Learning to plan with uncertain topological maps

Edward Beeching[1], Jilles Dibangoye[1], Olivier Simonin[1], and Christian Wolf[2]

[1] INRIA Chroma team, CITI Lab. INSA Lyon, France.
https://team.inria.fr/chroma/en/
[2] Université de Lyon, INSA-Lyon, LIRIS, CNRS, France.
{firstname.lastname}@insa-lyon.fr

Project page https://edbeeching.github.io/papers/learning_to_plan

1 Supplementary material

1.1 Example graphs

Figure 1 shows example graphs extracted from three different environments extracted with the method described in section 3.3 of the main paper.

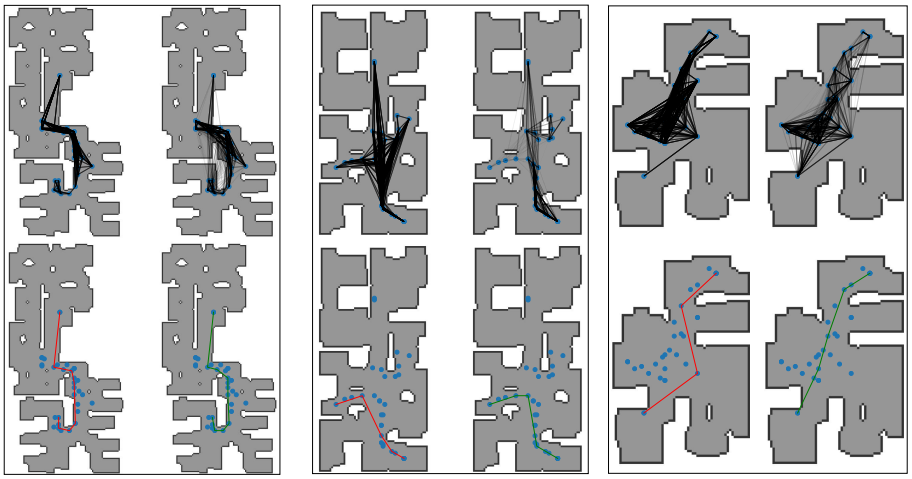


Fig. 1: Examples of top down maps and graphs from three environments. For each environment - Top-left: topological map with ground truth connectivities, top-right: topological map with connection probabilities estimated by the learned f_{link} function with line opacity corresponding to the link probability, bottom-left: ground truth shortest path between a source an target node, bottom-right: shortest path estimated by the Graph Neural Planner. Note the very bottom-right prediction connects two nodes that are not connected in the ground truth, this example would not be counted as a valid path during evaluation of the SPL metric.

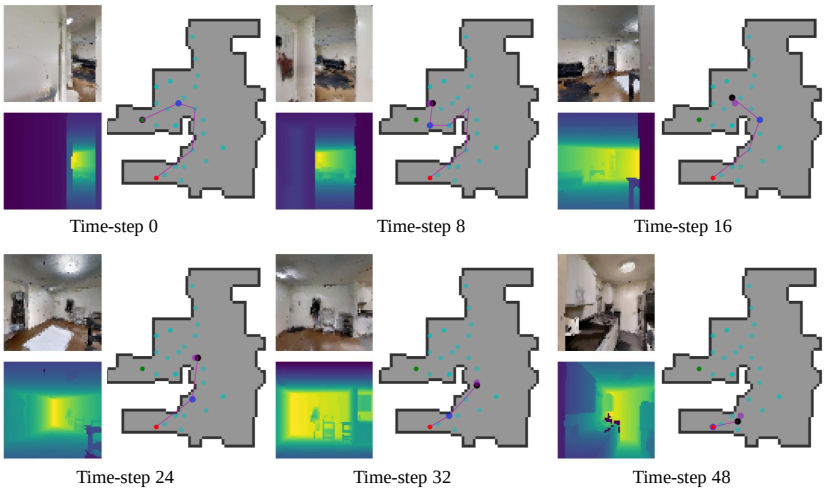


Fig. 2: Six time-steps from a rollout of the hierarchical planner (graph+local) in an unseen testing environment. For each time-step: left – RGB-D observation, right – map of the environment (unseen) with graph nodes, source node, target node, agent position(black), nearest neighbour to the agent, local point-goal provided by the high level planner and planned path.

