

Preferences over Sets

C. Domshlak

Dept. of Industrial Eng.
Technion, Haifa, Israel
dcarmel@ie.technion.ac.il

R. I. Brafman and S. E. Shimony* and Y. Silver

Dept. of Computer Sci.
Ben-Gurion University, Beer-Sheva, Israel
{ brafman, shimony, yaelsi } @cs.bgu.ac.il

Abstract

Research on preference elicitation and reasoning typically focuses on preferences over single objects of interest. However, in a number of applications the “outcomes” of interest are *sets* of such atomic objects. For instance, when creating the program for a film festival, editing a newspaper, or putting together a team, we need to select a set of films (resp. articles, members) that is optimal with respect to quality, diversity, cohesiveness, etc. This paper describes an intuitive approach for specifying preferences over sets of objects. An algorithm for computing an optimal subset, given a set of candidate objects and a preference specification, is developed and evaluated.

Introduction

The area of eliciting, modeling, and reasoning with preferences has experienced much interest in recent years. Most work in this area concentrates on specifying preferences over some universe of objects \mathcal{O} , and using this information for various tasks of preferential reasoning such as ordering a given subset of \mathcal{O} , identifying optimal objects within a subset of \mathcal{O} specified implicitly via hard constraints, etc. Examples of such tasks include selecting flights, presenting documents in their perceived importance order, customized product configuration for personal computers, etc.

In most practical applications, the universe of objects \mathcal{O} is combinatorial, that is, stands for a cross-product of a certain set of attributes. Since the size of such a universe \mathcal{O} is exponential in the number of attributes, the user cannot be expected to provide an explicit ordering over \mathcal{O} . The user can, however, provide some generic preferences over attribute values that would implicitly order \mathcal{O} . Much of the work in the area of reasoning with preferences has to do with processing a concrete class of statements that users find convenient to specify, adopting and justifying a certain semantics for these statements, and developing algorithms that use these statements to compare objects or select the best ones.

This paper deals with similar issues as well. However, unlike most past work which attempts to reason with preferences over *objects*, we are interested in similar forms of

reasoning over *sets of objects*. More specifically, the setting of our problem is: *Given a set of objects \mathcal{O} , elicit a (possibly partial) ordering over $2^{\mathcal{O}}$, and find at least one optimal subset of \mathcal{O} with respect to this ordering.*

In the rest of this section we explain why this capability is needed, and why some simple solutions are inadequate. In the following sections we consider a general approach for specifying set-preferences, and a more concrete instantiation of this approach that is based on CP-nets (Boutilier *et al.* 2004), and their extension with relative importance relations in the TCP-nets model (Brafman & Domshlak 2002). Then, we consider the problem of computing an optimal subset of a given set of objects, suggest a scheme for addressing it, which we refer to as *preference-based CSP generation*, and provide some initial empirical evaluation of its performance on a database of movies.

Our need for specifying preferences over sets arose naturally in the context of work on using preferences as a tool for personalizing and adapting online newsletters and multimedia presentations. Viewed abstractly, a newsletter is a collection of articles, and a personalized newsletter should provide the reader with a *preferred collection* of articles, possibly subject to various size and layout constraints. Similarly, setting up a committee or a task-force, or coming up with a program for a film festival requires addressing a similar optimal subset selection problem. Of course, in each case, what constitutes a good subset can be quite different.

A naive solution to the problem of subset selection would be to order all the objects and choose the set of top-most objects satisfying whatever hard constraints we have. This solution is quite likely to be inappropriate, since the overall attractiveness of a set of objects is rarely just a flat “accumulation” of attractiveness of these objects. For instance, given some articles we are less interested in some other articles (e.g., due to overlap in their content), while other groups of articles may complement each other. Similarly, when selecting team members, we know that some members are mutually enhancing, while others can be mutually detrimental. Likewise, in choosing the film festival program, we might wish to balance the choice of genre, topic, etc.

