Planning and Acting in the Real World

Michael Rose

In which we see how more expressive representations and more interactive agent architectures lead to planners that are useful in the real world.

Planners that are used in the real world for planning and scheduling operations generally extend both the representation language and the way the planner interacts with the environment.

Time, Schedules, and Resources

This section covers methods for representing and solving planning problems that include temporal and resource constraints. The approach is to divide the overall problem into a planning phase in which actions are selected and a scheduling phase in which temporal information is added to the plan to ensure that it meets resource and deadline constraints.

11.1.2 | Solving Scheduling Problems

For a given problem, suppose we wish to minimize plan duration (makespan). We must find the earliest start times for all the action consistent with the ordering constraints supplied with the problem.

We can apply the **critical path method** to the graph to determine the possible start and end times of each action. A **path** through a graph representing a partial-order plan is a linearly ordered sequence of actions beginning with start and ending with finish.

The **critical path** is that path whose total duration is longest. The path is critical because it determines the duration of the entire plan - shortening other paths doesn't shorten the plan as a whole, but delaying the start of anny action on the critical path slows down the whole plan.

For actions that are not on the critical path, there is a window of time known as the **slack** which is the quantity $LS - ES$, where LS is the latest possible start time and ES is the earliest possible start time. Together the ES and LS times for all the actions constitute a **schedule** for the problem.

The complexity of the critical path algorithm is $O(Nb)$ where N is the number of actions and b is the maximum branching factor into or out of an action. Mathematically, critical path problems are easy to solve because they are defined as a conjunction of linear inequalities on the start and end times. When we add constraints, the conjunctions become disjunctions and out problem becomes NP-hard.

One simple but popular heuristic is the **minimum slack** algorithm: on each iteration, schedule for the earliest possible start whichever unscheduled action has all its predecessors scheduled and has the least slack. Then update the ES and LS times for each affected action and repeat. This resembles the minimum-remaining-values heuristic in constraint satisfaction.

11.2 | Hierarchical Planning

We can plan at a higher abstraction level, and then decompose each problem into subproblems.

We can define **angelic semantics** using the reachable set of a HLA. Given a state s, the reachable set for a HLA h, is the set of states reachable by any of the HLAs implementations. The key idea is that the agent can choose which element of the reachable set it ends up in when it executes the HLA; thus a HLA with multiple refinements is more powerful than the same HLA with fewer refinements.

A high level plan - a sequence of HLAs - achieves its goal if its reachable set intersects the set of goal states.

For **demonic semantics**, each member of the reachable set has to be a goal state.

11.3 | Planning and Acting in Nondeterministic Domains

As in search, we can extend planning to handle partially observable, nondeterministic, and unknown environments. The methods are similar:

- sensorless (or conformant) planning for environments with no observations

We can convert a sensorless planning problem to a belief state planning problem. This works as it did in chapter 4.4.1, the main differences are that the underlying physical transition model is represented by a collection of action schemas and the belief state can be represented by a logical formula instead of an explicitly enumerated set of states.

- contingency planning for partially observable and nondeterministic environments

Contingent planning is the generation of plans with conditional branching based on percepts. Given a mechanism for computing exact or approximate belief states, we can generate contingent plans with an extension of the AND-OR forward search over belief states.

- online planning and replanning for unknown environments

The online agent has a choice of how carefully to monitor the environment.

- action monitoring: before executing an action, the agent verifies that all the preconditions still hold.
- plan monitoring: before executing an action, the agent verifies that the remaining plan will still succeed.
- goal monitoring: before executing an action, the agent checks to see if there is a better set of goals it could be trying to achieve.

11.4 | Multiagent Planning

A multibody problem is still a standard single agent problem as long as the relevant sensor information collected by each body can be pooled - either centrally or within each body - to form a common estimate of the world state that then informs the execution of the overall plan. In this case, the multiple bodies act as a single body.

If communication isn't working, this becomes a **decentralized planning** problem. There are also some systems which are a mixture of centralized and decentralized planning.

11.5 | Summary

- many actions consume resources such as money, gas, or raw materials. It is convenient to treat these as numeric resources in a pool. Actions can generate and consume resources, and it is usually cheap and effective to check partial plans for satisfaction of resource constraints before attempting further refinements.
- Time is one of the most important resources. It can be handled by specialized scheduling algorithms, or scheduling can be integrated with planning.
- Hierarchical task network planning allows the agent to take advice from the domain designer in the form of high level actions that can be implemented in various ways by lower level action sequences. The effects of HLAs can be defined with angelic semantics, allowing provably correct high level plans to

be derived without consideration of lower-level implementations. HTN methods can create the very large plans required by many real world applications.

- Standard planning algorithms assume complete and correct information and deterministic, fully observable environments. Many domains violate this assumption.
- Contingent plans allow the agent to sense the world during execution to decide what branch of the plan to follow. In some cases, sensorless or conformant planning can be used to construct a plan that works without the need for perception. Both conformant and contingent plans can be constructed by search in the space of belief states. Efficient representation or computation of belief states is a key problem.
- An online planning agent uses execution monitoring and splices in repairs as needed to recover from unexpected situations, which can be due to nondeterministic actions, exogenous events, or incorrect models of the environment.
- Multiagent planning is necessary when there are other agents in the environment with which to cooperate or compete. Joint plans can be constructed, but must be augmented with some form of coordination if two agents are to agree on which joint plan to execute.
- This chapter extends classic planning to cover nondeterministic environments. Chapter 17 describes techniques for stochastic environments: Markov decision processes, partially observable markov decision processes, and game theory. Chapter 21 shows that reinforcement learning allows an agent to learn how to behave from past successes and failures.