Integrating top-down planning with a bottom-up approach that learns to locally combine actions

Blazej Bulka and **Marie desJardins**
Department of Computer Science
University of Maryland Baltimore County
{bulka1, mariedj}@cs.umbc.edu

Abstract

This paper describes two open problems in planning and presents preliminary ideas that could lead to potential solutions. The problems on which the paper focuses are (1) combining top-down and bottom-up reasoning to create hierarchical plans, and (2) representing domain knowledge in domains that are unpredictable or too complex to described or reasoned about precisely. The first problem is particularly relevant for plan repair and replanning. The second is relevant in situations with environmental unpredictability or imperfect information about the world.

Combining top-down and bottom-up approaches

Organizing tasks into a hierarchical structure is a widely used approach in planning. It reduces the complexity of the planning problem by hiding irrelevant details at a time, and allows the reuse of tasks or solutions (*sub-task sharing*). The most widely used planning approach resulting in hierarchical plans is HTN planning (Sacerdoti 1975). It starts with the most abstract tasks and refines them by decomposing them with successively more primitive tasks, building a hierarchical task network from the top downwards. It is supposed to reflect the high-level reasoning of humans, who often have abstract goals, and search for actions that can potentially satisfy them. Moreover, the search for decompositions may be deferred for some parts of the plan, avoiding commitments before they are necessary, and allowing easier replanning later. Unfortunately, when a planner is faced with multiple possible decompositions of abstract tasks, unforeseen conflicts among decomposed tasks (e.g., due to different times of deferred decompositions) may be too computationally expensive, especially if an optimal plan is sought.

In practice, humans often exhibit a different approach to planning. When faced with a situation (possibly an unexpected one), they start thinking in terms of categories of primitive actions that are currently available to them, trying to predict what kind of more abstract results they can achieve. They may have a potential high-level goal in mind, but instead of decomposing the goal, they start aggregating available options to form higher-level intermediate goals, in the hope that they will find a solution compatible with the initial goal. This resembles building a hierarchical task network from the bottom up, or from the "inside out." Such behavior is also found in reactive planning and during plan repair, since this bottom-up approach allows a quicker response time by the planner, and can "fill gaps" created by the unforeseen events that triggered the plan repair procedure. Unfortunately, the reactive approach may create suboptimal plans, and can be more computationally intensive, if many intermediate levels have to be created by aggregating lower-level tasks.

The goal of our current research is to develop hybrid methods that combine the two approaches. Key issues include determining where the boundary between the two approaches should be and merging the two (or more) parts of the task network.

Representation and manipulation of knowledge

An important aspect of most planning approaches is how planner's knowledge (operators, their effects, preconditions and constraints) is represented. The more expressive the representation of this knowledge, the wider the range of planning domains that can be expressed. Unfortunately, an increase in expressiveness also leads to an increased cost of planning. The level of detail required to plan successfully for many real-world domains often leads to intractability. Examples of knowledge that are often difficult to represent include space (e.g., spatial limitations or effects of actions), time, amounts (e.g., money), and complex processes. Especially in the last case, the description of an action may not be tractable when an action is atomic from the point of view of the planner but still contains an internal process with intricate consequences. An example may be an action of applying for a construction permit, where the processes occurring at the issuing agency are complex and not controllable by the planner. (Sometimes an explicit description of such an action would border on the description of the action's algorithm in pseudocode.)

An alternative approach would be to decrease the expressiveness of planner's knowledge, acknowledging that it will never be able to exactly predict the action outcomes (e.g., due to either insufficient specification of the action or the inherent uncertainty of the outcome). The planner would have then to learn when an action can be used the most effectively. A similar idea was proposed by Gil (1994).

