Preference-based Problem Solving for Constraint Programming

Ulrich Junker

ILOG
1681, route des Dolines
06560 Valbonne
France
ujunker@ilog.fr

Abstract. Combinatorial problems such as scheduling, resource allocation, and configuration have many attributes that can be subject of user preferences. Traditional optimization approaches compile those preferences into a single utility function and use it as the optimization objective when solving the problem, but neither explain why the resulting solution satisfies the original preferences, nor indicate the trade-offs made during problem solving. We argue that the whole problem solving process becomes more transparent and controllable for the user if it is based on the original preferences. We show how the original preferences can be used to control the problem solving process and how they can be used to explain the choice and the optimality of the detected solution. Based on this explanation, the user can refine the preference model, thus gaining full control over the problem solver.

1 Introduction

Although an impressive progress has been made in solving combinatorial optimization problems such as scheduling, resource allocation, and configuration, optimization methods lack easy acceptance in industry. Expert users often want a detailled control over the choice of a solution and are reluctant to sacrifice existing practices for gaining small improvements in optimality. The existing practices may govern the whole decision making process, in particular if it is done manually. As an example, we consider the problem where multiple customer orders are given and each order specifies a product that needs to be produced. We further suppose that these products can be produced in different ways, on different machines, and at different times. A schedule chooses a job (i.e. an alternative in a production plan) for each customer order and allocates a machine and a time to each activity in this job while respecting temporal constraints and resource constraints. Usually, the experts will not determine an arbitrary schedule, but may apply standard choices, for example when choosing the job or the machines. Nevertheless, the standard choices may be abandoned if they are not feasible or if they are in conflict with more important choices. If an automatized scheduling system ignores these standard choices and the existing

Dagstuhl Seminar Proceedings 04271
Preferences: Specification, Inference, Applications
http://drops.dagstuhl.de/opus/volltexte/2006/399

practices, then experts may be reluctant to accept it even if the system produces a schedule that optimizes global objectives such as tardiness minimization.

As standard choices can be abandoned if there is a clear reason for this, they do not have the character of constraints, but represent preferences. These preferences are not formulated on global properties of the resulting schedule, but on individual choices. There may be preferences for choosing a production plan for a given product, preferences for choosing a resource, and so on. These individual preferences easily appear if the possible options for a choice are individuals with specific characteristics. Resource allocation problems may have many attributes of this kind. The situation is similar for configuration problems, which can have user preferences on all attributes that are exposed to the user. Those individual preferences are typically treated by a MAUT (Multiple attribute utility theory) approach and are aggregated into a single numerical utility function, which complements other optimization objectives such as cost minimization. The utility function usually has an additive form, namely that of a weighted sum of subutility functions, which well fits into a mixed integer programming (MIP) approach.

The quality of a MIP solution is then measured with respect to the utility function, but not with respect to the original preferences. For example, a degree of optimality of 99.8% means that as many preferences as possible have been satisfied except for 2%. Unfortunately, this explanation does not make sense to the end users who specified the original preferences. If the solution does not give the most preferred choices to the end users, they would like to know whether these most preferred choices are infeasible or whether they are in conflict with other choices that have been made. Explanations of the trade-offs may help users to accept the solution or to revise the preferences.

We therefore argue that solutions of optimization methods need to be enhanced by explanations of optimality in order to become acceptable for the users. Moreover, these explanations should be given in terms of the original preferences and in a form that is comprehensible for non-optimization specialists. Let us consider a simple configuration problem, namely that of choosing a vacation destination. Suppose that Hawaii is preferred to Florida for doing windsurfing, but that Florida has been selected in the solution. The explanation may be that the option Hawaii is infeasible since it is too far away. Or it may be that Hawaii leads to high hotel costs and thus penalizes subsequent choices. Or still another reason may be that a more important choice was that of including visits to attraction parks among the vacation activities, which is not possible in Hawaii. These explanations exhibit the trade-offs and the importance orderings that generated the solution and that justify it.

The explanations of optimality unveil other problems of the MAUT approach. Firstly, the MAUT approach assumes completeness of the preferences on the individual attributes. If the user has only specified a partial ordering, it will implicitly be completed when compiling the preferences into a utility function. For example, the user may prefer Hawaii to Florida, Hawaii to the Cote d'Azur, and the Cote d'Azur to the Mexico. We may compile this into a utility of 3 for

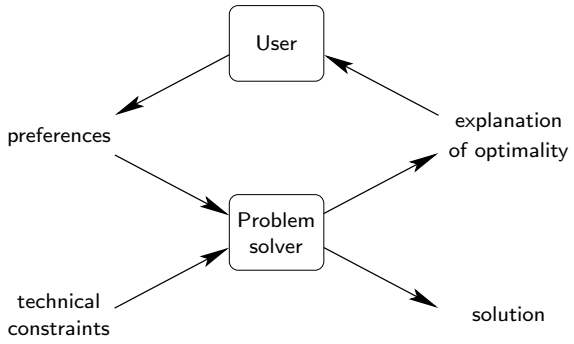

Fig. 1. An interactive optimization process driven by explanations and preferences.

Hawaii, 2 for the Cote d'Azur and for Florida and 1 for the Mexico. This implies that Florida is preferred to the Mexico, although the user has not stated this. If the options Hawaii and Cote d'Azur are not possible, then only the option Florida may be considered and this even if it may defeat the best choices for other attributes. Hence, implicitly chosen preferences are a problem if they are in conflict with true user preferences. It is therefore very important to expose these conflicts to the user.

Secondly, the MAUT approach will rank multiple solutions in the same way, although the end users may not consider all solutions as indifferent. Weight adjustments are then necessary to differentiate the solutions and to achieve that trade-offs are made in a way as expected by the users. As small changes in weights can have a tremendous impact on the result, this process is difficult to achieve manually. Furthermore, there are trade-offs that cannot be reached by weight adjustments. An additive utility function can only characterize those solution as optimal that are on the convex hull of the solution space. However, there are Pareto-optimal solutions that represent valid trade-offs, although they do not belong to the convex hull. Often they may even represent better compromises than the MAUT-solutions (such as the leximin-optimal solutions in [4]). Finally, additive utility functions suppose complete preferential independence and are not able to deal with context-dependent preferences.

Although the completeness and independence assumptions of the MAUT model are of great benefit for the optimizer, they may hinder fruitful interactions between the users and the optimizers. The users may start with rather incomplete preferences and may want to refine them dependent on the solutions they get. The preference model should be sufficiently expressive and flexible to allow users to control the problem solving behaviour and its outcome.

In this paper, we follow the vision in [2] and argue that preferences can directly be used to control the problem solving process. Hence, we don't compile the preferences into a utility function, but design problem solving methods that directly use the original preferences. Concretely, we pursue a multi-objective

optimization approach to achieve these capabilities. The approach consists in decomposing the whole problem into alternative sequences of single-criterion optimization problems which can be solved by standard optimizers. The chosen sequence gives information that explains the optimality of the solution. Based on the explanation, the user can either accept the solution or modify the preferences. The problem solver in turn modifies the solution correspondingly. Preferences thus allow the user to interact with the problem solver and to control its behaviour, while avoiding overconstrained situations. The whole approach has successfully been applied to configuration problems [8], but is of interest for constraint programming in general.

