Search and Inference in AI Planning

Héctor Geffner

ICREA & Universitat Pompeu Fabra
Paseo de Circunvalacion 8
08003 Barcelona, Spain
hector.geffner@upf.edu

Abstract. While Planning has been a key area in Artificial Intelligence since its beginnings, significant changes have occurred in the last decade as a result of new ideas and a more established empirical methodology. In this invited talk, I will focus on Optimal Planning where these new ideas can be understood along two dimensions: branching and pruning. Both heuristic search planners, and SAT and CSP planners can be understood in this way, with the latter branching on variables and pruning by constraint propagation, and the former branching on actions and pruning by lower bound estimations. The two formulations, however, have a lot in common, and some key planners such as Graphplan can be understood in either way: as computing a lower bound function and searching backwards from the goal, or as performing a precise, bounded form of variable elimination, followed by backtracking. The main limitation of older, so-called Partial Ordered Causal Link (POCL) planners, is that they provide smart branching schemes, in particular for temporal planning, but weak pruning rules. Indeed, the computation and even the formulation of good lower bounds for POCL plans is far from trivial. However, the pruning that cannot be obtained by the use of good monolithic lower bounds, can often be achieved by simple propagation rules over a suitable constraint-based formulation. We show this to be the case for CPT, currently the best domain-independent temporal planner, and then explore briefly further branching and pruning variations in parallel and conformant planning.

1 Introduction

AI Planning studies languages, models, and algorithms for describing and solving problems that involve the selection of actions for achieving goals. Most work so far has been devoted to *classical planning* where actions are deterministic, and plans are sequences of actions mapping a fully known initial situation into a goal. Other variants considered, however, are *temporal planning,* where actions have durations and some can be executed concurrently, and *contingent* and *conformant planning*, where actions are not deterministic, and their effects may or may not be observable. In each case, the form and semantics of plans can be defined precisely [1], the key problem is *computational:* how to search for plans effectively given a compact description of the task (e.g., in Strips).

2 Branching and Pruning: Heuristic Search and SAT

The search for optimal plans, like the search for optimal solutions in many intractable combinatorial problems, can be understood in terms of *branching* and *pruning*. Both Heuristic Search and SAT (and CSP) approaches in planning can be understood in this way; the former branches on actions and prunes by extracting and using lower bounds [2], the latter, branches on variables and prunes by constraint propagation and consistency checking [3]. The two approaches for taming the search, however, are closely related, and indeed, current SAT approaches [4] work on the encoding extracted from the *planning graph* [5]: a structure that can be interpreted as representing both a *heuristic function* and a *precompiled* theory.

Simplifying a bit, the planning graph can be thought as a sequence of layers P_0, A_0, P_1, $\ldots A_1$, $\ldots$ such that each layer P_i contains facts and each layer A_i contains actions. If computed in a state s, the facts in the first layer P_0 are the ones that are true in s, while then, iteratively, the actions in layer A_i are the ones whose preconditions are in P_i, and the facts in layer P_{i+1} are the ones added by actions in A_i (for this construction, no-op actions are assumed for each fact p with pre and postcondition p; see [5]). This is actually the *relaxed planning graph* also called the *reachability graph*, a simplification of the graph computed by Graphplan that ignores the so-called *mutex* information.

It is easy to show in either case that the heuristic function $h(s) = i$ where i is the index of the first layer P_i that contains the goals is a *lower bound* on the number of actions that are needed for achieving the goals from s. A generalized formulation of this class of lower bounds is given in [2], where a family of admissible heuristic functions $h^m(s)$ for a fixed integer $m = 1, 2, \ldots$ are defined that recursively approximate the cost of achieving a set of atoms C by the cost of the achieving the most costly subset of size m in C. Relaxed reachability corresponds to h^m with $m = 1$, while mutex reachability corresponds to h^m with $m = 2$.

From a logical perspective, if L_0, L_1, L_2, $\ldots$, refer to the collections of fact variables at time 0, action variables at time 0, fact variables at time 1, and so on, it is easy to verify that all the clauses in *planning theories* involve variables in the same layer L_i, or variables in adjacent layers. The *stratified* nature of these theories suggests a stratified form of inference: starting with the set of clauses Γ_i in the first layer L_i for $i = 0$, iteratively compute the sets of consequences $\Gamma_{i+1} = C_{i+1}(\Gamma_i \cup T_{i,i+1})$ over the next layer L_{i+1}, for $i = 0, 1, \ldots$, using Γ_i and the clauses $T_{i,i+1}$ that involve variables in both layers. If $C_{i+1}(X)$ is defined as the set of *prime implicates* of size no greater than m in L_{i+1} that follow from X, then the derived clauses turn out to be in correspondence with the clauses obtained from the planning graph: $m = 1$ yields a correspondence with the relaxed planning graph, while $m = 2$ yields a correspondence with the planning graph with pairwise mutexes. This inference is polynomial in the context of planning theories, where it corresponds precisely to a form of *bounded form of variable elimination* [6], where variables are eliminated in *blocks* inducing constraints of size no greater than m; see [7] for details.

