Preference Elicitation with Subjective Features

Craig Boutilier
Department of Computer Science
University of Toronto
Toronto, ON, Canada
cebly@cs.toronto.edu

Kevin Regan
Department of Computer Science
University of Toronto
Toronto, ON, Canada
kmregan@cs.toronto.edu

Paolo Viappiani
Department of Computer Science
University of Toronto
Toronto, ON, Canada
paolo@cs.toronto.edu

ABSTRACT

Utility or preference elicitation is a critical component in many recommender and decision support systems. However, most frameworks for elicitation assume a predefined set of features (e.g., as derived from catalog descriptions) over which user preferences are expressed. Just as user preferences vary considerably, so too can the features over which they are most comfortable expressing these preferences. In this work, we consider preference elicitation in the presence of *subjective* or *user-defined features*. We treat the problem of learning a user's feature definition as one of *concept learning*, but whose goal is to learn only enough about the concept definition to enable a good decision to be made. This is complicated by the fact that user preferences are unknown. We describe computational procedures for identifying optimal alternatives w.r.t *minimax regret* in the presence of both utility and concept uncertainty; and develop several heuristic query strategies that focus on reduction of *relevant* concept and utility uncertainty. Computational experiments verify the efficacy of these strategies.

1. INTRODUCTION

Assessing the preferences of users is a critical component in any decision support or recommender system: the ability to tailor recommendations to the needs and desires of a particular user is indeed a hallmark of intelligent decision support. Within AI, decision analysis, and operations research, a variety of systems and methodologies have been developed for *preference elicitation*: engaging in some "dialog" with a user to determine her preferences. These range from *conversational recommender systems* [4, 11] that provide considerable navigational control to a user as they explore the space of possible products/options, to adaptive *elicitation systems* that ask the user a sequence of questions about her preferences or utility function [2]. In what follows, we focus primarily on elicitation.

Typically one assumes the existence of a set of *universal* or *catalog features* over which user preferences are specified. For instance, in product configuration, preferences are assumed to be defined in terms of product features and specifications (e.g., color, engine size, fuel economy, available options, etc. in the case of a car). However, users can exhibit significant variation in the features over which their preferences are most naturally expressed, and these features may not be present among the set of catalog features. For instance, in the automotive domain, some users may be concerned about the "degree of safety" of a car, but different users may have different notions of safety in mind.[1] In recent work [3] we have developed a model for *subjective feature elicitation* that queries users about the feature in question, so that preference tradeoffs can be made involving the new feature. This model casts the problem as one of *concept learning* [1, 5]. However, unlike traditional concept learning, the aim is to learn *just enough* about the concept definition to make a good or optimal decision on the user's behalf. Specifically, it is assumed that the user's underlying utility function is known and can be used to render judgements of relevance [3].

In this paper, we extend this model to allow utility uncertainty. Indeed, while subjective feature elicitation is an important component of the process, genuine preference elicitation in the presence of subjective features requires interacting with a user when both the feature definitions and user preferences are unknown.

We present a model for preference elicitation with subjective features that allows for the simultaneous elicitation of user utility and user features, making appropriate tradeoffs between the two types of information. We use *minimax regret* [2, 10] as our decision criterion given concept and utility uncertainty, allowing good or optimal decisions to be made without complete specification of either component. We present several heuristic techniques for querying concept definitions and utility functions that reduce minimax regret quickly. In contrast to standard concept learning models, we aim to reduce "relevant" concept uncertainty w.r.t. the utility model, rather than learn an accurate concept definition for its own sake. And partially elaborated concept definitions influence the choice of utility queries as well. This provides a fully integrated preference elicitation methodology that allows a user to dynamically (and partially) specify their own utility-bearing product features.

2. FEATURE & UTILITY UNCERTAINTY

We consider *subjective features* that are "objectively" definable using catalog attributes, but where the definition varies from user to user. For instance, the notion of a *safe* car may be different for a parent with small children, a young, single professional interested in high-performance vehicles, and a family that takes frequent trips to the mountains. The *concept* in question, namely, safety, has *personalized definitions*.[2] The user also has *preferences* for safety (as she does for other car attributes) represented in the form of a util-

[1] E.g., a driver with a young family may define safety in terms of tires, air bags, child restraints, etc., while a high-performance driver refers to the braking system, roll bars, etc.