The paper is organized as follows. We first introduce constraint satisfaction problems with preferences. We then step by step introduce our optimization approach. We start with atomic optimization steps, which optimize the preferences of a single criterion. We then show how the atomic optimization steps for multiple criteria can be sequenced, which leads us to lexicographic optimization. Different sequences of the criteria lead to different extreme solutions, which we characterize in terms of a Permute-operator. The user can choose among the sequences by imposing a partial importance ordering between criteria. As this lexicographic approach does not characterize compromises between conflicting criteria, we finally turn our attention to Pareto-optimal solutions. We can characterize them by alternative sequences of atomic optimization steps as well if we introduce auxiliary (binary) criteria that limit the penalization of the less important criteria. For each of these optimization problems, we give a solved form and discuss how partial preferences are completed to obtain this solved form. In this paper, we do not discuss the algorithms for effectively computing solutions, but focus on the information that justifies a solution. For each optimization problem, we define an explanation of optimality that helps the user to modify the preference model and the problem solver outcome.

2 Combinatorial Problems with Preferences

2.1 Variables and Domains

Throughout this papers, we consider a finite set of variables $\mathcal{X}$ where each variable $x \in \mathcal{X}$ has a domain $D(x)$. For example, consider three variables x_1, x_2, x_3 of a vacation configuration example. The domain of x_1 contains the possible activities of the vacation, the domain of x_2 contains the possible vacation destinations, and the domain of x_3 contains the possible hotel chains:

$$D(x_1) := \{\mathit{Casino}, \mathit{Cliff\text{-}Diving}, \mathit{Film\text{-}studios}, \mathit{Sea\text{-}parc}, \mathit{Wind\text{-}surfing}\}$$
$$D(x_2) := \{\mathit{Acapulco}, \mathit{Antibes}, \mathit{Honolulu}, \mathit{Los\ Angeles}, \mathit{Miami}\}$$
$$D(x_3) := \{\mathit{H1}, \mathit{H2}, \mathit{H3}, \mathit{H4}, \mathit{H5}, \mathit{H6}\}$$

Each value $v \in D(x)$ defines a possible value assignment $x = v$ to x. A set that contains exactly one of those value assignments for each variable in $\mathcal{X}$ and that contains no other elements is called an *assignment* to $\mathcal{X}$. For example, a

Activity x_1	**City** x_2
Casino	Antibes
Cliff-diving	Acapulco
Film Studios	Los Angeles
Sea-Parc	Antibes
Sea-Parc	Los Angeles
Wind-Surfing	Antibes
Wind-Surfing	Honolulu
Wind-Surfing	Miami

Table 1. Constraint c_1.

City x_2	**Hotel chain** x_3
Acapulco	H2
Acapulco	H6
Antibes	H3
Antibes	H5
Honolulu	H3
Honolulu	H5
Los Angeles	H2
Los Angeles	H4
Los Angeles	H6
Miami	H1
Miami	H4

Table 2. Constraint c_2.

wind-surfing vacation in Honolulu's H3 chain is represented by the following assignment:

$$\sigma_1 := \{x_1 = \textit{Wind-surfing}, x_2 = \textit{Honolulu}, x_3 = \textit{H3}\}$$

The set of all assignments to $\mathcal{X}$ is called the *problem space* of $\mathcal{X}$ and we denote it by $\mathcal{S}(\mathcal{X})$. Given an assignment S to $\mathcal{X}$ we can project it to a subset Y of the variables by choosing the value assignments to elements of Y:

$$S \mid Y := \{(x = v) \in S \mid x \in Y\} \tag{1}$$

For example, projecting the assignment σ_1 to the vacation activity and the vacation destination results into the following subset:

$$\sigma_1 \mid \{x_1, x_2\} := \{x_1 = \textit{Wind-surfing}, x_2 = \textit{Honolulu}\}$$

2.2 Constraints

We can restrict the problem space of $\mathcal{X}$ by defining constraints on variables in $\mathcal{X}$. A constraint c has a scope $X_c \subseteq \mathcal{X}$ and a 'relation' which we express by a set R_c of assignments to the scope X_c. This set can be specified explicitly in form of a table where each column corresponds to a variable in X_c, each row corresponds to an assignment in R_c, and the value v from a value assignment $(x = v) \in S$ is put in the cell for column x and row S. Tables 1 and 2 show two compatibility constraints of the vacation example. The first constraint describes the activities that are possible in a city and has the scope $\{x_1, x_2\}$. The second constraint shows which hotel chain is available in which city and has the scope $\{x_2, x_3\}$.

The relation R_c can also be specified by a logical formula that involves the variables from X_c and the operations from a given mathematical structure over

City x_2	**Region** z_2
Acapulco	Mexico
Antibes	Cote d'Azur
Honolulu	Hawaii
Los Angeles	California
Miami	Florida

Table 3. Criterion for city.

Hotel x_3	**Price** z_3	**Quality** z_4
H1	40	Economic
H2	60	Basic
H3	100	Basic
H4	100	Standard
H5	150	Standard
H6	200	Luxury

Table 4. Criteria for hotels.

the variable domains (such as arithmetic operations, boolean comparisons, and boolean operations). In this case, the relation R_c contains all assignments S that imply the formula. An example is the following constraint on the variables x, y, z:

```
if x >= 10 then z = x*y;
```

A constraint satisfaction problem CSP for $\mathcal{X}$ is given by a finite set of constraints $\mathcal{C}$ the scopes of which are all subsets of $\mathcal{X}$. A CSP is finite if all its domains and relations are finite. A constraint c is satisfied by an assignment S to $\mathcal{X}$ iff $S \mid X_c$ is an element of R_c. An assignment S is a *solution* of $\mathcal{C}$ iff it satisfies all constraints of $\mathcal{C}$. If a CSP has no solution then it is called inconsistent. It is often convenient to replace a CSP $\mathcal{C}$ by the conjunction $\bigwedge_{c\in\mathcal{C}} c$ of its constraints. Similarly, we can replace a conjunction $\bigwedge_{i=1}^{k} c_i$ be the set of its conjuncts $\{c_1, \ldots, c_k\}$. Hence all definitions and propositions referring to constraints C_1, C_2 can also be applied to CSPs and vice versa.

2.3 Criteria and Preferences

A CSP can have multiple solutions. In general, the user of a CSP application will not be satisfied with a randomly chosen solution, but prefers certain solutions to others. Since there may be an exponential number of solutions, we cannot expect that a system first generates all of them and then asks the user to express her preferences directly on the solution space. However, it is not necessary to express the preferences directly between the alternatives such as the solutions. In decision theory, preferences are formulated on criteria. A criterion is simply a mathematical function that maps an alternative to some value.

We directly encode those functions in our constraint language. Let Ω be a domain. A criterion z with domain Ω is an expression $f(x_1, \ldots, x_n)$ where $x_1, \ldots, x_n$ are variables from $\mathcal{X}$ and f is a function of signature $D(x_1) \times \ldots \times D(x_n) \to \Omega$. The function f is formulated with the operators of the constraint language (e.g. sum, min, max, conditional expression) or by a table. A criterion can also be equal to a variable x_i, namely if is formulated with the help of the identity function.