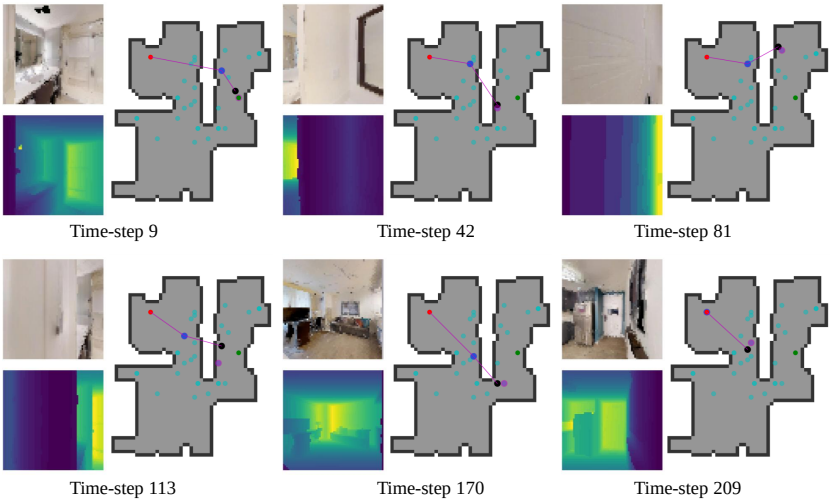
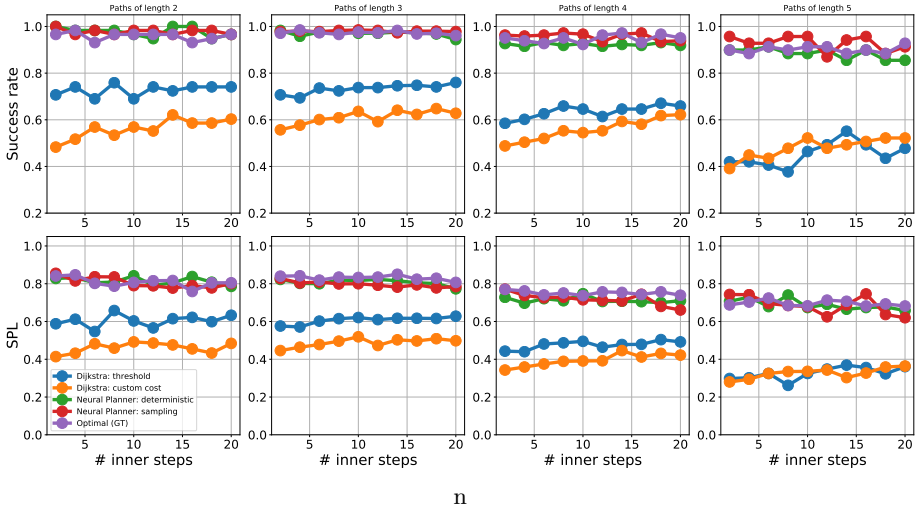


Fig. 3: Failure case - Six time-steps from a rollout of the hierarchical planner (graph+local) in an unseen testing environment. For each time-step: left – RGB-D observation, right – map of the environment (unseen) with graph nodes, source node, target node, agent position(black), nearest neighbour to the agent, local point-goal provided by the high level planner and planned path.



n

Fig. 4: Performance of the hierarchical agent with varied low level inner loop steps for a range of high-level path lengths.

1.2 Examples of Graph planner trajectories

Figures 2 and 3 of this document show additional rollouts of the hierarchical planner, complementary to figure 6 of the main paper.

1.3 Effect of the length m of the local policy

In figure 4 of this document we show the effect of the parameter m of the local policy, described in section 3, page 6, of the main document, when evaluated on a limited random subset of the validation data (1,200 problem instances). The m parameter limits the maximum number of steps the local policy can take before giving control back to the high-level graph planner. We recall that the local policy can also decide to terminate the inner loop earlier through an explicit *STOP* action. We see that performance of the planner and policy is comparable up to 20 time-steps, which means the computationally costly planning step can performed less frequently than the low level control of the local policy, without a reduction in performance.

1.4 Hyper-parameters

Table 1 provides the hyper-parameters for the three different neural models used in this work:

- the explorative policy used to create the graphs, trained through RL;
- the graph based high-level planner, trained in a supervised way, and
- the local policy, trained through RL.

Table 1: Hyper-parameters for the exploratory policy, node linkage function and Neural Planner

Exploratory policy and F_{link}	
Simulator resolution	64×64
optimizer	Adam: betas=(0.9, 0.999), eps=1e-5
learning rate	2.50e-04
weight decay	0.0
parallel agents	16
GRU hidden size	512
Entropy coef	0.001
Advantage normalization	True
Generalized Advantage Estimation	True
Minibatch size	4 (trajectories)
PPO CLIP	0.1
num environment steps	200 M
TBPTT	128
Neural Planner	
optimizer	Adam: betas=(0.9, 0.999), eps=1e-8
learning rate	0.001
weight decay	0.0001
batch size	32
num epochs	500
dataset size	36,000
GRU size	256
Feature size	512
Learning rate decay: 0.1 every 120 epochs	
GNN steps	6
Multilayer GRU Depth	2
Gradient norm clipping	2.0
Local policy	
Simulator resolution	64×64
optimizer	Adam: betas=(0.9, 0.999), eps=1e-8
learning rate	2.50e-04
weight decay	0.0
parallel agents	16
GRU hidden size	512
Entropy coef	0.001
Advantage normalization	True
Generalized Advantage Estimation	True
Minibatch size	4 (trajectories)
PPO CLIP	0.1
num environment steps	200 M
TBPTT	128