Autonomous Search Over Probabilistic Scene Graphs

Isaac Remy[1], Danny Broyles[2], Karen Leung[2]

[1]Department of Electrical and Computer Engineering, [2]Department of Aeronautics and Astronautics, University of Washington

Abstract and Motivation

Developing efficient search capabilities for robots in cluttered environments with limited prior information presents a significant and crucial challenge. This task has wide-ranging applications, including home assistance (e.g., searching for urgently-needed medication in a household setting) and search and rescue operations (e.g., searching for lost/injured persons in the wilderness). The key objective is to devise an optimal search strategy that maximizes the accumulation of target objects while minimizing distance traveled. To address this, our autonomous search agent utilizes reinforcement learning (RL) to generate a search policy capable of navigating through a scene-graph of the environment's *containers*, effectively locating as many target objects as possible despite the lack of prior knowledge regarding the specific containers where they are stored.

Background

Consider a home assistance robot tasked with locating a specific object (e.g., a T.V. remote) in a cluttered household living room. The robot will have access to, or generate, a map of the environment with specific containers identified (couch, table, etc).

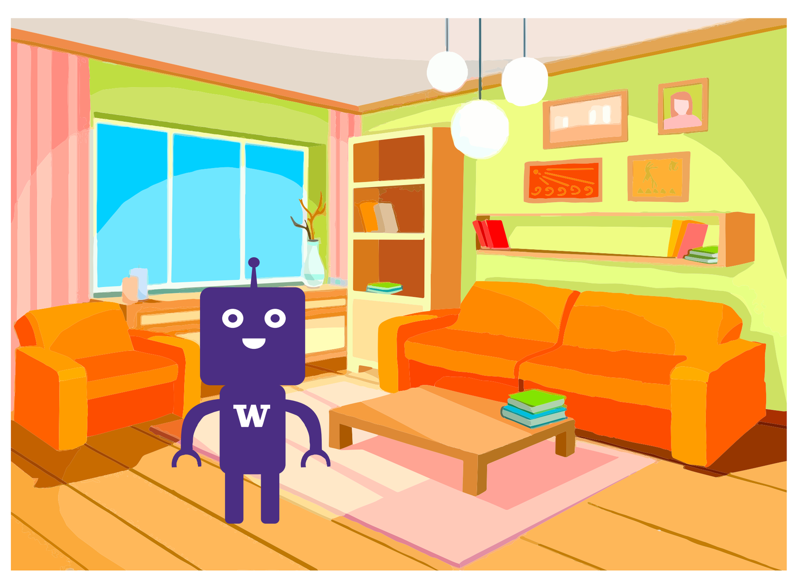
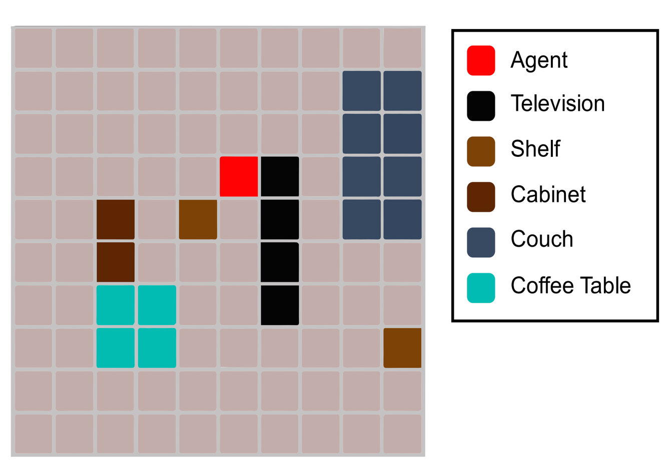


Fig 1. Typical living room. "Container" abstractly describes storage entities, such as furniture, rooms, or buildings

Fig 2. Grid-world representation of a living room with containers. Living room layouts are randomly generated; robotic agent is in red.



Problem Formulation and Proposed Solution

Environment Representation

Scene-graphs, originating in the field of computer graphics, provide powerful and condensed encodings of real-world environments; we end up with a "container graph".

Fig 3. Construction of scene-graph for room shown in Fig. 1.

