Preference-based Search for Configurable Catalogs

Paolo Viappiani, Boi Faltings
Artificial Intelligence Laboratory (LIA)
Ecole Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland
paolo.viappiani@epfl.ch

Abstract

We consider the problem of preference-based search for configurable products. We focus on example-based tools that interactively show both candidates (best options given current preferences) and suggestions to stimulate preference expression. Suggestions are generated to maximize the chance of eliciting further preferences using our *lookahead* principle. We present Probabilistic Constraint Optimization problems (ProbCOPs) that allow to reason about the uncertainty of the preference models and show how the generation of suggestions can be formulated as a single optimization problem.

Introduction

Preference-based search is an interactive process that helps users identify the most preferred option, called the *target* option, based on a set of preferences that they have stated on the attributes of the target.

An interesting technique for letting users volunteer their preferences is an interaction where the system shows proposed options and lets users express their preferences as *critiques* stimulated by these examples. This technique is called *example* or *candidate* critiquing, and has been explored by several authors (Linden, Hanks, & Lesh 1997; Smyth & McGinty 2003; Shimazu 2001).

The advantage of this kind of system is that the preference model is acquired interactively, and preferences are more likely to be correct if the user sees concrete examples.

In fact, psychological studies have shown that people construct their preferences (Payne, Bettman, & Johnson 1993) while learning about the available choices. When users are questioned about their preferences, they are likely to give incorrect answers, based on *means-objective* (Keeney 1992) that distract from the true target choice. In a user study (Viappiani, Faltings, & Pu 2006a) example-critiquing achieved higher accuracy than a traditional tool like the form-filling (where the user is asked to answer a set of questions), accuracy increases from 25% to 70%.

Example critiquing achieves higher decision accuracy when the displayed options are complemented with suggestions chosen to inform users about available choices (Pu, Viappiani, & Faltings 2006). The cognitive effort is comparable to simple interfaces such as a ranked list (Pu & Kumar 2004).

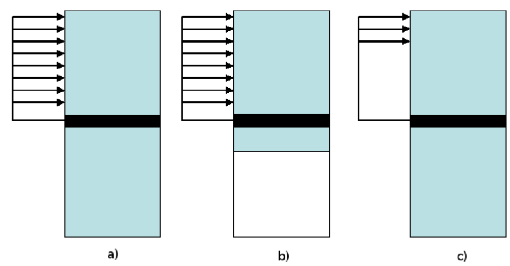

Figure 1: Intuitively the best suggestions are options that can become optimal if a new preference is stated. Model-based suggestions are computed by evaluating a given object with a set of dominators. In the original formulation (a) these are all the currently better options; in an approximation (b) possible suggestions are considered only in a subset of the space; the `top-escape` strategy (c) compare each option with few current best ones.

Example critiquing with suggestions

We consider and example-critiquing framework in which

1. Preferences are stated as reactions to displayed options (as "The price should be less than 600"). Such critiques are user-motivated.
2. These critiques are used as feedback in the next interaction cycle to generate a new set of displayed items.

If certain preferences are missing from the current model of the user, the system may provide solutions that do not satisfy those unknown preferences. If the user is aware of all of her preferences, she can realize the necessity to state them to the system. However this is not usually the case, because the user might not know all the available options.

Moreover, stating a preference costs some user effort and she would make that effort only if she perceives this as beneficial.

The influence of current examples is known as the *anchoring effect* (Tversky 1974).

To enable the user to refocus the search in another direction, many researchers have suggested to display alternatives or diverse examples, in addition to the best options (candidates).

In fact, in one user study it was observed that the majority of critiques (79%) were a reaction to seeing an additional opportunity rather than seeing unsatisfactory examples (Viappiani, Faltings, & Pu 2006b).

Different strategies for suggestions have been proposed in the literature. Linden (Linden, Hanks, & Lesh 1997) used extreme examples, where some attribute takes an extreme value. Others use diverse examples as suggestions (Smyth & McClave 2001; Smyth & McGinty 2003; Shimazu 2001).

However, an extreme example might often be an unreasonable choice: it could be a cheap flight that leaves in the early morning, a student accommodation where the student has to work for the family, an apartment extremely far from the city. Moreover, in problems with many attributes, there will be too many extreme or diverse examples to choose from, while we have to limit the display of examples to few of them.

