CP-nets with Indifference

(Invited Paper)

Thomas E. Allen
University of Kentucky
Lexington, KY 40506–0633
Email: thomas.allen@uky.edu

***Abstract*—Conditional preference networks (CP-nets) offer a potentially compact representation of qualitative preferences. Much of the research on CP-nets limits attention to strict preferences over binary variables. We extend the work of previous researchers to allow modeling preferences over multi-valued variables over some of which the preference holder may be indifferent. We show how to leverage the power of SAT solvers to learn and reason with such CP-nets. We also consider the possibility of exponentially long flipping sequences, showing why in practice this is unlikely to be problematic.**

I. INTRODUCTION

Preferences have been studied in philosophy, economics, psychology and computer science and have a wide range of applications, including e-commerce, recommender systems and control. As a motivating example, consider a home automation system designed to provide support to persons of advanced age who wish to continue living independently. The system enforces a set of (hard) *constraints*, such as that indoor temperature must be within a range commensurate with human health and that rooms must be sufficiently well-lit when the subject is moving through the house. Aside from these, however, the system allows the subject maximal determination over her environment. That is, subject to constraints, control of the home is governed by the subject's preferences. We expect that such preferences will sometimes be *conditional*; for example, the subject may prefer to converse by video with a friend on a particular night of the week, but play a favorite video game on some other night.

Certain problems are inherent in such an application. First, the system needs some way to obtain a model of the subject's preferences, either through *active* elicitation or *passively* observing her actions over time. Once this model is available, the system, when presented with some pair of alternatives that the subject may not have considered previously, should be able to determine which if either the subject would prefer more. A closely related third problem involves determining from the model which of all possible outcomes the subject would prefer most. These problems involving preferences are known as the *learning*, *reasoning*, and *optimization* problems, respectively.

A variety of methods have been proposed for modeling preferences, including generalized additive independence (GAI) value functions [1], [2] and soft constraints [3], [4]. A third, qualitative method, the one we consider here, is that of *conditional preference networks* (CP-nets) [5]. We define CP-nets formally in Section II, but at this point it is helpful to introduce them with an example. Suppose an automated home has windows with blinds that the system can open and close automatically, as well as sensors that report weather conditions in realtime. Suppose also that the subject prefers the blinds be open during the day and closed at night, with the exception that she prefers the blinds always be open when it is snowing and always closed when it is raining. As it turns out, she is equally happy with snow and fair weather, but dislikes rain. The subject's preferences as described can be modeled with the CP-net in Figure 1. In terms of this simple model, three *features* contribute to the subject's happiness—weather, time of day, and the state of the blinds—with 12 possible outcomes in all (fair day with blinds closed, rainy night with blinds closed, etc.). Such features are modeled as variables with finite domains. For example, the variable WEATHER has a multivalue domain consisting of *fair*, *rain* and *snow*, while the other two variables are binary. Each variable is represented in the CP-net as a *node*. The *edges* from WEATHER and TIME to BLINDS indicate that the preference over BLINDS depends on these other two features. Each node is annotated with a *conditional preference table* (CPT). The CPT specifies the local preferences over a variable given the values of all *parent* variables. Note that the subject's preference over WEATHER is *unconditional* since it does not depend on any other factor. Moreover, the CPT for TIME is empty, since we have no information on the subject's preferences for day versus night.

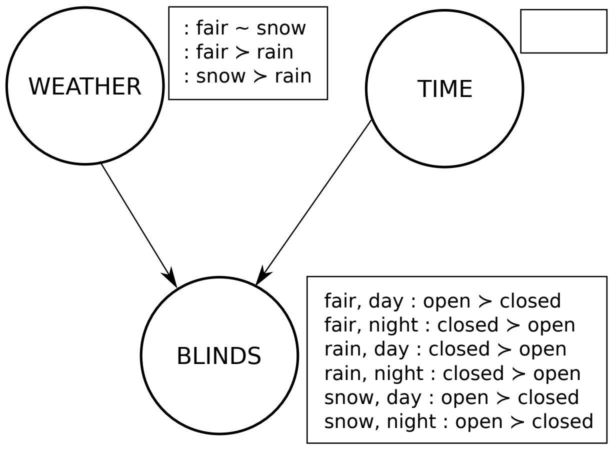
WEATHER
: fair ~ snow
: fair > rain
: snow > rain
TIME
BLINDS
fair, day : open > closed
fair, night : closed > open
rain, day : closed > open
rain, night : closed > open
snow, day : open > closed
snow, night : open > closed

Fig. 1. A Simple CP-net

In this paper we model lack of information as *incomparability*. Care should be taken to distinguish incomparability from *indifference*. While we are told that the subject is equally satisfied with fair and snowy weather, we should not assume, in the absence of additional information, that she is also equally happy with day and night. She may well prefer one to the other. Moreover, indifference here is transitive, while incomparability is not. Finally, it is worth noting that the preferences here are modeled *deterministically* rather than as probability distributions.

Each method of modeling preferences—GAI functions, soft constraints, CP-nets, and so on—has its strengths, weaknesses and unique characteristics. The choice of a preference modeling language for an application such as the one described above may be compared to an engineer's choice of a programming language or software application; various factors such as the specifics of the project and the practitioner's familiarity with the available tools may influence this decision. CP-nets have much appeal. At present, however, much of the discussion of CP-nets assumes strict preferences over binary domains. In our view, this poses a barrier to their adoption.

The remainder of the paper is organized as follows. In Section II we formalize our notions of preferences and CP-nets. In Section III we argue that strict preferences over binary domains are inadequate in many settings and that a richer class of CP-nets, such as we advocate here, is needed. In Section IV we demonstrate how to learn such CP-nets from observations with the help of a SAT solver. In Section V we show how a SAT solver can also be used to reason with such CP-nets using SATPlan. In Section VI we consider the possibility that flipping sequences may be exponentially long, but why in practice this is unlikely to present a problem.