[2] A distinct form of subjective feature involves aesthetic judgements or subjective tastes. In the car domain, two people may have different opinions as to whether a car looks "aggressive" or "cute," without being able to articulate that preference. As a result, collaborative filtering techniques [8] are best-suited to helping user's navigate through products with such features.

Submitted to 3rd ACM RecSys, New York, 2009

ity function: it is both the user's utility function over this extended attribute space, as well as her personal definition of safety, that determines the optimal vehicle. As such, the recommender system must engage in *both preference elicitation and feature elicitation* to make a suitable recommendation.[3]

This leads to interesting tradeoffs in elicitation. One could engage in feature elicitation using well-known concept learning techniques [1] and then, with a full definition in hand, move to preference elicitation (e.g., using techniques mentioned above); but this could be wasteful. For instance, suppose we learn that safety requires attribute X_i to be true (e.g., have side airbags) but know nothing else about the concept. If we engaged in preference elicitation simultaneously and ascertained that no cars in the user's price range satisfy X_i—or that other more important features must be sacrificed to attain X_i—then the full concept definition is not needed for optimal allocation.

Conversely, we could engage in preference elicitation, using the subjective feature as an attribute without knowing its definition, and then engage in feature elicitation to determine a final recommendation. However, without some idea of the concept definition much more preference information will be elicited than necessary. This suggests that *interleaved* feature and utility elicitation can be much more effective.

In this section, we first formalize our basic model of utility and concept uncertainty; we then define the minimax regret decision criterion for this case; finally, we develop a MIP formulation for solving the computing minimax regret. We turn to the question of elicitation in the next section.

2.1 Basic Model

As above, assume features $\mathcal{X} = \{X_1, ...X_n\}$, which we take to be Boolean for ease of exposition, and a feasible product set $\mathbf{X} \subseteq Dom(\mathcal{X})$. The user's utility for any product $\mathbf{x} \in \mathbf{X}$ is decomposed into two components. First, the user has some utility or *reward* w.r.t. catalog features. We denote this by function $r(\mathbf{x}; w)$ where w denotes the parameters of this reward function. In what follows, we assume r is additive over $\mathcal{X}$ (this is not critical, only that r is linear in whatever parameterization w we adopt).

The user also has a preference for configurations satisfying some target concept c. Concept c is an unknown Boolean function over $\mathcal{X}$: $c(\mathbf{x}) = c(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.[4] We suppose that c is drawn from a particular function class/hypothesis space H (e.g., the set of conjunctive concepts). We treat identification of c as a problem of concept learning [1], with some query set Q that can be used to refine the target concept. For instance, membership queries would be quite natural (e.g., "do you consider the following car to be safe?").[5] A value or *bonus* w_b is associated with any $\mathbf{x}$ s.t. $c(\mathbf{x})$ holds, representing user utility for concept satisfaction.

Assuming utility independence for concept satisfaction relative to other preferences, we define the utility of $\mathbf{x}$ under concept c and reward/bonus weight vector (or utility parameters) w to be:

$$u(\mathbf{x}; w, c) = r(\mathbf{x}; w) + w_b c(\mathbf{x})$$

In other words, the utility of $\mathbf{x}$ is its reward, plus the bonus w_b if $\mathbf{x}$ satisfies c. The optimal configuration is $\mathbf{x}^*_{w,c} = \arg\max u(\mathbf{x}; w, c)$.

Since c is definable in terms of catalog features, we could in principle elicit utilities using only catalog features. However, allowing a user to articulate her preferences in terms of natural composite features can dramatically reduce the burden of elicitation; furthermore, the addition of such aggregate features with suitable definitions can greatly increase the degree of (conditional) utility independence in a model.

[3]In cases where a very small number of definitions exist that tend to apply to specific user *types*, one could imagine predefining these features and quickly discriminating them. However, our aim is to allow more open-ended feature definition.

[4]Allowing multivalued concepts is straightforward.

[5]Other query types (e.g., equivalence queries) are less natural in this domain, but may play a role in others.

