User-Involved Tradeoff Analysis in Configuration Tasks

Pearl Pu, Boi Faltings, and Pratyush Kumar

Swiss Federal Institute of Technology
[1]Human Computer Interaction Group
[2]Artificial Intelligence Laboratory
CH-1015 Lausanne, Switzerland
{pearl.pu, pratyush.kumar, boi.faltings}@epfl.ch

Abstract. We describe configuration systems using constraint problem solving formalisms where feasible products are computed by constraint problem solvers. A feasible product is a configuration of constituent components that violates none of the configuration constraints and meets users' preferences as much as possible. To help users find desirable products, a system must possess and understand users' preference model as well as value functions. A constraint-based multi-attribute optimization problem (MOP) assigns utility functions to the set of feasible configurable products so that optimal ones stand out. Due to the incompleteness and uncertainties of user's preference models, optimal solutions are difficult to compute in practical settings if systems do not constantly interact with users and refine user's preference models. While building four user-involved MOPs, we have accumulated a set of interaction principles that optimize user and system collaboration while computing optimal solutions. In particular, we will concentrate here on our approaches and solutions to address tradeoff tasks in interactive MOPS.

1 Introduction

To perform complex tasks, such as searching the web for suitable products or services, planning a trip, or scheduling resources, people increasingly rely on computerized configuration systems to find outcomes that best satisfy their needs and preferences. However, preferences and desires get into conflicts and choosing the best product is a decision problem, or more specifically a tradeoff problem. Many automated decision support systems cannot satisfactorily help users make tradeoffs without a fully specified value function. On the other hand, a fully specified value function for all decision outcomes is difficult to establish for the following reasons:

- Users' preference models are incomplete. Therefore, it is hard to state value functions for preferences that do not exist.
- Users' beliefs about desirability are ephemeral and uncertain. As we studied our users in choosing vacation packages, we observed that they often start with a high value function on low-cost deals. However, as they discovered features that they found more important to them than price, they would change the initial value function.

Therefore, non-classical cases of tradeoff analysis are a fundamental problem of growing importance, especially if humans are becoming increasingly involved in the decision process. We investigate how to give maximum information support to users and enable them to make a happy choice. We are therefore concerned with the following questions:

- What are the types of non-classical tradeoffs tasks (task analysis and taxonomy)?
- What information do users need when they perform these tradeoff tasks (need assessments)?
- How to design interfaces that optimally present and explain this information to users?
- When there is too much information to present, how to structure the information?

We have built four user-involved configuration systems. During this process, we have found some answers to the above-mentioned questions. We have identified and accumulated a set of interaction principles that aim at augmenting and optimizing user and system collaboration while computing desirable outcomes and making tradeoffs. We have tested them with formative user studies throughout the implementation cycles.

2 Related Works

Previous work in the area of user preference elicitation and tradeoff analysis in a configuration tasks have generally followed two main approaches, based on the classical and modern decision theory. The classical theory deals with the idea of formulating a perfect model of users' preference utility function. Given a perfect utility function, the classical theory can accurately predict final configuration of a configurable item. Classical decision theory [Keeney] treats tradeoff problems under the assumption that a machine is able to help a human to externalize a value structure and use it to evaluate decision outcomes. A popular method to elicit such value function is to ask users to choose a set of outcomes and infer the model from their choices. This process can be lengthy and cognitively demanding.

Behavioral decision theory (Payne et al 1993, Carenini and Poole 2002), on the other hand, is very concerned with decision makers' behavior. Many years of studies have pointed out the adaptive and constructive nature of human decision making. Although individuals clearly aim at maximizing the accuracy of their decisions; they are often willing to tradeoff accuracy to reduce cognitive effort. The following key properties of a decision maker's behavior are fundamental validations to our hypotheses: 1) stating preferences is a process rather than a one-time enumeration of preferences that do not change over time; 2) user involved preference construction is likely to be more effective than using default or implicit models if a user is to understand and accept the solution outcomes (Carenini and Poole 2002); and finally 3) decision makers, when facing an unfamiliar task, are quite opportunistic in choosing their strategies by reassessing metagoals, and are likely to benefit from a

flexibility to choose between fundamental and means objectives (see also Keeney 1992).

Current research in the field has widely established the fact that consumer decision behavior is largely adaptive in nature. A consumer, in most scenarios, does not have a perfect formulation of his utility function, and it is during the decision process that hidden preferences come up to the fore front. These hidden preferences affect the decision process, as a result the decision maker in a sense adapts himself to the decision making process. The decision task and decision environment have a considerable effect on the adaptive decision making process.

Several decision support systems have taken a similar approach, such as FindMe (Burke et al 1997), ATA (Linden et al 1997), and Apt Decision (Shearin and Lieberman 2001). All of them are concerned with decision support for the search of multi-attribute products. FindMe promotes assisted browsing, relying domain knowledge in the process, and aims at reducing complexities in the multitude of product dimensions and data sources for users. Tradeoff analyses and tweaking (vs. example critiquing) are two main components from their system that are similar to ours. However, we investigate a precise concept for the minimal critiquing context in which tradeoff can effectively happen, while their tradeoff component is mainly used for explanation, and tweaking for adjusting values. Apt Decision uses learning techniques to synthesize a user's preference model by observing their critique of apartment features. Its main objective is profiling and predicting what a user wants in apartment searches. Adaptive decision making in AptDecision uses examples to elicit hidden preferences of users regarding apartments. User profiling is done through the process of example critiquing. Users decide features important to them by browsing through the examples to discover new features of interest and revising their preference model subsequently.