II. Preferences and CP-nets

Preferences involve a set of *outcomes* $\mathcal{O}$ and a *subject* S, assumed here to be an individual. We assume $\mathcal{O}$ is finite and can be factored into *variables* $\mathbf{V} = \{X_1, X_2, \ldots, X_n\}$ with associated *domains*: $\mathrm{Dom}(X_i) = \{x^i_1, x^i_2, \ldots, x^i_{d_i}\}$, where $d_i = |X_i| > 0$. For simplicity we sometimes denote such variables in uppercase, with their respective values in lowercase (e.g., $A = \{a_1, a_2\}, B = \{b_1, b_2, b_3\}$), or for specific examples use descriptive labels, with variables in SMALLCAPS and values in *italics*: e.g., FRUIT $= \{$*apple*, *banana*, *tomato*$\}$. The values that a variable X_i can take may be *constrained* to a subset of its domain. When a variable is constrained to *just one* value, we say the value has been *assigned* to the variable and write $X_i = x^i_j$. We may also constrain and assign multiple variables $\mathbf{X} \subseteq \mathbf{V}$ in this manner. One can observe that an assignment to all variables $\mathbf{X} = \mathbf{V}$ (a *full instantiation*) designates a single outcome $o \in \mathcal{O}$. We denote by $\mathrm{Asst}(\mathbf{X}) = \mathrm{Dom}(X_1) \times \mathrm{Dom}(X_2) \times \cdots \times \mathrm{Dom}(X_k)$ the set of possible assignments to $\mathbf{X} \subseteq \mathbf{V}$. Conversely, we define $\mathbf{o}[i]$ as the *projection* of a set $\mathbf{o}$ of outcomes onto variable X_i. We also define $\mathbf{o}[\mathbf{X}]$ as the projection of $\mathbf{o}$ onto a set of variables $\mathbf{X}$ and $\mathbf{o}[-i]$ as the projection of $\mathbf{o}$ onto $\mathbf{V} \setminus X_i$.

Definition 1. *A* (weak) preference relation $\succsim_S$ *is a partial order (a reflexive, antisymmetric, transitive relation) on a set of outcomes or their projection by a subject* S*.*

Where there is no possibility of confusion, we drop the subscript and use $\succsim$ as an infix operator. Informally, where $o_1, o_2 \in \mathcal{O}$, $o_1 \succsim o_2$ means that the subject considers the first outcome at least as good as the second. If it is also true that $o_2 \succsim o_1$, we say the subject is *indifferent*—equally satisfied with either outcome—and write $o_1 \sim o_2$. If $o_2 \not\succsim o_1$, we say the subject *strictly prefers* the first outcome to the second, written $o_1 \succ o_2$. If $o_1 \not\succsim o_2$ and $o_2 \not\succsim o_1$, the outcomes are *incomparable*, written $o_1 \bowtie o_2$. We use $\precsim$ and $\prec$ analogously. Table I shows how the $\succsim$ operator can be used to classify the relationship of a pair of outcomes in one of four ways.

TABLE I
Four Relational Classes of Outcome Pairs

$o_1 \succsim o_2$	$o_2 \succsim o_1$	Interpretation	Notation
F	F	incomparability	$o_1 \bowtie o_2$ or $o_2 \bowtie o_1$
F	T	strict preference	$o_2 \succ o_1$ or $o_1 \prec o_2$
T	F	strict preference	$o_1 \succ o_2$ or $o_2 \prec o_1$
T	T	indifference	$o_1 \sim o_2$ or $o_2 \sim o_1$

Definition 2. *A* ranking (chain) $\mathbf{r} = \langle r_1, r_2, \ldots, r_\ell \rangle$ *of outcomes or their projection is* monotonically worsening *if* $j < k$ *implies* $r_j \succsim r_k$*,* monotonically improving *if* $r_j \precsim r_k$*,* strictly worsening *if* $r_j \succ r_k$*, and* strictly improving *if* $r_j \prec r_k$*. We call any such ranking a* flipping sequence *if* r_j *and* r_{j+1} *differ in the value of just one variable* $X_i \in \mathbf{V}$ *for all* $j < \ell$*.*

Example 1. *Let* $\mathbf{V} = \{A, B, C, D\}$*. If* $A = \{a_1, a_2\}$ *and* $B = \{b_1, b_2, b_3\}$*, then* $a_1b_2 \succ a_2b_2 \sim a_2b_1 \succ a_1b_1$ *is a monotonically worsening flipping sequence from* a_1b_2 *to* a_1b_1 *on* $\{A, B\}$*. Such a sequence is a proof that* $a_1b_2 \succsim a_1b_1$*.*

Recall that a subject's preferences over some variable may depend on the value of another variable or set of variables. If such preferences consistently hold under the *ceteris paribus* assumption, they can be expressed formally as *conditional preference rules* and *tables*.

Definition 3. *A* conditional preference rule (CPR) *on a variable* $X_i \in \mathbf{V}$ *is an expression of the form* $\mathbf{u} : x^i_j \succsim x^i_k$ *or* $\mathbf{u} : x^i_j \not\succsim x^i_k$*, where* $\mathbf{u}$ *is an assignment to* $\mathbf{U}$ *(the* parents *of* X_i*), where* $\mathbf{U} \subseteq \mathbf{V} \setminus X_i$ *and* $x^i_j, x^i_k \in \mathrm{Dom}(X_i)$*. Note that* $\mathbf{U} = \emptyset$ *when the preferences over* X_i *are unconditional.*

Definition 4. *A* conditional preference table $\mathrm{CPT}(X_i)$ *of a variable* $X_i \in \mathbf{V}$ *is the set of all CPRs defined on* X_i*. We say that a CPT is* complete *if, after applying transitive closure, no pair of values from the domain of* X_i *is incomparable for any assignment to parent nodes; otherwise it is* incomplete*.*

Example 2. *Let* $\mathbf{V} = \{A, B\}$*,* $A = \{a_1, a_2, a_3, a_4\}$ *and* $B = \{b_1, b_2, b_3\}$*, such that the preference over* B *depends on the value of* A*. Specifically, whenever* $A = a_2$*, the subject strictly prefers* b_1 *to* b_2 *and* b_3*, but is indifferent given a choice* *between b_2 and b_3; i.e., $b_1 \succ b_2 \sim b_3$. This set of preferences can be expressed as an incomplete CPT with six CPRs:*

$$\begin{array}{ll} a_2 : b_1 \succsim b_2 & a_2 : b_2 \not\succsim b_1 \\ a_2 : b_1 \succsim b_3 & a_2 : b_3 \not\succsim b_1 \\ a_2 : b_2 \succsim b_3 & a_2 : b_3 \succsim b_2, \end{array}$$

which we may write equivalently (as in Figure 1) as:

$$\begin{array}{ll} a_2 : & b_1 \succ b_2 \\ a_2 : & b_1 \succ b_3 \\ a_2 : & b_2 \sim b_3. \end{array}$$

or even, where the ranking is complete, $a_2 : b_1 \succ b_2 \sim b_3$.

