Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)

The Model-based Approach to Autonomous Behavior: A Personal View

Hector Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, SPAIN
hector.geffner@upf.edu

Abstract

The selection of the action to do next is one of the central problems faced by autonomous agents. In AI, three approaches have been used to address this problem: the *programming-based approach*, where the agent controller is given by the programmer, the *learning-based approach*, where the controller is induced from experience via a learning algorithm, and the *model-based approach*, where the controller is derived from a model of the problem. Planning in AI is best conceived as the model-based approach to action selection. The models represent the initial situation, actions, sensors, and goals. The main challenge in planning is computational, as all the models, whether accommodating feedback and uncertainty or not, are intractable in the worst case. In this article, I review some of the models considered in current planning research, the progress achieved in solving these models, and some of the open problems.

Approaches to Autonomous Behavior

At the center of the problem of intelligent behavior is the *problem of selecting the action to do next.* In AI, three different approaches have been used to address this problem. In the *programming-based approach*, the controller that prescribes the action to do next is given by the programmer, usually in a suitable high-level language. In this approach, the problem is solved by the programmer in his head, and the solution is expressed as a high-level program. In the *learning-based approach*, the controller is not given by a programmer but is induced from experience: the agent's own experience, in reinforcement learning, or the experience of a 'teacher', in supervised learning schemes. Finally, in the *model-based approach*, the controller is not learned from experience but is derived automatically from a model of the actions, sensors, and goals. The controller is the solution to the model.

The three approaches to the action selection problem are not orthogonal, and exhibit different virtues and limitations. Programming agents by hand, puts all the burden on the programmer that can't anticipate all possible contingencies, and often result in systems that are brittle. Learning methods have the greatest promise and potential, but tend to be limited in scope. In particular, reinforcement learning methods do not provide a principled solution to the problem of learning when the state of the system is not observable. Last, model-based methods, require a model of the actions, sensors, and goals, and face the computational problem of solving the model; a problem that is computationally intractable even in the simplest case, where information is complete and actions have deterministic effects.

While planning is often defined as the branch of AI concerned with the "synthesis of plans of action to achieve goals", planning is best conceived as *the model-based approach to action selection*, a view that defines more clearly the role of planning in intelligent autonomous systems. Programming-based approaches, on the other hand, correspond to systems whose control is hardwired and thus less flexible. Learning or *model-free* approaches, have the greatest flexibility, but also the most challenging computational problem. Moreover, this flexibility is often be result of learning and using a model, and thus do not dispense with the need for effective model-based methods. The distinction that Dennett (1996) makes between 'Darwinian', 'Skinnerian', and 'Popperian' creatures, mirrors quite closely the distinction between hardwired (programmed) agents, agents that learn, and agents that use models respectively. The contrast between the first and the latter corresponds also to the distinction made in AI between reactive and deliberative systems, as long as deliberation is not equated to logical reasoning. Indeed, as we will see, the inferences captured by model-based methods that scale up are not logical but heuristic, and follow from approximations.

Planning Models

A wide range of models used in planning can be understood as variations of a *basic state model* featuring:[1]

- a finite and discrete state space S,
- a *known initial state* $s_0 \in S$,
- a set $S_G \subseteq S$ of goal states,
- actions $A(s) \subseteq A$ applicable in each $s \in S$,
- a *deterministic* transition function $f(a, s)$, and
- positive *action costs* $c(a, s)$.

[1] For lack of space, I omit references; see (Russell and Norvig 2010).

This is the model underlying *classical planning* where it also assumed that action costs $c(a, s)$ do not depend on the state, and hence $c(a, s) = c(a)$. A solution or *plan* in this model is a sequence of applicable actions that map the initial state into a goal state. The cost of a plan is the sum of the action costs, and a plan is optimal if it has minimum cost.

Domain-independent classical planners accept a compact description of models of this form, in languages such as Strips, ADL, or PDDL, and automatically produce a solution (plan) as the output. A classical plan $\pi = a_0, \ldots, a_n$ represents an *open-loop controller* where the action to be done at time step i depends just on the step index i. The solution of models that accommodate uncertainty and feedback, on the other hand, produce *closed-loop controllers* where the action to be done at step i depends on the actions and observations collected up to that point. These models can be obtained by relaxing some of the assumptions in the model above.

A *Markov Decision Process (MDP)*, for example, is a state model where the state transition function $f(a, s)$ is replaced by state transition *probabilities* $P_a(s'|s)$, and the next state s', while no longer predictable with certainty, is assumed to be *fully observable.* A solution to an MDP must drive the system state towards a goal state with certainty, and its form is a function or *policy* π that maps states into actions. The cost of a policy is the expected cost to go from the initial state to the goal.

