Preferences, Planning, and Control

Ronen I. Brafman
Department of Computer Science
Ben-Gurion University
brafman@cs.bgu.ac.il

Abstract

Preference handling is a problem of much theoretical and practical interest. In planning, preferences arises naturally when one considers richer notions of goals, as well as over-subscribed planning problems. In knowledge representation, it is a core issue with much recent work on preference languages and algorithms. In system design, preferences can be used to control choices and provide a personalized experience or adapt to varying context. In this talk I will discuss some of my work, together with many colleagues, in these areas. I will consider some of the challenges we face when designing a preference specification formalism and describe a simple graphical input language, CP-nets - which attempts to address some of these challenges. Surprisingly, CP-nets are closely related to an important analysis tool in planning - the causal graph, and the problem of inference in these networks has important links to the question of the complexity of plan generation. Moreover, the problem of finding a preferred plan given a rich goal specification can be solved by using techniques developed for constrained optimization in CP-nets. But CP-network are inherently a propositional specification language, whereas many control applications require a relational language. Time permitting, I will explain why this problem arises naturally in intelligent control applications. I will show how some recent and richer relational languages can be used to address this problem, and how closely they are related to probabilistic relational models.

Preferences play a central role in decision-making as they serve to guide our choice among alternative decisions. Planning problems are a special type of sequential decision problems. Thus, there are obvious relationships between the two areas. What is less obvious is that there are an important technical relationship between reasoning about preferences in some models and planning. This relationship has led to some new insights about the complexity of planning. In this talk, I will discuss some of these relationships: the more obvious relationships between preferences and the notion of a goal, and the less obvious relationship between the problem of planning and the problem of inference in preference reasoning, as well as between graphical structures for planning and preference. The work I will describe is the result of collaborations with quite a few people, including Craig Boutilier, Yury Chernyavsky, Holger Hoos, and David Poole, and most notably, Carmel Domshlak, whose thesis (Domshlak 2002) was the first to explore the relationship between causal graphs and CP-networks and made important contributions in both areas.

Let us start with the obvious relationship between preferences and goals. Early work in AI focused on the notion of a *goal*—an explicit target that must be achieved—and this paradigm is still dominant in AI problem solving. But as application domains become more complex and realistic, it is apparent that the dichotomic notion of a goal, while adequate for certain puzzles, is too crude in general. The problem is that in many contemporary application domains, e.g. information retrieval from large databases or the web, or planning in complex domains, the user has little knowledge about the set of possible solutions or feasible items, and what she typically seeks is the best that's out there. But since the user does not know what is the best achievable plan, she typically cannot characterize it or its properties specifically. As a result, she will end up either asking for an unachievable goal, getting no solution in response, or asking for too little, obtaining a solution that can be substantially improved. Of course, the user can gradually adjust her stated goals. This, however, is not a very appealing mode of interaction because the space of alternative solutions in such applications can be combinatorially huge, or even infinite. Moreover, such incremental goal refinement is simply infeasible when the goal must be supplied off-line, as in the case of autonomous agents (whether on the web or on Mars). Hence, what we really want is for the system to understand our preferences over alternative choices (that is, plans, documents, products, etc.), and home-in on the best achievable choices for us.

But what are preferences? Semantically, the answer is simple. Preferences over some domain of possible choices order these choices so that a more desirable choice precedes a less desirable one. We shall use the term *outcome* to refer to the elements in this set of choices. Naturally, the set of outcomes changes from one domain to another. Examples of sets of possible outcomes could be possible flights, vacation packages, cameras, end-results of some robotic application (e.g, pictures sent from Mars, time taken to complete the DARPA Grand Challenge, etc.). Orderings can have slightly different properties, they can be total (i.e. making any pair of outcomes comparable) or partial, strict (i.e. no two outcomes are equally preferred) or weak but that is about it. A

bit more formally, we say that a *preference* relation over a set Ω is a transitive binary relation $\succeq$ over Ω. If for every $o, o' \in \Omega$ either $o \succeq o'$ or $o' \succeq o$ then $\succeq$ is a *total* order. Otherwise, it is a *partial* order (i.e. some outcomes are not comparable).

Unfortunately, while the semantics of preference relations is pretty clear, working with preferences can be quite difficult, and there are a number of reasons for that. The most obvious issue is the cognitive difficulty of specifying a preference relation. Once multiple aspects of an outcome matter to us, ordering even two outcomes can be cognitively difficult because of the need to consider tradeoffs and interdependencies between various attributes. As the size of the outcome spaces increases, our cognitive burden increase, and additional computational and representational issues come in. How do we order a large set of outcomes? How can the user effectively and efficiently communicate such an ordering to the application at hand? How do we store and reason with this ordering efficiently?