The user should be motivated to state new preferences by options that are reasonable choices (given the previously stated preferences) and have a potential of optimality (a new preference is required to make them optimal).

This was expressed in the *lookahead principle* (Pu, Viappiani, & Faltings 2006):

> Suggestions should not be optimal under the current preference model, but should provide a high likelihood of optimality when an additional preference is added.

Model-based suggestions are calculated based on that principle. Results show that such suggestions are highly attractive to users and can stimulate them to express more preferences to improve the chance of identifying their most preferred item by up to 78% (Viappiani, Faltings, & Pu 2006c).

The lookahead principle was implemented by considering Pareto-optimality: suggestions are evaluated according to their probability of becoming Pareto-optimal. The advantage of this method is that it is qualitative, in the sense it does not rely on any particular parametrization: an option is Pareto-optimal if there is no other option that is better or equally preferred with respect to all the preferences and strictly better for at least one preference.

To become Pareto-optimal, the new preference has to make the current solution escape the dominance with better solutions (the "dominators"), as shown in Figure 1a. The computation of suggestions with this strategy requires a preliminary analysis of the available options to identify the dominators and then to make, for each of the options in the database, a series of checks to evaluate the chance that the option will become Pareto-optimal. This strategy has worst case complexity $O(n^2)$.

For large databases, an approximation (Viappiani & Faltings 2006) has been proposed, that considers possible suggestions only in a subset of options (Figure 1b). This strategy is based on the observation that suggestions retrieved with the look-ahead principle are not evenly distributed: in more than 60% of the cases, they are among the 25% top ranked options with respect to the current preferences (this is not surprising: since suggestions are options that should be reasonable, we will expect them to be not too far in the ranking from the current best options). This method can be used for reducing the computation time in large catalogs.

However such an approximation might not be enough for configurable catalogs, as they can be extremely large (the number of options is in general exponential in the number of attributes).

In this paper we consider the `top-escape` strategy that looks for the options that have the highest possibility of being better (thus escaping dominance) than the current best options displayed as candidates (Figure 1c). This method can be efficiently implemented in an single optimization problem and it is therefore suitable for configurable catalogs.

In the next section we introduce the notation and terminology for configurable catalogs.

Configurable Catalogs

Electronic catalogs often have the form of configuration systems where options are the many possible feasible solutions that satisfy the requirements. These configurable catalogs can be model as Constraint Satisfaction Problems (CSP).

An option or product is defined over a set of attributes that are the variables of the CSP. Given the variables $\mathcal{V} = \{v_1, .., v_n\}$, each variable v_i can take values in a particular domain $D_i = \{d_i^1, .., d_i^m\}$, m being the number of values in that domain. An assignment a is the binding between variables and domain values. If each variable is assigned to a value in its domain, the assignment is total.

Assignments are written as $a = (v_1 = \hat{d}_1, v_2 = \hat{d}_2, ..)$ or shortly as vectors $(\hat{d}_1, \hat{d}_2, ..)$. We use $v_i(a)$ to identify the value that an assignment associates with one variable.

Electronic configurable catalogs pose restrictions to the feasible combinations, that are usually the same for every user. These are represented by a set of hard constraints $\mathcal{HC}$; they can be unary or binary.

A unary constraint c_i on variable v_i is a function from D_i to $\mathbf{Bool}$, stating that a given value in D_i can be assigned to variable v_i; a binary constraint $c_{i,j}$ between variables v_i and v_j is a function from $D_i \times D_j$ to $\mathbf{Bool}$, asserting that a given combination of values can be assigned to v_i and v_j.

A total assignment that satisfies all hard-constraints $\mathcal{HC}$ is called a **solution**. As we use $CSPs$ to represent electronic catalogs, a solution corresponds to an option.

COPs or SoftCSPs are an extension of CSPs that consider constraints that can be partially satisfied. Such constraints are called *soft* (in opposition to standard *crisp* constraints) and are functions from combinations of domain values to numeric weights.