It would be interesting to push this approach to the extreme by providing the planner with an *opaque* representation of knowledge that cannot be analyzed (a *black-box approach*). Such *opaque knowledge* could be contained within pieces of executable code that can behave in an "agent-like" manner. That is, the code could be placed in an environment that simulates a possible state of the world before the action is applied. By executing an appropriate method of the code, it would respond whether preconditions are satisfied, and what the potential states of the world after the application of the action are. Moreover, it is possible to envision pieces of code interacting with each other, trying to determine the potential compatibility of the actions they represent (e.g., whether one action can satisfy a precondition of another without introducing a conflict). This would allow the planner (or the pieces of code themselves) to learn about their mutual effects (an approach similar to *Generic Learning Modules* (Levinson 1996)). Additionally, this approach allows reasoning about complex actions that involve internal processing and their outcomes that are not entirely predictable.

Given the agent-like behavior of these actions, or pieces of code, they could exhibit autonomy when interacting with other actions, spontaneously aggregating into groups and generating the bottom-up behavior described earlier (i.e., the agents would reason about what they can do together given a state of the world). Moreover, some of the actions could represent abstract actions that would observe the current aggregations of more primitive actions (another analogy to the idea of *Generic Learning Modules*, which included reinforcement hierarchies). Whenever the aggregation satisfies the conditions of the abstract action, they would ascertain that the abstract action is feasible in current conditions. Our goal is to develop a planning framework in which this "emergent" approach to creating a hierarchy of tasks could be integrated with more abstract, classical top-down planning.

Learning interactions The previously presented idea of opaque representation of actions, which analyze the state of the world before their execution and produce a derived state of the world after their execution, is closely related to state space search. Such a search may be very expensive without an efficient heuristic. One approach would be to allow the agents to learn an efficient heuristic. We hope to build on ongoing research in learning appropriate heuristics based on previous experiences (Likhachev & Koenig 2005). Another approach would be for agents to learn which actions can be applied together given the current situation. This approach resembles case-based reasoning, in which the current situation is compared with the previously seen ones, and if there is sufficient similarity, the previous solution is adjusted appropriately (Leake 1996).

Discussion

The ideas presented above may lead to improvements in many circumstances. Integration of top-down and bottom-up approaches may lead to efficient action in environments with uncertainty, while lacking the disadvantage of purely reactive planning. Moreover, aggregation and reorganization of actions need not stop after the parts of the plan grown from the bottom meet the part of the plan decomposed from the top, since continued planning may lead to an improved plan. However, once the two parts of the plan meet, the planner is ready to provide a solution to the planning problem at any time, which may be useful for time constrained or "anytime" reasoning.

We hope that the presented solution will be able to generate plans with alternative paths of actions to satisfy a goal (or a high-level abstract action), which will improve the performance of replanning. The learning of interactions among actions will also produce faster responses because agents will not analyze the situation fully; rather, they will follow a plan that was previously successful (similar to a *playbook approach* (Bowling, Browning, & Veloso 2004)). Additionally, the multi-agent nature of the planning process may also ease the distribution of planning.

Conclusion

The ideas presented in this position paper are in a very early stage of development. Our intention is to indicate the existence of interesting problems in this area, as well as some potentially novel ways to solve them. The "agent-based" approach is essentially a distributed planning approach. However, it differs from previous approaches in that it takes neither a top-down nor a bottom-up perspective, but provides a flexible framework in which the strengths of both these approaches can be combined. The idea of moving away from explicit representations is a radical one, but may lead to novel planning methods that are applicable to complex domains that do not readily lend themselves to traditional representations.

References

Bowling, M.; Browning, B.; and Veloso, M. 2004. Plays as effective multiagent plans enabling opponent-adaptive play selection. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS-04)*.

Gil, Y. 1994. Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML-94)*.

Leake, D. 1996. *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press/MIT Press.

Levinson, R. 1996. General Game-Playing and Reinforcement Learning. *Computational Intelligence* 12(1):155 – 176.

Likhachev, M., and Koenig, S. 2005. A Generalized Framework for Lifelong Planning A*. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-05)*.

Sacerdoti, E. 1975. The Nonlinear Nature of Plans. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 206 – 214.