Probabilistic Integration of Planning, Action, and Perception

Pat Langley (LANGLEY@FLAMINGO.STANFORD.EDU)
Robotics Laboratory, Computer Science Department
Stanford University, Stanford, CA 94305 USA

Introduction

An intelligent agent must integrate three central components of behavior – planning, action, and perception. Different researchers have explored alternative strategies for interleaving these processes, typically assuming that their approach is desirable in all domains. In contrast, we believe that different domains require different interleaving schemes. In this paper we identify three continua along which these strategies can vary, show how one can represent each spectrum in terms of probabilistic information, and outline how an agent might learn the best position along each continuum through experience with a particular domain.

We are exploring these issues in the context of ICARUS, an integrated architecture for controlling physical agents (Langley et al., 1991). The framework's central data structure is the grounded plan, which it represents as a sequence of segments that correspond to qualitative 'states'. That is, each segment specifies an interval of time – a continuous sequence of situations – during which the signs of a set of observed derivatives are constant; thus, state boundaries between segments occur between pairs of situations in which the sign of one or more derivatives change. A segment also specifies a set of primitive actions that should occur while the segment is active. A problem is simply an abstract plan that has not yet been completely grounded.

Each segment S in a plan specifies a set of `start` conditions (the situation at the beginning of S), a set of `while` conditions that must hold for each situation in the segment, and a set of `until` conditions (the situation at the end of S). In summary, a plan consists of a sequence of `repeat-until` statements, augmented by `while` constructs that detect violations of constraints on the segments. Unlike most work in qualitative physics, these plan segments can also contain numeric information about the positions and velocities of agent parts and manipulated objects at the transitions between segments, the rates of change for these variables within each segment, and the time steps required for each segment.

Embedding Action in Planning

One central issue in controlling an integrated agent concerns when to initiate execution of a plan. In principle, one can start to execute a plan as soon as its initial steps have been grounded in primitive motor actions, even though later parts of the plan remain abstract. In some domains, this scheme works because one can often find a reasonable way to generate the remaining plan steps after execution has begun. In other domains, this scheme can lead to failure to achieve one's goal, making it preferable to delay execution until the entire plan has been grounded in primitive actions.

We propose to represent the continuum between these two extremes by storing the conditional probability that, given an initial portion of a plan has been successfully grounded, the remaining portion can also be grounded. Upon reaching a planned state with sufficiently high probability, the agent initiates execution on the assumption that it can successfully ground the remaining steps (and thus execute them to achieve the goal) when necessary. In everyday terms, the agent decides to cross that bridge when it comes to it.[1] One can also store the expected cost of plan failures and take this information into account in deciding when to execute.

We intend to use a simple learning mechanism to acquire such probabilities from experience. The agent starts by assuming the chances of grounding each step in a plan P are independent and equal. Thus, given an initial success probability of 0.8 for each step, a four-step plan would appear to have a probability of $0.8^3 = 0.512$ after the first step has been grounded, a probability of $0.8^2 = 0.64$ after two successful steps, and a probability of 0.8 when only the last step remains to be specified.

However, each time the agent attempts to ground the remainder of the plan P, it updates the statistics on the conditional probabilities of success. In some cases, the score associated with a plan step will gradually increase until it approaches one, leading the agent to begin execution when it grounds this step in primitive actions. For other steps, the probabilities will remain low or even decrease, causing the agent to remain conservative or to become even more cautious than it was at the outset. The result of learning will depend on the characteristics of the environment in which the agent finds itself.

Perceptual Attention during Execution

A second issue involves when to sense the environment during plan execution. Most recent work has assumed that an agent should be purely reactive, in that it senses it surroundings on every time step. However, humans appear able to execute motor skills in either reactive (closed-loop) or automatic (open-loop) mode. In the latter, they carry out skills without feedback about

[1] Bresina (personal communication, 1990) has proposed similar ideas for interleaving planning and execution.

the effects of their actions to ensure proper execution. Langley, Iba, and Shrager (1994) present an analysis of reactive and automatic behavior which shows that, if one takes the cost of sensing and the probability of errors into account, automatic execution is clearly preferable for some domains.