Work in preference handling attempts to address these issues in order to supply tools that help us design systems acting well on our behalf, or help us act well. Preference *elicitation* methods aim at easing the cognitive burden of ordering a set of outcomes, or finding an optimal item. These methods include convenient languages for expressing preferences, questioning strategies based on simple questions, and more. Preference *representation* methods seek ways of efficiently representing preference information. The two are closely related because a compact representation requires less information to be specified, making its elicitation easier. Preference handling *algorithms* attempt to provide efficient means of answering common queries on preference models.

One of the basic assumption of most work on preference handling is that the object of preference have some structure. Typically, we assume that each object is described by a vector of values of a set of attributes $X_1, \ldots, X_n$, and that our level of preference depends on the value of these attributes only. Thus, denoting the domain of attribute X_i by $\mathcal{D}_i$, we shall assume that the set of possible outcomes is a subset of $\mathcal{D}_1 \times \mathcal{D}_2 \times \cdots \times \mathcal{D}_n$.

One strand of work on preference handling focuses on qualitative preference languages. The motivation is simple. We are usually able to communicate our preferences to service providers, such as our travel agent, using simple natural language statements. These statements do not quantify values or utilities, but simply express things like: "I prefer an isle seat to a window seat," or "I prefer to leave early morning, than later." If our travel agent is able to make, or suggest choices on our behalf based on such statements, why can't an artificial travel agent do the same? Of course, one could argue that this can be said of any task that humans perform, and we know how far we can emulate human intelligence. But notice that preference statements are very specialized. There are just a few constructs, and the type of reasoning required is very limited and focused — exactly the properties required of a domain for the deployment of useful expert systems.

There are various types of natural qualitative preference statements one could think of, but, arguably, the simplest and most natural take the form:"I prefer ϕ to ψ" where ϕ and ψ are two properties of the outcomes we care about. Although these properties could be quite general, we will focus on simple properties of the form $X_i = x_i$. Moreover, we will assume that both ϕ and ψ are conditions on the same attribute. Thus, examples of such statements are "I prefer a window seat to an isle seat", "I prefer flyinb British Airways to KLM," and "I prefer non-stop flights to flights with a stop-over." Here we have assumed that *seat-type*, *airline*, and *number of stop-overs* are some of the attributes of the flight objects.

Slightly richer, is the class of conditional statements of this kind. That is, statements of the form: "If α holds, I prefer ϕ to ψ." For example: "In Coach, I prefer a vegetarian meal to a non-vegetarian meal." CP-nets (Boutilier et al. 2004a) are a graphical formalism for representing and reasoning about such statements. The set of conditional statements is represented by an annotated directed graph whose nodes correspond to attributes. An edge from X_i to X_j denotes the fact that some of the preferences for the value of X_j are conditioned on the value of X_i. So if I say that "On KLM I prefer a vegetarian meal to a non-vegetarian meal," then *airline* is a parent of *meal-type* in my CP-net. Of course, the graph itself does not reflect all of the information in the statements, and so nodes are annotated by this information. Every node is associated with a conditional preference table. In this table, for any (or some) values of its parents, we record our preference over the node's values. Thus, if my only preferences for meal type are: "On KLM I prefer a vegetarian meal to a non-vegetarian meal, and on other airlines I prefer a non-vegeterian meal," then the CPT for node *meal-type* will include the information: KLM: non-veg $\prec$ veg, $\neg$KLM: veg $\prec$ non-veg.

A language requires a semantics as well. In our case, we need to explain what preference relation a CP-net induces over the set of possible outcomes. Basically, each statement of the form "If α holds, I prefer ϕ to ψ" provides us some information about outcomes that can be compared. Specifically, it says that given any two outcomes o_1, o_2 that satisfy α, such that o_1 satisfies ϕ and o_2 satisfies ψ, and otherwise, o_1 and o_2 are identical, it is the case that o_1 is preferred to o_2. Thus, given the statement "On KLM I prefer a vegetarian meal to a non-vegetarian meal" we can compare any two flights on KLM that are identical, except for the meal served. Thus, both must have the same type of seat, the same duration, etc. We cannot compare two British Airways flight, because they do not satisfy the context condition. We cannot compare two KLM flights that differ in some attribute other than *meal-type*, e.g., *duration*. Finally, the preference relation induced by a CP-net is the transitive-closure of the union of the preference relations induced by each statement separately.