Recall that we model a lack of information as incomparability. If a CPT contains no rule $\mathbf{u} : x_j^i \succsim x_k^i$, then we assume $\mathbf{u} : x_j^i \not\succsim x_k^i$. This has important implications for incomplete CPTs. In particular, if a CPT is empty, as in Figure 1, then we assume that $x_j^i \bowtie x_k^i$ holds unconditionally for all $j, k \leq d_i$, $j \neq k$. Moreover, if no flipping sequence can be found from o_1 to o_2 or vice versa, we say that $o_1 \bowtie o_2$.

Definition 5. *A* CP-net *$\mathcal{N}$ is a directed graph. Each node represents a variable $X_i \in \mathbf{V}$ and is annotated with a CPT describing the subject's preferences over the domain of X_i given its dependencies. An edge (X_h, X_i) indicates that the preferences over X_i may depend directly on the value of X_h.*

We define the *size* of a CPT as the number of CPRs it contains and the size of a CP-net as the sum of the sizes of its CPTs. The set of all variables $\mathbf{U} \subseteq \mathbf{V} \setminus X_i$ on which the preference over X_i may depend are called the *parents* of X_i, denoted $\mathrm{Pa}(X_i)$. While in general CP-nets may contain cycles, here we assume $\mathcal{N}$ is acyclic.

A CP-net $\mathcal{N}$ induces a second type of graph known as the *preference graph* $\mathcal{P}$. The latter consists of a vertex (node) for each outcome $o_j \in \mathcal{O}$ with directed edges between a pair of nodes *only if* they differ in the value of just one variable $X_i \in \mathbf{V}$. A directed edge from the vertex of o_k to that of o_j, where $o_j[-i] = o_k[-i]$ and $o_j[i] \neq o_k[i]$, indicates that $o_j \succsim o_k$. One may observe that the preference graph is a subgraph of the n-dimensional *polytope* consisting of all possible outcomes. (When restricting to binary domains, the resulting polytope is a *hypercube*.) For this reason, we sometimes refer informally to the induced preference graph as the *polytope* of $\mathcal{N}$. Moreover, a monotonically worsening flipping sequence from o_j to o_k corresponds to the existence of a path along directed edges from the vertex of o_k to that of o_j in the preference graph. It can be seen that the number of vertices in the preference graph is exponential in the number of nodes n in the corresponding CP-net. For this reason, CP-nets are a type of *compact preference representation*.

III. The Value of Indifference

Most discussions of CP-nets assume strict preferences over binary domains. However, such assumptions are often too restrictive for even simple problems. Consider for example a child's preferences over a set of blocks in a nursery. Each block is of a particular color and labeled with a single letter and numeral. The child prefers the *green* blocks to those of other colors, but if a block is *green*, she is indifferent as to the letter or numeral. Such preferences cannot be modeled with a CP-net, however, if we require strict rankings for the conditional preference rules. A learning algorithm that did not take into account the possibility of indifference would thus either output an unnecessarily complex model or infer erroneously that the child's preferences were inconsistent. In our automated home example a similar situation could emerge if the subject preferred only that the television be on so as to provide background noise, but was indifferent as to the channel or specific programming. Our model should not force us to misinterpret such indifference as irrationality.

The tacit assumption that human preferences are strict and complete is also too strong from a psychological standpoint. Popova et al. [6] offer perspective as quantitative psychologists on the tendency in the artificial intelligence community to assume strict, complete rankings even when these do not exist in data. The authors argue that this tendency can introduce biases, particularly when large, complex datasets are massaged so as to extract strict, complete rankings: "treating preferences as strict linear orders or strict weak orders may require researchers to impute vast amounts of information not provided by the voters or raters." An example of this in the quantitative preference literature (not mentioned in [6]) is the choice to eliminate "unfamiliar or low frequency items" from the familiar sushi dataset of Kamishima in order to produce complete rankings [7], a precedent followed by subsequent researchers. By explicitly allowing for indifference and modeling incomplete information as incomparability, we can be more faithful to such datasets.

Third, the combinatorial space of the reasoning and learning problems can potentially be reduced if the model permits indifference and incomparability. Consider a customer perusing a wine menu with dozens of alternatives. The customer enjoys wine, but does not consider himself a connoisseur. His preference over wines depends on the entrée. He hopes to order the grilled salmon and has heard that a Pinot grigio would be a good accompaniment; his "second" choice would be any of the many white wines on the menu. But as to *which* Pinot grigio or white wine, he is unable to state a preference. If indifference and incomparability are allowed, the customer's preferences are considerably simplified. In essence we have

$$\text{salmon} : \text{Pinot grigio} \succ \text{other white} \succ \text{everything else}$$

with all Pinot grigio wines and other white wines collapsed into *equivalence classes*. Similarly, "everything else" can be omitted from the CPT where salmon is the entrée, hence collapsing those alternatives into an *antichain*. On the other hand, the magnitude of the learning and reasoning problems (not to mention the customer's frustration) are significantly increased if we require him also to compare each of a dozen Merlots with each Cabernet, given salmon as the entrée.

More abstractly, consider a variable for which the CPT is empty (all values are incomparable) or over which the subject is completely indifferent. In either case, if the domain represents a set of available alternatives, it can be seen that the values could be merged to create a domain of size one, effectively eliminating the variable from the model altogether. Since the complexity of the learning and reasoning problems depends on the number of variables and the size of the domains, eliminating a variable shrinks the resulting search space, possibly leading to more efficient solutions. In short, far from adding unnecessary complexity, allowing for indifference, incomparability and incompleteness over multivalue domains could yield simpler models.

IV. Learning with Indifference

CP-net models can be explicitly constructed, actively elicited or learned passively from observational data. Each approach has its strengths and weaknesses. The first requires significant expert knowledge; while the graphical structure of CP-nets is intuitive, developing a custom network for some complex domain—correctly modeling the variables, identifying dependencies, deciding which preferences must be fully modeled and which are unlikely to prove useful—may not always be practical. The second method may require a lengthy intake process before the application can be used, but provides a model at the outset, whereas the third method may require days or weeks of observations to infer what the subject wants. Here we take the third approach, extending the work of Dimopoulos et al. [8] to learn CP-nets from outcome comparison data. The original algorithm assumes strict preferences over binary domains. We show how it can be extended to learn CP-nets with weak local preferences over multivalue domains.