Linden et al (1997) described a preference elicitation method using travel planning as its example domain. Initially only few user preferences need to be expressed. The ATA system (automated travel assistant) uses a constraint solver to obtain several optimal solutions. Five of them are shown to the user, three optimal ones in addition to two extreme solutions (least expensive and shortest flying time). User preferences are modeled as soft constraints in the CSP formalism. A candidate critiquing agent (CCA), similar to our example critiquing, constantly observes user's modification to the expressed preference, and refines the elicited model in order to improve solution accuracy.

We have developed similar techniques for a range of applications rather than the travel domain alone, such as conceptual design, COMIND (Pu and Lalanne 1996, 2000, 2002), resource scheduling, ICARUS (Pu and Melissargos 1997), travel planning, SmartClient Travel (Pu & Faltings 2000, Torrens et al 2002), and vacation package finder, VacationPlanner (Jurca and Pu 2001). ATA displays only five solutions, which is not diverse enough to guarantee the inclusion of the most optimal solutions, especially when the current user deviates from the standard one (see Faltings, Torrens and Pu, the same issue). Furthermore, by not showing near optimal solutions, users are discouraged from opportunistic and creative discovery of outcomes. Using compact visualization methods, we are able to effectively display up to thirty solutions in SmartClient Travel, and more in other systems. Finally, we are more concerned with interaction design principles that can guide users to state hidden

preferences, revise ill-defined ones, and focus on fundamental objectives. This is only possible by investigating user issues in a number of application domains.

Several recent works use constraint satisfaction techniques to automate the process of decision making. They also employ an interactive style to elicit preferences. Freuder et al (2000) introduced Exemplification CSPs by using an interactive version of the MAC algorithm to elicit user's value assignments to network parameters. O'Sullivan et al (2001) described machine learning techniques to help users formulate new constraints. The process involves asking users to provide acceptable solutions (positive examples) from which the system attempts to generalize the constraints exemplified. Freuder and O'Sullivan proposed to model tradeoffs as additional constraints in CSP-based configuration systems. The main goal of their work is to propose appropriate tradeoffs automatically. The conclusion is that arc-consistency based tradeoff proposals are sufficient for finding optimal tradeoffs. The assumption is, however, that users have strong preferences. That is, the issues of discovering hidden preferences and treating uncertainties in users' beliefs of their desires are ignored for the moment. The idea of system assisted tradeoffs can be made even more powerful with concept of hidden preference elicitation. Users often are not aware of all the attributes of the configuration problem that they are solving, and when these attributes are revealed to them in a step by step process, they formulate new preferences and change their existing preferences to reflect the new information. In ways, our work and theirs can be considered as two parts of the same system. The work described here allows users to perform tradeoff while discovering their hidden preferences and value functions. According to business negotiation theories [Pruitt 1981, Unit 1999 and Luo et al 2003], this process must be done extensively before the actual negotiation itself. Likewise in tradeoff, exploration and discovery are important before tradeoff itself. On the other hand, if a user repeatedly performs similar tradeoff tasks, we can use the techniques suggested by Freuder and O'Sullivan to automate this process. When the user finds a generalization not acceptable, then a negative example can be inferred to refine the version space of the constraint being acquired. Thus, we hypothesize that even thought it is easier to assist a user already having strong preferences [Freuder and O'Sullivan], the process of hidden preference elicitation should make the task as simple for naïve users, who have relatively less formulated preferences.

Bowen (2001) provides a formal notion of interactive CSP, motivated by the introduction of additional constraints so that only a single solution is generated. A specialization CSP is a set of these constraints. We can see adding hidden constraints as a form of specialization CSP. We intend to incorporate these approaches and extend our category of principles. However, we must first validate them with user studies as we did with our own approaches.

3 Defining Configuration Tasks Using CSP

We use constraint problem solving (CSP formulation) techniques as an underlying reasoning engine for our decision support systems. This is largely due to the natural modeling of multiple attribute products and solution search within the CSP framework. A constraint satisfaction problem (CSP), which underlies the decision

generation process, is characterized by a set of n variables $X_1,...,X_n$ that can take values in associated discrete domains $D_1,..,D_n$, and a set of m constraints C1,..,Cm, which we assume to be either unary (one variable) or binary (two variables), and which define the tuple that are allowed for the variable or pair of variables. Besides integrity constraints that can never be violated, a CSP may also include soft constraints. These are functions that map any potential value assignment into a numerical value that indicates the preference that this value or value combination carries. Solving a constraint satisfaction problem means finding one, several or all combinations of complete value assignments such that all integrity constraints are satisfied. When soft constraints are present, it also involves finding optimally preferred assignments.

We are concerned with a definition of attributes, criteria, preferences, and constraints from the user/system interaction point of view. A feature or component attribute is the aspect of a product (or solution) for which users would like to express preferences. A component attribute is a physical aspect of a product (e.g., the screen size of a PC), while a feature attribute is an abstract concept for a product such as the assimilability of a product (easy, hard). Optimization criteria are those attributes used for the evaluation of decision alternatives, and can be a function. For example, the total flying time of a trip is the sum of all of the flying segments plus the transfer time at intermediate airports. We can then state a criterion for the trip. For example, it should not take longer than x number of hours.