Definition 1 *A soft constraint (unary and binary) associates a weight to a combination of value assignments to one or more variables.*

- *A soft unary constraint φ_i on variable v_i is a function from D_i to $[0, 1]$, expressing the weight of each value D_i that can be assigned to variable v_i*

- *A soft binary constraint $\varphi_{i,j}$ between variables v_i and v_j is a function from $D_i \times D_j$ to $[0,1]$, expressing the weight of a given combination of values that can be assigned to v_i and v_j*

Since a given soft constraints always applies to the same variables, with a little abuse of notation we will write $\varphi_{i,j}(s_1)$ instead $\varphi_{i,j}(v_i(s_1), v_j(s_1))$.

It is natural to use soft constraints to represent the preferences of the user. In this paper we deal with preferences that involve only one variable, thus are unary soft constraints.

Definition 2 *A COP (Constraint Optimization Problem) is a tuple* $(\mathcal{V}, \mathcal{D}, \mathcal{HC}, \mathcal{SC}, +, \mathbf{Min})$ *where*

- $\mathcal{V} = \{v_1, .., v_n\}$ *is a set of variables,*
- $\mathcal{D} = \{D_1, .., D_n\}$ *is a set of domains,*
- $\mathcal{HC}$ *a set of hard constraints,*
- $\mathcal{SC}$ *a set of soft constraints*
- $+$ *is an aggregating operator*
- `Min` *is a selection function*

The operation $+$ is used to aggregate individual soft constraints into an overall score W:

$$\mathbf{W}(s) = \sum_{\varphi \in \mathcal{SC}} \varphi(s) \tag{1}$$

The **optimal solution** s_o of the COP ($\mathcal{V}$,$\mathcal{D}$,$\mathcal{HC}$,$\mathcal{SC}$,$+$,`Min`) is the solution of the CSP defined by ($\mathcal{V}$,$\mathcal{D}$,$\mathcal{HC}$) such that $\mathbf{W}(s_o) = Min_s[\mathbf{W}(s)]$. A COP that uses the standard $+$ and Min is also called *WeightedCSP*.

In this paper, we will use COP in two circumstances. Firstly, we will use COP to directly represent a preference-based search task, in which $\mathcal{SC}$ models the current preferences, that are known to the system. The optimal solution corresponds to the best option given the current user preferences (the candidates). The formulation is isomorphic to generalized additive utility models. However, since the soft constraints will necessarily be a inaccurate representation of the preferences, we compensate by showing not just one, but a set of k best candidate solutions.

Secondly, in the context of generating suggestions according to the `top-escape` strategy, we will instantiate a COP in which the weights represent the individual contribution of combinations of values to the quality of the suggestion. The quality of a suggestion is evaluated by the probability of optimality considering the uncertainty of possibly missing (unstated) preferences. We assume independence between the probabilities of having preference on different attributes, so that they can be combined by a standard sum.

Top-escape strategy for suggestions

In the original formulation of the look-ahead principle, each potential suggestion is evaluated by the probability of becoming Pareto-optimal when a new preference is considered. For this to happen the option has be strictly better than any dominating option with respect to this new preference.

The `top-escape` strategy reformulates the look-ahead principle by considering as suggestions the solutions that have the highest probability of being better (escape dominance) than the set of top options S_{top}.

The generation of suggestions according to this strategy can be made by solving a single optimization problem; it is therefore feasible even for very large configuration catalogs. We need to solve an auxiliary optimization problem in which each possible value d_i of the variable v_i is weighted by the probability that d_i is better than the value of the options in S_{top}.

The idea is similar to (Hebrard *et al.* 2005), where a problem called `MostDistant`, consisting in finding the most diverse solution of a CSP with respect to a set of previously retrieved solutions, is solved as an auxiliary constraint problem.

The uncertainty about possibly missing preferences is represented by prior distribution that can represent previous use of the interface, as in the original formulation of model-based suggestions. A precise knowledge of the prior might be not necessary, as model-based suggestions give a large increase of decision accuracy even when the distribution is assumed to be uniform (Viappiani, Faltings, & Pu 2006c).