A similar problem was addressed by (Brafman & Friedman 2005; Brafman, Domshlak, & Shimony 2004). They handled such conditional preferences for the appearance of an article by considering a model in which, for *each* article,

*Partially supported by the Lynn and William Frankel Center for Computer Science, and by the Paul Ivanier Center for Robotics and Production Management.
Copyright © 2006, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

the outcome explicitly specifies whether this article is included in the newsletter or not. Using a CP-net, it was possible to model the conditional preferences one might have for seeing one article given that some other articles are present or absent. Such CP-nets have one node for each article, and it is clear that in most application domains this solution cannot scale up, nor does it address real needs. First, the pool of possible articles (or potential team members or films) is often huge, and we cannot expect an editor to actually specify a preference model that explicitly mentions each and every one of them. Second, new articles constantly appear, and we cannot expect an editor to constantly update and modify the preference model to make it up to date. In short, what we really want is a *static* preference specification approach that can be used for *dynamically* selecting a subset of objects, where these objects are typically *unknown* at preference elicitation time. This is what the work presented here sets out to accomplish.

Background on TCP-nets

The TCP-nets model, an extension of CP-nets (Boutilier *et al.* 2004), and its corresponding preference language play a central role in the concrete language we propose. This section reviews their essential features. For a more in-depth description of TCP-nets we refer the reader to (Brafman & Domshlak 2002).

Language: Each TCP-net captures a collection of preference statements that specify a preference ordering over a universe of objects \mathcal{O} . Objects in \mathcal{O} are described in terms of some set of attributes $\mathcal{X} = \{X_1, \dots, X_n\}$, i.e., we have $\mathcal{O} = \times \text{Dom}(X_i)$. The TCP-nets model supports two types of statements over \mathcal{X} , namely (conditional) preference for values of a variable, e.g., “if the car is a sports car, I prefer black to red as its color,” and (conditional) relative importance of different variables, e.g., “for a sports car, performance is more important to me than reliability.” The intuitive meaning of value preferences is straightforward. Importance statements are used to inform us about tradeoffs, i.e., “preference over compromises”. For instance, if reliability is more important to me than performance, it means that if I have to compromise on either reliability or performance, I would rather compromise on performance.

Semantics: Reasoning about the ordering induced by such statements on \mathcal{O} requires a commitment to a concrete logical interpretation of these natural language statements. The TCP-nets model adopts the *ceteris paribus* (all else equal) semantics for statement interpretation (Hansson 2001). In this conservative semantics, a statement “I prefer $X = x_1$ to $X = x_2$ ” means that given any two objects that are identical except for the value of X , the user prefers the one assigning x_1 to X to the one assigning x_2 . If these two objects differ on some other attribute as well, then they cannot be compared based on this preference statement alone. Similarly, a statement “ X is more important than Y ” means that given two objects that are identical except for their values on X and Y , the user prefers the one assigning a better value to X than the one assigning a better value to Y . Again, if these two objects differ on some other variable, too, they can no

longer be compared based on this statement alone. Conditional statements have the same semantics, except that they are restricted to comparisons between elements that satisfy the condition. Thus, “I prefer $X = x_1$ to $X = x_2$ given that $Y = y_1$ ” is interpreted exactly as above, but only for objects that satisfy $Y = y_1$.

Clearly, each preference statement induces a preference relation over \mathcal{O} . The “global” preference relation specified by a collection of such statements (and thus, by the TCP-net) corresponds to the transitive closure of the union of these “local” preference relations. If the user provides us with consistent information about her preferences, then the binary relation induced by the TCP-net on \mathcal{O} is a strict partial order (i.e., transitive, irreflexive, and antisymmetric). Note that this order is rarely complete, and thus typically not all pairs of objects are comparable with respect to a TCP-net.

Representation: In TCP-nets, such a collection of preference statements is represented using an annotated graph. The nodes of the graph correspond to the attributes \mathcal{X} and the edges provide information about preferential and relative importance dependencies between the variables. TCP-nets have three edge types. The first type of (directed) edge captures preferential dependence; such an edge from X to Y implies that user preference over values of Y vary with values of X . The second (directed) edge type captures relative importance relations; such an edge from X to Y implies that X is more important than Y . The third (undirected) edge type captures conditional importance relations, holding only when certain other variables have particular values. Each node X in a TCP-net is annotated with a conditional preference table (CPT) describing the user’s preference order over $\text{Dom}(X)$ for every possible value assignment to the parents of X (denoted $Pa(X)$). In addition, each undirected edge (X, Y) is annotated with a conditional importance table (CIT) describing the relative importance of X and Y given the values of certain conditioning variables.