2.2 Minimax Regret

During preference elicitation, we are uncertain about the true utility w and the true user concept c. As a result, we cannot generally identify the optimal product $\mathbf{x}^*_{w,c}$; but we can still make a decision with partial utility and concept information. Let W be the set of feasible utility functions, those consistent with any prior information we have about user preferences and user query responses. W is generally a convex polytope given by linear constraints on utility parameters (as discussed below). Let *version space* $V \subseteq H$ represent our current set of consistent hypotheses w.r.t. c [9], i.e., those that respect any prior knowledge about the concept and responses to queries (as discussed below). Define *minimax regret* w.r.t. utility and feature uncertainty:

Definition 1 *Given utility space W and version space V, the* max regret *of $\mathbf{x} \in \mathbf{X}$, the* minimax regret *of (W, V) and the* minimax optimal configuration *are:*

$$MR(\mathbf{x}; W, V) = \max_{w \in W} \max_{c \in V} \max_{\mathbf{x}' \in \mathbf{X}} u(\mathbf{x}'; w, c) - u(\mathbf{x}; w, c) \quad (1)$$

$$MMR(W, V) = \min_{\mathbf{x} \in \mathbf{X}} MR(\mathbf{x}; W, V) \quad (2)$$

$$\mathbf{x}^*_{W,V} = \arg\min_{\mathbf{x} \in \mathbf{X}} MR(\mathbf{x}; W, V) \quad (3)$$

Should we recommend option $\mathbf{x}$, max regret $MR(\mathbf{x}; W, V)$ bounds (tightly) how far this decision could be from optimal. Intuitively, an adversary selects the user's utility function w and the intended subjective feature definition c to maximize the difference in utility between our choice $\mathbf{x}$ and the optimal choice $\mathbf{x}^*_{w,c}$ (notice that the adversary's maximizing configuration must be optimal under (w, c)). A minimax optimal choice is any product that minimizes max regret in the presence of such an adversary, and its max regret is the minimax regret given our current uncertainty.

2.3 Computing Regret: Conjunctive Concepts

We assume that the underlying configuration problem is represented as a MIP $\max_{\mathbf{x} \in \mathbf{X}} u(\mathbf{x})$. We can then incorporate utility uncertainty (in the form of a bounded polytope W) into the MIP following [2], and feature uncertainty in the form of a version space V following [3]. However, in the latter case, the formulation depends critically on the form of the concept and query classes one admits. We illustrate the formulation for the case of (nonmonotone) conjunctive concepts with membership queries.

Assume target c is a conjunction of literals over variables X_j. Memberships queries ask whether $\mathbf{x} \in c$ for some product $\mathbf{x}$. Let E^+ (E^-) be the set of positive (negative) examples acquired by these queries, and (nonempty) V the induced version space. Instead of representing V using most general and most specific concepts, we encode E^+ and E^- directly in our MIP below (e.g., negative examples can directly represent most general concepts [6]).

Constraint Generation We formulate the minimax problem Eq. 2 as a semi-infinite minimization. Let $(X_1, \cdots, X_n)$ be configuration variables over our n features: its instantiation will denote the minimax optimal product. Let constant $b(\mathbf{x}, w, c) = w_b$ if $c(\mathbf{x})$ and 0 otherwise. Let indicator variable I^c, for each $c \in V$, denote that configuration $(X_1, \cdots, X_n)$ satisfies c; and write $x^j \in c$ (resp., $\overline{x}_j \in c$) to denote that variable X_j occurs positively (resp.,

negatively) in c. Then $MMR(V)$ is given by:

$$\min \quad \delta$$

$$\text{s.t. } \delta \geq r(\mathbf{x}^*_{w,c}) - r(X_1, \cdots X_n) + b(\mathbf{x}^*_{w,c}, c) - w_b I^c \quad \forall c \in V, \forall w \in W \tag{4}$$

$$I^c \leq X_j \qquad \forall c \in V, \forall x_j \in c \tag{5}$$

$$I^c \leq 1 - X_j \qquad \forall c \in V, \forall \overline{x}_j \in c \tag{6}$$

For any *fixed* concept c and utility function $w \in W$, the adversary maximizes the regret of $(X_1, \cdots, X_n)$ with witness product $\mathbf{x}^*_{w,c}$. The MIP above chooses a configuration that minimizes against the "worst-case" choice of the adversary (with (4) ensuring MMR is as great as regret given any $c \in V, w \in W$; and (5, 6) encoding whether $(X_1, \cdots, X_n)$ satisfies c).