Uncertainty in the preference model

We are dealing with an interactive process in which preferences are iteratively revealed. Initially we have an empty set of known preferences and all that we know is the probability distribution of the possible preferences. The interaction is mixed-initiative and the user can state a preference on any of the attributes at any given point. As the interaction goes on, more preferences become known to the system and the number of variables with uncertain preferences decreases.

At any given point, the set of attributes can be partitioned in two parts: V_k are attributes in which the preference is known, while V_u the set of attributes for which the preference is unknown. For each attribute in V_u we consider the set of possible preferences and the relative distribution.

Preferences are represented by soft constraints, whose weight can be interpreted by the degree of preference. We suppose that the system knows that each user has preference value functions in a known parameterized family. Here we assume a single parameter θ, but the method can be generalized to handle cases of multiple parameters.

We indicate with $q_i(\theta, v_i(s))$ or $q_i(\theta, s)$ the parameterized value function of preference r_i on variable i (we use the letter r to represent preferences and q to represent value functions). For any given user and any preference r_i, we assume the system knows the probability distribution $p_i(\theta)$, that might be integrating prior beliefs or past experiences with the present user.

Probabilistic Constraint Optimization

We define probabilistic constraint optimization problems (ProbCOP). This definition is used to model uncertainty of preferences in a preference-based search task that takes place in a configurable scenario.

Definition 3 *A Prob-COP is a tuple* $(\mathcal{V}, \mathcal{D}, \mathcal{HC}, \mathcal{SC}, \Pi)$*, where*

- $\mathcal{V}$ *is a set of variables,*

- *$\mathcal{D}$ is a set of domain values,*
- *$\mathcal{HC}$ a set of hard constraints,*
- *$\mathcal{SC}$ a set of soft constraints, representing the known preferences*
- *Π a set of probability distribution π_i of uncertain soft constraints*

As said previously we represent the possible preferences by a set of parameterized cost functions. The probability distribution π_i represents an uncertain preference. It assigns a probability $p_i(\theta)$ to each value of the parameter θ in $q_i(\theta, v_i(s))$, the value function of a possible preference on variable v_i.

Note that $\mathcal{V}$ and $\mathcal{HC}$ combined are a traditional CSP and $\mathcal{V}$, $\mathcal{HC}$ and $\mathcal{SC}$ a COP problem.

We use ProbCOP as a formalism for representing the preference-based search task. At any given moment, ProbCOP models the information that the system has about the user: the preferences previously stated (SC), the uncertainty about possible missing preferences (Π).

How to compute top-escape suggestions

Considering that the ProbCOP $(\mathcal{V}, \mathcal{D}, \mathcal{HC}, \mathcal{SC}, \Pi)$ models the current preference-based search task, we initially retrieve a set of top-k solutions (S_{top}) by solving the COP problem $(\mathcal{V}, \mathcal{D}, \mathcal{HC}, \mathcal{SC}, +, Min)$ underlying the ProbCOP (without considering the uncertain preferences).

The goal is to find s such that the probability p^{esc} that s escapes S_{top} (s is preferred to all solutions in S_{top} according to the new preference) when a new preference is stated is maximized. In fact the probability p^{esc} is a weighted sum of the individual probabilities p_i^{esc} that refer to escaping S_{top} in the case of having the new preference on variable v_i. This happens when $v_i(s)$ is better than any $v_i(s_{top})$ for $s_{top} \in S_{top}$, for any $v_i \in V_u$.

For each value d in the domain of a variable $v_i \in V_u$ let $\delta_i(d, E)$ be the probability that a new (unary) preference on variable v_i makes d preferred to each of the values E, where E is the set of values taken by solution in S_{top}, $E = \{v_i(s) | s \in S_{top}\}$. The value of $\delta_i(d, E)$ is found by integrating over possible θ in the probabilistic distribution of possible preferences for variable v_i, π_i. This can be expressed using the Heavyside step function $H(x) \equiv$ **if** $(x > 0)$ **then** 1 **else** 0:

$$\delta_i(d, E) = \int [\prod_{d' \in E} H(q_i(\theta, d') - q_i(\theta, d))] p_i(\theta) d\theta$$

The suggestions can be found by solving a COP where $\delta_i(d, E)$ are the weights of the soft constraints.