A *Partially Observable MDP (POMDP)* further extends MDPs by relaxing the assumptions that states are fully observable. In a POMDP, a set of observation tokens $o \in O$ is assumed along with a *sensor model* $P_a(o|s)$ that relates the true but hidden state s of the system with the observable token o. In POMDPs, the initial state of the system is not known, but is characterized by a probability distribution, and the task is to drive the system to a final, fully observable target state. Solutions to POMDPs are closed-loop controllers of a different form that map *belief states* into actions, with optimal solutions reaching the target state with minimum expected costs. Belief states are probability distributions over the states.

While MDPs and POMDPs are commonly described using positive or negative *rewards* rather than positive *costs*, and using *discount factors* rather than *goal states*, simple transformations are known for translating discounted reward MDPs and POMDPs into equivalent *goal MDPs* and *goal POMDPs* as above, that are strictly more expressive (Bertsekas 1995; Bonet and Geffner 2009).

MDP and POMDP models are also useful when the uncertainty in the system dynamics or in the feedback is represented by means of *sets* rather than *probability distributions*. Planning with a dynamics and feedback represented in this qualitative manner is called *contingent planning*. Solutions to these models can be represented in many ways, including functions of sets of states into actions, contingent plans, sets of rules, and finite-state machines. *Conformant planning* is a special case of contingent planning where there is no sensing, and a plan, that must achieve the goal for every possible initial state and transition, is like in classical planning, an action sequence.

The languages used for representing conformant, contingent, MDP, and POMDP planning problems in compact form, are minor variations of the languages used for representing classical planning problems, in particular PDDL.

Status

Since all the models considered in planning are intractable in the worst case, the main challenge in planning is *computational:* how to scale up to large and complex problems. In this sense, the research agenda in planning is not too different from the agenda concerning other intractable models in AI such as SAT, CSPs, and Bayesian Networks. In all cases, blind methods don't work, and for domain-independent solvers to scale up, they must automatically *recognize* and *exploit* the structure of the problems by some form of inference, for guiding or pruning the search for solutions.

Classical Planning

The good news in planning over the last decade, is that planning over the simplest types of models, *classical planning*, works. Namely, solvers accept problems involving hundreds of actions and variables, and produce good solutions quickly.[2] The sheer size of a problem is no longer an impediment to its solution. This progress is the result of a renewed emphasis on experimentation, and new ideas (Blum and Furst 1995; Kautz and Selman 1996; McDermott 1996; Bonet, Loerincs, and Geffner 1997). State-of-the-art classical planners manage to solve large problems using heuristic functions that are derived automatically from the problem encodings (Richter, Helmert, and Westphal 2008).

The model underlying classical planning is simple, but as we will see below, quite flexible too. Actions in planning can be activities or policies of any sort that can be characterized deterministically in terms of pre and postconditions. While non-deterministic effects are not represented, they can often be handled in a natural way. Some of the best planners in the MDP competitions held so far, for example, are not MDP solvers, but classical planners that ignore all but one of the possible outcomes, and replan from scratch when the system is observed off its expected trajectory (Geffner 2002; Yoon, Fern, and Givan 2007).

Beyond Classical Planning

For dealing with non-classical planning models, two types of approaches have been pursued: a *top-down* approach, where *native solvers* are developed for more expressive models, and a *bottom-up* approach, where the power of classical planners is exploited by means of suitable *translations.*

MDP and POMDP planners are examples of native solvers for more expressive models. Interestingly, recent MDP and POMDP planners make heavy use of heuristic search methods as well, and progress on the solution of both

[2] My focus throughout is on satisficing planning, not optimal planning. Satisficing planners search for solutions that are good but not necessarily optimal.

types of models has been significant in recent years. A limitation of these planners in comparison with classical planners, however, is that *inference* is performed at the level of *states* and *belief states*, rather than at the level of *variables*, taking the form of *value function updates*.

Translation-based approaches address features that are absent from the classical model such as *soft-goals*, *plan-constraints*, *uncertainty*, and *partial feedback*, by compiling them away. For example, *soft-goals* represent formulas that if achieved along with the goal, entail a positive utility. The task is to compute plans that maximize overall utility, defined as the sum of the utilities achieved minus the plan cost. Soft-goals, thus express preferences, as opposed to hard goals. Interestingly, however, soft-goals can be compiled away easily and efficiently resulting in standard classical planning problems (Keyder and Geffner 2009). Likewise, *plan constraints* expressible as LTL formulas, such as "if a tool is used, it must eventually be returned", that appear in recent versions of PDDL, can be compiled away effectively too: the LTL formula is converted into a Buchi automata that is then merged with the planning domain (De Giacomo and Vardi 2000).

