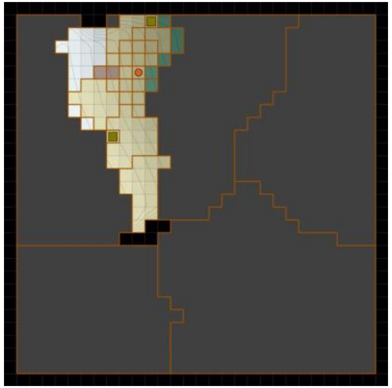
- After updating the region graph, determine the next target location:
  - Closest required resource, if visible
  - Otherwise, closest unobserved region
- Plan and follow a least-cost route to the target location

No region clustering

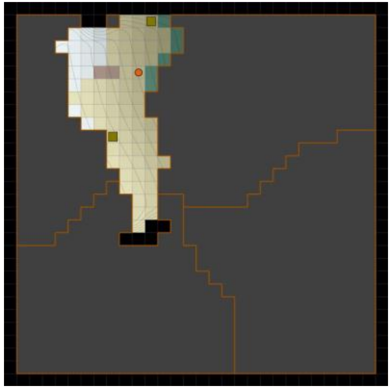
Region clustering without local region memory



Region clustering with local region memory



Region clustering only for unobserved regions



Future Work

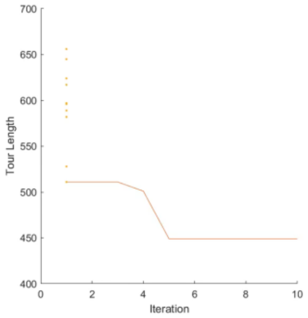
The full potential of the CMM framework extends beyond this work.

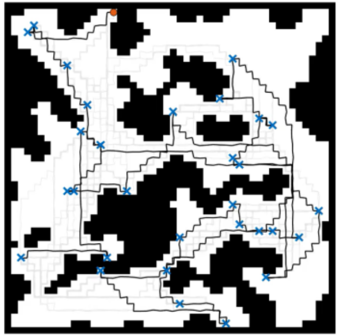
Advanced agent strategies:

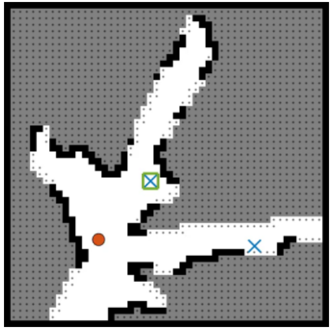
- Ant Colony Optimization
- Monte Carlo Tree Search

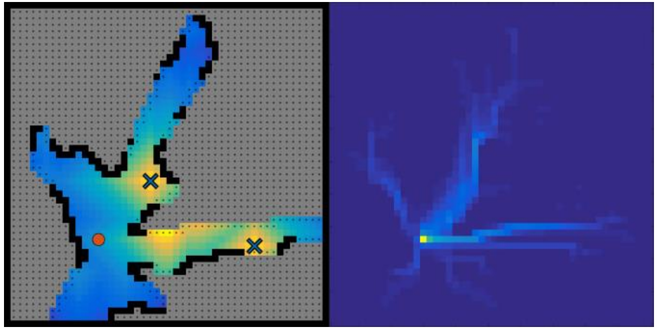
Anticipatory analysis:

- Generate synthetic trajectories for a particular agent profile
- Predict how the agent will behave in a new environment









Conclusion

The CMM framework is a useful tool for studying sequential multicriteria decision-making problems with uncertainty.

The pathfinding problem is a versatile problem domain that can be configured in many different ways.

The set of Pareto optimal solutions gives a broad overview of the options available to a decision-maker.

There's still lots of opportunity for future work!

Thank You

Thanks to my advisor, Dr. Keller,
To my committee, Dr. Popescu, Dr. Skubic, and Dr. Zare,
To NGA and ARO/NVESC for supporting me during this work,
And to my parents, friends, family, and everyone else.