In the vacation example, we want to express preferences on the activity, the vacation region, and the price and the quality of the hotel chain. We therefore

introduce four criteria z_1, z_2, z_3, z_4 and their respective domains $\Omega_1, \Omega_2, \Omega_3, \Omega_4$. The criterion z_1 is equal to the vacation activity x_1 and has the domain $\Omega_1 := D(x_1)$. The other criteria are defined via a table (see Tables 3 and 4) and have the following domains:

$$\Omega_2 := \{\textit{California}, \textit{Cote d'Azur}, \textit{Florida}, \textit{Hawaii}, \textit{Mexico}\}$$
$$\Omega_3 := [0, 1000]$$
$$\Omega_4 := \{\textit{Economic}, \textit{Basic}, \textit{Standard}, \textit{Luxury}\}$$

We can evaluate the expression $f(x_1, \ldots, x_n)$ if an assignment σ for the variables in $\mathcal{X}$ is given. We denote the resulting value by $z(\sigma)$.

The domain Ω of a criterion z describes the possible outcomes in which the user is interested in. The user can compare these outcomes and formulate preferences on them. Preferences are modelled in form of a preorder $\succsim$ that can be decomposed into a strict part $\succ$ and indifference $\sim$. If $\omega_1 \succsim \omega_2$ holds for two outcomes $\omega_1, \omega_2 \in \Omega$, then this means that the outcome ω_1 is at least as preferred as ω_2. Please note that we do not require that the preference order $\succsim$ is complete. Hence, if may be the case that neither $\omega_1 \succsim \omega_2$, nor $\omega_2 \succsim \omega_1$ hold. In this case, the preference between ω_1 and ω_2 is not specified, meaning that the user can refine the preference order later on.

A preorder is a transitive and reflexive relation. It is not necessary that the user specifies this order exhaustively. Given a relation $R \subseteq \Omega \times \Omega$ of user preferences, we obtain a preorder $\succsim$ by determining the reflexive and transitive closure of R. The result is the smallest preorder that contains the specified user preferences. For example, suppose that the user prefers Hawaii at least as California, California at least as the Cote d'Azur and as Florida, and the Cote d'Azur at least as Mexico. Similarly, the user prefers cliff diving at least as sea parc visits, sea parc visits at least as casino visits and at least as wind-surfing, and casino visits and wind-surfing at least as visits of film studios. Furthermore, the user prefers casino visits at least as wind-surfing and vice versa. Figures 2 and 3 show these preferences (straight arcs and dotted-dashed arcs) and the corresponding preorders (any arc) in a graphical form.

We are mainly interest in the strict part $\succ$ of this preorder, namely the set of all pairs (ω_1, ω_2) in $\Omega \times \Omega$ such that $\omega_1 \succsim \omega_2$ holds, but not $\omega_2 \succsim \omega_1$. The absence of a strict preference between two outcomes can either signify indifference or incompleteness. The strict part of a preorder is a strict partial order, i.e. an irreflexive and transitive relation. We write $\omega_1 \succeq \omega_2$ as a short-hand for $\omega_1 \succ \omega_2$ or $\omega_1 = \omega_2$. The relation $\succeq$ is a subset of the preorder $\succsim$, but the inverse does not hold in general. In the example, the strict parts are obtained by suppressing the dotted-dashed arcs and the reflexive arcs of the form (ω, ω).

We also consider the case where the preorder $\succsim$ is complete. The strict part of a complete preorder is a ranked order, i.e. a strict partial order satisfying the following property for all outcomes $\omega_1, \omega_2, \omega_3$: if $\omega_1 \succ \omega_2$ then either $\omega_3 \succ \omega_2$ or $\omega_1 \succ \omega_3$. Ranked orders can be represented by utility functions u that map assignments to a numerical value such that $u(\omega_1) > u(\omega_2)$ iff $\omega_1 \succ \omega_2$. A complete preorder that is additionally anti-symmetric is a total order. The

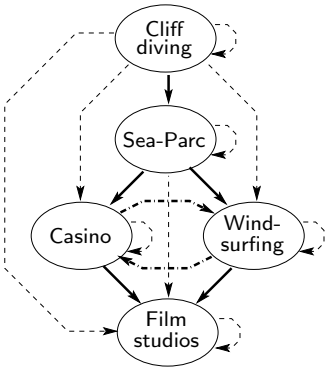

Fig. 2. Preferences $\succsim_1$ on activities.

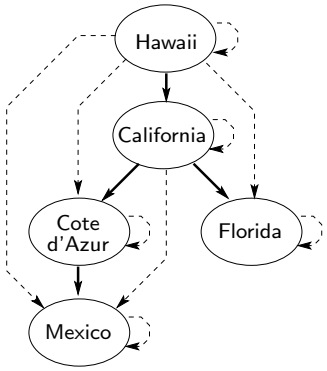

Fig. 3. Preferences $\succsim_2$ on regions.

strict part of a total order is a strict total order, i.e. a strict partial order that is complete on all outcomes that are different.

The user can formulate preferences on multiple criteria. Consider m criteria $z_1, \ldots, z_m$ with domains $\Omega_1, \ldots, \Omega_m$. Furthermore, consider a strict partial order $\succ_i$ for each domain Ω_i. We say that the pair $p_i := \langle z_i, \succ_i \rangle$ of the i-th criterion and the i-th order is a *preference*. This terminology is, for example, used in [9]. If a preference $\langle z_i, > \rangle$ uses the increasing order $>$, then we abbreviate it by *maximize*(z_i). Similarly, we abbreviate a preference $\langle z_i, < \rangle$ based on the decreasing order by *minimize*(z_i). For example, price minimization is expressed as *minimize*(z_3) in the vacation example. Preferences thus generalize optimization objectives as used in CP or MIP.

2.4 Wishes

Wishes are constraints that should be satisfied if possible. A wish can be modelled by a binary criterion, namely the truth value of the constraint, and an implicit preference ordering that prefers true to false. Given a constraint c, we define its truth value z_c for an assignment σ as follows:

$$z_c(\sigma) := \begin{cases} 1 \text{ if } \sigma \text{ satisfies } c \\ 0 \text{ otherwise.} \end{cases} \tag{2}$$

A wish for c is a preference $\langle z_c, > \rangle$. We abbreviate it by *wish*(c).

3 Preference-based Problem Solving

Combinatorial problems with preferences are classically solved by compiling all preferences in to a single utility function and by determining a solution of the constraints that has maximal utility. The utility function is also used to measure

the quality of the solution that is returned by the optimizer. A 100% optimality means that as many preferences as possible have been satisfied. However, the optimizer does not give an explanation of optimality in terms of the original preferences. If the user specified a preference $\langle z, \succ \rangle$, she wants to know whether the criterion z has its best value in the solution. If not, the user wants to get an explanation why no better outcome has been obtained for the criterion. If the criterion z is in conflict with other criteria then the explanation should indicate this conflict and the trade-off that has been made.

We show that explanations of this kind can be produced if the problem solving process is based on the original preferences. If there is only a single preference, then we can set up an optimization problem that directly optimizes this preference. In this case, we don't obtain any mismatch between the preferences and the optimization objectives. If multiple preferences are given, we therefore treat them step by step in a given sequence. In each step, we solve a classical optimization problem that optimizes one of the original preferences. The sequential approach imposes that some criteria get priority over the others. We therefore treat alternative sequences and are thus able to determine the extreme solutions where trade-offs are always decided in favour of more important criteria. In order to produce other trade-offfs, we introduce auxiliary criteria and wishes.

