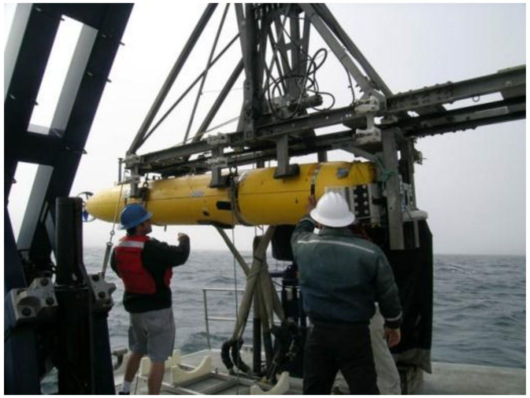
Plan-based Policy Learning for Autonomous Feature Tracking

Maria Fox, Derek Long and
Daniele Magazzeni

KING'S College LONDON

Interesting Problem in the Ocean Sciences

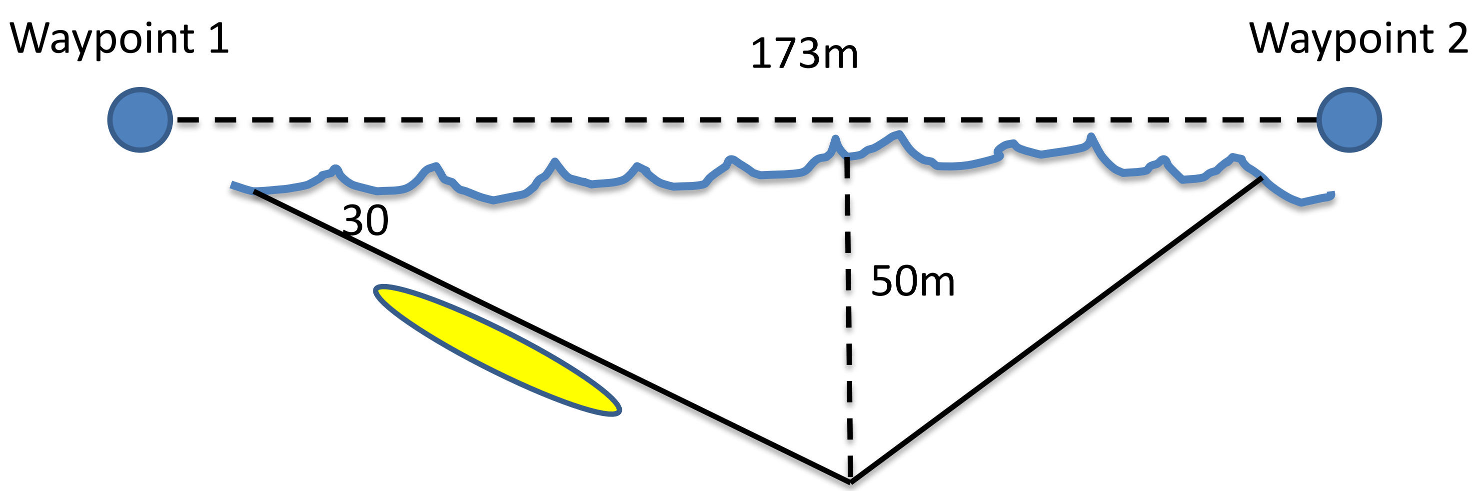


- Harmful algal blooms are huge patches of algae that come up from the bottom and bloom on the surface.
- They are associated with widespread marine mortality events and shellfish poisonings.
- Our specific task is to use an AUV to follow a particular contour on the surface defined by a chlorophyll concentration.
- The problem involves intelligent decision-making, but combinatorial reasoning cannot be done on board.
- Plan-based policy learning produces robust, lightweight, intelligent trackers.

M B A R I

Patch-tracking

- Why can't the AUV do edge-following?
  - The edge is not distinct but might be dispersed over several metres
  - The AUV does not have high manoeuverability relative to the contour of the edge.