A preference is a statement about a desired condition on an attribute or a criterion. For example, “I would like to leave Geneva after 10am” is a user preference. Most user preferences are stated, although some can be inferred, such as those used in collaborative filtering. User’s preferences can be also inferred from demographical analysis, default logic, and case-based reasoning (Ha and Haddawy 1998). For example, knowing that a traveler is booking a business trip, we can infer that his most important criterion is to be at destination on time. In this paper, we focus on user stated preferences only.

A constraint is a statement about a condition whose violation will lead to user’s task failure. Being in Hamburg at 2pm for a business meeting is a constraint if that meeting cannot be pushed back. If the user fails to find a flight to satisfy that constraint, he will not be able to achieve his main objective. In most of our systems, user constraints are modeled as hard preference to achieve a uniform treatment of constraints and preferences. Besides user specified constraints, our system also deals with integrity constraints, those conditions that must be satisfied in the configuration process. For example, integrity constraints would state that there must be at least 30 minutes for changing flights within Europe, and 60 minutes for international ones. Integrity constraints are part of domain knowledge, and are elicited from domain experts during the construction of the system.

All of these variables are uniformly known as decision parameters, since they effect how users make decisions. We distinguish incomplete (or hidden) decision parameters from ill-specified ones. Ill-specified parameters give rise to conflicting solutions, assuming that attributes are not independent. An ill-specified parameter can emerge when users are too concerned with means objectives rather than fundamental objectives (Keeney 1992). For example, a user immediately restricting himself to the choice of minivans will not be able to explore station wagon possibilities. If his

fundamental objective is to have enough baggage space for his family, then the minivan preference is an ill-specified one.

4 Taxonomy of Tradeoff Tasks

Before presenting the taxonomy for tradeoff tasks (TOT), we describe some results from interviewing users and studying how they perform tradeoff tasks in conceptual design, select vacation packages and plan a trip. This process, called task analysis, must precede the design and implementation of any interactive system. Task analysis also includes needs assessment, which analyze users' information needs and functional requirements. The result of task analysis is a task model describing the structure and component breakdowns of the tasks being studied.
We participated in a product design course on conceptual design and design tradeoffs. We interviewed and worked with professors who taught conceptual design, as well as practicing designs of mechanical systems. A number of students from that course then became involved in the interface design of a conceptual design system, COMIND. Such participatory design is a way to bring functional and information requirements directly to the interaction design.

For the travel planning domain, we interviewed students, staff, and secretaries at various times in order to establish a task model. In addition, we surveyed 10 commercial on-line flight reservation systems (Jurca 2000). The task model only differs slightly from the one obtained in the conceptual design domain. We will, however, combine them into one reference model.

4.1 Tradeoff Task taxonomy

From the task analysis, we have identified five types of tradeoff tasks:

1. Tradeoff when desires are in conflict (value tradeoff) – users have specified a set of preference values which cannot be satisfied at the same time. Tradeoff, that is the revision of certain preference values, is necessary to obtain feasible configurations.
2. Tradeoff when users are uncertain about value functions (utility tradeoff) – while doing tradeoff and especially when users can visualize the set of decision outcomes, they change the importance attached to a certain criteria. In vacation planning especially, user will often relax the price preference in order to keep vacation options in exotic places. This is a form of the first type of tradeoff except users are manipulating the value function rather than the individual assigned values. That is, if a user is repeatedly relaxing the price preference, we may be able to infer that his preference for destination is stronger than that for price. Modeling value functions is very close to modeling users. We must do this with extreme care in order to keep users in control.
3. Tradeoff when a decision maker is undecided (outcome tradeoff) – users are ambivalent about outcomes of a decision process. In this case, we observed that users are likely to elicit additional preferences so that some outcomes stand out more.

4. Tradeoff at the constraint level (constraint tradeoff) – when one or a set of constraints give rise to zero solutions or very few solutions, diagnosing the problem at the value assignment level is inefficient. Rather, relaxing the constraints or introduce additional values will help. Which constraints to relax, or whether to relax constraints or introduce values, are some of the tradeoff tasks a user faces.
5. Tradeoff between model accuracy and decision effort (accuracy vs. effort tradeoff) – users often prefer accepting a decision outcome and deal with the consequences, rather than improving the preference model with great effort (see Luce 1998). While we do not deal with emotional aspect of decision making and decision justifiability, we must respect users' cognitive ceiling and design our interactive tradeoff systems respectively.

Accuracy and effort tradeoff depicts how much patience a user has to be engaged in doing the task, and serves as cognitive limit to all the other tradeoff tasks. Value tradeoffs are local adjustment of values. When the number of tradeoff value exceeds a threshold (unknown parameter), users can no longer keep track of the value dependencies. They will adopt the strategies of doing tradeoffs at the solution level. Revising value functions while users are searching products requires learning users' preferences from his behavior. This can be seen as a super class of tradeoff activities from the value tradeoff. Constraint tradeoff is a special case and will not be treated here.

5 Preference Revision – Value Tradeoff

Not only preferences change, preferences should change if we believe that we are adaptive decision makers. Value tradeoff requires revising the original stated preferences for one or more tradeoff variables in order to find feasible compromises. Consider someone looking for a vacation package in Mexico in December. He prefers to explore wildlife in Mexico rather than sun bathing on the beaches. His budget is around $1000. He has not considered kayaking so far. Table 1 shows several packages in the catalog, which match the destination Mexico. Taking into account the preferences on budget and vacation features, there is definitely a tradeoff to make: pay almost 10 times more and spend twice as much time to explore wildlife for one vacation package, or pay less and stay at the beach. Alternatively, the user can choose the kayaking package which matches neither the budget nor the feature. But in order to find these deals, the user cannot prematurely state the budget preference.