3.1 Atomic Optimization Step

Consider a single preference $\langle z, \succ \rangle$. We are interest in those solutions of the constraints C that assign a $\succ$-maximal value to the criterion z. A solution σ assigns a $\succ$-maximal value v to the criterion z iff there is no other solution σ^* that assigns a better value v^* to z, i.e. a value that satisfies $v^* \succ v$. To characterize those solutions, we introduce an operator, denoted by $\mathit{Max}(\langle z, \succ \rangle)$, that maps a constraint C to a new constraint that is satisfied by exactly the solutions that assign $\succ$-maximal values to z. Hence, $\mathit{Max}(\langle z, \succ \rangle)(C)$ denotes the optimization problem that need to be solved. As it concerns a single criterion, it need not be decomposed further and thus represents an atomic optimization step.

If $>$ is a total order, then the preference $\langle z, > \rangle$ can be modelled by an ordinal utility function u that maps each possible outcome ω in the domain of z to a unique numeric utility value $u(\omega)$. We then obtain a classical optimization problem, namely that of maximizing $u(z)$. This problem can be solved by standard optimizers (such as constraint-based Branch-and-Bound), which find a solution of maximum value u^* for $u(z)$. Since we supposed that $>$ is a total order, there is a unique outcome ω^* in the domain of z that has the utility u^*, Hence, each solution of C that is optimal w.r.t. the preference $\langle z, > \rangle$ satisfies the constraint $C \wedge z = \omega^*$ and the inverse is true as well. Hence, the following equivalence holds for preferences with total orders:

$$\mathit{Max}(\langle z, > \rangle)(C) \equiv C \wedge z = \omega^* \tag{3}$$

We can thus characterize the entire set of optimal solutions by the constraint $C \wedge z = \omega^*$ and replace the original problem $\mathit{Max}(\langle z, > \rangle)(C)$ by this constraint

without loosing any optimal solution. Subsequent optimization steps can then further reduce the set of optimal solutions of $Max(\langle z, > \rangle)(C)$. We also say that $C \wedge z = \omega^*$ is the *solved form* of the optimization problem $Max(\langle z, > \rangle)(C)$.

Once the optimization problem has been solved, the user may ask for explanations of optimality such as

1. Is ω^* a best value in Ω?
2. Why can't z have a value better than ω^*?
3. Why hasn't the value ω been chosen for z?

The answer to the first question is yes iff Ω does not contain another value ω^{**} such that $\omega^{**} > \omega^*$. If ω^* is a best value, then the criterion has obtained its overall optimal value meaning that the preferences of the user has been addressed to complete satisfaction. Otherwise, there are better values ω^{**} in Ω and the user might ask why none of those values have been assigned to z. The set of those values is characterized by the unary constraint $z > \omega^*$, which can easily be encoded in a constraint solver (e.g. by removing all values smaller than or equal to ω^* from the domain of z). Since ω^* is the optimal value, the constraint $C' := C \wedge z > \omega^*$ is inconsistent. For the purpose of explanation, we represent the conjunctive constraint C' by the set of all its conjuncts and determine a conflict for C', i.e. a minimal subset X' of C' that is inconsistent. The conflict can, for example, be computed by the QuickXplain-algorithm [7]. Since C is assumed to be consistent, the conflict X' needs to contain the constraint $z > \omega^*$. The other elements $X := X' \cap C$ of the conflict then explain why z can't have a value better than ω^* w.r.t. the order $>$. These constraints X defeat any value for z that is better than the optimum ω^*. The user might also ask why z has not obtained a specific value ω. If ω is greater than the optimum ω^*, then the choice $z = \omega$ is defeated by X. Otherwise, ω is smaller than the optimum ω^* and has not been chosen for that reason. In order to give the right answer, the explanation of optimality needs both include the defeaters X as well as the ordering $>$.

Let σ be a solution of the problem $Max(q)(C)$ with total preferences $q := \langle z, > \rangle$. An *explanation of the Max(q)-optimality* of σ is a triple (q, ω^*, X) such that ω^* is equal to the optimum $z(\sigma)$ and X is a minimal subset of the set of conjuncts of C for which $X \cup \{z > \omega^*\}$ is inconsistent.

The atomic optimization problems of our vacation examples all allow the best value for all the preferences except for price minimization. The optimum of the problem $Max(minimize(price))(C)$ is 40 and an explanation of optimality is $(minimize(price), 40, \{c_2\})$.

For total orders, we are now able to transform the optimization problem into a solved form and to explain the optimality.

3.2 Alternative Optimizations

User preferences will usually be partial orders, in particular at the beginning of the interactive problem solving process. Hence, there may be two solutions σ_1 and σ_2 of $Max(\langle z, \succ \rangle)$ that assign incomparable values ω_1 and ω_2 to the criterion

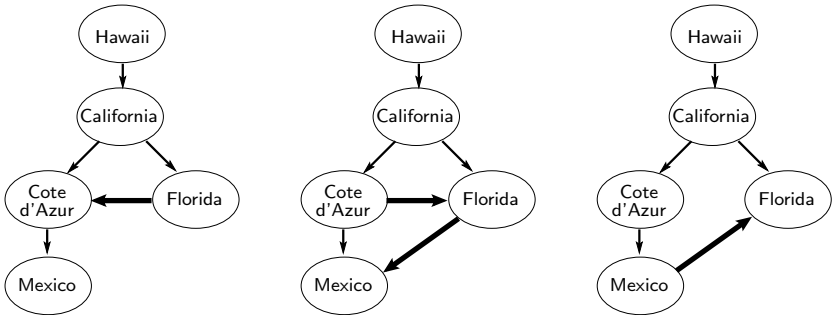

Fig. 4. The three linearizations $>_{2,1}, >_{2,2}, >_{2,3}$ of the region preferences.

z. These values are neither equal, nor one is preferred to the other, meaning that $\omega_1 \neq \omega_2$, $\omega_1 \not\succ \omega_2$, and $\omega_2 \not\succ \omega_1$ all hold. As a consequence, the solved form of $\mathit{Max}(\langle z, \succ \rangle)(C)$ will be a disjunction of assignments of the form $z = \omega_i$. If Ω^* is the set of all values that are assigned to the criterion z in the different solutions of the optimization problem, then the solved form of the problem is as follows:

$$\mathit{Max}(\langle z, \succ \rangle)(C) \equiv \bigvee_{\omega^* \in \Omega^*} C \wedge z = \omega^* \tag{4}$$

We use the partial preferences order $\succ_2$ on the vacation regions to give an example. Suppose that the cities Honolulu and Los Angeles are not possible due to booking problems. We thus obtain the problem $\mathit{Max}(\langle z_2, \succ_2 \rangle)(C')$ where $C' := C \wedge x_2 \neq \mathit{Honolulu} \wedge x_2 \neq \mathit{Los\ Angeles}$. In this case, Ω^* contains the regions Florida and Cote d'Azur and the solved form is $C' \wedge (z_2 = \mathit{Florida} \vee z_2 = \mathit{Cote\ d'Azur})$.

This characterization does not yet give us a way to compute the solved form as a standard optimizer needs a total order (or a corresponding utility function as discussed in the previous sub-section). However, we can reduce the original optimization problem for partial orders to optimization problems for total orders. We just choose a total order $>$ on Ω that is a superset of $\succ$. We call this a linearization of $\succ$ and we denote the set of all linearizations of $\succ$ by $\tau(\succ)$. For example, we obtain three linearizations $>_{2,1}, >_{2,2}, >_{2,3}$ for the preferences $\succ_2$ on the vacation region (cf. Figure 4). Thanks to the linearizations, we are able to transform the optimization problem for partial orders into alternative optimization problems on total orders since the following property holds:

$$\mathit{Max}(\langle z, \succ \rangle)(C) \equiv \bigvee_{> \in \tau(\succ)} \mathit{Max}(\langle z, > \rangle)(C) \tag{5}$$

As a consequence, we can use the solved forms of the alternative optimization problems in order to obtain the solved form for the optimization problem of the partial orders. Furthermore, the explanations of optimality of the different

alternatives allow us to define an explanation of optimality for the original problem. Consider of a solution σ of the problem $Max(p)(C)$ with partially ordered preferences $p := \langle z, \succ \rangle$. An explanation of the $Max(q)$-optimality of σ is also an *explanation of the Max(p)-optimality* of σ iff $q = \langle z, > \rangle$ and $>$ is a linearization of $\succ$. As such a linearization $>$ of $\succ$ exists, the explanation of $Max(p)$-optimality also exists.