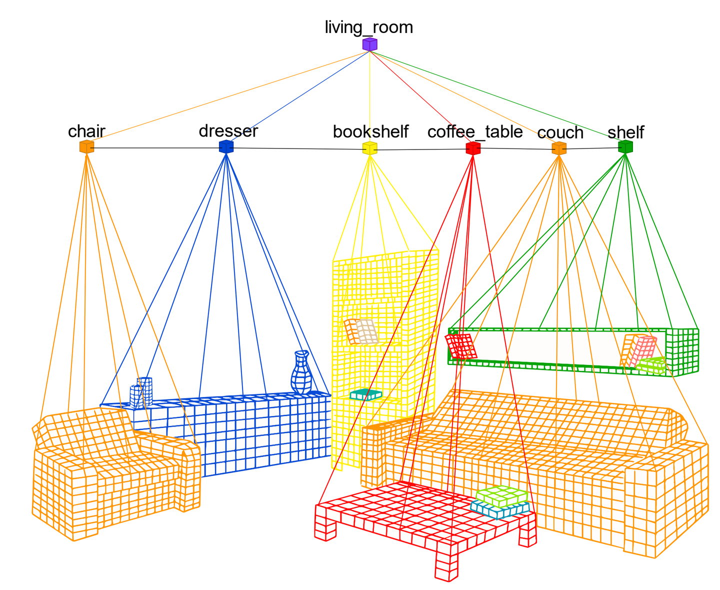
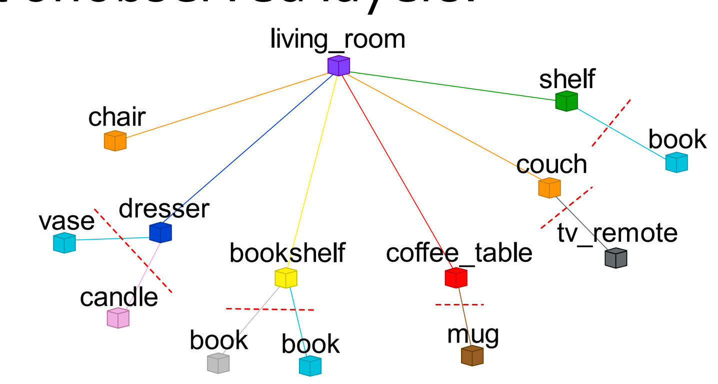


Fig 4. Final container graph. Red dashed lines represent unobserved layers.



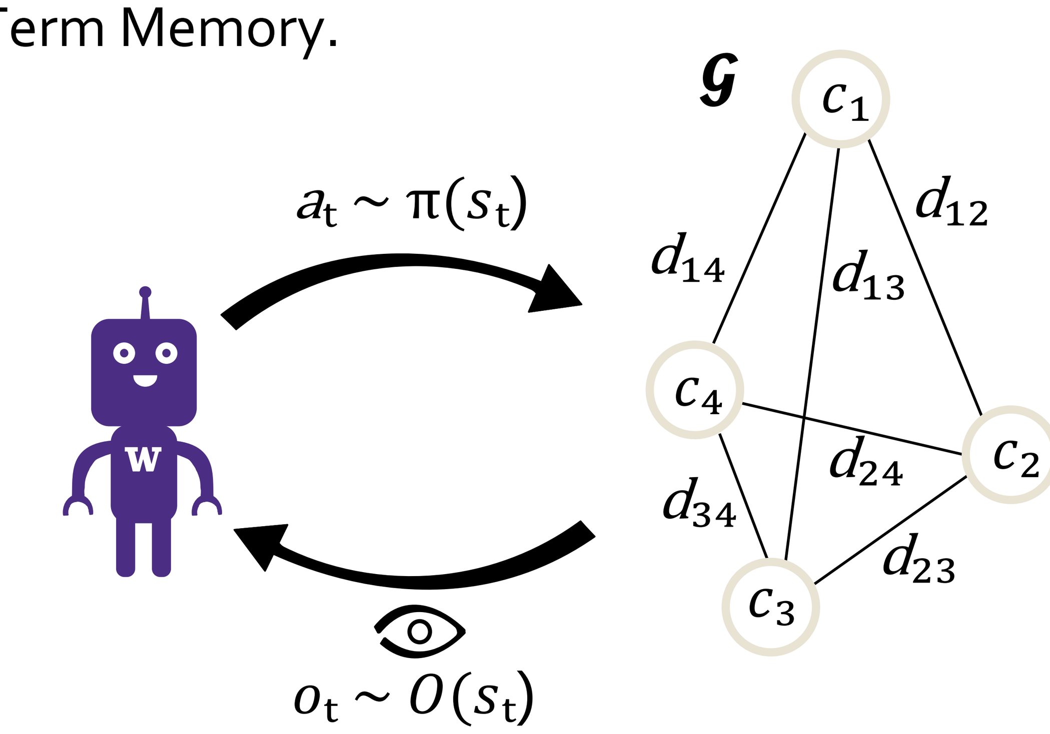
RL Agent

Our agent runs the popular Proximal Policy Optimization algorithm on top of a recurrent neural network using Long-Short Term Memory.