Definition 4 *Given a ProbCOP $(\mathcal{V}, \mathcal{D}, \mathcal{HC}, \mathcal{SC}, \Pi)$ and a set of solutions S_{top}, the* Top-escape COP $(\mathcal{V}, \mathcal{D}, \mathcal{HC}, \mathcal{SC}_{esc}, +, Max)$, *where $\mathcal{SC}_{esc}$ is such that $\forall v_i \in V_u : \varphi_i^{esc}(d) = \delta_i(d, E)$, and $E = \{v_i(s) | s \in S_{top}\}$.*

The optimal solution of *Top-escape COP* is a solution with the highest probability of escaping dominance with a set of top dominators, when a new preference is added to the user model. It can be solved with branch and bound techniques.

If we want to consider the possibility that the user might have hidden conditional preferences (as a preference on variable v_i conditioned to v_j, represented as a binary soft constraint), we can define and compute $\delta_{i,j}(d_i, d_j, E)$ in a similar way.

Evaluation

	average time	
method	random CSP	laptop CSP
Lookahead strategy	0.82s	6.15s
Top-escape strategy	0.02s	0.16s

Table 1: Execution times for the original look-ahead strategy and for the top-escape strategy.

We can compare the top-escape suggestions strategy with the original look-ahead strategy based on Pareto-optimality.

We considered random generated problems and an electronic catalog of laptop computers. Random CSPs were created with a number of variables varying between 3 and 7, number of values between 2 and 5. The laptop CSP consists in a catalog of computers with 10 variables, that span between 2 and 15 values, with 528 solutions.

We generated 3 candidates (top options) as S_{top} for the top-escape strategy. We generated 3 suggestions with both methods.

As we can see, the look-ahead strategy need significantly less computation time.

Variants and Improvements

Lex-Top-Cost strategy

The suggestions generated according to `top-escape`, might be actually far from the current best option in the ranking. This means that a top-escape suggestion can be very good in showing some new interesting opportunities but it might be poor with respect to the previously stated preferences.

An improvement is to select, among solutions that have the same chance of escaping S_{top}, those that are better given the current preferences. In other words, this method sorts solutions in a lexicographic order by the probability of escaping top-dominance and the utility of current preferences. It means that a solution s_1 is considered a better suggestion than s_2 if s_1 has higher total probability of top-escaping or if the probability is equal but the current cost is lower.

For this strategy, we define a COP called *Lex-Top-Cost*. We have two sets of soft constraints $\mathcal{SC}_{esc}$ (representing the probability of escaping dominance when a new uncertain preference is stated), defined over variables in V_u, and $\mathcal{SC}_{current}$ (representing the satisfaction of current preferences), defined over variables in V_k. The evaluation function $Eval_{Top-Cost}$ constructs a vector (p, c), where p represents the sum of the probabilities and c the sum of the individual value functions.

$$Eval_{Top-Cost} = \left(\sum_{\varphi_i \in SC_{esc}} \varphi_i, \sum_{\varphi_j \in SC_{current}} \varphi_j \right) \quad (2)$$

The selection function is Lex, such that $Lex[(p, c), (p', c')] = (p, c)$ if $p > p'$ or $p = p' \wedge c < c'$.

Definition 5 *Given a ProbCOP* $(\mathcal{V}, \mathcal{D}, \mathcal{HC}, \mathcal{SC}, \Pi)$ *and a set of solutions* S_{top}*, the* Lex-Top-Cost COP $(\mathcal{V}, \mathcal{D}, \mathcal{HC}, \mathcal{SC}_{esc} \cup \mathcal{SC}_{current}, \mathtt{Eval_{Top-Cost}}, \mathtt{Lex})$*, where* $\mathcal{SC}_{current} = \mathcal{SC}$ *and* $\mathcal{SC}_{esc}$ *is such that* $\forall v_i \in V_u : \varphi_i^{esc}(d) = \delta_i(d, E)$*, and* $E = \{v_i(s) | s \in S_{top}\}$*.*

Iterative strategy

Escaping dominance with the top-ranked solutions is often not enough to become Pareto-optimal, as there might be other solutions that also dominate. In the *iterative* strategy, we also try to find a most representative sets of top options so that breaking dominance is likely to ensure Pareto-optimality.