As Iba and Langley (1987) have noted, closed-loop and open-loop behavior are merely extreme points along a continuum. An agent can take different positions on this continuum by varying the frequency with which it compares situations in the environment with `while` and `until` conditions. This is turn can be determined by information stored on plan segments about the probability of external interruptions, the variation in timing, and the cost of sensing. Briefly, segments that have a high probability of completion without error, that run for predictable lengths of time, and that have high sensing costs are likely candidates for automatic execution. One can also reduce execution costs by selectively attending to features based on statistical information.

For a given plan, one should be able to determine the appropriate position along this spectrum through learning. Suppose the agent has retrieved or generated a plan, but that this structure contains no information about the number of time steps required for execution or the probability of successful execution. Practice occurs in closed-loop, reactive mode, since this provides the feedback necessary to determine when errors occur and when a segment is complete. On every practice run, the agent updates its statistics about the time taken for each segment, the probability that the segment completed successfully without intervention, and the cost of sensing.

For some segments, external factors commonly interfere with successful completion, so the associated probability is low. In such cases, the agent never moves beyond closed-loop mode, since open-loop processing does not provide the sensory information needed to detect and recover from errors or interference. For other segments, such interference is rare, giving a high probability of successful completion, but the variance in timing is high. Again the agent must remain in closed-loop mode, in this case to detect completion. However, in some segments the probability of completion is high and the timing variance is low. For such components of a plan, the agent will gradually come to automatically execute the primitive actions the estimated number of time steps, without stopping to check the `while` or `until` conditions, monitoring execution infrequently or not at all, and thus freeing perceptual resources for other tasks.

Perceptual Interruptions to Planning

A final issue concerns the impact of perception on the planning process. During sensing, an agent may recognize a situation that suggests a new problem, which it then passes on to the planning module. We posit the existence of problem-generating triggers or reflexes that take the same form as stored plans, but that match against environmental states rather than state-goal pairs. In such situations, the agent can ignore the new problem or attend to it, either abandoning its previous task or putting it on hold. The outcome of this decision depends on the relative importance of the two tasks and the expected cost of interruption and recovery.

To resolve such conflicts, we propose that the architecture associate priorities with every state in memory. On each cycle, the agent attends to the problem with the final state having the highest priority, using an agenda to focus its problem-solving attention. Rather than using an explicit depth-first search regimen, planning passes control from problem to subproblem through propagated priority scores, causing the parent to become suspended until the child has been solved or abandoned. At the same time, the perceptual process continually adds new problems to the agenda, based on stored plans whose initial state matches the environmental state. If the retrieved problem has higher priority than the currently active one, the agent suspends the latter and pursues the more urgent goal. Once this task has been handled, control passes back to the original problem, unless another one has taken over in the meantime.

The agent needs some means to estimate the priority distributions associated with each stored plan. For this, we propose that it collect statistics about the observed desirability of the final state for each plan once it has been achieved. We assume that certain states have innate priorities associated with them, but that most are initially neutral. However, states that occur along the path to a state with an innate priority come to acquire their own priority distributions through the propagation that occurs during planning. This gives an effect similar to that in reinforcement learning; over time, the expected score for a state that occurs early along a path will approximate the innate score for the final state.

Conclusion

In summary, our design for an integrated architecture gives a central role to probabilistic information about the probability of successfully completing plans, the probability of executing them without errors, and the relative importance of the final states. Different environments reflect different distributions, which in turn recommend different strategies for interleaving of perception, action, and planning. However, simple learning algorithms should let an agent infer this distributional information from experience, and thus approach the optimal strategy as it becomes familiar with a domain. We are implementing these ideas in the ICARUS architecture, and we plan to evaluate them in domains that require an integrated agent, such as the control of a flight simulator.

References

Iba,W., & Langley,P. (1987). A computational theory of motor learning. *Computational Intelligence,3*,338–350.

Langley, P., McKusick, K. B., Allen, J. A., Iba, W. F., & Thompson, K. (1991). A design for the ICARUS architecture. *SIGART Bulletin, 2*, 104–109.

Langley, P., Iba, W., & Shrager, J. (1994). *Reactive and automatic behavior in plan execution*. Unpublished manuscript, Robotics Laboratory, Stanford University, Stanford, CA.