In Figure 1 we see an example of a CP-net and the preference order it induces over the set of possible outcomes taken from (Boutilier et al. 2004a). The CP-net in Figure 1(a) expresses my preferences for evening dress. This network consists of three variables J, P, and S, standing for the jacket, pants, and shirt, respectively. I unconditionally prefer black to white as a color for both the jacket and

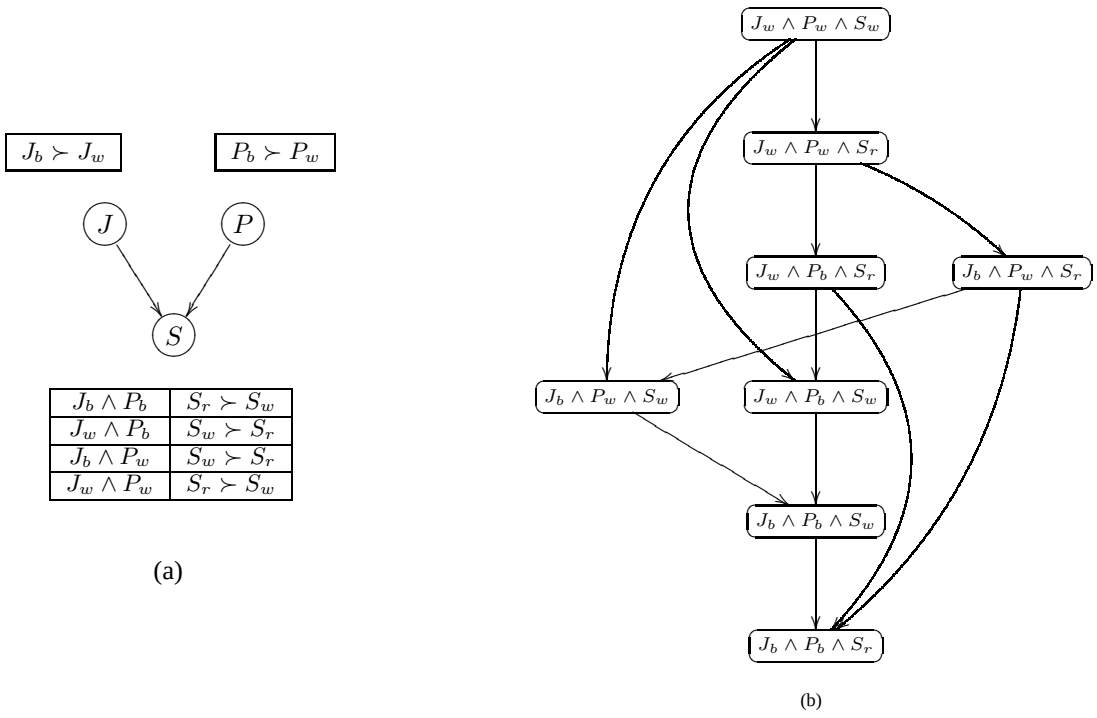

Figure 1: (a) CP-Net for "Evening Dress": Jacket, Pants and Shirt; (b) the induced preference graph.

the pants, while my preference between the red and white shirts is conditioned on the *combination* of jacket and pants: if they have the same color, then a white shirt will make my outfit too colorless, thus I prefer a red shirt. Otherwise, if the jacket and the pants are of different colors, then a red shirt will probably make my outfit too flashy, thus I prefer a white shirt. Figure 1(b) shows the corresponding preference graph. For example, we see an edge from the node representing a black jacket, white pants, and red shirt, to the node representing black jacket, white pants, and white shirt. It is justified because we see in the CPT of variable S that a white shirt is preferred to a red one when the jacket is black and the pants are white.

It is important to remember that on top of what you see in Figure 1(b) a transitive-closure operation must be applied to get the actual preference order. Alternatively, we can say that o is preferred to o' if there is a path from o' to o in this graph. For example, a black jacket and pants and a white shirt are preferred to a black jacket, white pants, and red shirt. Note also that the ordering over outcomes is partial. This is typical of CP-nets because their semantics is conservative.

Now that we have a formalism for specifying preferences, we can specify richer goal notions. For example, our planning domain describe the problem of building some product, say a PC. It includes operators for ordering various products from various places, and for assembling different parts together. Using a preference language, we can define an ordering over the properties of the final PC built. For example, we may prefer AMD to Intel for the CPU, and we may prefer a wireless mouse to a wired mouse, etc. In fact, often, our preferences are conditional: "If I we get a premium soundcard then I prefer premium speakers, but otherwise, I prefer standard speakers."

Being able to specify preferences is nice, but can we plan with them? Here we combine two techniques — planning as CSP (Do and Kambhampati 2001) and preference-based constrained optimization (Boutilier et al. 2004b). That is, we cast the problem of finding an optimal plan as a problem of finding a preferred solution to the CSP induced by the planning problem. This can actually be done in an elegant way given preferences in the form of a CP-net. For details, see (Brafman and Chernyavsky 2005).