Schematic Example: Figure 1 depicts a TCP-net over four variables—binary E, C, T and ternary F . Standard directed edges in this graph capture preferential dependencies, doubly-directed edges capture relative importance, and undirected edges capture conditional relative importance; \succ denotes preference over variable values; and \rightarrow denotes variable importance. The graph shows that the preference over values of C depends on F ’s value, while the preference over values of T depends on the value of both F and E . Actual preferences are provided in CPTs. For example, when $F = f_0$, we prefer C to be *false*. Additionally, an importance edge from F to E indicates that F ’s value is more important to us than that of E . Finally, the undirected edge between C and D and its associated CIT specify an importance relation between C and D conditional on E .

The structure of the TCP-net graph plays an important role in determining the consistency of preference specification and in reasoning about preference, although the user need not be aware of this structure (Brafman & Domshlak 2002). Here we note that not all sets of preference statements representable as TCP-nets are consistent. That is, some TCP-nets may correspond to a strict binary relation on \mathcal{O} that is not antisymmetric. However, (Brafman & Domsh-

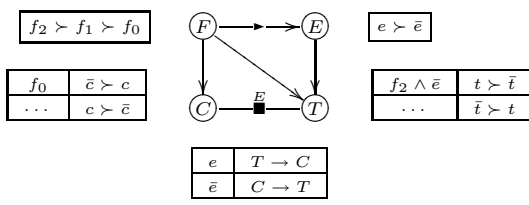


Figure 1: A schematic example of a TCP-net.

lak 2002) specifies a wide class of TCP-nets for which consistency is guaranteed, which can be easily recognized by the user interface. In what follows, we restrict ourselves to networks of this class.

Expressing Preferences over Subsets

Our goal is to provide a convenient tool for specifying preference over subsets of a set \mathcal{O} that has the same attribution structure $\mathcal{O} = \times_{X_i \in \mathcal{X}} \text{Dom}(X_i)$ as before. Our elicitation procedure consists of the following steps:

1. Obtain from the user properties of sets of objects that affect her preference over these sets.
2. Allow the user to express preference statements in terms of these properties.
3. Construct a preference representation model over these properties (a TCP-net in our case) that captures the information provided by these statements.

Below we describe a language that (in our opinion) efficiently addresses the major needs of qualitative preference specification over sets, and discuss some semantic issues involved in such preference specification. Subsequently, we discuss a computational scheme for selecting an optimal subset of \mathcal{O} , given a collection of such “set-preference” statements represented by a TCP-net.

Set-Preference Specification Language

It seems that most properties of sets of attributed objects that affect user preferences over such sets (informally) take the form “at least one object with $C = c$, and $D = d_1$ or $D = d_2$,” “the number of items with $C = c$ ”, etc. Formally, define the set $\overline{\mathcal{X}}$ of primitive propositions as

$$\overline{\mathcal{X}} = \{X = x \mid X \in \mathcal{X}, x \in \text{Dom}(X)\}$$

Let $\mathcal{L}_{\overline{\mathcal{X}}}$ be the propositional language defined over $\overline{\mathcal{X}}$ with the usual logical operators. Note that we can consider objects of \mathcal{O} as models of this language, and thus it makes sense to write $o \models \varphi$ where $o \in \mathcal{O}$ and $\varphi \in \mathcal{L}_{\overline{\mathcal{X}}}$.

While various basic properties of object sets can be considered, in our work we found that two classes of properties cover all our needs¹. The first class has the form $\langle |\varphi| \rangle$, where $\varphi \in \mathcal{L}_{\overline{\mathcal{X}}}$ and $\text{Dom}(\langle |\varphi| \rangle) = \mathbb{Z}^*$. Given a subset $O \subseteq \mathcal{O}$, $\langle |\varphi| \rangle(O)$ denotes the number of objects in O that satisfy φ , i.e., $\langle |\varphi| \rangle(O) = |\{o \in O \mid o \models \varphi\}|$. Using the property, the

¹Our framework can be extended to support other, though possibly not all, forms of properties as well.

user is able to express her preference on the number of objects in the selected subset that satisfy φ . The second class of properties has the form $\langle |\varphi| \text{ REL } k \rangle$, where $\varphi \in \mathcal{L}_{\overline{\mathcal{X}}}$, REL is a relational operator over integers, and $k \in \mathbb{Z}^*$ is a non-negative integer. While properties in the first class can take any non-negative integer value, the properties in this class are naturally Boolean; $\langle |\varphi| \text{ REL } k \rangle(O)$ is assigned the truth value of $|\{o \in O \mid o \models \varphi\}| \text{ REL } k$.