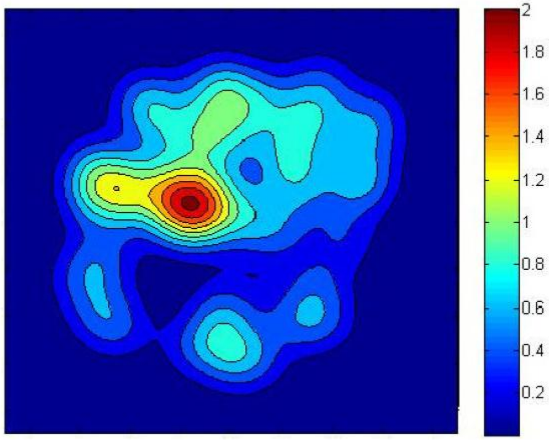


Our Approach

- We use the same policy-learning strategy as we used in our work on multiple battery management (ICAPS 2011):
  - Sample patch instances
  - Use a planner to solve them offline
  - Execute the plans against the instances to construct <policy state,action> examples for training
  - Learn a decision tree classifier that maps states to actions
  - Evaluate the performance of the learned policy on new simulated instances.

The Simulator

- To build the planning instances we generate patch contours using a simulator constructed at MBARI.

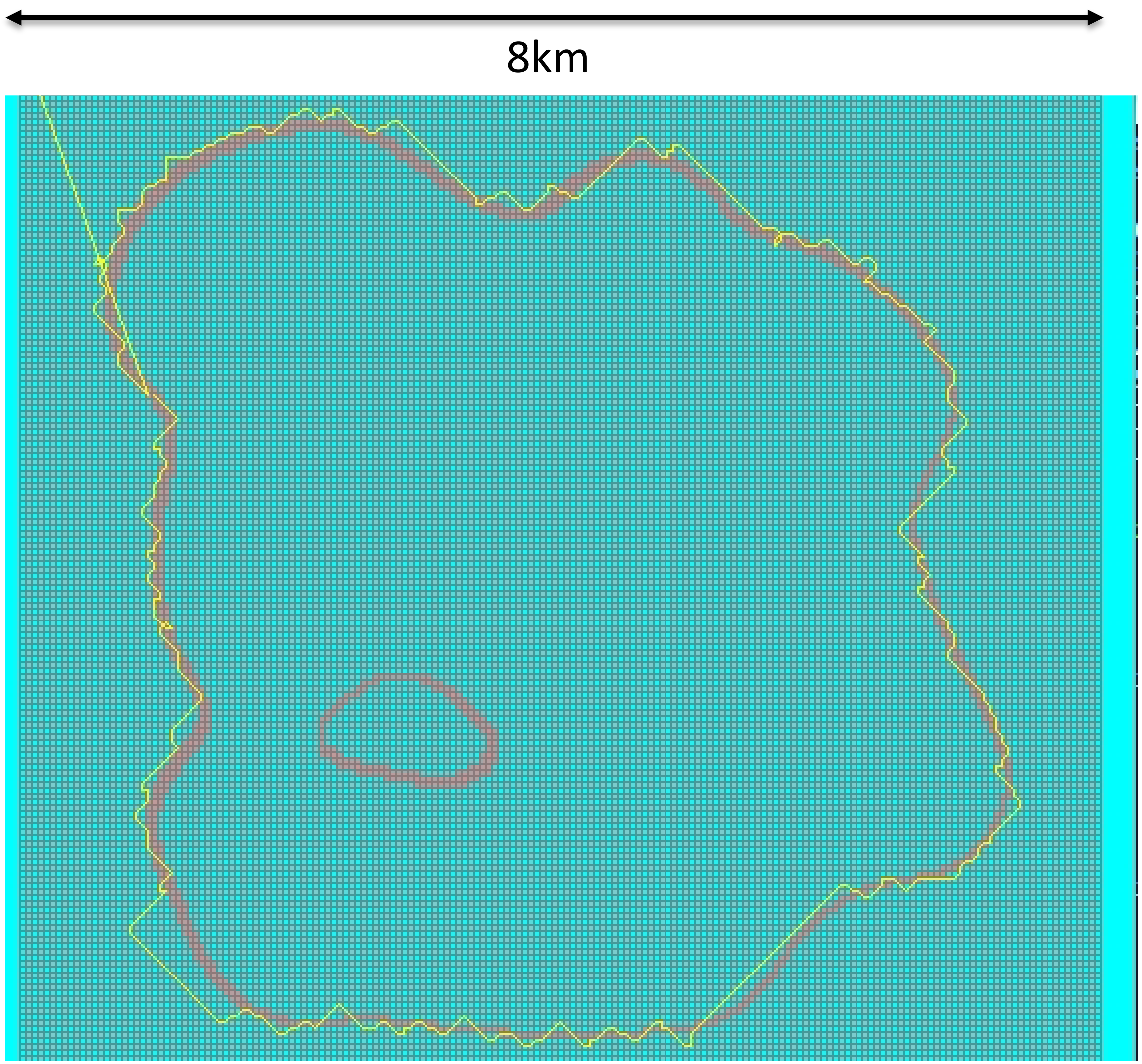


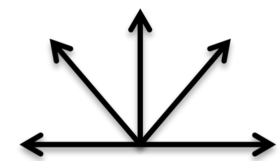
- Red regions signify high chlorophyll