Here we are only able to provide an overview of the algorithm in [8]; the reader is referred to the original paper for details. That algorithm takes as its input a set of binary variables and a set of pairwise comparisons between outcomes. It outputs an acyclic CP-net consistent with the input or reports failure. The algorithm initializes an empty CP-net and attempts to extend it one node at a time. It first searches for variables for which the preferences are unconditional (i.e., nodes with 0 parents). Search then continues over the unadded variables for those with 1 parent, 2, and so on, from the added nodes, until each variable has been added to the CP-net with edges from parent nodes and a CPT consistent with comparison data. Since edges are added only from existing to newly added nodes, no cycle is introduced. The authors show that learning a CP-net from comparison data is NP-hard even under certain simplifying assumptions; in the worst case, the algorithm they propose could require exponential time in the number of variables in the input.

A critical step involves determining whether a particular set of previously added nodes can be the parents of a given variable. To decide this, the algorithm uses a SAT reduction in an effort to find a CPT for the variable that is consistent with the comparison data. (For this we have adapted the authors' notation, particularly that involving the Boolean variables, to make it consistent with our own in the subsequent discussion.) Each pair of outcomes in the database is factored as

$$o_i \succ o_j \;\equiv\; \mathbf{u}_i v_i \mathbf{w}_i \succ \mathbf{u}_j v_j \mathbf{w}_j, \text{ where } v_i \neq v_j$$

such that v_i and v_j are the values of the variable under consideration, $\mathbf{u}_i$ and $\mathbf{u}_j$ are the values of the prospective parents, and $\mathbf{w}_i$ and $\mathbf{w}_j$ are the values of the other (uninvolved) variables. For each combination of parent values, a Boolean variable $z_{\mathbf{u}_k}$ is defined such that the variable is true iff the CPR $\mathbf{u}_k : v_i \succ v_j$ is included in the target node's CPT. Since variables are binary (v_i and v_j represent *all* the values of the variable under consideration) and all preferences are strict, only one Boolean variable and its complement are needed for each combination of parent variables:

$$\begin{aligned} z_{\mathbf{u}_k} &\equiv \mathbf{u}_k : v_i \succ v_j \\ \neg z_{\mathbf{u}_k} &\equiv \mathbf{u}_k : v_j \succ v_i. \end{aligned}$$

Given this mapping, for each pair of outcomes, the appropriate clause is conjuncted to formula ϕ (initially $\phi = \emptyset$) from the following:

$$\begin{aligned} (\mathbf{u}_i : v_i \succ v_j) \vee (\mathbf{u}_j : v_i \succ v_j) &\equiv z_{\mathbf{u}_i} \vee z_{\mathbf{u}_j} \\ (\mathbf{u}_i : v_i \succ v_j) \vee (\mathbf{u}_j : v_j \succ v_i) &\equiv z_{\mathbf{u}_i} \vee \neg z_{\mathbf{u}_j} \\ (\mathbf{u}_i : v_j \succ v_i) \vee (\mathbf{u}_j : v_i \succ v_j) &\equiv \neg z_{\mathbf{u}_i} \vee z_{\mathbf{u}_j} \\ (\mathbf{u}_i : v_j \succ v_i) \vee (\mathbf{u}_j : v_j \succ v_i) &\equiv \neg z_{\mathbf{u}_i} \vee \neg z_{\mathbf{u}_j} \end{aligned}$$

Observe that the resulting formula ϕ is 2SAT, which can be solved in linear time using Tarjan's algorithm for finding strongly connected components. If ϕ is satisfiable, a mapping from a satisfying assignment provides a CPT for the variable, which is then added as a node to the CP-net along with edges from the newly discovered parent nodes.

The authors note in [8] that while binary variables are assumed, "the same basic idea applies for non-binary variables." This is indeed the case, even when indifference is allowed. The search over variables for parent nodes proceeds just as before; however, the SAT reduction must be adapted. In extending to weak preferences over multivalue domains, the comparisons in the database may be of four types: In addition to $o_1 \succ o_2$ and $o_2 \succ o_1$, the user may also specify $o_1 \sim o_2$ and $o_1 \bowtie o_2$. From these we obtain the truth values of $o_1 \succsim o_2$ and $o_2 \succsim o_1$ as in Table I. This database of outcome comparisons is then decomposed as in the original algorithm:

$$o_i \succsim o_j \;\equiv\; \mathbf{u}_i v_i \mathbf{w}_i \succsim \mathbf{u}_j v_j \mathbf{w}_j, \text{ where } v_i \neq v_j.$$

One can observe that the resulting clauses are 2CNF as before. The mapping to Boolean variables is also similar; however, in our case we have CPRs for each combination of parent values *and all ordered pairs from the domain of the variable under consideration*. We thus define Boolean variables $z_{\mathbf{u}_k,i,j}$ that are true iff the CPR $\mathbf{u}_k : v_i \succsim v_j$ is included in the CPT of the variable under consideration. Then for each pair of outcomes, we conjunct to ϕ the appropriate clause from the following:

$$\begin{aligned} (\mathbf{u}_i : x_i \succsim x_j) \vee (\mathbf{u}_j : x_i \succsim x_j) &\equiv z_{\mathbf{u}_i,i,j} \vee z_{\mathbf{u}_j,i,j} \\ (\mathbf{u}_i : x_i \succsim x_j) \vee (\mathbf{u}_j : x_i \not\succsim x_j) &\equiv z_{\mathbf{u}_i,i,j} \vee \neg z_{\mathbf{u}_j,i,j} \\ (\mathbf{u}_i : x_i \not\succsim x_j) \vee (\mathbf{u}_j : x_i \succsim x_j) &\equiv \neg z_{\mathbf{u}_i,i,j} \vee z_{\mathbf{u}_j,i,j} \\ (\mathbf{u}_i : x_i \not\succsim x_j) \vee (\mathbf{u}_j : x_i \not\succsim x_j) &\equiv \neg z_{\mathbf{u}_i,i,j} \vee \neg z_{\mathbf{u}_j,i,j} \end{aligned}$$

Fig. 2. Algorithm PRINT-TRANSITIVITY-RULES(X)

```
for $\mathbf{u} \in \mathrm{Asst}(\mathrm{Pa}(X_i))$ do
    for all $\{\{a, b\}, c\} \leq d_i$ do
        $z_{\mathbf{u},a,b} \wedge z_{\mathbf{u},b,c} \Rightarrow z_{\mathbf{u},a,c}$
    end for
end for
```