Example 1 Consider the following example of newsletter editing. Let the various articles \mathcal{O} be schematically described in terms of four attributes: *Country*, *Format* (news, interview, opinion, etc.), *Topic* (politics, weather, etc.), and *Emotion* (positive, negative, neutral, etc.) The editor in charge of selecting the content for the newsletter specifies four properties that affect her content preference:

$$\begin{aligned} F &= \langle |(\text{format} = \text{news})| \rangle \\ E &= \langle |(\text{emotion} = \text{neutral}) \vee (\text{emotion} = \text{negative})| \leq 2 \rangle \\ C &= \langle |(\text{country} = \text{Iraq}) \wedge (\text{topic} = \text{politics})| \geq 2 \rangle \\ T &= \langle |(\text{topic} = \text{culture}) \vee (\text{emotion} = \text{positive})| > 1 \rangle \end{aligned}$$

Observe that the property F is of the first, multi-valued class $\langle |\varphi| \rangle$, while E , C , and T are of the second, Boolean class $\langle |\varphi| \text{ REL } k \rangle$. Consider the following subset O of articles:

	format	country	topic	emotion
o_1	news	Iraq	politics	neutral
o_2	news	U.S.A.	weather	negative
o_3	interview	Iraq	economy	positive
o_4	opinion	France	culture	positive

For this subset $O \in \mathcal{O}$ we have $F(O) = 2$ (due to o_1 and o_2), $E(O) = \text{true}$ (only o_1 and o_2 satisfy φ_E), $C(O) = \text{false}$ (only o_1 satisfies φ_C), and $T(O) = \text{true}$ (due to o_3, o_4). \square

Now, consider a set of properties $\mathcal{P} = \{P_1, \dots, P_m\}$ specified as above over some (not necessarily pairwise distinct) formulas $\varphi_1, \dots, \varphi_m \in \mathcal{L}_{\overline{\mathcal{X}}}$, respectively. Observe that we can treat each P_i as a variable (Boolean or multi-valued, depending on its specification,) and each subset $O \subseteq \mathcal{O}$ provides a complete assignment to \mathcal{P} . That is, abstractly, we can view each subset O of \mathcal{O} as a vector of values, \mathbf{p}_O , for P_1, \dots, P_m , and abusing notations, we have a correspondence between the power set $2^{\mathcal{O}}$ and the abstract set of outcomes $\mathcal{O}_{\mathcal{P}} = \times_{\mathcal{P}} \mathcal{D}(P_i)$. Moreover, any preference order over $\mathcal{O}_{\mathcal{P}}$ implicitly induces a preference order over $2^{\mathcal{O}}$. What is nice about a preference order over $\mathcal{O}_{\mathcal{P}}$ is its abstractness — the user specifying it does not need to know the actual content of \mathcal{O} . The user need only know what properties of object sets she cares about, and express these properties in terms of the attributes \mathcal{X} . This means that a single static preference order over $\mathcal{O}_{\mathcal{P}}$ can be used in reasoning over different, dynamically changing sets of actual objects \mathcal{O} . In fact, by using $\mathcal{O}_{\mathcal{P}}$ instead of $2^{\mathcal{O}}$ we have reduced our problem of specifying preferences over subsets to that of specifying preferences over attributed objects. Given this reduction, we can use our favorite qualitative or quantitative specification language and representation model. We chose to use TCP-nets, but other choices are possible.

Example 2 Continuing Example 1, the editor states that:

1. The number of news articles should be maximal (i.e. $f_2 \succ f_1 \succ f_0$).
2. Having at least two articles on the political situation in Iraq (c) is better if there is at least one news article, otherwise fewer than two Iraq articles \bar{c} are better.
3. Preference on number of non-positive articles (E) is less important than preference on the number of news articles (F).
4. If the general prevalent emotion of the collection is not too negative (e), then T , the topic of some articles, is more important than the country (C), otherwise the importance is reversed.
5. If negative emotion prevails (\bar{e}) and there are two news articles, it is better to have more than article of a positive note or about culture (t). Otherwise, \bar{t} is preferred.