Because CP-nets induce a partial order, then there may be a few optimal solutions to the CSP. These solutions are such that no one of them is comparable to the other (otherwise, we could eliminate some of them as non-optimal). As we noted, there is an elegant way of finding a single solution that requires only solving a CSP. However, if we seek multiple solutions, we must call a subroutine that compares different candidates. This ability to compare outcomes is of general interest in reasoning about preferences, and one natural question is how difficult it is to answer such *dominance queries*, as they are known, and whether we can find efficient algorithms to do this.

We have seen that o is preferred to o' *iff* there exists a directed path from o' to o in the preference graph induced by the CP-net. This path is called *an improving flipping sequence* because each edge denotes the change in value of a single variable (a value *flip*). Recall that the preference statement that induces this edge has the form "If α then I prefer x_i to x_i' as the value of X_i." Thus, given any outcome o sat-

isfying α and $X_i = x_i'$ we can improve it by changing X_i to x_i.

But the problem of finding a path in some structured state space is quite similar to planning. In fact, we can view each preference statement as inducing an action. The statement "If α then I prefer x_i to x_i' as the value of X_i" can be viewed as an action that has $\alpha \wedge X_i = x_i'$ as its preconditions, and $X_i = x_i$ as its effect. Thus, preference statements can be viewed as defining a special type of planning domain in which each operator affects a single variable only. From this planning perspective, the CP-network is describing the possible causal relationships between different propositions in the planning domains. Edges link preconditions to their effects. In fact, this structure is known in the planning literature under various names, and most typically, as the *causal graph* of the planning domain (see, e.g., (Knoblock 1994; Williams and Nayak 1997).

So now we see that there are two closely related computational problems: deciding whether $o \prec o'$ holds, and finding a plan from o to o'. Their complexity must be intimately related, too. Indeed, we now know that this complexity is closely related to the topology of this graph. It was known early on that if the graph is a tree and all variables are boolean, then there is a linear-time procedure for planning and for answering the dominance query. In (Brafman and Domshlak 2003), it is show that this is still tractable when the induced undirected graph is a tree, and otherwise, the problem becomes NP-hard.

Of course, from the planning perspective, action with single effects and Boolean-valued variables are a strong limitation. Can the tractability results be extended to more general cases, such as multi-values variables. Unfortunately, Domshlak and Dinitz show that this is not the case (Domshlak and Dinitz 2001). But fortunately, all is not lost. Helmert (Helmert 2006) shows how, building on these ideas, one can generate interesting heuristics and design a very efficient planner.

One of the main problems for some of the intractability results in planning in general, and here as well, is the fact that some planning problems have only exponentially long solutions. What if we restrict out attentions to problem for which there are reasonable plans. In terms of planning complexity, we can ask whether there are classes of planning problems that are tractable in terms of their output complexity. It turns out that the complexity of STRIPS planning in general can be linked to a property of the causal graph, its tree-width. For graphs with constant tree-width, the output complexity of planning is polynomial! See (Brafman and Domshlak 2006) for the details.

So we see how a problem of reasoning about preference induced research on planning complexity, the result of which shed light on complexity issues in both fields. And this line of research that started originally in preference reasoning has started to make a real impact on our understanding of the complexity of planning via causal graphs.

References

Boutilier, C.; Brafman, R.; Domshlak, C.; Hoos, H.; and Poole, D. 2004a. CP-nets: A tool for representing and reasoning about conditional *ceteris paribus* preference statements. *Journal of Artificial Intelligence Research (JAIR)* 21:135–191.

Boutilier, C.; Brafman, R.; Domshlak, C.; Hoos, H.; and Poole, D. 2004b. Preference-based constrained optimization with CP-nets. *Computational Intelligence,Special Issue on Preferences in AI and CP* 20(2):111–136.

Brafman, R., and Chernyavsky, Y. 2005. Planning with goal preferences and constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 182–191.

Brafman, R. I., and Domshlak, C. 2003. Structure and complexity of planning with unary operators. *Journal of Artificial Intelligence Research* 18:315–349.

Brafman, R. I., and Domshlak, C. 2006. Factored planning: How, when, and when not. In *AAAI*, 809–814.

Do, M. B., and Kambhampati, S. 2001. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artificial Intelligence* 132(2):151–182.

Domshlak, C., and Dinitz, Y. 2001. Multi-agent off-line coordination: Structure and complexity. In *Proceedings of Sixth European Conference on Planning (ECP)*, 277–288.

Domshlak, C. 2002. *Modeling and Reasoning about Preferences with CP-nets*. Ph.D. Dissertation, Ben-Gurion University, Israel.

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

Knoblock, C. 1994. Automatically generating abstractions for planning. *Artificial Intelligence* 68(2):243–302.

Williams, B., and Nayak, P. 1997. A reactive planner for a model-based executive. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 1178–1185.