- There are many contours defined by different chlorophyll levels
- The outermost contour defines what we call a "standard" patch.

(simulator developed by Mike Godin, Research Engineer, MBARI)

Example Plan

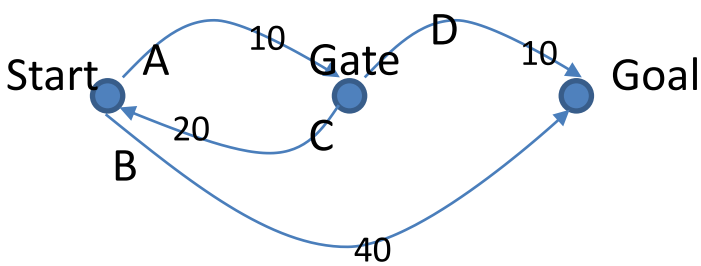
We solve 2000 of these, each plan contains about 500 actions



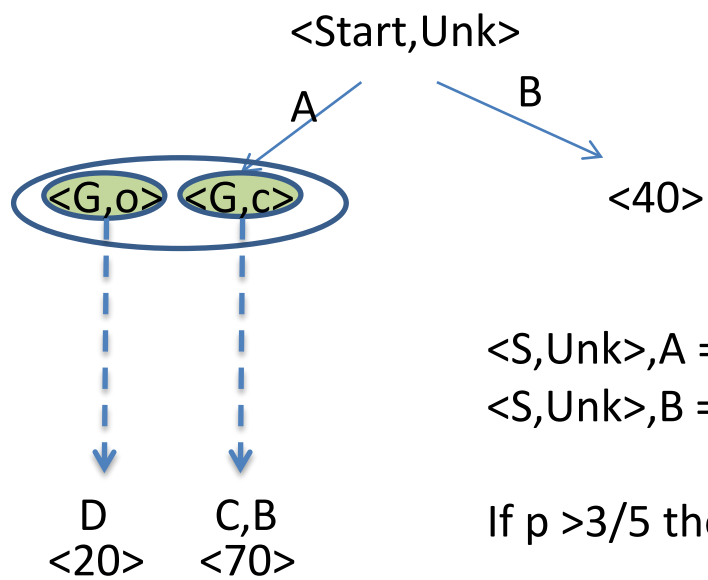


Max confusion = 5

Hindsight Optimisation



Gate is open with probability p



<S,Unk>,A = 20p + 70(1-p) = 70-50p

<S,Unk>,B = 40

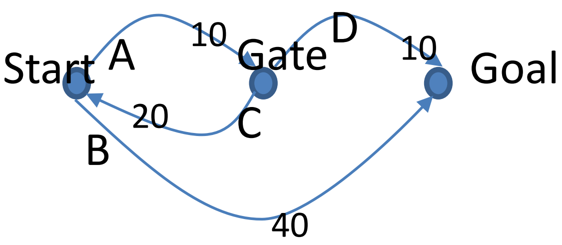
If p >3/5 then A is better, otherwise B

Final policy for p > 3/5:

- <S,Unk> $\rightarrow$ A
- <S,c> $\rightarrow$ B
- <G,o> $\rightarrow$ D
- <G,c> $\rightarrow$ C

HOP samples and plans from every intermediate state.