In our example, we consider a solution σ_1 that chooses Florida (i.e. $z(\sigma_1) = Florida$) and a solution σ_2 that chooses the Cote d'Azur (i.e. $z(\sigma_2) = Cote\ d'Azur$). There are three linearizations of the partial preferences. The first one, namely $>_{2,1}$ prefers Florida to the Cote d'Azur and thus helps to explain the optimality of σ_1. The other two, namely $>_{2,2}, >_{2,3}$ prefer Cote d'Azur to Florida. Any of them can be used in an explanation of the optimality of σ_2 such as the following one:

$$\xi := (\langle z_2, >_{2,2} \rangle, Cote\ d'Azur, \{x_2 \neq Honolulu, x_2 \neq Los\ Angeles\})$$

If this explanation is submitted to the user, she can ask why the other regions have not been selected. The options Hawaii and California are defeated by the conflict that is reported by the explanation. The option Mexico has been discarded since the proposed option Cote d'Azur has been preferred by the user. And the option Florida has been discarded since the linearization $>_{2,2}$ prefers the Cote d'Azur. If the user is not satisfied with the last response, she can refine the preference model and add a preference $Florida \succ_2' Cote\ d'Azur$. This will eliminate the current solution. Hence, the linearizations of the user preferences are an important part of the explanation as they give hints to the user how to modify undesired solutions.

According to formula 5, it is necessary to consider all linearizations of the strict partial order $\succ$ in order to compute the solved form of the optimization problem with partial preferences. However, we have seen that some of those linearizations lead to the same solution and the preference-based search method [6] exploits this fact and explores only a subset of the linearization when computing the preferred solutions.

3.3 Lexicographical Approach

User preferences do not concern a single criterion, but multiple criteria. We therefore consider the preferences $p_1, \ldots, p_n$ on n criteria. We suppose that p_i has the form $\langle z_i, \succ_i \rangle$. Each preference gives rise to an optimization operator $Max(p_i)$. The question is how these optimization operators should be composed, in particular since the preferences may be in conflict. The preferences are in conflict on the constraints C iff the conjunction $\bigwedge_i Max(p_i)(C)$ has no solution. In this case, we need to find a trade-off between the preferences. The easiest way to start with is to treat the preferences in a given order and to decide trade-offs in favour of the preferences that are considered first. Lexicographic optimization defines an ordering on the solution space based on this importance principle. Consider two assignments σ_1 and σ_2. We consider the tuples of values

that both assignments assign to the criteria and define a lexicographical order $\succ_{lex}$ between them. The relation

$$(z_1(\sigma_1), \ldots, z_n(\sigma_1)) \succ_{lex} (z_1(\sigma_2), \ldots, z_n(\sigma_2)) \tag{6}$$

holds iff there exists a k such that $z_k(\sigma_1) \succ_k z_k(\sigma_2)$ and $z_i(\sigma_1) = z_i(\sigma_2)$ for $i = 1, \ldots, k-1$. A solution σ of C is a *lexicographically optimal solution* of C iff there is no other solution σ^* of C such that $(z_1(\sigma^*), \ldots, z_n(\sigma^*))$ is lexicographically better than $(z_1(\sigma), \ldots, z_n(\sigma))$. We introduce the lexicographic optimization operator $\mathit{Lex}(p_1, \ldots, p_n)$ that maps a constraint C to a new constraint C' that is satisfied by the lexicographically optimal solutions of C.

Similar to equation 5, we can transform the problem $\mathit{Lex}(p_1, \ldots, p_n)(C)$ into a solved form by considering a linearization for each strict partial order $\prec_i$. We define $\tau(\langle z_i, \succ_i \rangle)$ as the set of all preferences $\langle z_i, >_i \rangle$ for which $>_i$ is a linearization of $\succ_i$. Furthermore, we define $\tau(p_1, \ldots, p_n)$ as the Cartesian product $\tau(p_1) \times \ldots \times \tau(p_n)$. The following equivalence holds between lexicographic optimization problems with partial preferences and the problems obtained by linearizing these preferences:

$$\mathit{Lex}(p_1, \ldots, p_n)(C) \equiv \bigvee_{(q_1, \ldots, q_n) \in \tau(p_1, \ldots, p_n)} \mathit{Lex}(q_1, \ldots, q_n)(C) \tag{7}$$

A lexicographic optimization problem $\mathit{Lex}(q_1, \ldots, q_n)(C)$ with total preferences can then be transformed to a sequence of single-criterion optimization problems which can be solved by a standard optimizer:

$$\begin{aligned} \mathit{Lex}(q_1)(C) &\equiv \mathit{Max}(q_1)(C) \\ \mathit{Lex}(q_1, \ldots, q_n)(C) &\equiv \mathit{Lex}(q_2, \ldots, q_n)(\mathit{Max}(q_1)(C)) \end{aligned} \tag{8}$$

The solved form has the form $C \wedge z = \omega_1^* \wedge \ldots z = \omega_n^*$. In the vacation example, the problem $\mathit{Lex}(\langle z_1, \succ_1 \rangle, \langle z_2, \succ_2 \rangle, \mathit{minimize}(z_3))(C)$ has the solved form $C \wedge z_1 = \mathit{Cliff\ diving} \wedge z_2 = \mathit{Mexico} \wedge z_3 = 60$.

Explanations for lexicographical optimality are sequences of explanations for single-criterion optimization problems. Consider a solution σ of the problem $\mathit{Lex}(p_1, \ldots, p_n)(C)$. A sequence $(\xi_1, \ldots, \xi_n)$ is called an *explanation of the $\mathit{Lex}(p_1, \ldots, p_n)$-optimality* of σ iff there exists totally ordered preferences $(q_1, \ldots, q_n) \in \tau(p_1, \ldots, p_n)$ such that ξ_i is an explanation of $\mathit{Max}(q_i)$-optimality of the i-th optimization problem $\mathit{Max}(q_i)(C \wedge z_1 = z_1(\sigma) \wedge \ldots \wedge z_{i-1} = z_{i-1}(\sigma))$. Explanations of $\mathit{Lex}(p_1, \ldots, p_n)$-optimality always exist and can easily be produced when solving the problem.