Modeling these statements as a TCP-net results precisely in the TCP-net of Figure 1. \square

Proceeding with obtaining and modeling preference statements over \mathcal{P} , we use the semantics of TCP-nets to order the elements of $\mathcal{O}_{\mathcal{P}}$, and this induces an ordering over $2^{\mathcal{O}}$: $O \subset \mathcal{O}$ is preferred to $O' \subset \mathcal{O}$ iff the (unique) property vector $\mathbf{p}_O \in \mathcal{O}_{\mathcal{P}}$ associated with O is preferred to the (unique) property vector $\mathbf{p}_{O'} \in \mathcal{O}_{\mathcal{P}}$ associated with O' . For example, consider the subsets $O_1 = \{o_1\}$ and $O_2 = \{o_1, o_2\}$. These subsets differ in property F (number of news format items), since $F(O_1) = 1$ and $F(O_2) = 2$. Upon examination, it is evident that $E(O_1) = E(O_2) = true$, because both subsets contain fewer than 2 items. Likewise, $C(O_1) = C(O_2) = false$ and $T(O_1) = T(O_2) = false$, the latter because no item in either O_1 or O_2 is about culture or has positive emotion. Therefore, we can use the *ceteris paribus* semantics of TCP networks to decide that O_2 is preferred to O_1 , since the network specifies $f_2 \succ f_1$, and “everything else” is equal.²

Optimal Subset Selection

One of the most important tasks of preferential reasoning is that of preference-based constrained optimization (PCO)—finding a preferentially optimal object that satisfies a given set of hard constraints. In our context, we consider PCO extended to set preferences, and focus on finding an optimal subset of a given set \mathcal{O} of objects.

As noted, our set-preference specification scheme can be used to lift any language for specifying preferences over objects into a language for specifying preferences of sets of objects. It would be great to have a PCO algorithm for set preferences that is as general, too. A naive PCO algorithm would simply generate all candidate subsets and compare them to each other. Unfortunately, this approach is impractical for all but a few applications, as it requires generating all possible subsets of \mathcal{O} . For instance, in our experiments we work with a database of 3200 movies, and generating all subsets of such a database is obviously unrealistic.

²It is possible to specify logically dependent properties. This does not have any logical implications on the semantics, although it could be exploited to make reasoning more efficient by pruning inconsistent combinations.

This is where our choice of representing the set preferences using a TCP-net N over certain subset properties \mathcal{P} becomes crucial. An attractive property of TCP-nets is that they come with a convenient PCO algorithm. This algorithm is able to examine feasible solutions *in an order consistent with the preference relation induced by the network* (Brafman & Domshlak 2002). That is, any solution generated in the future is guaranteed not to dominate the current solution.³ Although the technical details of this algorithm are quite involved, the basic scheme is a standard depth-first search, as follows:

1. Let L be a sorted list of TCP-net variables ordered in a way consistent with the TCP arcs.
2. Starting from the first unassigned variable V , assign to V the most preferred (untried) value given the assignment to its predecessors, according to $CPT(V)$.
3. Backtrack to the previous variable if required.

The statement “if required” above means either if some constraint is violated, or alternately if we wish the algorithm to enumerate outcomes in order of desirability, in which case we backtrack at any complete assignment. Observe that each leaf node in this search-tree corresponds to a complete assignment, and leaf nodes are generated in a non-decreasing order of preference. In our example (Figure 1) the list L consists of the variables: (F, E, T, C) .⁴ The algorithm will begin by assigning f_2 , the most preferred value, to F , followed by assigning *true* to E . Given the values f_2, e we prefer and assign *false* to T . Since the assignment to F is not f_0 , the value *true* is preferred for C . The result is the complete assignment $\{f_2, e, c, \bar{t}\}$. If for some reason (see below) the assignment is infeasible, we backtrack and re-assign C , this time to the less preferred value *false*, etc.

Unfortunately, in our case, a complete assignment to the TCP-net is only a set of properties. Whether there is a feasible subset with these properties and what this subset is, is not immediately clear – there is no simple and general way of constructively selecting a subset with arbitrary properties. However, it is still possible to efficiently adapt the PCO algorithm for TCP-nets to our setting, as we explain below.

The Basic Idea

As shown above, the PCO algorithm for TCP-nets proceeds by searching over a tree of partial assignments. Each partial assignment in our case maps into a set of properties. Each set of properties maps into a set of feasible subsets that satisfy these properties. The root node of the search tree has an empty set of properties associated with it, and thus corresponds to the set of all feasible subsets. Each node has the same property values as its parents, plus one additional property value. Thus, the subsets associated with each node are also associated with its parent, but not necessarily vice versa. If we explicitly maintain the set of subsets associated with

³Because TCP-nets induce a partial order, the latter does not imply that solutions in the future are strictly less preferred.