Table 1. Different vacation packages for Mexico in December

Vacation in Mexico in December, prefer wildlife	3 days	6 days	7 days
	$142 - $332	$2590	$1290
	beach	wildlife	kayaking

5.1 Exploring Search Spaces by Value Tradeoff

Suppose a configuration system elicits users' preferences in a particular order. If these preferences will eventually lead to conflicts, the discovery of this conflict depends on the order. In the vacation example, if the price is set to $2000 in the beginning, the wildlife deal will not be shown. In other words, the user is never informed of the possibility in which by paying a bit more, he is able to get his dream vacation. To overcome this problem, we have used the dynamic query technique (Shneiderman) and implemented an exploratory tool that enables users to discover his needs and preferences by informing him of all relevant decisions.

Fig. 1 is an interface of VacationPlanner (Jurca and Pu 2001). All attributes of a vacation package (destination, date, max price, max duration, youngest accepted age, and etc.) are displayed on the left-side panel. Users express preferences on these attributes by clicking on the desired check boxes or moving sliders into the right positions. The system will display the packages matching selected preferences. When a user states $2000 preference on price, for example, he sees immediately that the wildlife deal is no longer available because that feature is no longer selectable (grayed out). At this point, he can decide whether wildlife is a more fundamental objective, or his budget.

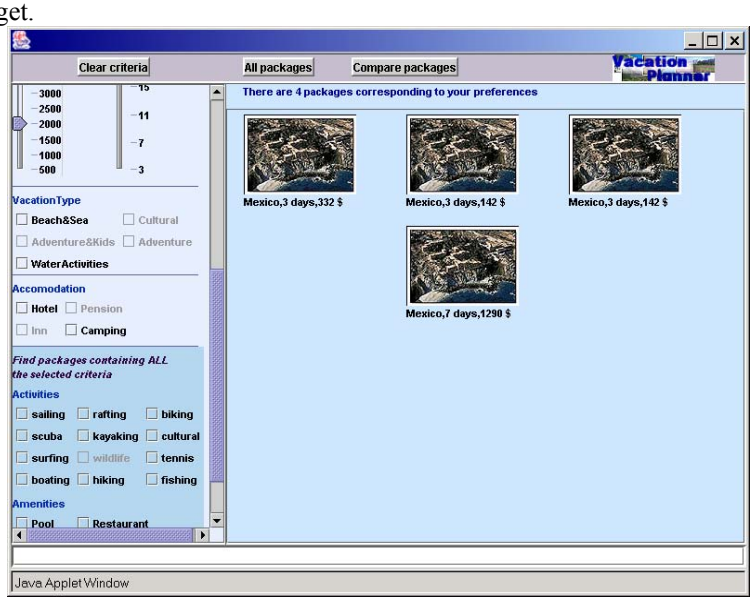

Fig. 1. Putting a low price for a vacation in Mexico will exclude the possibility of exploring wildlife (change the screen shot)

5.2 Partial Satisfaction and Example Critiquing

Displaying the whole set of flight combinations between two is often impossible, and is also not desirable. Making tradeoffs in this case is intertwined with the search

process. That is, rather than making a tradeoff with current solutions, the user would post a preference in order to see near-by solutions for tradeoff purposes. Suppose I only see the beach and kayaking deals and I would like to find a vacation to explore wildlife. I click on beach to choose another alternative: wildlife. Even though the solution with wildlife violates my earlier budget preference, it is still desirable for the system to show that package. Again, by doing so, the system informs the user of an important decision. Highlighting the fact that the price preference has been violated is even more informative for our decision makers as shown in Table 2.

Table 2. Combining search with tradeoffs

destination	duration	price	feature
Mexico	3 days	$332	beach
Mexico	6 days	$2590	Wildlife

We have implemented this tradeoff feature in SmartClient- and Isy-Travel, our configuration system for travel planning. Users can freely pose preferences, some of them being inconsistent. A partial satisfaction CSP algorithm based on soft constraints [Torrens et al] is used to compute near-best solutions where preferences over certain attributes have been violated. An explanation mechanism is used to highlight dependent attributes and their respective assigned values. Users can choose to keep the proposed solution, or retract inconsistent preferences.

In Figure 2, a user has stated a preference on both departure and arrival time for segment 1. He usually prefers to fly out of Geneva after 8:00 am, but for this trip he has to be at Hamburg before 11 am. The flight shown in red (the highlighted lines) violates the first preference, but satisfies the second. The violated attributed has been highlighted. Another solution (not highlighted) shows that leaving after 8:00 am will get him to Hamburg after 12 am. At this point, he must decide whether he will leave earlier, or he pushes back the meeting time.

