Reusable sub-goal based Monte Carlo planning algorithm

Beomjoon Kim

January 5, 2019

1 Introduction

Motivation:

- In many complex robot planning problems, parts of the solutions of the entire planning problem may be known in advance because they were learned, planned, or specified by a human.
- For instance, for a cooking problem, a human might hint the planner that the object must be washed before placed on a stove, or a hierarchical planner might determine that a swept volume to the object should be cleared first, for the simple problem of picking an object up.
- We assume that these hints are represented as a sequence of sub-goals that define the associated sub-planning problems that, if you can find feasible solutions to these sequence of sub-planning problems, then you can find a feasible solution to the entire problem.
- The sub-goals allow us to shorten the planning horizon, and focus the sampling effort on manipulating a particular set of objects in achieving the immediate sub-goal. Moreover, if we have sub-goals for a smaller but recurring planning problem, then we may re-use the same sub-goals whenever the smaller planning problem needs to be solved in a larger planning problem.
- For example, in many robot mobile-manipulation domains, the navigation-among-movable-obstacles (NAMO) problem constantly arise when the robot tries to pick-and-place (or push) an object to clear obstacles out of the swept volume. If we can specify sub-goals for this problem, then any planning problem that requires solving NAMO as a sub-problem would be more efficient.
- We do not make assumptions about where the sub-goals come from, but we require them to be such that achieving one sub-goal do not interfere with the feasibility of achieving the subsequent sub-goals, but it affects their optimality.

- Given our assumption on the sub-goals, one might naively try to solve the entire problem by solving sub-problems independently. However, not all feasible solutions are equal: some solutions are better than the others, and solving them independently and locally is likely to yield a feasible yet poor solution. Our goal is to produce a solution that is not only feasible, but globally optimal.

Planning algorithm

- We propose an anytime planning algorithm that, given a sequence of sub-goals and a reward function, computes a plan that optimizes the sum of the rewards along the trajectory for a deterministic planning environment with hybrid action space.
- Our algorithm is a Monte Carlo planning algorithm that updates the estimate of its value function based on roll-outs of trajectories.
- Traditionally, the Monte Carlo planning has been applied to Markov Decision Processes with discrete action spaces.
- There are some algorithms for continuous action spaces, such as HOOT and SOOP, which places a continuous-armed bandit agent, Hierarchical Optimistic Optimization (HOO), in the case of MDP, and black-box function optimizer, Simultaneous Optimistic Optimization (SOO), in the case of deterministic environment, at each node, much like how UCB agent is placed at each node for UCT to solve an MDP with discrete action space. They do not work for high dimensions.
- The performance of the optimization-agent at each node heavily influences the performance of the planning algorithm.
- We propose a new black-box function optimization algorithm that can work in high dimensions, and analyze the regret.
- To make use of the sub-goals, we also propose a root-switching technique that searches for the solution to the sub-goal from the root, but once found, switches the root to the resulting goal node, and work towards the next goal. Once we found a feasible solution, we switch the root back to the original root node in order to find a better solution, while keeping all of the past roll-out branches.

Format of the sub-goals.

- The sub-goals are specified using the predicates used in operator preconditions or the predicate used to define the goal. This is based on the observation that the preconditions naturally define the associated sub-planning problem that needs to be solved if the associated operator is to be used in the entire planning problem.

Using our algorithm

- Our algorithm can be used as a stand-alone algorithm, or can be integrated with a classical task-planner which will generate sub-goals for our planner; but exact usage is out of scope of this paper.

Experiments

- Our hypotheses are: (1) using sub-goals reduce the planning time (2) our algorithm can globally optimize the reward function, (3) Our algorithm is better than existing Monte Carlo planning algorithms, and (4) there exists larger problems that can benefit from sub-goal specification of recurring smaller sub-problems.
- For (1), compare my algorithm with and without root switching
- For (2), plot a reward curve vs time
- For (3), compare with SOOP, HOOT, and stripstream(?)
- For (4), show how packing and NAMO sub-goals are re-used in the bigger problem