Markov Decision Process Breakdown

- Observation: $\mathcal{G}(c,\ d),\ c \in \boldsymbol{C},\ d \in \boldsymbol{D}$
- Actions: $c \in \boldsymbol{C}$
- Reward function:

$$r(s_t, a_t) = \lambda_o \sum_{k=0}^{N_c} obj_k^* - \lambda_a ||c_t - c_{t-1}||$$



Goal: Plan trajectory informed by learned container probabilities and distances

Preliminary Results

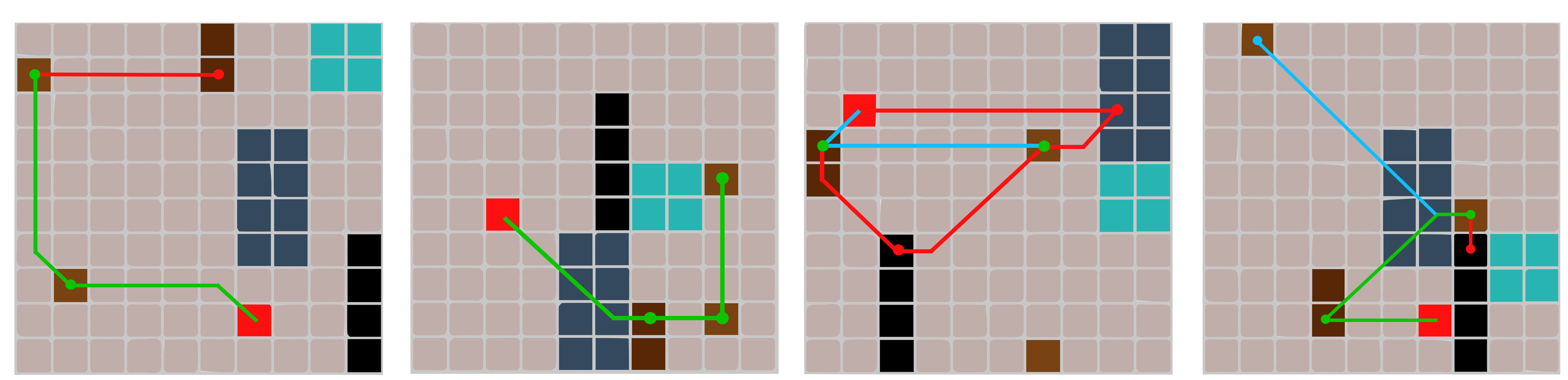


Fig 5. Comparison of optimal (blue) and learned (red) trajectories after training. Green represents a common trajectory. Circles represent a container being "opened".

Analysis and Future Work

Performance Plots

Figures 6 and 7 track the mean reward and episode length of the agent during training against an oracle for two different values of distance metric $\boldsymbol{\lambda_a}$. The oracle performance represents the maximum reward possible for every training episode; the oracle has full access to all object locations.