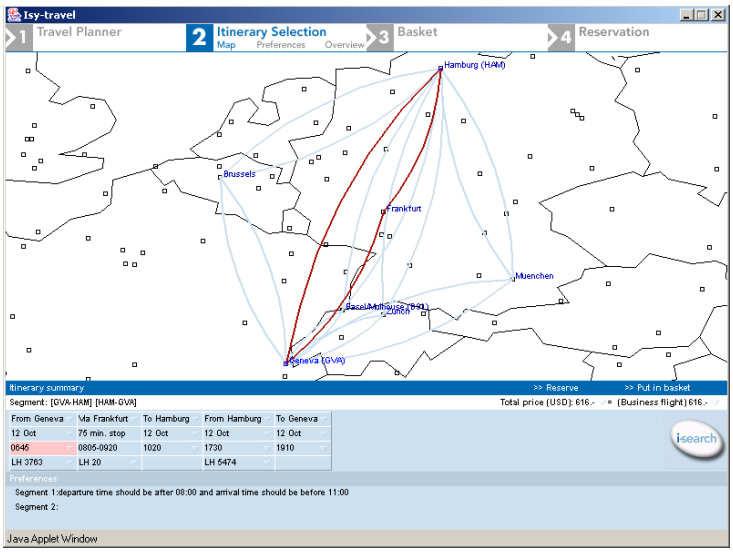

Fig. 2. Conflict values can be revised via tradeoff analysis show decision context

5.3 Quantitative and Non-linear Functions

Some tradeoff functions are highly quantitative. If the aspect ratio is almost constant, then the underlying tradeoff function is predictable, such as the case of using one dollar to buy 3 donuts, and 3 dollars to buy 9 donuts. When one dollar buys 3 donuts, but three dollars buy 12 donuts, it is harder to predict how many dollars it would take to buy 15 donuts, or how many donuts one would get by spending 5 dollars. Such scenarios require means to forward and backward propagate value changes for the dependent variables in tradeoff scenarios, so as to facilitate decision task.

In the case of travel planning, the departure time and arrival time of a segment (a trip consists of multi-segments if it involves one or more intermediate airports) are almost linear. In this case, the tradeoff analysis can be supported via a visual interface called the parallel coordinates, as shown in Figure 3. If a user desires a late departure, a late arrival is expected. If a user wants to arrive before a certain time, the parallel coordinates show the feasible departure times (back propagation). By setting ranges on the range sliders (vertical bars), a user can learn about tradeoff value dependencies and then make informed decisions.

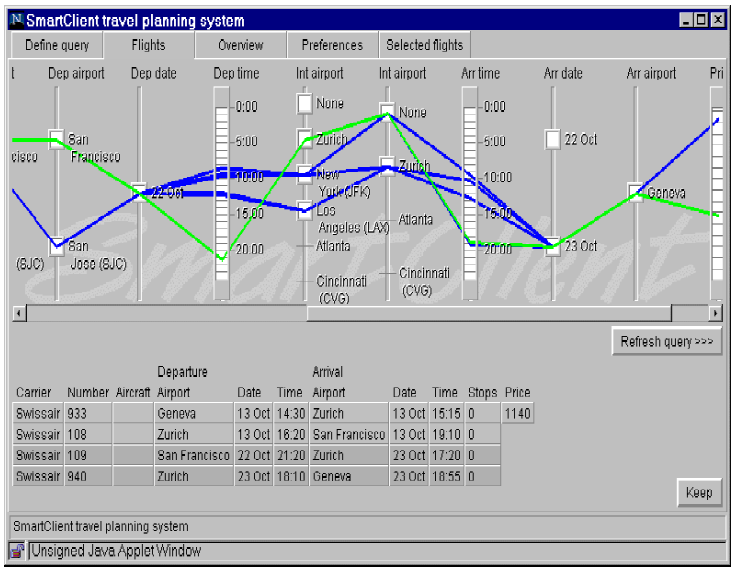

Fig. 3. Value tradeoff in parallel coordinates

For value tradeoff, we propose the following interaction principles:

1. *Use exploratory tools to help users discover their preferences.* Following studies from decision behavior theories, users want to discover their preferences, rather than being modeled as a system. Exploratory tool is the key to building configuration systems that box the products, not the user.
2. *Combine search with tradeoff.* While searching for similar solutions of a configuration, users like to post preferences in the solutions themselves. When preferences are in conflict, partial satisfaction can effectively inform users of meaningful tradeoff problems.
3. *Visual feedback:* Since some preference values do not have an obvious relationship with decision outcomes, an immediate feedback of results is important and allows users to correctly access how to proceed with the incremental process of decision making.

5.4 Tradeoff Context and Minimal Context

In most cases, it is effective to use an entire example solution as a critiquing context, because it accommodates all attributes and it is easy to implement. There are two situations for which showing an entire solution is not ideal: 1) when a solution contains too much detail to show, and 2) attributes influencing decision objectives are not explicit in the example solution. For the first situation, business trips for example may well contain more than the usual outbound and inbound segments. Flying from Geneva to San Francisco with a visit to Denver for a couple of days is a three-

segment trip. In the second situation, choosing a PC feature, such as advanced rather than (rudimentary, basic, high volume), entails to a certain configuration of PC components. When displaying a solution to users, the feature attributes do not become explicit in the example solutions.

We define the dependent attribute cluster to be a set of dependent attributes whose values influence each other in the decision process. Such a cluster is the minimal critiquing context. If a decision problem entails a number of such clusters, we should allow users to critique them one at a time for an effective management of display complexity and cognitive load. As an example, we can have several critiquing contexts in the travel domain:

- cabin class, price
- departure time, arrival time, airline, intermediate airport for the outbound
- departure time, arrival time, airline, intermediate airport for the inbound

This is because the choice of cabin class influences price and vice-versa. Departure time, on the other hand, influences the arrival time, the choice of airlines, and the intermediate airport (if they are available). Given a CSP based decision system, it is possible to detect such clusters either by examining the network structure, inferring these cluster structure from solutions (not trivial), or performing domain knowledge engineering.

