7 Equivalences of Optimization Criteria

The presented algorithms for MDPs with unavoidable dead ends are significantly more complicated than those for MDPs with unavoidable ones. Nonetheless, intuition tells us that for a given tuple \((S, A, T, C, G, D, s_0)\), solving it under the infinite-penalty criterion (i.e., as an iSSPUDE) should yield the same policy as solving it under the finite-penalty criterion (i.e., as an fSSPUDE) if in the latter case the penalty \(D\) is very large. This can be stated as a theorem:

**Theorem 7.** For iSSPUDE and fSSPUDE MDPs over the same domain, there exists the smallest finite penalty \(D_{\text{thres}}\) s.t. for all \(D > D_{\text{thres}}\) the set of optimal policies of fSSPUDE (with penalty \(D\)) is identical to the set of optimal policies of iSSPUDE.

*Proof sketch.* Although the full proof is technical, its main observation is simple — as \(D\) increases, it becomes such a large deterrent against hitting a dead end that any policy with a probability of reaching the goal lower than the optimal \(P^*\) starts having a higher expected cost of reaching the goal than policies optimal according to iSSPUDE's criterion.

As a corollary, if we choose \(D > D_{\text{thres}}\), we can be sure that at any given state \(s\), all optimal (\(J^*\)-greedy) policies of the resulting fSSPUDE will have the same probability of reaching the goal, and this probability is \(P^*(s)\) according to the infinite-penalty optimization criterion (and therefore will also have the same conditional expected cost \([J^*[P^*]]\))

This prompts a question: what can we say about the probability of reaching the goal \(J^*\)-greedy policies if we pick \(D \leq D_{\text{thres}}\)? Unfortunately, in this case different greedy policies may not only be suboptimal in terms of this probability, but even for a fixed \(D\) each may have a different, arbitrarily low chance of reaching the goal. For example, consider an MDP with three states, \(s_0\) (the initial state), \(d\) (a dead end), and \(g\) (a goal). Action \(a_d\) leads from \(s_0\) to \(d\) with probability 0.5 and to \(g\) with probability 0.5 and costs 1 unit. Action \(a_g\) leads from \(s_0\) to \(g\) also with probability 1, and costs 3 units. Finally, suppose we solve this MDP as an fSSPUDE with \(D = 4\). It is easy to see that both policies, \(\pi(s_0) = a_d\) and \(\pi(s_0) = a_g\), have the same expected cost, 3. However, the former reaches the goal with probability 0.5, while the latter always reaches it. The ultimate reason for this discrepancy is that the policy evaluation criterion of fSSPUDE is oblivious to policy's probability of reaching the goal, and optimizes for this parameter only indirectly, via policy's expected cost.

To summarize, we have two ways of finding an optimal policy in the infinite-penalty case, either by directly solving the corresponding iSSPUDE instance, or by choosing a sufficiently large \(D\) and solving the finite-penalty fSSPUDE MDP. We do not know of a principled way to choose \(D\), but it is typically easy to guess by inspecting the MDP. Thus, although the latter method gives no a-priori guarantees, it often yields a correct answer in practice.

8 Experimental Results

The objective of our experiments was to find out the most practically efficient way of finding the optimal policy in the presence of unavoidable dead ends and infinite penalty for visiting them, by solving an iSSPUDE MDP or an fSSPUDE MDP with a large \(D\). To make a fair comparison between these methods, we employ very similar algorithms to handle them. For both classes, the most efficient optimal solution methods are heuristic search techniques, so in our experiments we assume knowledge of the initial state and use only algorithms of this type.

To solve an fSSPUDE, we use the implementation of the LRTDP algorithm, an instance of the FIND-AND-REVISe heuristic search framework for SSPs, available in the miniGPT package [Bonet and Geffner, 2005]. As a source of admissible heuristic state costs/goal-probability values, we choose the maximum of atom-min-forward heuristic [Haslum and Geffner, 2000] and SixthSense [Kolobov et al., 2011]. The sole purpose of the latter is to soundly identify many of the dead ends and assign the value of \(D\) to them. (Identifying a state as a dead end may be nontrivial if the state has actions leading to other states.)

Since solving iSSPUDE involves tackling two MDPs, a MAXPROB and an SSP, to instantiate the SHS schema (Algorithm 2) we use two heuristic search algorithms. For the MAXPROB component, we use a specially adapted version [Kolobov et al., 2011] of miniGPT’s LRTDP, equipped with SixthSense (note that the atom-min-forward heuristic is cost-based and does not apply to MAXPROB MDPs). For the SSP component, we use miniGPT’s LRTDP, as for fSSPUDE, with atom-min-forward; SixthSense is unnecessary because SSP has no dead ends.

Our benchmarks were problems 1 through 6 of the Exploding Blocks World domain from IPPC-2008 [Bryce and Buffet, 2008] and problems 1 through 15 of the Drive domain from IPPC-06 [Buffet and Aberdeen, 2006]. Most problems in both domains have unavoidable dead ends. To set the \(D\) penalty for the fSSPUDE model, we examined each problem and tried to come up with an intuitive, easily justifiable value for it. For all problems, \(D = 500\) yielded a policy that was optimal under both the finite-penalty and infinite-penalty criterion.

Solving the fSSPUDE with \(D = 500\) and iSSPUDE versions of each problem with the above implementations yielded the same qualitative outcome on all benchmarks. In terms of speed, solving fSSPUDE was at least an order of magnitude faster than solving iSSPUDE. The difference in used memory was occasionally smaller, but only because both algorithms visited nearly the entire state space reachable from \(s_0\) on some problems. Moreover, in terms of memory as well as speed the difference between solving fSSPUDE and iSSPUDE was the largest (that is, solving iSSPUDE was comparatively the least efficient) when the given MDP had \(P^*(s) = 1\), i.e. the MDP had no dead ends at all or had only avoidable ones.