Regret constraints for most $w \in W, c \in V$ will be inactive, so we use constraint generation to search through the space of adversarial utility functions and concepts. Let $Gen \subseteq W \times V$ be a (small) set of (w, c)-pairs (initially a single pair); we solve a relaxed MIP using only constraints for those $(w, c) \in Gen$. Let δ^* and $\mathbf{x}^*$ be the solution to the relaxed MIP. We test for violated constraints by solving the max regret problem $MR(\mathbf{x}^*; W, V)$, detailed below. If $MR(\mathbf{x}^*, W, V) > \delta^*$, the utility-concept pair (w', c')—produced as a witness in max regret computation—offers larger regret for $\mathbf{x}^*$ than any $(w, c) \in Gen$; indeed, it corresponds to the maximally violated constraint in the relaxed MIP. So we add (w', c') to Gen and resolve. If $MR(\mathbf{x}^*; W, V) = \delta^*$, $\mathbf{x}^*$ is the optimal solution to $MMR(W, V)$.

Generating Violated Constraints We compute the maximally violated constraint for the MIP above by solving the max regret problem $MR(\mathbf{x}^*; W, V)$ for the current relaxed solution $\mathbf{x}^*$. This too can be formulated as a MIP that, given $\mathbf{x}^*$, chooses an (adversarial) concept c, utility w and configuration. Details are omitted, but we use a standard reformulation to convert quadratic terms into a continuous variable, giving us a linear objective (similar to [2]).

3. SIMULTANEOUS FEATURE AND UTILITY ELICITATION

While minimax regret provides an appealing means for making recommendations under utility and feature uncertainty, our aim is to learn enough about a user's preferences and underlying concept to make good (or even optimal) recommendations, while asking as few queries as possible. In this section, we develop several heuristic query strategies that can quickly reduce $MMR(W, V)$. We begin with a discussion of several forms of queries.

Query Types With respect to explicit concept queries we restrict attention to membership queries of the form "does $\mathbf{x}$ satisfy concept c?" (e.g., "Do you consider car $\mathbf{x}$ to be safe?"). Each membership query gives rise to a positive or negative concept example, and the version space can be encoded in a variety of ways depending on the hypothesis class [6].

Especially natural are *comparison queries*: a user is asked if she prefers one product $\mathbf{x}$ to another $\mathbf{y}$. Such comparisons can be localized to specific subsets of attributes as well, depending on the form of the utility model and can be generalized to choice sets over more than two products.

Responses to these and other common queries impose linear constraints on W when subjective features are absent. But the situation becomes more complicated when feature uncertainty is added. If a user states that she prefers $\mathbf{x}$ to $\mathbf{y}$ in response to a comparison query, we can no longer impose simple constraints on W. The greater utility of $\mathbf{x}$ could be due to its satisfaction of the subjective feature; but we this cannot be reflected in the weight vector alone. This can introduce complicating constraints that tie W and V together. One simple solution to this problem is to ask concept queries whenever one asks a comparison query. More precisely, if a user is asked whether she prefers $\mathbf{x}$ or $\mathbf{y}$, a membership query can be asked of each outcome at the same time (e.g., "is $\mathbf{x}$ safe?"). This is reasonably natural, since the assessment of preference likely involves some cognitive assessment of the subjective feature in question. We call such a query a *combined comparison/membership (CCM)* query. This allows us to impose simple valid linear constraints; e.g., if $\mathbf{x}$ is preferred and satisfies the concept, while $\mathbf{y}$ does not, then we have $w\mathbf{x} + b - w\mathbf{y} > 0$.

However, if we want a pure comparison query without the corresponding membership queries, we can still impose valid (and complete) *conditional constraints* on W, based on the whether $\mathbf{x}, \mathbf{y}$ satisfy the concept, thus linking W and V. In the case of conjunctive concepts, we can linearize these conditional constraints without the introduction of new variables.