Consider a solution σ_1 of the vacation configuration problem $\mathit{Lex}(\langle z_1, \succ_1 \rangle, \langle z_2, \succ_2 \rangle, \mathit{minimize}(z_3))(C)$. An explanation of optimality is (ξ_1, ξ_2, ξ_3) where

$$\begin{aligned} \xi_1 &:= (\langle z_1, >_{1,1} \rangle, \mathit{Cliff\ diving}, \{\}) \\ \xi_2 &:= (\langle z_2, >_{2,1} \rangle, \mathit{Mexico}, \{z_1 = \mathit{Cliff\ diving}\}) \\ \xi_3 &:= (\langle z_3, < \rangle, 60, \{z_2 = \mathit{Mexico}\}) \end{aligned}$$

We can depict this explanation in a graphical form. Each triple ξ_i is represented by a node. There is an edge from ξ_i to ξ_j if the defeaters of z_j contain a constraint

of the form $z_i = \omega_i$ meaning that the optimal value for z_i helped to defeat the better values for z_j. Figure 7 shows the explanation of the lexicographical optimality of σ_1 in graphical form. This graphical form exhibits the conflicts between the criteria. The edges go from more important criteria to less important criteria and show that the conflicts have been resolved in favour of the more important criteria.

3.4 Alternative Sequentializations

Whereas lexicographical optimization is one of the fundamental approaches of multi-criteria optimization, few attention has been paid to the choice of the importance ordering. Most approaches use a static importance ordering among the criteria. However, it is often natural to try out different orderings of the criteria. Thus, a user can get a clear idea of the extreme solutions before exploring compromises.

We characterize each family of extreme solutions by a permutation π of the n preferences $(p_1, \ldots, p_n)$. Given such a permutation, we optimize the criteria in the ordering $p_{\pi_1}, \ldots, p_{\pi_n}$. Let Π be the set of all those permutations. We introduce a new operator, called $\mathit{Permute}(p_1, \ldots, p_n)(C)$, that maps the constraint C to a constraint C' that is satisfied by all extreme solutions of C. This constraint is equivalent to a disjunction of lexicographic optimization problems:

$$\mathit{Permute}(p_1, \ldots, p_n)(C) \equiv \bigvee_{\pi \in \Pi} \mathit{Lex}(p_{\pi_1}, \ldots, p_{\pi_n})(C) \tag{9}$$

As explanations of the different lexicographic optimization problems preserve the ordering of the criteria, it is straightforward to combine the explanations of those problems. Consider the solution σ of the problem $\mathit{Permute}(p_1, \ldots, p_n)(C)$, An explanation $(\xi_{\pi_1}, \ldots, \xi_{\pi_n})$ of the $\mathit{Lex}(p_{\pi_1}, \ldots, p_{\pi_n})$-optimality of σ is an *explanation of the* $\mathit{Permute}(p_1, \ldots, p_n)$*-optimality* of σ iff π is a permutation.

The vacation example $\mathit{Permute}(\langle z_1, \succ_1 \rangle, \langle z_2, \succ_2 \rangle, \mathit{minimize}(z_3))(C)$ has three extreme solutions that are justified by different importance orderings. Solution σ_1 chooses Cliff-Diving in Mexico with hotel costs of 60$ based on the order z_1, z_2, z_3. Solution σ_2 proposes Wind-Surfing in Hawaii with hotel costs of 100$ by following the order z_2, z_1, z_3. Solution σ_3 offers Wind-Surfing in Florida with hotel costs of 40$ based on the order z_3, z_2, z_1. Figure 5 displays the possible combinations of the criteria values (in a way that is similar to the micro-structure of constraint networks). The three extreme solutions are represented by thick lines. As an example of an explanation, we give one for the optimality of the last solution:

$$\begin{array}{l}((\langle z_3, < \rangle, 40, \{\}),\\ \ (\langle z_2, >_{2,1} \rangle, \mathit{Florida}, \{z_3 = 40\}),\\ \ (\langle z_1, >_{1,1} \rangle, \mathit{Wind\ surfing}, \{z_2 = \mathit{Florida}\}))\end{array}$$

This explanation exhibits the importance ordering of the criteria and thus explains how conflicts between preferences are resolved.

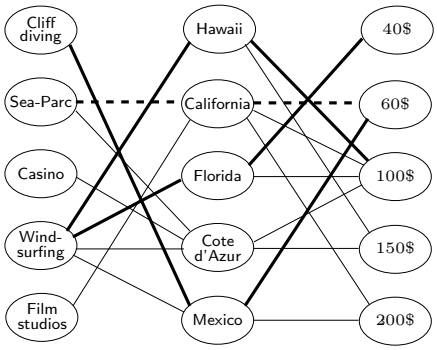

Fig. 5. Combinations of criteria.

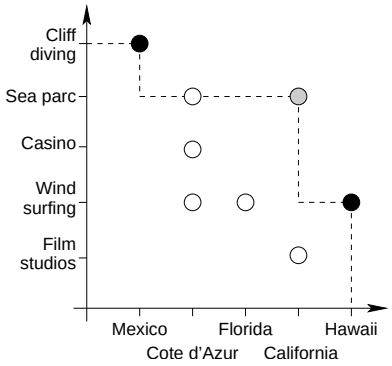

Fig. 6. Trade-off between z_1 and z_2.

3.5 Importance Preferences

Thanks to the explanations, the user can inspect the conflicts between criteria and the way they have been resolved. If the user is not satisfied with such a conflict resolution, she can change it by reordering the criteria. For example, suppose that the user is not satisfied with a vacation package of a high price. The user now learns that this high price is caused by a very good quality rating, which was chosen to be more important. The user wants to give price minimization a higher importance than quality maximization and thus expresses an importance ordering between the price criterion z_3 and the quality criterion z_4. We formalize this importance ordering in terms of a strict partial order $I \subseteq \mathcal{Z} \times \mathcal{Z}$ on the criteria set $\mathcal{Z} := \{z_1, \ldots, z_n\}$. In our example, we have

$$I := \{(z_3, z_4)\} \tag{10}$$

We now use this importance ordering to restrict the set of extreme solutions. We consider only those permutations π of the preferences $p_1, \ldots, p_n$ that respect the importance ordering. More important criteria need to be ranked first. Hence, the permutation π respects the importance ordering I iff the following property holds for all i, j:

$$(z_{\pi_i}, z_{\pi_j}) \in I \text{ implies } i < j \tag{11}$$

Let $\mathit{\Pi}(I)$ be the set of all permutations π that respect I. Furthermore, we introduce a variant of the permute-operator $\mathit{Permute}(p_1, \ldots, p_n : I)(C)$ that is restricted to those extreme solutions that are obtained by the permutations in $\mathit{\Pi}(I)$:

$$\mathit{Permute}(p_1, \ldots, p_n : I)(C) \equiv \bigvee_{\pi \in \mathit{\Pi}(I)} \mathit{Lex}(p_{\pi_1}, \ldots, p_{\pi_n})(C) \tag{12}$$

Preferences between criteria may eliminate extreme solutions, but do not add new ones:

$$I \subseteq I^* \text{ implies } \mathit{Permute}(p_1, \ldots, p_n : I^*)(C) \Rightarrow \mathit{Permute}(p_1, \ldots, p_n : I)(C) \tag{13}$$

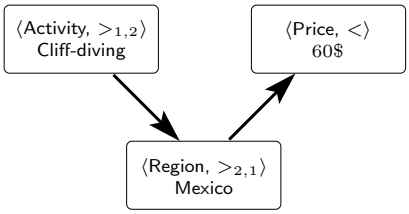

Fig. 7. Explanation of lexicographic optimality of σ_1.

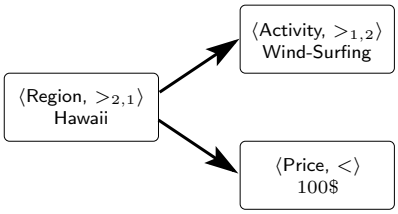

Fig. 8. Explanation of lexicographic optimality of σ_2.