⁴Actually the ordering between T and C depends on the value of E , since the conditional importance arc has E as a conditioning element, but we ignore this issue here for simplicity.

each node, we obtain a conceptually simple adaptation of the PCO algorithm for TCP-nets to our context. The adapted algorithm explicitly searches the abstract space of $\mathcal{O}_{\mathcal{P}}$, as well as the concrete dual space of $2^{\mathcal{O}}$.

Although the above is a simple scheme, it is unlikely to work in practice because it maintains a partition over the set of all feasible subsets. Even if we assume subsets have size bound of k , we require space of $O(|\mathcal{O}|^k)$ for running this algorithm. One way to overcome this problem is to represent this set of subsets compactly, e.g., using variants of BDDs. However, we do not envision object subsets in applications of our interest exhibiting strong locality or order supporting compact storage in this form. Instead, we propose representing these sets *implicitly* by associating a constraint satisfaction problem (CSP) with each node of the search tree to implicitly represent the associated set of subsets. The benefit of using this approach is twofold. First, its space requirements are polynomial. Second, in this approach we never generate all subsets associated with a node in this search tree. Indeed, to generate an optimal subset, we just need to determine for each leaf of the search tree (i.e. a complete value assignment to set properties) whether it has a solution, and if so, we just need one solution. Explicit sets, or BDDs for that matter, do not provide such space guarantees.

Preference-based CSP Generation

We now explain how to generate the CSP associated with each node of the search tree. The resulting algorithm can be viewed as generating a tree of CSPs in some preferred order.

As stated above, each (possibly partial) value assignment \mathbf{p} to our set properties \mathcal{P} , implicitly represents the associated set $\{O\}_{\mathbf{p}}$ of subsets of \mathcal{O} by solutions to a certain CSP $\mathcal{C}_{\mathbf{p}}$. This CSP has one variable V_o for each object $o \in \mathcal{O}$ with $Dom(V_o) = \{0, 1\}$. In our running example, the CSP variables are $\{V_{o_1}, V_{o_2}, V_{o_3}, V_{o_4}\}$. Each complete instantiation \mathbf{v} to these CSP variables naturally induces a subset $O \subseteq \mathcal{O}$, namely the set of items o for which $\mathbf{v}[V_o] = 1$. For example, the instantiation $\{V_{o_1} = V_{o_2} = 1, V_{o_3} = V_{o_4} = 0\}$ induces the subset $O = \{o_1, o_2\}$. Observe that this part of the $\mathcal{C}_{\mathbf{p}}$ specification is independent of \mathbf{p} . The constraints of $\mathcal{C}_{\mathbf{p}}$ are the ones naturally induced by \mathbf{p} , as follows. For each property $P_i \in \mathcal{P}$, we add one constraint that enforces $P_i = \mathbf{p}[i]$. The property P_i involves the cardinality of the number of objects m that satisfy ϕ_i , and requires that the relation $m \text{ REL } k$ hold for some given value k . Let O_{ϕ_i} be the set of all objects in \mathcal{O} that satisfy ϕ_i , and let $|O_{\phi_i}| = M$. The constraint enforcing $P_i = \mathbf{p}(i)$ is an M -ary constraint over the variables $\mathbf{V}_{\phi_i} = \{V_o | o \in O_{\phi_i}\}$. An assignment \mathbf{v} obeys the constraint just when $m \text{ REL } k$, where m is the number of variables $V_o \in \mathbf{V}_{\phi_i}$ that are assigned 1 in \mathbf{v} .