Fig 6. Agent performance when $\lambda_a = 0.05$

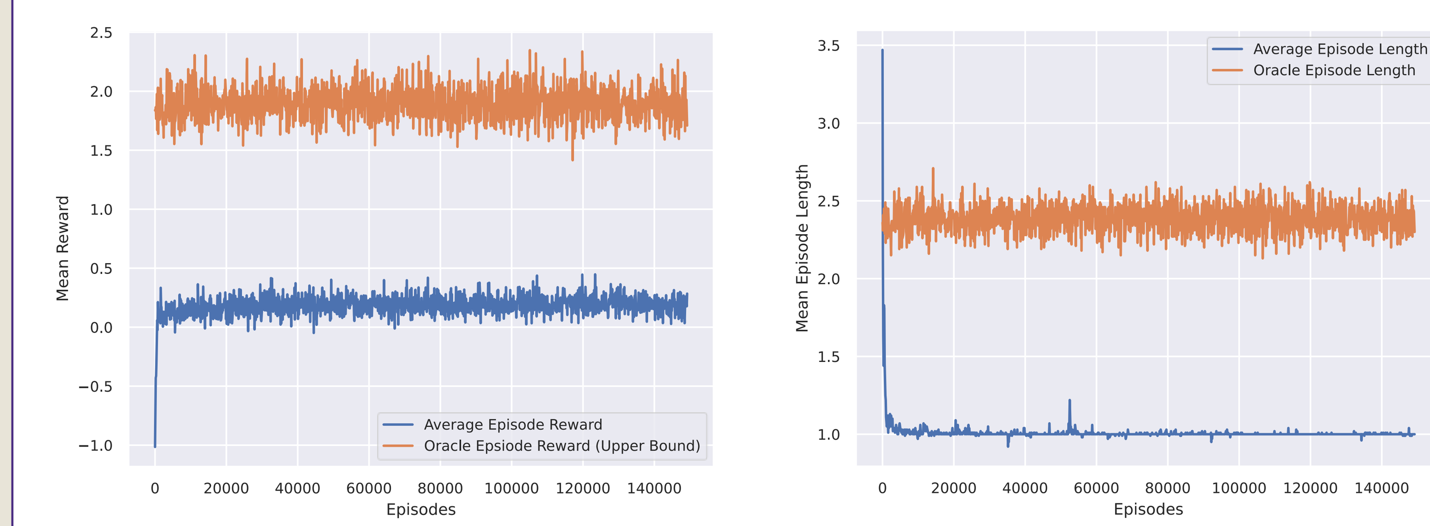
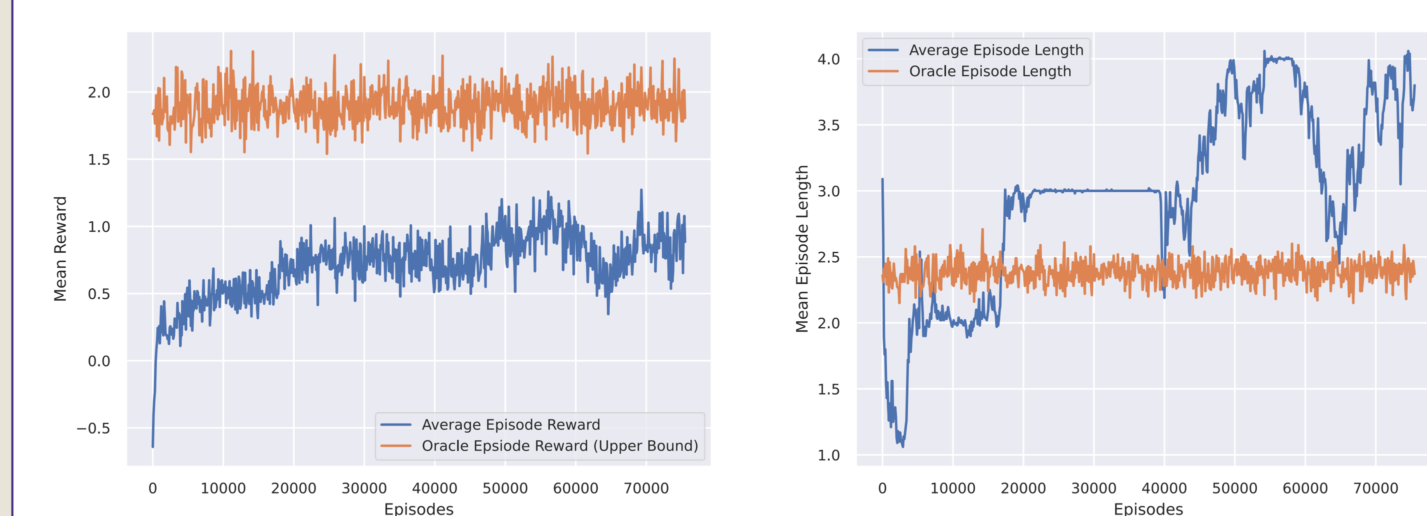


Fig 7. Agent performance when $\lambda_a = 0.02$



Takeaways

The penalty shaping for the distance makes a big difference in behavior, but how that penalty is shaped is contextually dependent. Further analysis also showed this problem is an Orienteering Problem variant, which is NP-hard.

Next Steps

For the next phase, we plan to utilize large language models for providing container/object priors and construct a Bayes' net model from those priors to do inference during planning.

Acknowledgements

We would like to thank Jakub Kowalewski for assistance with the codebase during early stages of development, and Professor Abhishek Gupta for providing guidance on the RL formulation.