Elicitation Strategies We now develop elicitation strategies for simultaneous utility and feature uncertainty.

For the choice of an appropriate comparison query, we adopt the *current solution strategy (CCSS)* [2]: given the minimax optimal solution $\mathbf{x}^*_{W,V}$ and the adversarial witness $\mathbf{x}^a$, the user is asked which of these two products is preferred.

To select membership queries, we examine two methods explored in [3]. The first is a simple *halving* strategy adapted from standard conjunctive concept learning: we ask random memberships queries until a positive example is found; then queries are asked by negating literals one by one in the (unique) most specific conjunctive hypothesis. Once a positive example is found, this converges to the true conjunctive concept using a number of queries linear in the number of catalog features.

We also explore the *current solution strategy for membership queries (MCSS)*: this selects a query based on which of the optimal product $\mathbf{x}^*_{W,V}$ and/or witness $\mathbf{x}^a$ satisfy the adversary's choice of concept c^a in the current solution. If $\mathbf{x}^*_{W,V}, \mathbf{x}^a \in c^a$, then CSS asks membership query $\mathbf{x}^a$; if $\mathbf{x}^*_{W,V} \notin c^a, \mathbf{x}^a \in c^a$, then CSS asks query $\mathbf{x}^*_{W,V}$; otherwise CSS asks a query depending on the whether $\mathbf{x}^a$ is V-consistent (see [3] for further details and motivation). MCSS will never as a membership query if the product in question is "certain" (i.e., has its concept status *determined* unambiguously by V).

Unlike the cases of pure utility or pure feature elicitation, in the simultaneous case, we must make a decision at each stage regarding which type of query to ask, membership or comparison.

In our "interleaved" strategies below, we decompose max regret of the current solution into *reward regret* and *concept regret* and use these measures to determine whether to ask a comparison (utility) query or a membership (concept) query, depending on which is larger. Let $(\mathbf{x}^*, \mathbf{x}^a, w, c)$ be the current solution. Max regret of $\mathbf{x}^*$ is $rr + cr$ (reward regret plus concept regret), where

$$rr = r(\mathbf{x}^a; w) - r(\mathbf{x}^*; w); \quad cr = w_b(c(\mathbf{x}^a) - c(\mathbf{x}^*))$$

Given this, we examine five query strategies. Two are *phased strategies* that first attempt to learn the concept and then refine the utility function. The first is dubbed *Ph(H,CCSS)* and initially uses the halving algorithm (membership queries) to determine the precise concept definition, and then uses CCSS (comparisons) to refine utility function uncertainty. The second phased strategy is *Ph(MCSS,CCSS)* and has the same form as the first, but uses MCSS to generate membership queries. Of course, MCSS can "stall" if the current solution is such that minimax optimal and adversarial products are V-certain. In such a case, a comparison query is asked. As

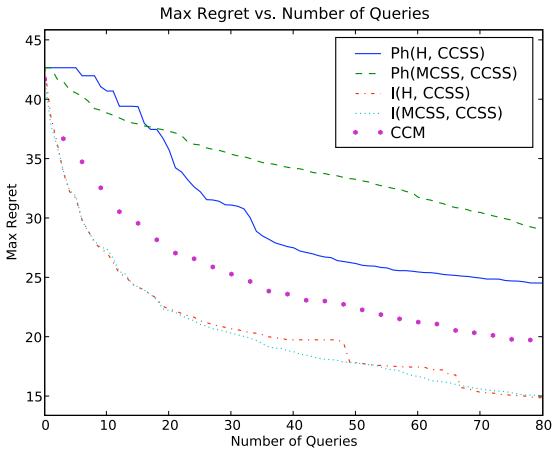

Figure 1: Minimax Regret vs. Number of Queries (30 variables, 90 constraints), averaged over 20 runs

such, we can view Ph(MCSS,CCSS) as a form of interleaving (see below), but with an absolute preference for membership queries if an MCSS-membership query is not vacuous. Our *interleaved strategies* ask a membership queries if concept regret exceeds reward regret at the current solution, and a comparison query if reward regret is greater. They always use CCSS to generate comparisons; but the first, *I(H,CCSS)*, generates membership queries via halving, while the second, *I(MCSS,CCSS)*, uses MCSS. Finally, the *CCM* strategy uses our combined comparison-membership queries, using CCSS to generate the comparison, and asking membership queries of both alternatives, $\mathbf{x}^*$ and $\mathbf{x}^\alpha$.