The importance ordering I also impacts explanations. Consider the solution σ of the problem $Permute(p_1, \ldots, p_n : I)(C)$, An explanation $(\xi_{\pi_1}, \ldots, \xi_{\pi_n})$ of the $Lex(p_{\pi_1}, \ldots, p_{\pi_n})$-optimality of σ is an *explanation of the* $Permute(p_1, \ldots, p_n : I)$*-optimality* of σ iff π is a permutation in $\Pi(I)$.

We now discuss the effect of importance preferences on the vacation example. Suppose that solution σ_3 has been submitted to the user. This solution proposes Wind-surfing in Florida with hotel costs of 40$ based on the ordering z_3, z_2, z_1. The user criticizes this explanation by stating that the choice of the vacation activity and the vacation region is more important than the price. The importance preferences $I_1 := \{(z_1, z_2), (z_1, z_3)\}$ are added to the preference model. As the order z_3, z_2, z_1 does not respect these importance preferences, solution σ_3 is no longer proposed. Solutions σ_1 and σ_2 still have valid importance orderings. Figures 7 and 8 give explanations of optimality of these solutions of the problem $Permute(\langle z_1, \succ_1\rangle, \langle z_2, \succ_2\rangle, minimize(z_3) : I_1)(C)$.

3.6 Trade-offs and Preference Limits

Extreme solutions are resolving conflicts between preferences completely in favour of the more important criteria. If two criteria z_1 and z_2 are in conflict, then z_1 gets its best value, while z_2 is completely penalized. Changing the order completely turns the balance around and the conflict is decided in favour of z_2 while penalizing z_1. However, it is also possible to use compromises and to trade a small improvement for z_2 against a small penalization of z_1 without changing the order of the criteria. Pareto-optimality captures all the possible trade-offs.

The notion of Pareto-optimality does not impose any importance ordering on the criteria. It extends the partial ordering on the different criteria to a partial ordering on the complete criteria space without making any particular assumption. An assignment σ^* *Pareto-dominates* another assignment σ iff 1.. σ^* and σ differ on at least one criterion z_k (i.e. $z_k(\sigma^*) \neq z_k(\sigma)$) and 2. σ^* is strictly better than σ on all criteria on which they differ (i.e. $z_i(\sigma^*) \neq z_i(\sigma)$ implies $z_k(\sigma^*) \succ_k z_k(\sigma)$). An equivalence definition consists in saying that σ^* dominates σ iff σ^* is at least as as good as σ on all criteria (i.e. $z_k(\sigma^*) \succeq_k z_k(\sigma)$) and there is at least one criterion where σ^* is strictly better than σ (i.e. $z_k(\sigma^*) \succ_k z_k(\sigma)$). Pareto-dominance defines a strict partial order on the criteria space.

A solution σ of C is a *Pareto-optimal solution* of C iff there is no other solution σ^* that Pareto-dominates σ. We introduce an operator $Pareto(p_1, \ldots, p_n)$ that maps a constraint C to a new constraint C' that is only satisfied by the Pareto-optimal solutions of C. Non-Pareto-optimal solutions clearly are not desirable since there are solutions that are better on one or more criteria while keeping the other criteria unchanged.

Figure 5 shows a Pareto-optimal solution for the vacation example which is not an extreme solution. This solution, which we name σ_4, proposes Sea-parc visits in California with Hotel costs of 60$. Hence, σ_4 does not choose the best value for any criteria, but it is Pareto-optimal since we cannot improve any criterion without getting a worse value for another criterion. Figure 6 shows the trade-off between the vacation activity z_1 and the vacation region z_2. The solution σ_4 is situated between the two extreme solutions σ_1 and σ_2.

As for lexicographical optimization, we can linearize the partially ordered preferences and transform a Pareto-optimization problem into a disjunction of Pareto-optimization problems with totally ordered preferences:

$$Pareto(p_1, \ldots, p_n)(C) \equiv \bigvee_{(q_1,\ldots,q_n)\in\tau(p_1,\ldots,p_n)} Pareto(q_1, \ldots, q_n)(C) \tag{14}$$

However, there is no direct way to transform a Pareto-optimization problem into a solved form even if it is based on total orders. One approach to solve those problems consist in generalizing optimization methods such as Branch-and-Bound search to a partial order. The approach is pursued in [5]. Branch-and-bound search for a partial order needs to maintain a whole Pareto-optimal frontier which might become rather inefficient. More importantly, this method does not produce explanations of optimality that exhibit the trade-offs between criteria. For this reason, we do not follow this approach, but seek ways to solve Pareto-optimization problems by alternative sequences of classical optimization steps. We observe that all extreme solutions are Pareto-optimal (see [6]):

$$Permute(p_1, \ldots, p_n)(C) \Rightarrow Pareto(p_1, \ldots, p_n)(C) \tag{15}$$

Hence, we can start with extreme solutions when determining Pareto-optimal solutions. An extreme solution is entirely characterized by an ordering of the criteria (and of the user preferences). However, these orderings do not characterize those compromises between two conflicting criteria where none of the two criteria gets its best value. We need to insert additional steps into the sequence of optimization problems. An extreme solution is always in favour for the most important criterion and completely penalizes the less important ones. For example, consider two conflicting preferences $\langle z_1, \succ_1\rangle$ and $\langle z_2, \succ_2\rangle$. Let σ be the extreme solution for the ordering z_1, z_2. The criterion z_1 gets its best feasible value, namely $z_1(\sigma)$, whereas z_2 is penalized and gets the value $z_2(\sigma)$. If we want to obtain a better value for the less important criterion z_2, then we need to limit its penalization before optimizing the more important criterion z_1. For this purpose, we add the constraint $z_2 \succ z_2(\sigma)$ before optimizing z_1. If this constraint is satisfiable, then the optimization of z_1 will produce the best solution for z_1

under the constraint $z_2 \succ z_2(\sigma)$. If such a penalization limit is infeasible, then it should be retracted. To achieve this, we represent penalization limits of the form $z_2 \succ z_2(\sigma)$ as wishes.

We introduce a wish for each criterion and for each possible value of this criterion:

$$\mathit{limits}(\langle z_i, \succ_i \rangle) := \langle \mathit{wish}(z_i \succeq_i \omega_1), \ldots, \mathit{wish}(z_i \succeq_i \omega_n) \rangle \tag{16}$$

Furthermore, we consider the set I of importance preferences stating that wishes for worse outcomes precede wishes for better outcomes:

$$I := \{(\mathit{wish}(z_i \succeq_i \omega), \mathit{wish}(z_i \succeq_i \omega^*)) \mid \omega^* \succ \omega, i = 1, \ldots, n\} \tag{17}$$

It is a common modelling technique in Operations Research to transform an integer variable into a set of binary variables. Hence, our transformation has nothing unusual except that it applies to the criteria and not to the decision variables. The Pareto-optimal solutions of the original model then correspond exactly to the extreme solutions of the binary model (cf. theorem 1 in [6]):

$$\mathit{Pareto}(p_1, \ldots, p_n)(C) \equiv \mathit{Permute}(\mathit{limits}(p_1), \ldots, \mathit{limits}(p_n) : I)(C) \tag{18}$$

This correspondence allows us to transform Pareto-optimal solutions into a solved form and to define explanations of optimality.