Plan-based policy learning



Build 1000 samples:

in 1000p of them, the gate is open. The optimal plan is A,D.

in 1000(1-p) of them the gate is closed and the optimal plan is B.

Play these plans out in simulation against the initial policy state <S,Unk>:

in 1000p cases, from state <S,Unk> we apply A

in 1000(1-p) cases, from state <S,Unk> we apply B

in 1000p cases, from state <G,o>, we apply D

Gate is open with probability p

Classify:

if p > 1/2 then <S,Unk> $\rightarrow$ A

otherwise <S,Unk> $\rightarrow$ B

and <G,o> $\rightarrow$ D.

Policy when p > 1/2:

<S,Unk> $\rightarrow$ A

<S,c> $\rightarrow$ B

<G,o> $\rightarrow$ D

<G,c> $\rightarrow$ C

Roll out against new samples:

When we execute A we will sometimes arrive in <G,c>.

No policy action for <G,c>!

Repair:

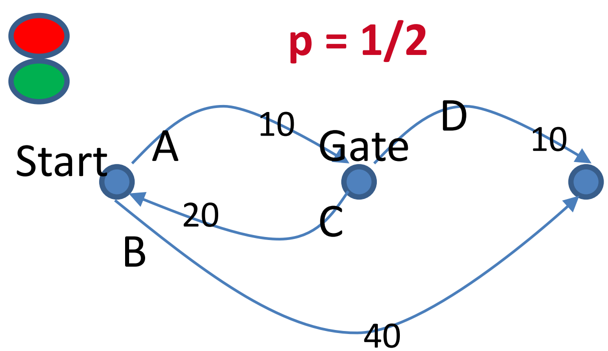
add <G,c> $\rightarrow$ C and reclassify:

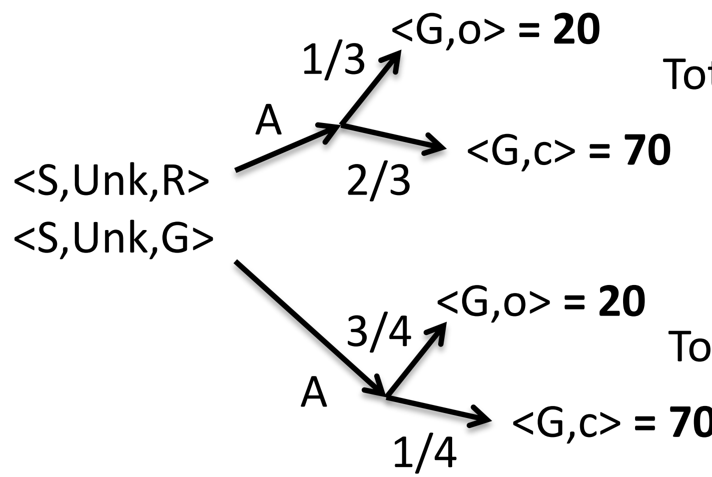
We learn to do this when p > 1/2, while HOP learns to do this only when p > 3/5, and it proposes B otherwise.

So HOP is optimal in this case, and PBL is not.

Adding a Light in HOP

- There might be observable variables that increase discrimination.