It has also been shown recently that *uncertain information* can be compiled away too in conformant problems with deterministic actions (Palacios and Geffner 2009). The translation maps a conformant problem P into a classical problem $K(P)$, whose solutions, computable with an off-the-shelf classical planner, are in correspondence. The complexity of the translation is exponential in a width parameter that is bounded and equal to 1 over most of the benchmarks. The ideas behind this translation have been used since to define an effective action selection mechanism for on-line planning in the presence of *partial observability* (Albore, Palacios, and Geffner 2009), and for computing solutions to planning problems with uncertainty and partial feedback, in the form of *finite-state controllers* (Bonet, Palacios, and Geffner 2009).

Challenges

In spite of the progress achieved, some open challenges remain for making the model-based approach practical in the design and analysis of autonomous systems. A choice of challenges and opinions follow.

Classical planning. The performance increase in classical planning over the last decade is less less the result of the new heuristics, than other forms of *inference,* such as *helpful actions* (actions deemed as directly relevant to the goal) and *landmarks* (fluents that must be achieved in the way to the goal), and new *search algorithms*, like greedy local searches for finding states with lower heuristic values (enforced hill climbing), and best-first searches with multiple queues (Hoffmann and Nebel 2001; Hoffmann, Porteous, and Sebastia 2004; Helmert 2006). One way to understand the difference between recent planning algorithms and traditional heuristic search algorithms, is that the latter do not aim to exploit the structure of states and heuristic values that are regarded as 'black boxes'. Ideas such as helpful actions and landmarks, on the other hand, are the result of looking at the fine propositional structure. I believe that *further progress in classical planning will depend more on new methods for exploiting this structure, than on more powerful heuristic estimators or search algorithms.*

Probabilistic planning. As argued above, scalable MDP and POMDP solvers will have to reason, not at the level of *states* and *belief states*, but at the level of *variables*.[3] MDP planners that use heuristic values derived from the propositional, factored representation of MDPs, like MDP planners based on classical replanners, are moves in that direction. The challenge is achieve both scalability and quality, and furthermore, to do this in the more difficult setting of POMDPs, where current heuristic estimators are not informative and replanning approaches do not apply (as it can't be determined then when to replan and from which state). Recent translation-based approaches to conformant and contingent planning are moves in that direction too, as the translations and the inferences required to solve the translations are carried out over propositional representations. A limitation of these approaches is that, by not taking probabilities into account, they are left with the task of minimizing cost in the worst case (optimally or heuristically), which is ill-defined in many domains of interest. Thus either, probabilities must be taken into account, or meaningful solution forms that do not demand bounded cost in the worst case, are needed (Daniele, Traverso, and Vardi 1999). I close this point by noting that there are very few domain-independent solvers able to represent and solve a problem such as the Wumpus (Russell and Norvig 2010).[4] A concrete challenge in this area is *to have a domain-independent solver that can solve the Wumpus and scale up to large instances.*

Learning. There are many roles for learning in model-based approaches, the first of which is learning the model itself from experience and partial observations. Learning, however, has also a role to play in the search for solutions; a role that has been crucial in the context of SAT, but has not been exploited yet in the context of planning. For example, consider an agent that has to deliver a large package to one of two cells A or B in a grid, by going to the cell and dropping the package. Furthermore, assume that A is closer to the agent than B but that A cannot be entered while holding a large package. Most current heuristics will drive the search toward A in a way resembling a fly that wants to get across a closed window. Unlike flies, however, search algorithms avoid revisiting the same states, and eventually would solve the problem after partially exhausting the space around A. A more intelligent strategy would be to note that the failed search around A is the result of an interaction ignored by the heuristic that should be fixed. This is precisely what SAT solvers do: they identify the causes for failure (backtracks) and fix them, while searching. Traditional heuristic methods cannot replicate this behavior because they ignore the structure of the heuristic function. Yet this structure is avail-

[3] Terms like 'factored MDPs' and 'factored POMDPs' refer to representations built on top variables, not to solution methods that work at the level of those variables.

[4] The only such solver that I'm aware of is a POMDP solver, GPT, that doesn't scale up to large instances.

able to heuristic search methods in planning, and they should eventually be able to exploit it. The challenge is *to account for an effective form of 'learning from backtracks' as in SAT, without having to use either CNF or CSP encodings* that are not suitable for planning over long horizons.