In practice however, attributes tend to be related in a complex dependency structure. In the above example, airline can be an attribute in the first group as well if we assume that airlines offer very different fares. If they are strictly modeled as dependent, then the three clusters will join and the minimal critiquing context becomes again the entire solution, thus possibly too complex for the multi-segment trips. A possible solution is to perform conditional preference elicitation i.e. after a user has stated a preference on airline, other attributes become independent and form clusters (see conditional preferences in Boutilier et al 1997). However, this forces users to state preferences in a particular order.

1. *Show minimal context:* If attributes form smaller dependent clusters, it is more optimal to use each cluster as the minimal critiquing context. This reduces the cognitive and display complexities for users.
2. *Show feature attributes in critiquing context:* If preferences were elicited on feature attributes, then the critiquing context should be enlarged to accommodate them as well.

6 Solution Tradeoffs – Eliciting Additional Preferences

Many decision problems involve selecting a single outcome (a vacation package, a trip, a spouse), although other decision problems can afford several choices (a range of values, a set of values). When several criteria compete to argue for the desirability of a single solution, a decision maker is unable to choose. This ambivalence causes many decision makers to seek additional preferences to justify their final choices,

although another equally known strategy is to discover additional constraints to eliminate choices.

Consider the case of selecting apartments whose current features include their size and their price. Figure 4 shows the available apartments and how they rank in the tradeoff space. Suppose users are unable to decide among the apartments lying on the Pareto curve (those nodes labeled black). Upon examining the apartments, a user discovered that one apartment is especially close to public transportation, which saves 20-30 minutes of commute for him every day. Even though it is rather small, he takes this apartment because it is cheaper than the biggest apartment and saves time. This example shows that discovering additional preferences will help a user choose good candidates in tradeoffs.

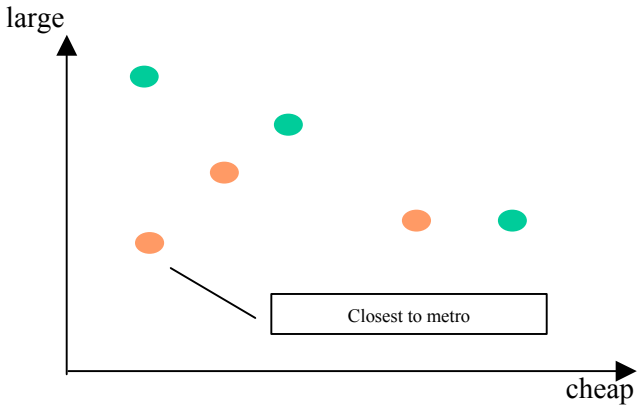

Fig. 4. Apartments in the tradeoff space on price and size, where three of them are Pareto optimal solutions.

In traditional decision theories, the notion of Pareto optimality is very important. A Pareto optimal solution is superior to all other solutions in at least in one criteria dimension. The solution set is divided into Pareto optimal solutions and dominated ones. Most researchers advocate displaying only Pareto optimal solutions. However, the above example shows that via exploration and discovery, a new criterion can push dominated solutions to the Pareto surface. The key for interface design is therefore to show solution details as well as how solutions present themselves in the tradeoff space. This maximizes the exploration and discovery needed to help the undecided decision makers.

We now show several implementations of tradeoff spaces in 2D, 3D, multidimensional and discrete cases. We will then discuss their effectiveness.

Figure 5 is a 2D tradeoff space for selecting flights where the x-axis is the total flying time and y-axis the overall preference satisfaction score (the higher the score the better a solution). These axes can be changed to other values, such as the total cost, and the departure and arrival times of the flights. Consider a user who wants to leave at 8am from Geneva airport. The only flights available from search leave either at 6:20am or 12am. Even though 6:20am is closer to 8am, getting there would require

driving since trains do not operate yet that early. Since parking a car is really costly for a two-week vacation, this user may choose the 12am flight.

Fig. 5. 2D Tradeoff space for selecting flights; x-axis is the total flying time and y-axis the general preference satisfaction score.

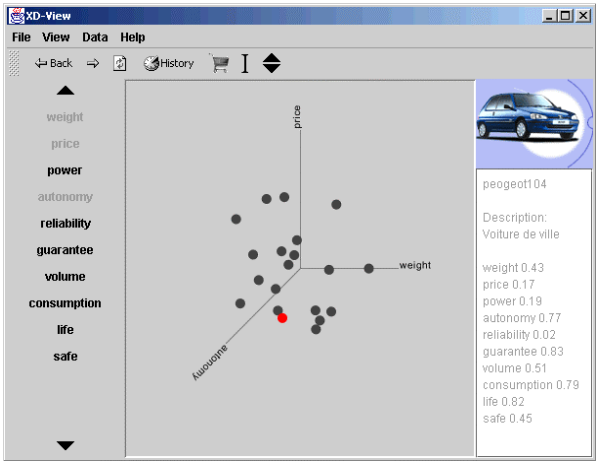

Fig. 6. 3D tradeoff space for a car configurator

Figure 6 shows a three dimensional tradeoff space. All of the criteria for tradeoff are displayed on the left panel. A user can select any three and view the results in the display. This 3D visualization supports rotation, navigation, and zooming. Although it is fun to use the 3D viewer, the advanced features such as navigation may frustrate a novice user. It is also more difficult to compare solutions when they are too close to together.