One additional difficulty must be addressed. With binary domains (whether restricting to strict preferences or not), the CPT will *always* be consistent, given how we have defined the Boolean literals. However, for multivalue domains, this is not the case. For these we must take into account the possibility of inconsistency through violation of transitivity. Consider, for example, that the solver could output a satisfying assignment mapping to CPRs such as:

$$\begin{array}{ll} a_1 : b_1 \succsim b_2 & a_1 : b_2 \not\succsim b_1 \\ a_1 : b_1 \not\succsim b_3 & a_1 : b_3 \succsim b_1 \\ a_1 : b_2 \succsim b_3 & a_1 : b_3 \not\succsim b_2, \end{array}$$

corresponding to $a_1 : b_1 \succ b_2 \succ b_3 \succ b_1$. To obviate such contradictions, we output to the solver (i.e., conjunct to ϕ) additional clauses to enforce transitivity (see Fig. 2). However, when the additional rules are transformed, we get

$$\neg z_{\mathbf{u},a,b} \vee \neg z_{\mathbf{u},b,c} \vee z_{\mathbf{u},a,c}$$

with the result that, for multivalue domains, the formula ϕ constructed by this method is 3SAT. Because of this, we call a solver (MiniSAT) that uses a heuristic for the NP-complete subproblem rather than using a known polynomial time algorithm. Moreover, as with the original algorithm, in the worst case the SAT solver could be called exponentially many times in the number of variables $|\mathbf{V}|$.

V. REASONING WITH INDIFFERENCE

Once a CP-net model is available, some method is needed for determining which of a pair of outcomes the subject will prefer, if either. Formally the question of whether a CP-net entails the dominance of one outcome over another $\mathcal{N} \models o_1 \succ o_2$ is known as *dominance testing* (DT). Goldsmith et al. have shown that in general this problem is PSPACE-complete [9]. As Boutillier et al. recognized, $\mathcal{N} \models o_1 \succ o_2$ iff there is a (strictly) improving flipping sequence from o_2 to o_1, and the search for such a sequence can be formulated as a STRIPS type planning problem [5]. Here we consider the more general question of *weak dominance*—i.e., whether $\mathcal{N} \models o_1 \succsim o_2$. One can observe that the latter corresponds to a *monotonically* worsening flipping sequence from o_1 to o_2 (or, identically, a monotonically improving flipping sequence from o_2 to o_1). By searching for flipping sequences in both directions (from o_1 to o_2 and vice versa), we can reason from the model whether the subject strictly prefers one outcome to the other, is indifferent, or whether no information can be obtained from the CP-net on the preference (see Table I).

Here, as with the learning problem in Section IV, we employ a reduction to SAT. However, unlike the problem of deciding

Fig. 3. Algorithm SAT-FLIP($\mathcal{N}, n, o_1, o_2, L$)

Input: CP-net $\mathcal{N}$ with n variables, outcomes o_1 and o_2, and L the longest allowed flipping sequence

Output: CNF formula ϕ that is satisfiable iff $\mathcal{N}$ entails a monotonically worsening flipping sequence from o_1 to o_2 of length $\ell \leq L$

```
for $t \leftarrow 1$ to $L$ do
    for $i \leftarrow 1$ to $n$ do
        $\mathfrak{JustOne}$(state $z$ at time $t$, node $i$)
    end for
end for
for $t \leftarrow 1$ to $L - 1$ do
    for $i \leftarrow 1$ to $n$ do
        for distinct $i, j \leq d_i$ do
            $\alpha_{t,i,j,k} \Rightarrow z_{t,i,j} \wedge z_{t+1,i,k}$
            for $h \leftarrow 1$ to $n$ s.t. $h \neq i$ do
                for $q \leftarrow 1$ to $d_h$ do
                    $\alpha_{t,i,j,k} \wedge z_{t,h,q} \Rightarrow z_{t+1,h,q}$
                    $\alpha_{t,i,j,k} \wedge \neg z_{t,h,q} \Rightarrow \neg z_{t+1,h,q}$
                end for
            end for
        end for
        $\alpha_{t,0} \wedge z_{t,i,p} \Rightarrow z_{t+1,i,p}$
        $\alpha_{t,0} \wedge \neg z_{t,i,p} \Rightarrow \neg z_{t+1,i,p}$
    end for
    $\mathfrak{JustOne}$(action $\alpha$ at time $t$)
    for each CPR in $\mathcal{N}$ of the form $\mathbf{u} : x_j^i \not\succsim x_k^i$ do
        $z_{t,p_1,u_1} \wedge \cdots \wedge z_{t,p_e,u_e} \Rightarrow \neg \alpha_{t,i,j,k}$
    end for
end for
$\alpha_{t,0} \Rightarrow \alpha_{t+1,0}$ for all $t \leq L - 2$
assert $o_1$ and $o_2$ as the states at $t = 1$ and $L$
```

whether a particular CPT is consistent with comparison data, the search for flipping sequences involves a series of actions and steps. In our approach, we thus make use of the *satisfiability as planning* (SATPlan) method proposed by Kautz and Selman [10]. One potential issue is that in the worst case flipping sequences—and hence the number of steps—could be exponential in the number of nodes n. We address this possibility in Section VI. In the meantime, we assume that flipping sequence length can be bounded by a constant $\ell_{\max}$.

The algorithm in Fig. 3 provides the SATPlan reduction. Its input is a CP-net, its number of nodes, the pair of outcomes to compare, and the longest flipping sequence to be considered. It outputs a CNF formula ϕ.

Consider the flipping sequence given in Example 1: $a_1b_2 \succ a_2b_2 \sim a_2b_1 \succ a_1b_1$. In the SATPlan reduction we regard such a sequence as a plan requiring four time steps. The first term a_1b_2 reflects the states of A and B at $t = 1$, a_2b_2 represents the states at $t = 2$, etc. The change in the state of A from a_1 to a_2 corresponds to a particular *action* that occurs at $t = 1$.