3 Branching and Pruning in POCL Planning

Partial Order Planners were common in AI during the 80's and early 90's but could not compete with Graphplan and successors in terms of performance [5]. The reason being that POP planners, and in particular Partial Order Causal Link (POCL) planners [8], provide a branching scheme particularly suited for temporal planning [9], but no comparable pruning mechanisms. This limitation has been addressed recently in [10], where an optimal temporal planner that combines a POCL branching scheme with strong pruning mechanisms has been formulated in terms of constraints. The key element that distinguishes this planner, called CPT, from previous constraint-based POCL planners is the ability to reason about *all actions in the domain* and not only *the actions in the current plan.* The latter planners do not infer anything about an action until it is included in the plan, and something similar occurs in the standard methods for solving Dynamic CSPs. Yet often a lot can be inferred about such actions even before any commitments are made; the lower bounds on the starting times of *all* actions as computed in the planning graph being one example. In order to perform these and other inferences, CPT represents and reasons with a set of variables associated with *all* the actions in the domain. By means of a suitable set of constraints, propagation rules, and preprocessing, CPT has been shown to be the top performing optimal temporal planner, approching the performance of the best SAT planners in the special case in which all actions have unit duration [10].

The inference capabilities of CPT are illustrated in [10] by means of a simple TOWER-n domain, where n blocks $b_1, \ldots, b_n$ that are initially on the table, need to be stacked in order with b_1 on top. This is trivial problem for people but not for an optimal *domain-independent* planner that fails to recognize the structure of the problem. Indeed, none of the optimal planners considered, including Graphplan, SAT, and Heuristic Search planners can solve instances larger than $n = 15$. CPT, on the other hand, solves these and larger instances, in a few seconds by *pure (polynomial) inference and no search.* Actually, in [11], it is shown that many of the standard benchmarks used in planning, including all instances of Blocks, Ferry, Logistics, Gripper, Miconic, Rovers and Satellite, are solved *backtrack free* by an extension of CPT that performs further but still polynomial inference in every node.

4 Further Variations on Branching and Pruning

Graphplan computes the planning graph once from the initial situation and then searches the planning graph backwards for a plan. In [12], an alternative branching scheme is considered based on forcing a selected action in or out of the plan at a given time. The planning graph is then recomputed in every node in a way compatible with the commitments made, and a node is pruned when its planning graph pushes the goal beyond planning horizon. It is then shown that this alternative branching scheme, that preserves the same lower bound mechanism as

Graphplan (the planning graph), does much better than Graphplan when many actions can be done in parallel. In [13], the same branching scheme is used for *conformant planning* where the plan must work for a number of possible initial states (the initial state is partially unknown). Then partial conformant plans are pruned when they become incompatible with the plan for some initial state. This is determined by *model-count* operations that are rendered efficient by a precompilation of the planning theory into a suitable logical form [14].

Clearly, branching and pruning go a long way in optimal problem solving, yet it is not *all* branching and pruning. Two other ideas that have been shown to be important as well in problem solving are *Learning* in both CSP/SAT [15] and State Models [16], and *Decomposition* [14, 17], in particular in problems that are harder than SAT.

Acknowledgments: Many of the ideas in CPT as well as all the code are due to Vincent Vidal. My work is supported in part by Grant TIC2002-04470-C03-02, MCyT, Spain.

References

1. Geffner, H.: Perspectives on AI Planning. In: Proc. AAAI-02. 1013–1023
2. Haslum, P., Geffner, H.: Admissible heuristics for optimal planning. In: Proc. AIPS-00. 70–82
3. Kautz, H., Selman, B.: Pushing the envelope: Planning, propositional logic, and stochastic search. In: Proc. AAAI-96, 1194–1201
4. Kautz, H., Selman, B.: Unifying SAT-based and Graph-based planning. In Proc. IJCAI-99, (1999) 318–327
5. Blum, A., Furst, M.: Fast planning through planning graph analysis. In: Proceedings of IJCAI-95, Morgan Kaufmann (1995) 1636–1642
6. Dechter, R.: Bucket elimination: A unifying framework for reasoning. Artificial Intelligence **113** (1999) 41–85
7. Geffner, H.: Planning graphs and knowledge compilation. In: Proc. KR-04. 662–672
8. Weld, D.S.: An introduction to least commitment planning. AI Magazine **15** (1994)
9. Smith, D., Frank, J., Jonsson, A.: Bridging the gap between planning and scheduling. Knowledge Engineering Review **15** (2000) 61–94
10. Vidal, V., Geffner, H.: Branching and pruning: An optimal temporal POCL planner based on constraint programming. In: Proc. AAAI-04. (2004) 570–577
11. Vidal, V., Geffner, H.: Solving simple planning problems with more inference and no search. In: Proc. CP-05. (2005)
12. Hoffmann, J., Geffner, H.: Branching matters: Alternative branching in graphplan. In: Proc. ICAPS-2003. (2003) 22–31
13. Palacios, H., Bonet, B., Darwiche, A., Geffner, H.: Pruning conformant plans by counting models on compiled d-DNNF representations. In: Proc. ICAPS-05. (2005)
14. Darwiche, A.: Decomposable negation normal form. J. ACM **48** (2001) 608–647
15. Dechter, R.: Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. Artificial Intelligence **41** (1990) 273–312
16. Bonet, B., Geffner, H.: Learning in DFS: A unified approach to heuristic search in deterministic, non-deterministic, probabilistic, and game tree settings. (2005)
17. Dechter, R.: AND/OR Search spaces for Graphical models. TR (2004)