Continuing the example, consider the TCP-net assignment $\mathbf{p} = \{f_2, e, c, \bar{t}\}$. The corresponding CSP has one cardinality constraint per property, as follows. f_2 means exactly 2 items of news format, inducing the constraint:

$$C_{f_2} : V_{o_1} + V_{o_2} = 2 \quad (1)$$

since o_1 and o_2 are the only items having news format. The assignment e induces the constraint:

$$C_e : V_{o_1} + V_{o_2} \leq 2 \quad (2)$$

as o_1 and o_2 are the only items to have either neutral or negative emotion. (C_e happens to be always satisfied.) The assignment c induces the (unsatisfiable) constraint:

$$C_c : V_{o_1} \geq 2 \quad (3)$$

Finally, \bar{t} induces the constraint:

$$C_{\bar{t}} : V_{o_3} + V_{o_4} \leq 1 \quad (4)$$

The algorithm to find an optimal subset of \mathcal{O} with respect to a TCP-net N is thus defined as follows:⁵

1. Set up the binary-valued CSP variables $\{V_o | o \in \mathcal{O}\}$. (Optionally, add all prior hard constraints.)
2. Incrementally generate complete property assignments $\mathbf{p} \in 2^{\mathcal{P}}$ in a non-increasing order of desirability according to N using the PCO algorithm.
3. For each assignment \mathbf{p} do:
 - (a) Set up the CSP constraints induced by \mathbf{p} .
 - (b) Let \mathbf{v} be a solution to the CSP, if one exists.
 - (c) If a solution was found, return the set of items o for which $\mathbf{v}[V_o] = 1$.
 - (d) Otherwise, retract the induced constraints, and backtrack in PCO.
4. Return 'fail' (this means an inconsistent specification).

In our running example, assuming no user-imposed hard constraints, the TCP-net is traversed top-down, resulting in a most desirable outcome $\mathbf{p} = \{f_2, e, c, \bar{t}\}$. As shown above, this results in an unsatisfiable CSP instance (since even C_c alone is unsatisfiable). Therefore, the CSP solver returns failure, and we backtrack to the next-best assignment to N .

As shown above, the PCO scheme now flips variable C_c , to get the next-best outcome $\mathbf{p}' = \{f_2, e, \bar{c}, \bar{t}\}$. The induced constraints for \mathbf{p}' are the same as for \mathbf{p} , except that C_c is replaced by $C_{\bar{c}}$, a constraint over V_{o_1} that requires at most one of these variables to be set to 1 (and thus $C_{\bar{c}}$ is always satisfied.) Solving the constraints $C_{f_2}, C_e, C_{\bar{c}}, C_{\bar{t}}$ results in one of the solutions: $\{V_{o_1} = V_{o_2} = V_{o_3} = 1, V_{o_4} = 0\}$ or $\{V_{o_1} = V_{o_2} = V_{o_4} = 1, V_{o_3} = 0\}$. The algorithm returns the first solution found, say $\{V_{o_1} = V_{o_2} = V_{o_3} = 1, V_{o_4} = 0\}$, corresponding to the subset $\{o_1, o_2, o_3\}$ of \mathcal{O} . Observe that based on the input, there is no way to prefer one of these latter results over another, and it is thus sufficient to return one of them arbitrarily.

Algorithm Improvements Several additional optimizations can be added to the above basic scheme: Since the enumeration of assignments \mathbf{p} generates complete assignments in a depth-first search over partial assignments, one could try to solve the CSP induced by a partial outcome \mathbf{p}'' . A failure to find a solution to $\mathcal{C}_{\mathbf{p}''}$ means that all assignments extending \mathbf{p}'' are unrealizable in \mathcal{O} , and this can be used to prune the search tree by backtracking earlier. Additionally, it is not necessary to redefine the CSP constraints from

⁵Detailed pseudo-code appears in a longer version of this paper. Any hard constraints imposed by the user beyond the preference structure (such as set cardinality or set-valued constraints (Bessiere *et al.* 2004)) can be naturally integrated into the algorithm.

scratch for each CSP instance. In a backtracking search over \mathbf{p} the induced constraints can be kept for each P_i for which $\mathbf{p}(i)$ has not changed from the last CSP instance. This was evident in our example, where only one induced constraint changed between \mathbf{p} and \mathbf{p}' .

Preliminary Analysis The optimization scheme discussed in this section requires potentially solving a large number of constraint satisfaction problems. Moreover, these CSPs involve multi-variable constraints, rather than just the standard binary constraints. However, these CSPs are not arbitrary, because constraints are incrementally added and removed as we traverse the tree of partial property assignments. Moreover, the multi-variable constraints in our CSP problems are all of a well-known and widely studied type called *cardinality constraints* (Hentenryck & Deville 1991), and this special case of constraints is known to be representable compactly (linear space). Thus, the space complexity issue is completely avoided, and the overall space requirements of this scheme are negligible. Algorithms for handling such constraints are a standard feature in many off-the-shelf CSP solvers, such as CLP (Puget 1994).