Total cost of A: 53.3

Total cost of A: 32.5

Total cost of B: 40

	RED	GREEN
OPEN	2/5	3/5
CLOSED	4/5	1/5

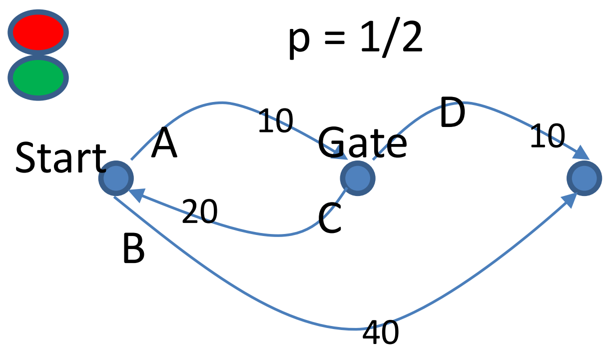
Policy fragment:

<S,Unk,R> → B
<S,Unk,G> → A

HOP has to recalculate the conditional probabilities in each state when the light is added.

Adding a Light in PBL

Note that we don't have to redo the planning step when we add the light!



Policy State:
<Location, GateStatus, LightStatus>

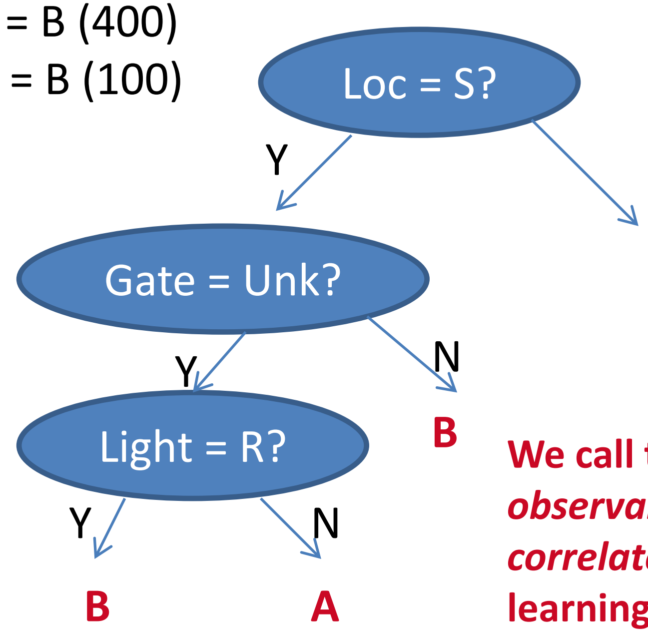
Play out the plans against the initial policy state:
<S,Unk,R> = A (200)
<S,Unk,G> = A (300)
<S,Unk,R> = B (400)
<S,Unk,G> = B (100)

	RED	GREEN
OPEN	2/5	3/5
CLOSED	4/5	1/5

We don't need to be given this explicitly

Policy fragment:

<S,Unk,R> $\rightarrow$ B
<S,Unk,G> $\rightarrow$ A

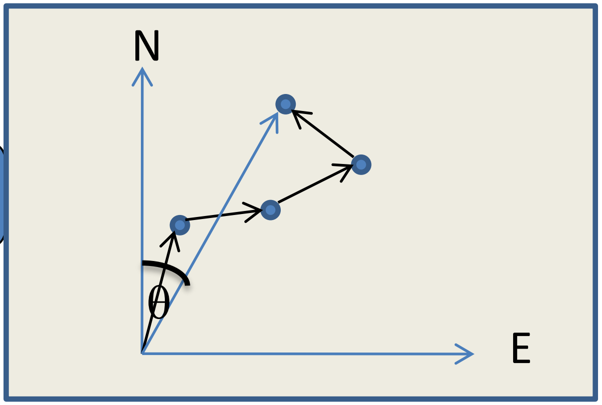


We call this *observable-correlate* learning

Observable-Correlate Policy Learning

- An informative set of state variables is essential to enable the classifier to structure the decision tree.
- The policy state consists of:
  - Average bearing over last 10 moves (angle from North)
  - The *count* of times each of the five actions (Left, Right, Forward, Forward-Left, Forward-Right) was performed in the plan so far (LC, RC, FC, FLC, FRC)
  - What chlorophyll level was last sensed
  - Facing direction (N,S,E,W)
  - Confusion level (0-max)

(θ, LC, RC, FC, FLC, FRC , R/W, Facing, Conf)