4. EMPIRICAL EVALUATION

We experimented with the five query strategies above, comparing them on randomly generated configuration problems. Queries are posed to simulated users, each of which possesses a randomly generated utility function and subjective feature used to answer queries. We measured the effectiveness of our strategies by the considering regret reduction in function of the total number of queries.

Large configuration problems were randomly generated, defined on 30 variables with 90 random binary constraints; and conjunctive concepts were generated from a pool of 10 variables, with an average concept size of 6.67 conjuncts. Fig. 1 illustrates the results when the bonus weight bound set to 10% of the utility uncertainty.

We see that both interleaved strategies, dominate the phased strategies by a significant margin. Both interleaved strategies start by asking the same comparison queries (and tend to ask quite a few comparison queries before any membership queries are ever asked), so their effectiveness is identical until the concept regret becomes dominant. When this happens, MCSS membership queries are more effective than halving queries, but only by a small margin. After 80 cycles, max regret is reduced to about a third of its original value. In contrast, the phased strategies fail to reduce regret significatively.

The combined strategy CCM performs worse than the interleaved strategies, but is still better than either phased approach. In this strategy each interaction counts as three queries, since the user must answer a comparison query and two membership queries. However, one asks a comparison and membership query of the same three outcomes, thus the cognitive cost might be significantly less than 3 queries. A "leftward compression" of the CCM curve would make the strategy seem somewhat more competitive.

Minimax regret computation is initially very fast (less than 1s), but is greatly affected from the conditional constrains (once 50 comparisons are incorporated, minimax regret computation requires more than one minute). From this perspective, the CCM query strategy offers the fastest computation time, as it never requires a conditional constraints.

Overall these results suggest that MMR a very effective means of determining good decisions in the face of simultaneous utility and feature uncertainty. Furthermore, it is a very effective driver of elicitation: the flexibility of interleaved strategies is, indeed, advantageous. Of course, more refined heuristics for choosing between membership and comparison queries could make interleaved strategies even more robust.

5. CONCLUDING REMARKS

We have presented a model for preference elicitation that allows a user to define her own subjective features over which she can express her preferences. Our interleaved strategies are especially effective at simultaneous elicitation of concepts and utilities, using regret to make appropriate choices among the different types of queries. Furthermore, optimal or near-optimal product recommendation is generally possible with far from complete concept definitions and utility information.

A number of important directions for future research remain. Further development of query strategies is one critical direction. Additional empirical and theoretical analysis of means to make suitable tradeoffs between membership queries and comparison queries is ongoing, as is the generalization of our computational and elicitation models to richer concept hypothesis spaces, and additional forms of concept queries and utility queries.

6. REFERENCES

[1] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.

[2] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artifical Intelligence*, 170(8–9):686–713, 2006.

[3] Craig Boutilier, Kevin Regan, and Paolo Viappiani. Online feature elicitation in interactive optimization. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning (ICML-09)*, Montreal, 2009. to appear.

[4] Robin Burke. Interactive critiquing for catalog navigation in e-commerce. *Artificial Intelligence Review*, 18(3-4):245–267, 2002.

[5] David Haussler. Learning conjunctive concepts in structural domains. *Machine Learning*, 4:7–40, 1989.

[6] Haym Hirsh. Polynomial-time learning with version spaces. In *AAAI*, pages 117–122, 1992.

[7] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Wiley, New York, 1976.

[8] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[9] Tom M. Mitchell. Version spaces: A candidate elimination approach to rule learning. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, pages 305–310, Cambridge, 1977.

[10] Leonard J. Savage. *The Foundations of Statistics*. Wiley, New York, 1954.

[11] Paolo Viappiani, Boi Faltings, and Pearl Pu. Preference-based search using example-critiquing with suggestions. *Journal of Artificial Intelligence Research*, 27:465–503, 2006.