The search time requirement is still an important issue, especially given that standard CSP heuristics do not do well in the presence of cardinality constraints. Prior work, such as (Solotorevsky, Shimony, & Meisels 1998) and others, developed specific search heuristics that effectively focus the search in CSPs with such cardinality constraints, greatly alleviating the problem. Empirical results presented next suggest that at least for some applications, the algorithm completes in reasonable time, for a non-trivial TCP-net and a non-trivial database. To get an initial sense of the performance of our algorithm, we implemented and tested it in on the AMDB database⁶ of over 3200 movies. The goal was to generate a program for a three day film festival. Preferences for this selection were rather complex (see the full paper for the precise TCP-nets used), and involved 14 properties in the larger network – which we believe to be a realistic size for real-world applications – and 5 properties in the smaller network. The algorithm was tested on four different subsets of this database, containing 20, 1000, 1600, and 3200 films, respectively. The resulting running times (in seconds) were:

$ \mathcal{O} $	20	1000	1600	3200
TCP-5	0.03	0.16	0.31	0.95
TCP-14	0.89	2.34	2.34	356.25

Given the inherent complexity of our subset optimization problem, the running times in this representative example are quite reasonable, and seem to indicate the feasibility of our approach. Moreover, we believe that there is much potential for improving runtime performance by using methods that better address the type of incremental CSP involved.

Conclusion

We motivated the need for specifying and reasoning with preferences over sets of objects. We formalized a generic approach for specifying such preference based on set properties, and considered a concrete version that uses TCP-nets

to express preferences over properties. Then, we adapted the TCP-net based PCO techniques to this context, obtaining a PCO algorithm that generates a sequence of CSPs, and which, unlike the naive generalization, requires polynomial space only. And while it is worst-case exponential time, an initial implementation exhibited reasonable runtime on an application based on realistic data. We believe running times can be further improved by better exploiting the special structure of this CSP sequence – where one constraint is added or removed at each step. We refer the reader to a longer version of this paper containing a more detailed explanation of the algorithm, as well as the domain and networks used in our empirical study.

To the best of our knowledge, (desJardins & Wagstaff 2005) is the only previous paper to deal with set preference specification. It suggests a language in which users indicate how desirable it is for a set to contain elements with diverse or specific values of an attribute. Our language is significantly more expressive than the scheme suggested in (desJardins & Wagstaff 2005). The diversity or specificity of an attribute’s values is just one of many properties a user can define and specify preferences over, in our formalism. Moreover, we allow the specified preferences to be conditional on the degree to which other properties are satisfied.

References

- Bessiere, C.; Hebrard, E.; Hnich, B.; and Walsh, T. 2004. Disjoint, partition and intersection constraints for set and multisets variables. In *CP-04*, 138–152.
- Boutilier, C.; Brafman, R.; Domshlak, C.; Hoos, H.; and Poole, D. 2004. CP-nets: A tool for representing and reasoning about conditional *ceteris paribus* preference statements. *JAIR* 21:135–191.
- Brafman, R. I., and Domshlak, C. 2002. Introducing variable importance tradeoffs into CP-nets. 69–76.
- Brafman, R. I., and Friedman, D. 2005. Adaptive rich media presentations via preference-based constrained optimization. In *Proceedings of the IJCAI-05 Workshop on Advances in Preference Handling*, 19–24.
- Brafman, R.; Domshlak, C.; and Shimony, S. E. 2004. Qualitative decision making in adaptive presentation of structured information. *ACM TOIS* 22(4):503–539.
- desJardins, M., and Wagstaff, K. 2005. DD-PREF: A language for expressing preferences over sets. In *AAAI-05*.
- Hansson, S. O. 2001. *The Structure of Values and Norms*. Cambridge University Press.
- Hentenryck, P. V., and Deville, Y. 1991. The cardinality operator: A new logical connective and its application to constraint logic programming. In *ICLP-91*.
- Puget, F. 1994. A C++ implementation of CLP. Technical Report, ILOG SOLVER collected papers, ILOG.
- Solotorevsky, G.; Shimony, S. E.; and Meisels, A. 1998. CSPs with counters: A likelihood-based heuristic. *Journal of Experimental and Theoretical AI* 10:117–129.

⁶<http://www.steffensiebert.de/amdb/>