We observe that the probability p_i^{esc} of escaping top-domination with option s_{top} (the current best option) when a missing preference is present on variable v_i is an upper bound for the probability of Pareto-optimality (p_i^{po}).

$$p_i^{esc}(s_1, s_{top}) = p[\varphi_i(s_1) < \varphi_i(s_{top})] \quad (3)$$

If we consider three options, say s_{top} , s_1 and s^*, where s_1 is dominated both by s_{top} and s^*, s_1 will become Pareto-optimal only if it escapes both dominance relations. In practice, we have to subtract from p_i^{esc} all the cases in which s^* still dominates s_1.

In general, the current option s_1 will be dominated by many solutions. Given the set S^* of dominators of s_1, the probability of escaping dominance with S_{top} but not becoming Pareto-optimal is the probability of having at least one dominator s^* that is better or equal to s_1 when the latter escapes s_{top}.

$$p_i^{po} = p_i^{esc}(s_{top}) * p[\nexists s^* \in S^* : \varphi_i(s^*) \leq \varphi_i(s_1) < \varphi_i(s_{top})] \quad (4)$$

To avoid the generation of all dominators of s_1, we approximate the probability by considering only a subset of S^*. We would like to consider a smaller set S^* that is large enough to give us a better approximation of the probability of Pareto-optimality. To do so, we look for k solutions s^* that maximize $p[\varphi_i(s^*) \leq \varphi_i(s_1) < \varphi_i(s_{top})]$. This will give us a handy set of dominators that give the highest contribution to the "correction" of the probability of becoming Pareto-optimal.

We define *top-block*, a COP where the weights of a soft constraint φ_i^{tb} represent the probability of any domain value being preferred to the value of the current suggestion when the latter is better than $v_i(s_{top})$.

Definition 6 *Given a ProbCOP* $(\mathcal{V}, \mathcal{D}, \mathcal{HC}, \mathcal{R}, \Pi)$*, we define* `top-block` ***COP*** $(\mathcal{V}, \mathcal{D}, \mathcal{HC}, \mathcal{SC}^{tb}, +, Max)$*, where* $\forall v_i \in V_u$*:* $\varphi_i^{tb} : d \rightarrow p[\varphi_i(d) \leq \varphi_i(s_1) < \varphi_i(s_{top})]$

Solving the top-block COP, we retrieve a set S^* that we can use to approximate the probability of becoming optimal of the current solution that was selected as suggestion s.

The strategy proceeds in the following way. We generate an initial suggestion s by considering the top-escape approximation, by solving *top-escape* CSP. We generate a set S^* of solutions by solving *top-block* CSP; we evaluate the approximated value of optimality of s by calculating the expression of equation (indicated in the algorithm as `eval-approx-po`). We replace the current s with one option in S^*. We repeat this indefinitely, stopping when there is no further gain in the (approximated) probability of becoming Pareto-optimal. Algorithm 1 shows how to retrieve suggestions in this way.

Algorithm 1: **The iterative approximated method for computing suggestions.**

```
s_top = solve(top-k)
s = solve(top-escape CSP, s_top)
S* = solve(top-block(s))
p_po(s)=eval-approx-po(s, S*)
while true do
    s_1 = choose(S*)
    S*_1 = solve(top-block(s_1))
    p_po(s_1)=eval-approx-po(s_1, S*_1)
    if p_po(s_1) < p_po(s) then
        Return s
    s = s_1
```

Example

We consider an example (Table 2) with three variables v_1, v_2 and v_3, all with domain $\{a, b, c\}$. The hard constraints require pairwise different values: $\forall_{i,j \in [1,3]} c_{i,j} : (v_i \neq v_j)$.

Suppose that preferences are:

- Initial Preference on v_1: r(a)=0, r(b)=1, r(c)=2 (or any other evaluation that prefers a to b and b to c).