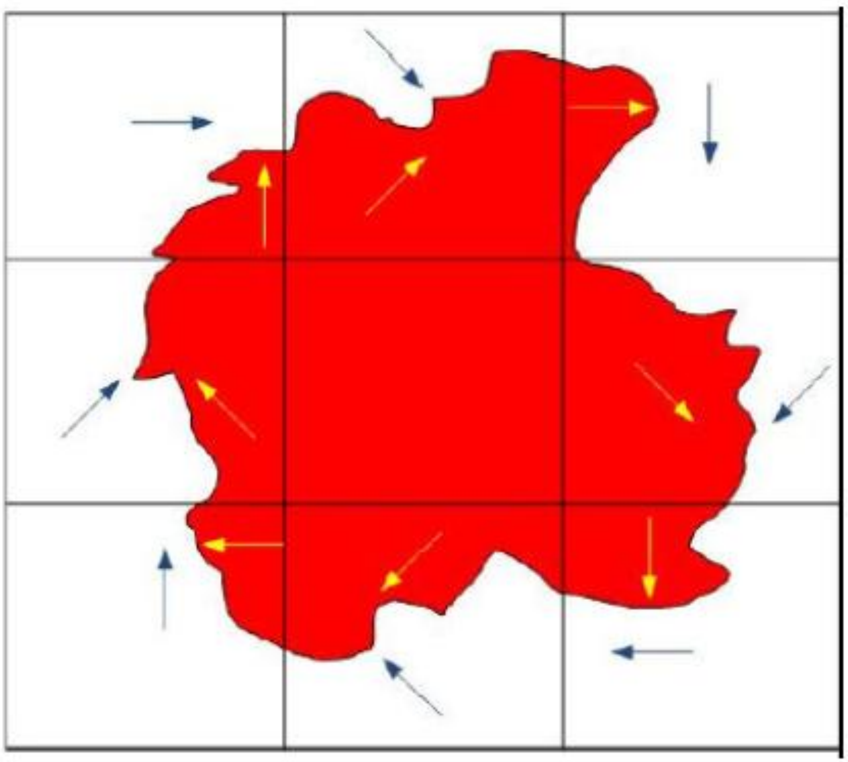
We don't retain the cell that is used in the plan state!

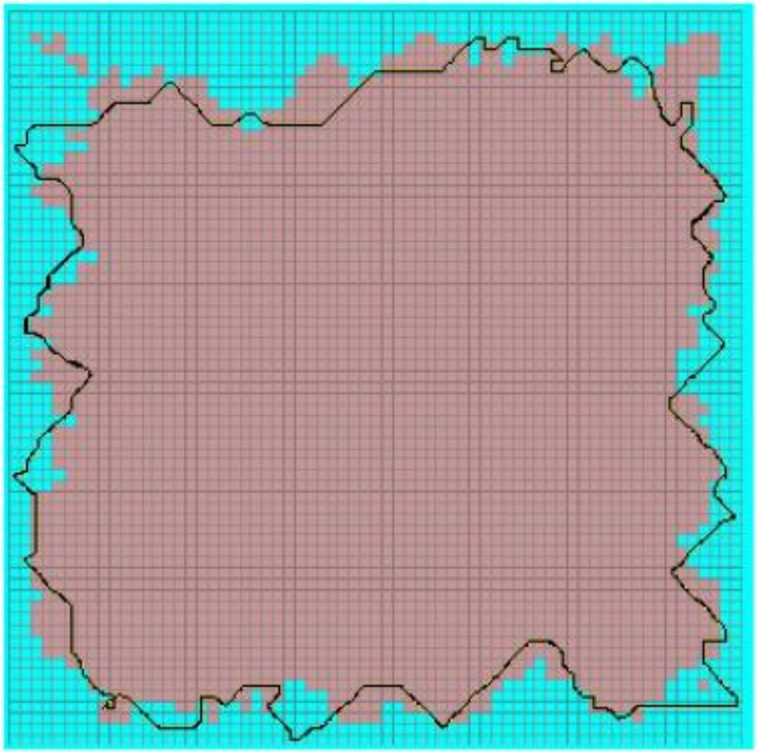
Why no cell? Why not HOP?