Interestingly, the original optimization steps of the form $\mathit{Max}(\langle z_i \succ_i \rangle)(C')$ have disappeared in this new characterization. If the result of this optimization if ω', then this step corresponds to the last successful wish on z_i, namely $\mathit{Max}(\mathit{wish}(z_i \succeq_i \omega'))(C')$. Each Pareto-optimal solution is thus characterized by a sequence of wishes and there are multiple wishes for the same criterion z_i. There are logical dependencies between wishes that allow us to speed up the solving process and to simplify the explanations. If $\mathit{wish}(z_i \succeq_i \omega)$ fails, then all wishes $\mathit{wish}(z_i \succeq_i \omega')$ for better outcomes $\omega' \succ \omega$ will also fail. Furthermore, if $\mathit{wish}(z_i \succeq_i \omega)$ succeeds and is directly preceded by a wish for the same criterion then it subsumes this previous wish and the previous wish can be removed from explanations of optimality.

Explanations of Pareto-optimality are thus obtained by determining subsequences of explanations for lexicographical optimality of the binary preference model. Let σ be a Pareto-optimal solution of $\mathit{Pareto}(p_1, \ldots, p_n)(C)$. Then σ is an extreme solution of $\mathit{Permute}(\mathit{limits}(p_1), \ldots, \mathit{limits}(p_n) : I)(C)$ and there exists an explanation $(\xi_1, \ldots, \xi_m)$ of optimality of this extreme solution. Let ξ_j be $(\langle \mathit{wish}(z'_j \succeq \omega'_j), > \rangle, v_j, X_j)$. We say that ξ_j is *relevant* iff the wish is successful (i.e. $v_j = 1$) and the next triple concerns a different criterion (i.e. $z'_{j+1} \neq z'_j$). An explanation of Pareto-optimality of σ is the sequence $(\xi_{j_1}, \ldots, \xi_{j_k})$ of the relevant triples from $(\xi_1, \ldots, \xi_n)$ in the original order, i.e. $j_1 < j_2 < \ldots < j_k$. These explanations can still contain multiple wishes for the same criterion. The last wish for the criterion in an explanation determines the value of the criterion. The other wishes limit the penalization of the criterion before optimizing other criteria. Hence, the explanation highlights the critical choices that need to be made in order to obtain the Pareto-optimal solution σ.

The solving algorithms for Pareto-optimal solutions in [6] is based in wishes and is easily able to provide these explanations of optimality. The explanations can also be obtained for Wassenhove and Gelders' algorithm for bicriterion optimization [10].

We give an explanation for the Pareto-optimal solutions σ_1 and σ_4 of the vacation example. The explanation for σ_1 consists of one wish for each criterion. Each of these wishes assigns the optimal value.

$$\begin{array}{l}((\mathit{wish}(z_1 \succeq_1 \mathit{Cliff\ diving}), 1, \{\}),\\ (\mathit{wish}(z_2 \succeq_2 \mathit{Mexico}), 1, \{z_1 \succeq_1 \mathit{Cliff\ diving}\}),\\ (\mathit{wish}(z_3 \leq 60), 1, \{z_2 \succeq_2 \mathit{Mexico}\}))\end{array}$$

When this explanation is presented to the user, she might criticize it by saying that the vacation region has been penalized too much by its defeater, which is the wish $w_1 := \mathit{wish}(z_1 \succeq_1 \mathit{Cliff\ diving})$. The user therefore adds a wish $w_2 := \mathit{wish}(z_2 \succeq_2 \mathit{Cote\ d'Azur})$ to limit this penalization. This wish needs to get higher importance than w_1 to be effective in all cases. The user therefore adds the importance statement (w_2, w_1) as well. This leads to a new solution, namely σ_4 and the following explanation:

$$\begin{array}{l}((\mathit{wish}(z_2 \succeq_2 \mathit{Cote\ d'Azur}), 1, \{\}),\\ (\mathit{wish}(z_1 \succeq_1 \mathit{Sea\ Parc}), 1, \{z_2 \succeq_2 \mathit{Cote\ d'Azur}\}),\\ (\mathit{wish}(z_2 \succeq_2 \mathit{California}), 1, \{z_1 \succeq_1 \mathit{Sea\ Parc}\}),\\ (\mathit{wish}(z_3 \leq 60), 1, \{z_2 \succeq_2 \mathit{California}\}))\end{array}$$

Explanations for Pareto-optimal, which consist of sequences of successful wishes, thus offer new ways to the user to explore the space of Pareto-optimal solutions.

4 Conclusion

We have shown that combinatorial optimization can directly use the original user preferences even if those preferences are incomplete. The solving process considers different ways to complete these preferences and optimizes a single criterion at a time, while exploring different importance orderings of the criteria. In doing so, the whole optimization process does not only result into an optimal solution, but also produces an explanation of optimality of the solution. Such an explanation indicates the conflicts between preferences and shows how they have been resolved. The user can examine this explanation and either accept the solution or refine the preference model. The refined preferences will eliminate the given explanation. If the solution has no further justification, the problem solver will thus produce another solution and the procedure is repeated.

As explanations are comprehensive and are formulated in the same "language" as the optimization problem, the user can react to all elements of the explanation and change the problem statement. For example, the user can relax constraints, refine preferences, add importance statements between preferences, or limit the penalization of the less important criteria. Thanks to the explanations, the problem solver behaviour becomes completely transparent to the user

and the user gains full control over the problem solver. We thus obtain an interactive problem solving process that consists of optimization, explaining, and preference elicitation.

As a consequence, the user is freed from tedious tasks such as adjusting the weights of an additive utility function. As the weights do not appear in any explanation of optimality, the meaning and effects of the weights are not very transparent to the user. The choice of the weights may even depend on the form of the criteria space, thus creating a strong dependence between constraints and preference model.

For the sake of brevity, we have neither discussed conditional preferences (cf. [1]), nor algorithms for computing the optimal solutions (cf. [6]). The tutorial [3] shows how other forms of preferences can be characterized in terms of optimization operators and describes different techniques to compute optimal solutions.

References

1. Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. Preference-based constrained optimization with cp-nets. *Computational Intelligence*, 20:137–157, 2004.
2. Jon Doyle. Prospects for preferences. *Computational Intelligence*, 20:11–136, 2004.
3. Jon Doyle and Ulrich Junker. Tutorial on preferences. AAAI-2004 tutorials, AAAI, 2004.
4. Matthias Ehrgott. A characterization of lexicographic max-ordering solutions. In *Methods of Multicriteria Decision Theory: Proceedings of the 6th Workshop of the DGOR Working-Group Multicriteria Optimization and Decision Theory*, pages 193–202, Egelsbach, 1997. Häsel-Hohenhausen.
5. Marco Gavanelli. An algorithm for multi-criteria optimization in CSPs. In *ECAI'02*, pages 136–140, Lyon, France, 2002. IOS Press.
6. Ulrich Junker. Preference-based search and multi-criteria optimization. *Annals of Operations Research*, 130:75–115, 2004.
7. Ulrich Junker. QuickXplain: Preferred explanations and relaxations for over-constrained problems. In *AAAI'04*, pages 167–172, 2004.
8. Ulrich Junker and Daniel Mailharro. Preference programming: Advanced problem solving for configuration. *AI-EDAM*, 17(1), 2003.
9. Werner Kießling. Foundations of preferences in database systems. In *28th International Conference on Very Large Data Bases (VLDB 2002)*, pages 311–322, 2002.
10. Luc N. Van Wassenhove and Ludo F. Gelders. Solving a bicriterion scheduling problem. *European Journal of Operational Research*, 4(1):42–48, 1980.