Π_2		Π_3	
θ	$p(\theta)$	θ	$p(\theta)$
$a \succ b \succ c$	0.16	$a \succ b \succ c$	0.25
$a \succ c \succ b$	0.16	$a \succ c \succ b$	0.25
$b \succ a \succ c$	0.16	$b \succ a \succ c$	0
$b \succ c \succ a$	0.16	$b \succ c \succ a$	0
$c \succ a \succ b$	0.16	$c \succ a \succ b$	0.25
$c \succ b \succ a$	0.16	$c \succ b \succ a$	0.25

Table 2: Probability distributions for the uncertain preferences on variable v_2 and v_3, expressed giving a probability to every single combination. Π_2 expresses that any combination is equally likely, while Π_3 represents equal chance of having either a or c as most preferred value. Here, the parameter θ of the distribution represents the possible preference order.

s	$\delta_2(v_2(s), \{b, c\})$	$\delta_3(v_3(s), \{b, c\})$	$p^{esc}(S_{top})$
(b,a,c)	0.33	0	0.16
(b,c,a)	0	0.5	0.25
(c,a,b)	0.33	0	0.16
(c,b,a)	0	0.5	0.25

Table 3: Example of calculation of the probability of escaping the top options $s_{o_1} = (a, b, c)$ and $s_{o_2} = (a, c, b)$. Given that the 2 top options take both values b and c on v_2 and v_3, the only positive contribution to the probability is given by value a, which is better than b and c with p=0.33 for v_2, and with p=0.5 for v_3.

	dominators S_d	δ_2	δ_3	p_{opt}
(b,a,c)	(a,b,c) (a,c,b)	0.33	0	0.16
(b,c,a)	(a,b,c) (a,c,b)	0	0.5	0.25
(c,a,b)	(a,b,c) (a,c,b) (b,a,c) (b,c,a)	0	0	0
(c,b,a)	(a,b,c) (a,c,b) (b,a,c) (b,c,a)	0	0	0

Table 4: Example of calculation of the probability of becoming Pareto-optimal. The solution (c,b,a) has no probability of becoming Pareto-optimal when a single preference is missing, because if the preference is on v_3 it is still dominated by (b,c,a), if the preference is on v_2, it does not escape dominance with (a,b,c).

- Preference Distribution for v_2: any permutation of a,b,c with equal probability, assigning 0 to the most preferred, 2 to the worst
- Preference Distribution for v_3: either c or a is the most preferred with equal chance, assigning 0 to the most preferred, 2 to the worst

Candidates Given the only known preference on v_1, in this case the two best configuration $s_{o_1} = (a, b, c)$ and $s_{o_2} = (a, c, b)$ with cost 0, that are also the Pareto-optimal.

Model-based Suggestions We consider the difference between the original method based on Pareto-optimality and the top-escape strategy.

The best suggestion according to the original definition of the lookahead principle (Table 4) is (b, c, a) that has 0.25 probability of becoming optimal, than (b, a, c) with 0.16.

The top-escape strategy constructs a COP with the following soft constraints: $v_2 \to \{a \to 0.33, b \to 0, c \to 0\}$, $v_3 \to \{a \to 0.5, b \to 0, c \to 0\}$. The approximate method retrieves either (b, c, a) or (c, b, a) as best suggestion. We can notice that (c, b, a) has no real chance of becoming Pareto-optimal by means of a single new preference statement (Table 3).

Lex-Top-Cost method constructs a COP that, in addition to the constraints of top-escape, separately evaluates the current cost $v_1 \to \{a \to 0, b \to 1, c \to 2\}$. The resulting best suggestion (b, c, a) has (prob,cost) $= (0.25, 1)$ preferred to (c, b, a) whose lex-evaluation is $(0.25, 2)$.

The iterative method would either retrieve (b, c, a) or (c, b, a) at the first step, as they have the same chance of escaping s_{top}. In the second case, it will retrieve (b, c, a) in S^*, because it is the solution that has the highest chance to be better than (c, b, a) when the latter is better than s_{o1} and s_{o2}. So, (c, b, a) will be considered the best suggestion by this method.

Evaluation

In this section we compare the strategies for generating suggestions for preference-based search for configurable products that we considered in this paper. We consider, as before, random generated problems and electronic catalog of laptop computers.