- We want the policy to be independent of the cell that the AUV is in
  - Conditioning on the cell would make the policy thousands of times bigger
- Possibly HOP could compute the conditional probabilities at each state by sampling from the simulator if it knew the AUV cell, but.....
- suppose HOP has access only to the policy state, how would it compute the conditional probability of being in the patch following a given next move?

Repairing the Policy with Default actions

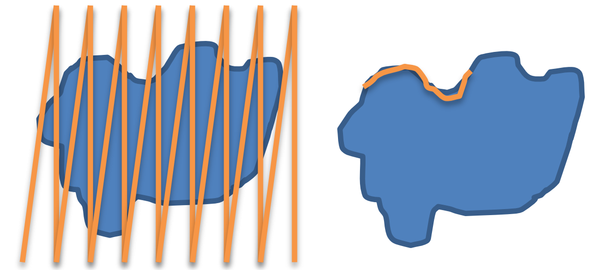
Area-based Default Action





The repaired policy traversing an unseen patch

Evaluation

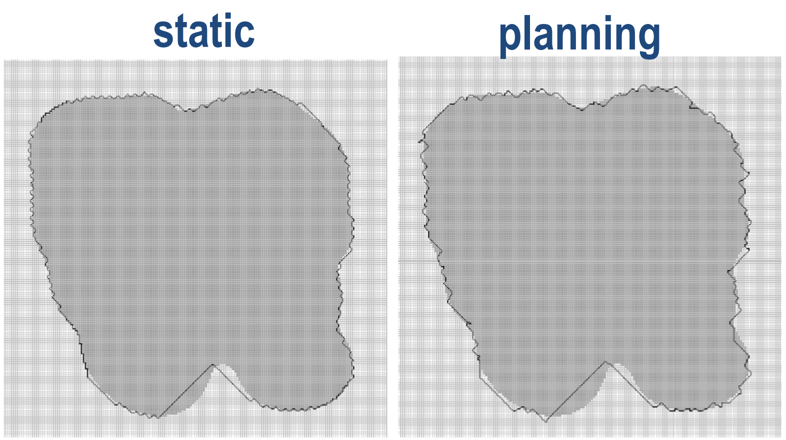


- For a fair evaluation we need:
  - good measures
  - good (non planning) opponent

How far is the patch from the path?

$$\Psi = \frac{\displaystyle\sum_{\forall e \in \mathcal{E} \cdot t^* \in \mathcal{T} \cdot \nexists t' \in \mathcal{T} \cdot ||e - t'|| < ||e - t^*||} (||e - t^*||^2)}{|\mathcal{E}|}$$

$$\Omega = \frac{\displaystyle\sum_{\forall t \in \mathcal{T} \cdot e^* \in \mathcal{E} \cdot \nexists e' \in \mathcal{E} \cdot ||t - e'|| < ||t - e^*||} (||t - e^*||^2)}{|\mathcal{T}|}$$



How far is the path from the patch?