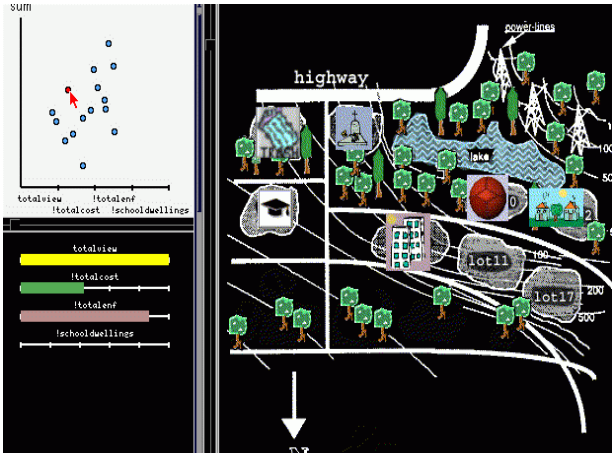

Fig. 7. Multidimensional tradeoff spaces

When the number of simultaneous tradeoff variables exceeds two, we implemented an interface as shown in Figure 7. The user task is to select a solution so that pieces of land are assigned for building different structures. During evaluation, four criteria are being considered: the total view score, the total cost, the noise levels, and the score for measuring how close residential buildings are to schools. We reformulated the optimization problem so that we are always maximizing. That explains the ! sign before some variables. It denotes the negation of that variable. So maximizing !totalcost is minimizing the total cost.

The y-axis represents the total scalar value a solution obtains. On the x-axis, we lay down all the individual criteria. These bottom bars act as "weights" and they attract solutions that score very high in the features that they represent. It is obvious that rearranging the bottom bars will give different displays of the solutions and the ambiguities that exist. However, a user can always consult the slider bars below the tradeoff space for detail. Conversely, changing the values of these bars, a user will see matching solutions in the tradeoff space. This interface overcomes the limitation of not displaying all tradeoff variables and their values in a single overview. It gives a notion of how solutions perform on individual accounts, as well as their details if ambiguities exist. It is fun to explore. Even though we still have to carry out more refined user studies to demonstrate its discovery capabilities, our informal user studies show that users found this interface very effective and have allowed them to discover many solutions that were evaluated as inferior by the initial standards that they had set. In the figure, the selected node is actually chosen by our test user as the best solution, even though it does not perform well in the absolute score (y-axis), on noise, or school distance features.

The interaction principles in this section on solution tradeoffs can be summarized as follows:

- *Show examples to provoke interaction:* Most users are not aware of or cannot articulate preferences. Solving ambivalent tradeoff problems requires discovering

and weighing these newly discovered features of certain solutions. An exploratory tool must provide examples as well as the tradeoff spaces.

- *Do not systematically eliminate dominated solutions:* allow ambivalent users to explore solutions nearby the "optimal" ones. Very often, surprises can be found to explain a tradeoff decision.

8 User studies: Combining Decision Behavior Research with User Requirement Analysis and Formative Usability Studies

Some existing findings from consumer behavior research and behavioral decision theory allowed us to avoid costly experiments to understand users' behaviors when they face decision tasks. However, we still analyzed usability issues by surveying existing commercial systems, such as the survey on travel planning (Jurca 2000). We also used online decision systems such as Personalogic in our user requirement studies. The characteristics, both in terms of shortcomings and strengths, were integrated into the design of Isy-travel and VacationPlanner. Earlier decision support systems were built from analysis done on commercial resource allocation systems employed in the airline industry. COMIND was based on extensive interviews with conceptual designers. The design principles reported here are therefore a result of consumer behavior literature, user requirement analyses, and most importantly testing of these principles in systems that we have built.

Formative user studies were also performed for all of the systems described here. It's a process involving writing scripts for each usage scenario, identifying potential users, conducting usability testing, analyzing results, and proposing design changes. The systems all went through several phases of change as a result of formative user studies.

For the future, we intend to conduct several comparative user studies on rather focused claims. For example, we believe that users select different outcomes under two different conditions: under the condition that they will view only solutions and under the condition that they tradeoff on values without ever seeing an example until the end. We can then further establish interaction principles that address these two conditions.

9 Conclusion

We have pointed out the importance of tradeoff tasks in CSP-based configuration systems. Via tasks analysis, we established taxonomy of various tradeoff tasks. We then focused our discussion on designing, implementing and testing a set of interaction principles that maximizes user and machine's collaboration in finding "optimal" solutions. A recurring theme in these principles is to provide tools to help users model themselves, rather than devising algorithms to model them. Through exploration, we turn the preference model incompleteness and user belief uncertainty to advantages, which is decision discovery. Through exploration, users discover knowledge of the configuration space and make better and more informed decisions.