We want to check whether the suggestions we find with the `top-escape` method are very different from those retrieved with the original formulation. To do so, we compare the actual probability of becoming Pareto-optimal (computed with with the original lookahead method) of the suggestions retrieved by each of the top-escape methods and use this as an evaluation of the performance of the strategies. For each simulation, a random preference model is generated and 3 suggestions are retrieved. The average probability of the 3 suggestions for each method is considered. The values are scaled upon the probability of optimality of the best possible suggestions (those retrieved with the exact method). This however is not a precise measure of the quality of the suggestions.

method	random CSP	laptop CSP	time
Look-ahead Pareto	100%	100%	6.15s
Top-escape	70%	21%	0.16s
Lex-Top-Cost	88%	38%	0.17s
Iterative	93%	40%	0.40s

Table 5: The evaluation of the different methods for the retrieval of suggestions. The methods are evaluated according to the probability that a suggestion becomes Pareto-optimal. The numbers are scaled so that the exact method gets 100%. Given a user model of preferences, 3 suggestions are retrieved with each of the methods; their true probability of optimality is considered. The computation time is given for the laptop configuration database.

Conclusions

We presented the problem of preference-based search and focused on example-based tools in which preferences are acquired by critiques of the user to displayed examples.

We considered the look-ahead principle (Pu, Viappiani, & Faltings 2006) for suggestions that stimulate preference expression and looked at implementation for configurable catalogs. The main difficulty to compute suggestions is to avoid the generation of all the solutions of the configuration problem.

We proposed a strategy, called `top-escape`, that looks for solutions that have the highest probability of being better than a few current best options at the top of the current ranking and can be modeled as a single optimization problem. We presented an improvement `lex-top-cost` that considers, among the solutions that have the same probability, those that are better given the current preferences. Then, we proposed a method that looks for the best dominators to approximate the probability of becoming Pareto-optimal, requiring to iteratively solve a series of COP problem.

The methods proposed seem to be sufficiently fast for practical applications.

References

Hebrard, E.; Hnich, B.; O'Sullivan, B.; and Walsh, T. 2005. Finding diverse and similar solutions in constraint programming. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI'05)*, 372–377.

Keeney, R. L. 1992. *Value-Focused Thinking. A Path to Creative Decision Making*. Cambridge: Harvard University Press.

Linden, G.; Hanks, S.; and Lesh, N. 1997. Interactive assessment of user preference models: The automated travel assistant. In *Proceedings of the Fifth Internation Conference on User Modeling (UM'97)*.

Payne, J.; Bettman, J.; and Johnson, E. 1993. *The Adaptive Decision Maker*. Cambridge University Press.

Pu, P., and Kumar, P. 2004. Evaluating example-based search tools. In *Proceedings of the ACM Conference on Electronic Commerce (EC'04)*.

Pu, P.; Viappiani, P.; and Faltings, B. 2006. Increasing user decision accuracy using suggestions. In *ACM Conference on Human factors in computing systems (CHI06)*, 121–130.

Shimazu, H. 2001. Expertclerk: Navigating shoppers buying process with the combination of asking and proposing. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, volume 2, 1443–1448.

Smyth, B., and McClave, P. 2001. Similarity vs. diversity. In *Proceedings of the 4th International Conference on Case-Based Reasoning (ICCBR'01)*, 347–361.

Smyth, B., and McGinty, L. 2003. The power of suggestion. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico*, 127–132.

Tversky, A. 1974. Judgement under uncertainity: Heuristics and biases. *Science* 185:1124–1131.

Viappiani, P., and Faltings, B. 2006. Design and implementation of preference-based search. In Springer., ed., *The 7th International Conference on Web Information Systems Engineering*, LNCS4255, 72–83.

Viappiani, P.; Faltings, B.; and Pu, P. 2006a. Evaluating preference-based search tools: a tale of two approaches. In *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06)*, 205–211. Boston, MA, USA: AAAI press.

Viappiani, P.; Faltings, B.; and Pu, P. 2006b. The lookahead principle for preference elicitation: Experimental results. In *Seventh International Conference on Flexible Query Answering Systems (FQAS)*.

Viappiani, P.; Faltings, B.; and Pu, P. 2006c. Preference-based search using example-critiquing with suggestions. *Journal of Artificial Intelligence Research (JAIR)* 27:465–503.