Hierarchies. Hierarchies form a basic component of HTNs, an alternative model for planning that is concerned with the encoding of the *strategies for solving problems.* Hierarchies, however, play no role in state-of-the-art domain independent planners that are completely flat. Yet, it is clear that most real plans are understood as involving low and high level actions. For example the action of picking up a block involves displacements of the gripper that must be opened and closed on the right block. A basic question that has not been fully answered yet is how these abstractions can be formed automatically, and how they are to be used to speed up the planning process. For instance, the standard blocks world is an abstraction of a problem where blocks are at certain locations, and the grid move between locations. This abstraction, however, is not adequate when the table has no space for all the blocks, or when the gripper cannot get past towers of a certain height. The open question is *how to automatically compile detailed planning description into more abstract ones that can be used to solve the original problem more effectively.* There is a large body of work on abstract problem solving that is relevant to this question, the latest of which is (Marthi, Russell, and Wolfe 2007), but no robust answer yet applicable to a wide range of problems.

Multi-agent planning. I've discussed planning models involving single agents, yet often autonomous agent, must interact with other autonomous agents. We do this naturally all the time: walking on the street, driving, etc. The first question is how plans and plan costs should be defined in such setting. This is a subtle problem and many proposals have been put forward, often building on equilibria notions from game theory. Two elements, however, are still missing from this body of work. First, *it should be possible to define a hierarchy of multi-agent models, starting with a simple 'classical model', as in single-agent planning.* For example, a model where each agent has a 'classical planning problem', knows the problems of the other agents, and at each time point knows what each agent has done so far. More interesting problems, accommodating limited forms of collaboration could then be defined by playing with the cost structure of the problems. Second, by suitable translations, *it should be possible for each agent to plan using state-of-the-art single-agent planners*. Moreover, the goals of the other agents could be hidden to the other agents, that could infer them from the observations gathered using plan recognition techniques, which as shown recently, can leverage on classical planning technology as well (Ramirez and Geffner 2010).

I think that these are all meaningful computational problems where significant progress can be achieved over the next few years. If you are looking for challenges in AI, consider joining the effort.

Acknowledgments. H. Geffner is partially supported by grant TIN2009-10232, MICINN, Spain.

References

Albore, A.; Palacios, H.; and Geffner, H. 2009. A translation-based approach to contingent planning. In *Proc. IJCAI-09*.

Bertsekas, D. 1995. *Dynamic Programming and Optimal Control, Vols 1 and 2.* Athena Scientific.

Blum, A., and Furst, M. 1995. Fast planning through planning graph analysis. In *Proceedings of IJCAI-95*.

Bonet, B., and Geffner, H. 2009. Solving POMDPs: RTDP-Bel vs. Point-based Algorithms. In *Proc. IJCAI-09*.

Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A robust and fast action selection mechanism for planning. In *Proceedings of AAAI-97*, 714–719. MIT Press.

Bonet, B.; Palacios, H.; and Geffner, H. 2009. Automatic derivation of memoryless policies and finite-state controllers using classical planners. In *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS-09)*.

Daniele, M.; Traverso, P.; and Vardi, M. Y. 1999. Strong cyclic planning revisited. In *Proceedings of ECP-99*.

De Giacomo, G., and Vardi, M. 2000. Automata-theoretic approach to planning for temporally extended goals. *Lecture notes in computer science* 226–238.

Dennett, D. 1996. *Kinds of minds*. Basic Books New York.

Geffner, H. 2002. Perspectives on artificial intelligence planning. In *Proc. AAAI-2002*.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *JAIR* 22(1):215–278.

Kautz, H., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proc. AAAI*, 1194–1201.

Keyder, E., and Geffner, H. 2009. Soft goals can be compiled away. *JAIR* 36:547–556.

Marthi, B.; Russell, S.; and Wolfe, J. 2007. Angelic semantics for high-level actions. In *Proc. ICAPS-07*.

McDermott, D. 1996. A heuristic estimator for means-ends analysis in planning. In *Proc. Third Int. Conf. on AI Planning Systems (AIPS-96)*.

Palacios, H., and Geffner, H. 2009. Compiling Uncertainty Away in Conformant Planning Problems with Bounded Width. *JAIR* 35:623–675.

Ramirez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proc. AAAI-2010*.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *Proc. AAAI*, 975–982.

Russell, S., and Norvig, P. 2010. *Artificial Intelligence: A Modern Approach*. Prentice Hall. 3rd Edition.

Yoon, S.; Fern, A.; and Givan, R. 2007. FF-replan: A baseline for probabilistic planning. In *Proc. ICAPS-07*.