We define a Boolean variable $z_{t,i,p}$ for the *state* of each variable X_i over time such that $z_{t,i,p}$ is true iff $X_i = x_p^i$ at time t, where $t \leq L$, $i \leq n$, and $p \leq |X_i|$. Just one such state applies to each variable X_i at each time t. For a set of literals Y, it is helpful to define $\mathfrak{JustOne}(Y)$, the rules specifying that *at least* one literal $y \in Y$ holds $(y_1 \vee y_2 \vee \cdots)$ and that *at most* one literal holds (for distinct $y, y' \in Y$, $\neg y \vee \neg y'$). Here we assert $\mathfrak{JustOne}(z_{t,i,p})$ for all t, i. We also define Boolean variables $\alpha_{t,i,j,k}$ for each possible *action*. Recall that in a flipping sequence, at each time t the value of just one variable changes. Thus, for all $t < L$, $i \leq n$, and distinct $x^i_j, x^i_k \in \mathrm{Dom}(X_i)$, there is a possible action corresponding to a monotonically worsening flip from x^i_j to x^i_k. We output these to the solver as rules of the form:

$$\alpha_{t,i,j,k} \Rightarrow z_{t,i,j} \wedge z_{t+1,i,k}. \tag{1}$$

We next output *framing rules* specifying that, if an action causes the value of some variable X_i to change, the values of every other variable X_h must remain unchanged.

$$\alpha_{t,i,j,k} \wedge z_{t,h,q} \quad \Rightarrow \quad z_{t+1,h,q} \tag{2}$$
$$\alpha_{t,i,j,k} \wedge \neg z_{t,h,q} \quad \Rightarrow \quad \neg z_{t+1,h,q} \tag{3}$$

Since the length of a flipping sequence may be strictly less than L, we also define *null actions* $\alpha_{t,0}$ specifying that no change occurs at time t or later:

$$\alpha_{t,0} \wedge z_{t,i,p} \quad \Rightarrow \quad z_{t+1,i,p} \tag{4}$$
$$\alpha_{t,0} \wedge \neg z_{t,i,p} \quad \Rightarrow \quad \neg z_{t+1,i,p} \tag{5}$$
$$\alpha_{t,0} \quad \Rightarrow \quad \alpha_{t+1,0} \qquad (t \leq L-2). \tag{6}$$

At each time step exactly one action or the null action must occur, so we assert $\mathfrak{JustOne}$ $\alpha_{t,i,j,k}$ or $\alpha_{t,0}$ for all $t < L$.

We further specify rules corresponding to the CPRs in the CPTs of $\mathcal{N}$. The relevant CPRs are those that occur in the *negative*. A rule of the form $\mathbf{u} : x^i_j \not\succsim x^i_k$, where $\mathbf{u} = u_1 u_2 \cdots u_e \in \mathrm{Asst}(\mathrm{Pa}(X_i))$, means a monotonically worsening flip cannot occur from x^i_j to x^i_k in X_i when that node's parents are assigned the values in $\mathbf{u}$; hence action $\alpha_{t,i,j,k}$ cannot occur under such circumstances. Let $X_{p_1}, X_{p_2}, \ldots$ denote the parents of X_i, such that $X_{p_1} = x^{p_1}_{u_1}, X_{p_2} = x^{p_2}_{u_2}$, etc. Observe that at time t in the flipping sequence, these assignments correspond to the state variables z_{t,p_1,u_1}, z_{t,p_2,u_2}, etc. We thus output:

$$z_{t,p_1,u_1} \wedge z_{t,p_2,u_2} \wedge \cdots \wedge z_{t,p_e,u_e} \Rightarrow \neg \alpha_{t,i,j,k}. \tag{7}$$

Finally, we output as assertions o_1 and o_2 as the states at times $t = 1$ and $t = L$ respectively.

Special care is required if CPTs may be incomplete. Proper interpretation of an incomplete CPT likely depends upon the application and how the CPT was constructed. Here we treat incomplete CPTs as incomplete information about the subject's preferences. As such, during preprocessing, we: (1) Check CPTs for consistency given a specific assignment to parents and raise an exception if an inconsistency is found. (2) Complete the transitive closure. (3) Otherwise, in the absence of a rule specifying that $\mathbf{u} : x^i_j \succsim x^i_k$ or inferred through closure, we assume $\mathbf{u} : x^i_j \not\succsim x^i_k$. That is, in the absence of information about the relationship between values x^i_j and x^i_k given an assignment to parents, we assume incomparability.

The algorithm in Fig. 4 shows how to use the reduction subroutine in a generalized search for a flipping sequence. It takes as its input a CP-net, its number of nodes, the pair of outcomes to compare, and a bound on flipping sequence length. It returns a monotonically worsening flipping sequence from o_1 to o_2 with length at most $\ell_{\max}$ or $\emptyset$ if no such sequence exists. We call this latter result *indeterminate* since a longer sequence could exist if $\ell_{\max} < |\mathcal{O}|$. The algorithm maintains an upper and a lower bound for flipping sequence length. The lower bound is initialized to the Hamming Distance between the two outcomes since the flipping sequence must be at least this length. Some suitable value L in this range is chosen, a CNF formula ϕ created, and the SAT solver called. If ϕ is satisfiable, a mapping from a satisfying assignment τ provides the desired flipping sequence. However, unless the upper and lower bounds are the same, the flipping sequence may not be minimal. (Indeed, it is possible that it contains a cycle, in which case a shorter flipping sequence *must* exist.) This may not be of concern if we only wish to show that $o_1 \succsim o_2$, but for some applications it may be desirable to find a minimal sequence. Search continues until a suitable sequence is found or is found not to exist.

Fig. 4. Algorithm FIND-FLIPPING-SEQUENCE($\mathcal{N}, n, o_1, o_2, \ell_{max}$)

Input: CP-net $\mathcal{N}$, outcomes o_1 and o_2, and $\ell_{\max}$ the longest flipping sequence to consider in the search

Output: monotonically worsening flipping sequence **FS** from o_1 to o_2 of length $\ell \leq \ell_{\max}$ (or indeterminate if none found)

```
1: FS ← $\emptyset$
2: $\ell_{\min}$ ← Hamming-Distance($o_1, o_2$)
3: repeat
4:     $L$ ← a suitable value in the interval ($\ell_{\min} \, .. \, \ell_{\max}$)
5:     $\phi$ ← SAT-FLIP($\mathcal{N}, n, o_1, o_2, L$)
6:     $\tau$ ← SAT-Solver($\phi$)
7:     if $\tau \neq \emptyset$ (if satisfiable) then
8:         FS ← sequence obtained from $z_{t,i,p}$ values in $\tau$
9:         $\ell_{\max}$ ← $L - 1$
10:    else
11:        $\ell_{\min}$ ← $L + 1$
12:    end if
13: until (FS $\neq \emptyset$ and $\ell_{\min} \geq \ell_{\max}$) or we give up
14: return FS (or $\emptyset$ if indeterminate)
```

VI. A Consideration of Long Flipping Sequences

The algorithms in Fig. 3 and 4 assume that the length of a flipping sequence can be bounded by a constant. However, Boutillier et al. [5] proved that—for CP-nets with domains of size 3 or greater and incomplete CPTs, such as we consider here—a minimal flipping sequence between outcomes may be exponentially long in the number of nodes in the worst case. Thus, the problem of reasoning with CP-nets is not even in NP: that is, even if we could somehow *guess* a correct flipping sequence, the problem of *verifying* it could turn out to be intractable.

