Quantitative Semantics + DIRL

Mithilesh Vaidya

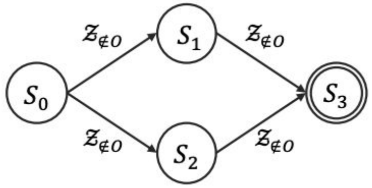
Based on: Jothimurugan, Kishor, et al. "Compositional reinforcement learning from logical specifications." Advances in Neural Information Processing Systems 34 (2021): 10026-10039.

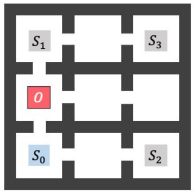
SpectRL language

- Specification: $\phi ::= \texttt{achieve}\ b \mid \phi_1\ \texttt{ensuring}\ b \mid \phi_1; \phi_2 \mid \phi_1\ \texttt{or}\ \phi_2$

- Maze task
  and
  Generated graph by SpectRL

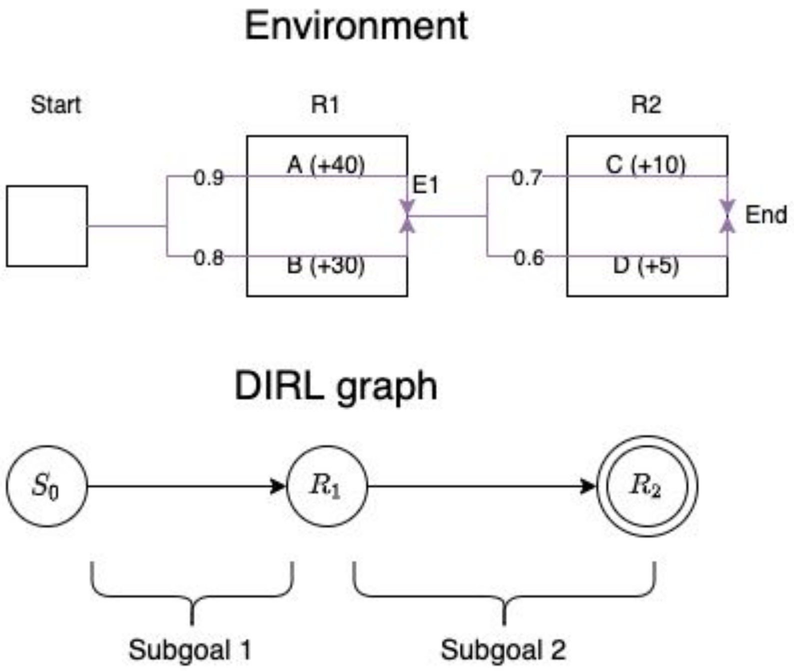
(achieve (reach S_1) or achieve (reach S_2);
achieve (reach S_3)) ensuring avoid O





- Policy learnt to maximize probability of satisfaction of ϕ by learnt trajectory
- **Key limitation: lack of quantitative semantics**
  Maximize P(success) AND minimize some function $f(s_0, s_1, \ldots, s_k)$

Example



- A(+x) denotes cost of taking the route via A
- Value on arrow denotes probability
- Inherent tension in **P(success)** and **Cost**

e.g. fast lane costs more but *P(reach)* is also higher

Note: User does not specify different subgoals for A and B (model needs to learn it)

Assumptions

Assumption 1: f(.) can be decomposed into a summation (or any known function) for each sub-goal i.e. for a task with *m* subgoals and a known function *g(.)*

$$f(s_1, s_2, \ldots, s_T) = g\,(f(s_{00/1}, \ldots, s_{k0}), f(s_{01}, s_{11}, \ldots, s_{k1}), \ldots, f(s_{0m}, s_{1m}, \ldots, s_{km/T}))$$

Assumption 2: f(.) shows some discreteness

Justification: Many real-world problems have inherent discreteness in cost

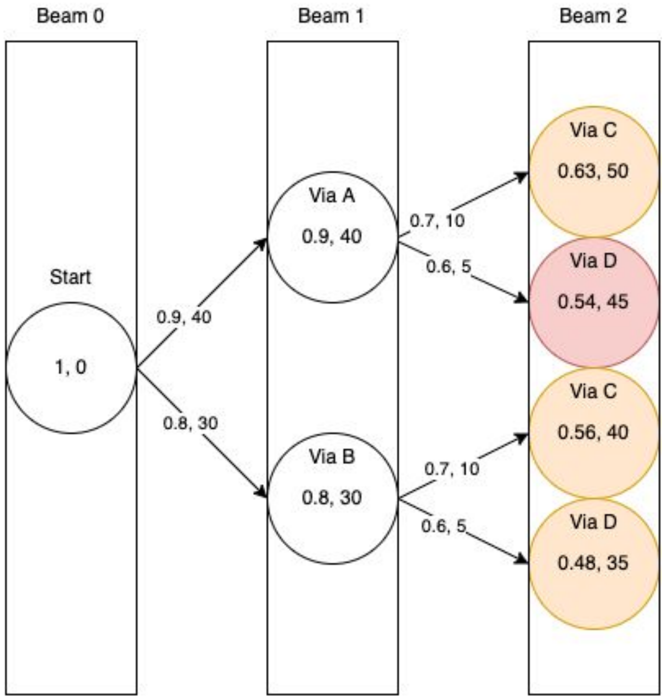
Approach

1. For **each** subgoal *m*, train K_m RL policies (one for each possible set of trajectories* and maintain: P(success) AND values of *f(.)* [K = 2 in eg]
2. **Cluster** the values of *f(.)* and use the mean as the representative.**
3. Use **beam search** to consider all possible candidates (scope for pruning)
4. Keep only *top-m candidates* depending on compute-performance tradeoff and proceed to next sub-goal

*k can be either determined automatically using clustering approaches or given as input by user if known beforehand

**If assumption holds and learnt RL policies indeed have distinct values of *f(.)*, the mean will serve as a good approximation (can also use min or other summaries)

Beam Search and Pruning



P gets multiplied, *f(.)* gets added
(could be any arbitrary aggregate)

Scope for pruning:

- Node 2 has lower *P* and higher *f(.)* than Node 3 → eliminate Node 2
- If given budget = 45, also eliminate node 1

Note: We don't need to maintain previous beams at each step since we have aggregated both *P* and *f(.)*

Key challenges

Challenge	Solution/Discussion
# nodes in beam may explode	● In the worst case, may have to explore all (just like SAT) → no way around it to guarantee optimality ● In practice, prune based on both: probability of success and cost
Learning distinct policies (one per cluster of *f*)	Let F_k be set of values of *f(.)* for policy *k:* ● F_k should have minimum variance (small intra-cluster distance) ● Sets F_k and F_l should be far from each other (large inter-cluster distance) Structure reward function accordingly to encourage exploration till we find a cluster and then exploit that cluster
Estimating K (number of RL policies/clusters of *f*) on-the-fly	● Threshold on inter-cluster distance distribution ● Elbow method in k-means

Thank you

Questions?