References

1. J. F. Allen, L. K. Schubert, G. M. Ferguson, P. A. Heeman, C. H. Hwang, T. Kato, M. N. Light, N. G. Martin, B. W. Miller, M. Poesio, and D. R. Traum. *The TRAINS project: A case study in building a conversational planning agent*. TRAINS Technical Note 94-3, University of Rochester, Department of Computer Science, September 1994.
2. C. Boutilier, R. Brafman, C. Geib, and D. Poole. *A Constraint-Based Approach to Preference Elicitation and Decision Making*. In AAAI Spring Symposium on Qualitative Decision Theory, Stanford, 1997.
3. Bowen, J. *The (Minimal) Specialization CSP: A basis for Generalized Interactive Constraint Processing*, in Proceedings of Workshop on User-Interaction in Constraint Processing at CP-2001. 2001.
4. R. Burke, K. Hammond, and B. Young. *The findme approach to assisted browsing*. In IEEE Expert, volume 12(4), pages 32--40, 1997.
5. Carenini G. and Poole D. *Constructed Preferences and Value-focused Thinking: Implications for AI research on Preference Elicitation*. AAAI-02 Workshop on Preferences in AI and CP: symbolic approaches - Edmonton, Canada, 2002.
6. J. Doyle and R. Thomason. *Background to qualitative decision theory*. AI magazine, 20(2), summer 1999.
7. Freuder E, Likitvivatanovong C and Wallace R. *A case Study in Explanation and Implication*, in Proceedings of CP-2000 Workshop on Analysis and Visualization of Constraint Programs and Solvers, 2000.
8. E.C. Freuder and B. O'Sullivan. Generating Tradeoffs for Interactive Constraint-Based Configuration.
9. D. Pruitt. *Negotiation Behavior*. Academic Press, 1981.
10. I. Unt. Negotiation Without a loser. Copenhagen Business School, 1999.
11. Jurca, A. and Pu, P. *Algorithms for Online Vacation Planning*. Technical Report, Swiss Federal Institute of Technology, Lausanne. p. 1-12, 2001.
12. Jurca, A. *Survey of Online Travel Planning Systems.* Technical Report, Swiss Federal Institute of Technology, Lausanne, 2000.
13. Keeney, R. and Raiffa, H., *Decision with multiple objectives: Preferences and value tradeoffs*. 1976: Cambridge University Press.
14. Keeney, R., *Value-Focused Thinking: A Path to Creative Decision Making*. 1992: Harvard University Press.
15. V. Ha and P. Haddawy. *Problem-focused incremental elicitation of multiattribute utility models*, in Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, pages 215--222, August 1997.
16. Ha, V. and Haddawy, P. *Toward Case-Based Preference Elicitation: Similarity Measures on Preference Structures*, in Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, 1998.
17. Horvitz, E., *Principles of mixed-initiative user interfaces*, in Proceedings of The ACM SIGCHI Conference on Human Factors in Computing Systems, ACM Press, p. 159-166, 1999.
18. Inselberg, A. and Dimsdale, B. *Parallel Coordinates: a Tool for Visualizing Multidimensional Geometry*, in Proceedings of IEEE Visualization '90, IEEE Computer Society, p. 361-378, 1990.
19. Lalanne, D. *Computer Aided Creativity and Multi-criteria optimization in Design*. Ph.D. thesis, the Swiss Institute of Technology (EPFL), Lausanne, 1998.

20. G. Linden, S. Hanks, and N. Lesh. *Interactive assessment of user preference models: The automated travel assistant*. In Proceedings of User Modeling '97, 1997.
21. Mackay, W.E., Holm, E., and Horn, S.B., *Who's in Control? Exploring human-agent interaction in the McPie Interactive Theater project*, in Proceedings of The ACM SIGCHI Conference on Human Factors in Computing Systems. 2001.
22. Payne, J.W., Bettman, J.R., and Johnson, E.J., *The Adaptive Decision Maker*. Cambridge University Press, 1993.
23. Pu, P. and Faltings, B. *Personalized Navigation of Heterogeneous Product Spaces using SmartClient*, in Proceedings of the International Conference on Intelligent User Interfaces, ACM Press, 2002.
24. Pu, P. and Faltings, B. *Enriching Buyers' experiences: the SmartClient Approach*, in Proceedings of The ACM SIGCHI Conference on Human Factors in Computing Systems, 2000.
25. Pu, P. and Lalanne, D. *Human and Machine Collaboration in Creative Design*, in Proceedings of the European Conference of Artificial Intelligence, 1996.
26. Pu, P. and Lalanne, D. *Interactive Problem Solving via Algorithm Visualization*, in Proceedings of the IEEE Information Visualization Symposium, IEEE Press, 2000.
27. Pu, P. and Lalanne, D. *Design Visual Thinking Tools for Mixed Initiative Systems*, in Proceedings of the International Conference on Intelligent User Interfaces, ACM Press, 2002.
28. Pu, P. and Melissargos, G. *Visualizing Resource Allocation Tasks*. IEEE Computer Graphics and Applications, 1997. **17**(4).
29. O'Sullivan, B., Freuder, E., and O'Connell, S. *Interactive Constraint Acquisition*, in Proceedings of Workshop on User-Interaction in Constraint Processing at the CP-2001, 2001.
30. Shearin, S. and Lieberman, H. *Intelligent Profiling by Example*. in Proceedings of the Conference on Intelligent User Interfaces, ACM Press, 2001.
31. Johnson, B. and Shneiderman, B. *Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures*, in *Visualization '91*, p. 284-291, 1991.
32. Torrens, M., Faltings, B., and Pu, P., SmartClients: *Constraint Satisfaction as a Paradigm for Scaleable Intelligent Information Systems.* International Journal of Constraints, 2002. **7**: p. 49-69.