Nevertheless, there is reason to believe that this worst-case scenario is rare. To test this, we generated a series of random CP-nets using the algorithms in Fig. 5 and 6. For each experiment we varied the density $\delta \in [0, 1]$ of the dependency graph—defined here as the ratio of directed edges in $\mathcal{N}$ to the maximum number of such edges possible in an acyclic graph with $|\mathbf{V}|$ nodes. We similarly varied the density $\varepsilon \in [0, 1]$ of the $\mathrm{CPT}(X_i)$ of each node—defined as the ratio of rules of the *positive* form $\mathbf{u} : x^i_j \succsim x^i_k$ (as opposed to the *negative* form $\mathbf{u} : x^i_j \not\succsim x^i_k$) to the total number

TABLE II
FIRST EXPERIMENT: $n = 4$, $d_i = 4$

Mean flipping sequence length over all outcomes					
ε	$\delta = 0.10$	0.30	0.50	0.70	0.90
0.10	1.8	2.2	2.6	2.9	3.0
0.30	2.0	2.7	2.9	3.0	3.0
0.50	2.6	3.5	3.4	3.0	3.0
0.70	3.0	3.3	3.3	3.1	3.0
0.90	3.0	3.5	3.4	3.1	3.0

Longest flipping sequence for any outcome pair					
ε	$\delta = 0.10$	0.30	0.50	0.70	0.90
0.10	4.0	4.0	4.0	4.0	4.0
0.30	5.0	6.5	6.1	5.0	4.0
0.50	7.2	9.7	7.9	5.3	4.0
0.70	8.3	9.0	8.1	5.3	4.0
0.90	8.5	8.7	7.5	5.4	4.0

TABLE III
SECOND EXPERIMENT: $n = 6$, $d_i = 3$

Mean flipping sequence length over all outcomes					
ε	$\delta = 0.10$	0.30	0.50	0.70	0.90
0.10	1.9	3.0	3.3	4.0	4.0
0.30	2.2	3.9	3.8	4.0	4.0
0.50	2.7	4.1	4.3	4.0	4.0
0.70	3.1	4.4	4.3	4.0	4.0
0.90	3.7	4.5	4.3	4.0	4.0

Longest flipping sequence for any outcome pair					
ε	$\delta = 0.10$	0.30	0.50	0.70	0.90
0.10	6.3	6.9	6.8	6.0	6.0
0.30	6.8	10.6	8.9	6.0	6.0
0.50	8.5	10.8	10.8	6.0	6.0
0.70	9.2	11.3	9.6	6.0	6.0
0.90	11.0	11.6	9.2	6.0	6.0

of possible rules. From each randomly generated CP-net $\mathcal{N}$ we then explicitly constructed the induced preference graph $\mathcal{P}$, which we represented as a sparse adjacency matrix. We then used the Floyd–Warshall algorithm to find the shortest path between all pairs of outcomes. (Recall from Section II that a flipping sequence corresponds to a path between the corresponding outcome vertices in the preference graph.) We ran $N = 10$ trials for each value of δ and ϵ and calculated the mean length of the longest flipping sequence in the induced preference graph, as well as the mean flipping sequence length over all pairs of outcomes.

Table II shows the results of this experiment for CP-nets with 4 variables each with a domain of size 4. While the induced preference graph consists of $4^4 = 256$ nodes, the mean length of a flipping sequence for a randomly selected pair of outcomes from a random CP-net was typically less than 4, while the longest flipping sequence typically ranged between 4 (the value of n) and 10. Table III shows similar results for CP-nets with 6 variables, each with a domain of size 3. While the preference graph for the latter consists of $3^6 = 729$ outcomes, the mean length of a minimal flipping sequence over all pairs of outcomes was typically less than 4.5, while the mean longest flipping sequence between any pair of outcomes was less than 12. It can also be observed that, as $\delta \to 1$, the length of the longest flipping sequence converges to n, as might be expected. We ran similar experiments for other choices of n and d_i, but did not come across any randomly generated instances in which the length of a flipping sequence was apparently exponential in n. We expect in future research to derive a more precise statistical model for the expected length of a flipping sequence in a random CP-net.

It is nonetheless possible that some class of randomly generated CP-nets or some domain in which CP-nets are learned exists for which the worst case is more common. Even so, a very long flipping sequence is unlikely to provide useful information about the preferences of a human subject. To see why, consider that we have a CP-net $\mathcal{N}$ representing the preferences of a subject and a pair of outcomes o_1, o_2 with a very long flipping sequence from o_1 to o_2. We observed in Section I that—for the class of CP-nets considered here—preferences are modeled deterministically rather than as probability distributions; however, it should be understood that this determinism is a *modeling* decision, not an intrinsic property of the subject's underlying preferences. Whether $\mathcal{N}$ is constructed by the subject, elicited through queries or learned from data, we assume the model reflects some margin of error. If such errors are small, they can be safely ignored. However, for very long flipping sequences, even small errors are problematic. Consider that in a flipping sequence, each transition (action) from one term (step) to the next is determined by a particular CPR, $\rho_1, \rho_2, \ldots, \rho_{\ell-1}$. (Note that some CPRs may be applied more than once.) Let $\eta_k \in (0, 1)$ be the probability of error associated with CPR ρ_k. We can then speak of the probability that a determined flipping sequence actually exists and observe that this probability tends to 0 as ℓ increases:

$$\lim_{\ell \to \infty} \prod_{k=1}^{\ell} (1 - \eta_k) = 0.$$

Informally, we can compare a flipping sequence to an electrical signal that weakens with distance; eventually the signal is no longer detectable. We are thus justified in assuming a bound $\ell_{\max}$ on the length of a flipping sequence and reasoning that if no monotonically worsening flipping sequence from o_1 to o_2 with length $\ell < \ell_{\max}$ exists, then it is probable that *no* such flipping sequence actually exists, in which case we report simply that $o_1 \not\succeq o_2$ (rather than indeterminate). Observe that this latter result depends on the values of η_k rather than n; thus we can drop use of the modifier *exponentially* and simply speak of *very* long flipping sequences; i.e., those for which $\ell > \ell_{\max}$. We believe in practice the value of $\ell_{\max}$ will turn out to be domain specific and as such is a model parameter that is best learned from data. We observe in passing that the analysis here is closely related to the concept of *noisy AND* [11] and also to PCP-nets, a recently proposed probabilistic extension of CP-nets [12], [13].