- Static Policy + loops-avoidance routine

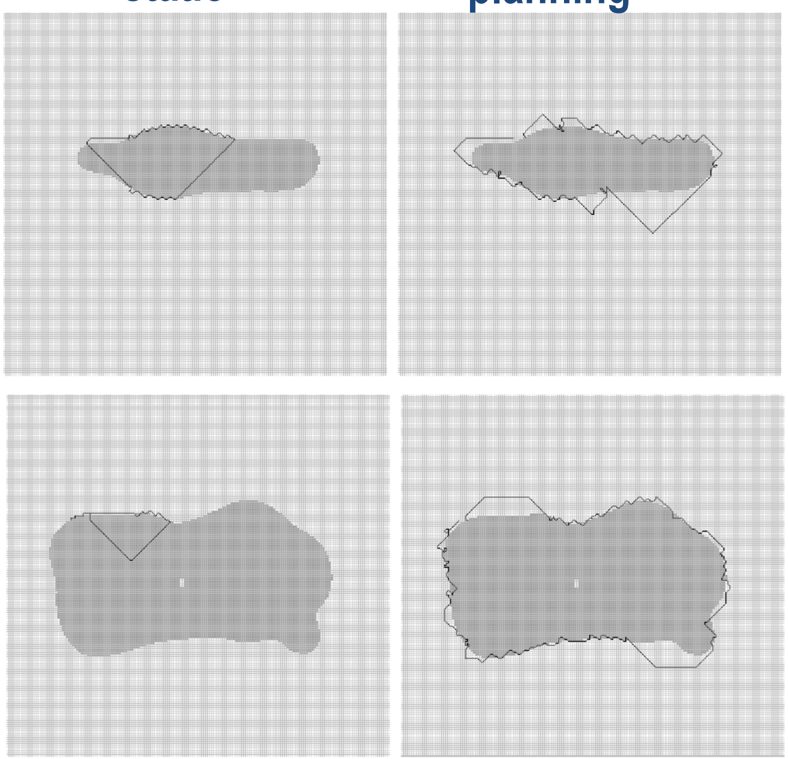
  "On every move, use the area-based default action"

Robustness Tests

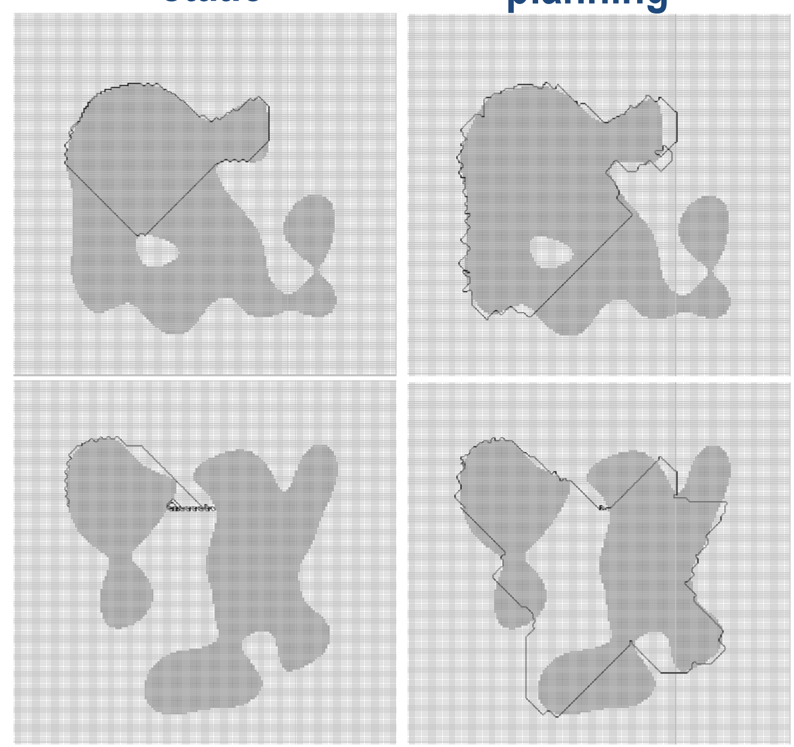
We perform 10,000 tests and average the results

Bloom	Policy	Ψ	Ω	Conf	Length
Standard	static	39.37	2.56	2.52	510.94
	planning	39.48	3.28	3.59	533.10
Horizontal	static	1741.18	51.72	19.93	328.77
	planning	187.83	15.08	14.67	387.23
Thin	static	56.95	45.14	45.14	314.39
	planning	25.09	35.80	9.71	353.60
Inner	static	1009.51	21.40	16.23	358.45
	planning	585.66	20.07	15.56	382.94

static planning



static planning



Conclusions

- Plan-based policy learning works well when:
  - It would be very difficult (or impossible) to calculate the conditional probabilities at intermediate states
  - The sampled instances can be usefully seen as planning problems
  - There is a gap between the plan state variables and the observable variables, and the relationship between them is not known.
- The experiments reported here were done in simulation, but we have now performed sea tests at MBARI with Kanna Rajan and Frederic Py, with promising initial results (work in progress).