Fig. 5. Algorithm GENERATE-RANDOM-CPNET$(\mathbf{V}, \delta, \varepsilon)$

```
Input:   V    variables with associated domains
         δ    upper bound on dependency graph density
         ε    lower bound on density of ≿
Output:  𝒩    randomly generated acyclic CP-net

1: initialize nodes for $\mathcal{N}$
2: let $\pi$ be a random permutation of the nodes in $\mathbf{V}$
3: initialize $E$ and add edges $(\pi(X_i), \pi(X_j))$ for all $i > j$
4: while $|E|/\binom{|\mathbf{V}|}{2} > \delta$ do
5:     delete an edge selected at random
6: end while
7: for all $X_i \in \mathbf{V}$ do
8:     $\mathrm{CPT}(X_i) \leftarrow$ GENERATE-RANDOM-CPT$(X_i, \varepsilon)$
9: end for
10: return $\mathcal{N}$
```

Fig. 6. Algorithm GENERATE-RANDOM-CPT(X_i, ε)

```
Input:   X_i   node of a CP-net with known parents
         ε     lower bound on ratio of ≿ rules given u
Output:  CPT   a complete, consistent CPT for X_i

1: CPT $\leftarrow \emptyset$
2: for $\mathbf{u} \in \mathrm{Asst}(\mathrm{Pa}(X_i))$ do
3:     $d_i \leftarrow |\,\mathrm{Dom}(X_i)|$
4:     add rules $\mathbf{u} : x^i_j \succsim x^i_j$ to CPT for all $j \leq d_i$
5:     while $\left(\left|\text{rules } \mathbf{u} : x^i_j \succsim x^i_k\right| - d_i\right) / \left(d_i^2 - d_i\right) < \epsilon$ do
6:         randomly select and add a rule $\mathbf{u} : x^i_j \succsim x^i_k \notin$ CPT
7:         take the transitive closure of all $\succsim$ rules given $\mathbf{u}$
8:     end while
9:     add rule $\mathbf{u} : x^i_j \not\succsim x^i_k$ for all $i, j$ s.t. $\mathbf{u} : x^i_j \succsim x^i_k \notin$ CPT
10: end for
11: return CPT
```

VII. Contributions and Future Research

We identified a need for CP-nets that can model preferences with multi-value domains over which a subject may be indifferent or unwilling to state a preference. We showed how to formulate CPRs and CPTs that explicitly model indifference and incomparability over multivalue domains. We next showed how to extend an algorithm designed to learn strict preferences over binary domains from example comparison data [8] to learn this richer class of CP-nets. As in the original algorithm, we used a SAT reduction to find CPTs consistent with comparison data. However, for multivalue domains, we showed that it was necessary to output additional clauses to ensure transitivity, resulting in a 3SAT rather than 2SAT instance. We further showed that SAT can be used to reason with this richer class of CP-nets using SATPlan. Our method for this assumes a bound on the length of a flipping sequence. This is potentially problematic, since, as [5] proved, it is possible to construct CP-nets for which the flipping sequences are exponentially long. Our data, however, suggest that such flipping sequences are rare. Moreover, we showed that, if the CPRs admit even some small probability of error, very long flipping sequences are unlikely to provide useful information about a subject's preferences. Future work includes comparing the performance of various learning and reasoning algorithms and extending our present research to encompass probabilistic as well as deterministic CP-nets.

References

[1] F. Bacchus and A. Grove, "Graphical models for preference and utility," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1995, pp. 3–10.

[2] P. Fishburn, "Preference structures and their numerical representations," *Theoretical Computer Science*, vol. 217, no. 2, pp. 359–383, 1999.

[3] S. Bistarelli, U. Montanari, and F. Rossi, "Semiring-based constraint satisfaction and optimization," *Journal of the ACM (JACM)*, vol. 44, no. 2, pp. 201–236, 1997.

[4] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier, "Semiring-based csps and valued csps: Frameworks, properties, and comparison," *Constraints*, vol. 4, no. 3, pp. 199–240, 1999.

[5] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole, "CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements," *Journal Artificial Intelligence Research*, vol. 21, pp. 135–191, February 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622467.1622473

[6] A. Popova, M. Regenwetter, and N. Mattei, "A behavioral perspective on social choice," *Annals of Mathematics and Artificial Intelligence*, pp. 1–26, 2013.

[7] T. Kamishima, "Nantonac collaborative filtering: recommendation based on order responses," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '03. New York, NY, USA: ACM, 2003, pp. 583–588. [Online]. Available: http://doi.acm.org/10.1145/956750.956823

[8] Y. Dimopoulos, L. Michael, and F. Athienitou, "Ceteris paribus preference elicitation with predictive guarantees," in *Proceedings of the 21st International Joint conference on Artificial intelligence (IJCAI-09)*, ser. IJCAI'09. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 1890–1895.

[9] J. Goldsmith, J. Lang, M. Truszczyński, and N. Wilson, "The computational complexity of dominance and consistency in CP-nets," in *Proc. IJCAI*, 2005.

[10] H. Kautz and B. Selman, "Planning as satisfiability," in *IN ECAI-92*. Wiley, 1992, pp. 359–363.

[11] F. J. Dıez and M. J. Druzdzel, "Canonical probabilistic models for knowledge engineering," Technical Report CISIAD-06-01, UNED, Madrid, Spain, Tech. Rep., 2006.

[12] C. Cornelio, J. Goldsmith, N. Mattei, F. Rossi, and K. B. Venable, "Updates and uncertainty in CP-nets," in *26th Australasian Joint Conference on Artificial Intelligence*, 2013, to Appear.

[13] D. Bigot, H. Fargier, J. Mengin, and B. Zanuttini, "Probabilistic conditional preference networks," *Septièmes Journées d'Intelligence Artificielle Fondamentale*, p. 57, 2013.