EPFL Technical Report No: IC/2004/54, June 17, 2004

Survey of Solving Multi-Attribute Decision Problems

Jiyong Zhang, Pearl Pu

Human Computer Interaction Group,
School of Computer and Communication Sciences,
Swiss Federal Institute of Technology, Lausanne (EPFL),
CH-1015, Lausanne, Switzerland
{jiyong.zhang, pearl.pu}@epfl.ch

Abstract

Finding the optimal solution of a Multi-Attribute Decision Problem (MADP) is a key problem for electronic commerce systems. In this paper, we formally define the multi-attribute decision problem, and we report our survey of four different methods (soft-CSP framework, multi-attribute decision theory, CP-network, and Heuristic strategies) which potentially could be used to solve the MADP, and their advantages and disadvantages will be discussed respectively.

Keywords: multi-attribute decision problem (MADP), constraint satisfaction problem (CSP), soft-CSP, multi-attribute utility theory (MAUT), CP-network, heuristic decision strategy

1. Introduction

Helping the buyer to find the optimal solution from the electronic product catalogs efficiently is a crucial task for the design of electronic commerce systems. Typically an electronic catalog is defined as an information system that provides access to a collection of product descriptions that an organization wants to offer [39]. The items of the catalog are identified by a set of attributes, and the whole set of items define the product space. Unlike traditional commerce, where the activities are carried out directly between human individuals or organizations, in e-commerce environments, the buyer (or the decision maker) interacts with the pre-designed computer systems so to get the information of products or services he wants. Usually the product or service information that the e-commerce system could provide is far more beyond the individual decision maker's requirement. Studies from economics have shown that the individual only has bounded rationality when making decisions due to his limited knowledge and computational capacity [37]. Therefore, the information provided by the e-commercial system should be highly "selective" to meet the preference of the individual decision maker.

In this paper we focus on the following problem: How to help the decision maker to determine the optimal solution(s) from a large set of available outcomes from an electronic catalog according to the decision maker's preferences? Typically an electronic catalog involves several attributes, and due to the existence of multi-attribute, the decision maker has to make tradeoffs between different attributes during the selection process when his preferences are not fully satisfied. We call this kind of problem a Multi-Attribute Decision Problem (MADP), which is formally defined as follows:

- a set of attributes $\mathbf{X} = \{x_1, ..., x_n\}$;
- a set of domain values $\mathbf{D} = \{D_1, ..., D_n\}$: where each $D_i (1 \le i \le n)$ is a set of possible values for attribute x_i. without ambiguity here we also use $\mathbf{D}$ to denote the space of all possible outcomes which is the Carthesian product of $\mathbf{D} = D_1 \times D_2 \times \cdots \times D_n$;
- a set of constraints $\mathbf{C} = \{C_1, ..., C_m\}$: where each $C_j (1 \le j \le m)$ is a constraint function on a subset of attributes $\mathbf{X}$ to restrict the values they can take;
- a set of available outcomes $\mathbf{O} = \{o_1, ..., o_l\}$: where each $o_j (1 \le j \le l)$ is an element of the possible outcome space $\mathbf{D}$, and $\mathbf{O}$ is a subset of $\mathbf{D}$. Usually the size of $\mathbf{O}$ is relatively small compared to the size of $\mathbf{D}$, but it is still too big for the decision maker to choose one by one. The solution(s) that the decision maker finally chooses must lie in this set.

- a set of the decision maker's preference statements $\mathbf{P} = \{P_1, P_2, \cdots P_t\}$: this part of information needs to be elicited from the decision maker before or during the interaction. Different decision makers may have different preference statements, and some preferences can be violated for the tradeoff purposes during the procedure of searching for the optimal solution.

To illustrate the MADP, here we describe a concrete example in the *apartment renting* domain. Suppose we are designing an e-commercial system providing the apartment renting service, and to simplify the discussion, we assume that the apartments we provided only have 5 distinct attributes: $\mathbf{X} = \{$*Type, Kitchen, Bathroom, Area, and Price*$\}$, and each attribute may take a certain set of values as listed below:

$D_{type} = \{house,\ apartment,\ studio\}$

$D_{Kitchen} = \{private,\ share,\ none\}$

$D_{Bathroom} = \{private,\ share,\ none\}$

$D_{Area} = [20, 200]\ m^2$

$D_{Price} = [300, 4000]\ CHF$

A set of general constraints can be obtained from the nature of the apartment renting domain, for example:

$C_{Type,Kitchen}$: If Type=*apartment*, then kitchen =*private* or *share*

$C_{Type,Area}$: If Type=*house*, then Area >100 m^2

$C_{Area,Kitchen,Bathroom}$: If Area > 120 m^2, then Kitchen = *private* and Bathroom = *private*

In this example, the set of available outcomes $\mathbf{O}$ is the list of the apartments provided by the system. It is natural to notice that $\mathbf{O}$ is only a subset of the total possible outcome space $\mathbf{D}$: for instance, the apartment with the biggest area size and lowest price is a possible outcome in $\mathbf{D}$, but most likely it cannot be offered by the e-commerce system.

As for the decision maker's preference statements, some of them can be generated by the commonsense held by most individuals, for example: "If everything being equal except price, the cheaper, the better", "if everything else be equal, I prefer the apartment with bigger area", etc. Some other preferences can be elicited and stated easily, for example: "I prefer the apartment with bathroom and the size of $100m^2$ ". There remains some compound preferences which are very useful but quite difficult to be elicited, such as "If the size is smaller than 50 m^2, I prefer having kitchen instead of bathroom when the price is the same". The decision maker's preference statements can be violated for the tradeoff purpose. Preference elicitation is a very crucial and difficult task for the decision support systems. More discussion on this topic can be found in [9, 30]. In this paper we assume that the decision maker's preferences have already been elicited.

Two types of problems are raised when we try to solve a MADP: one is to find the "optimal" solution among the outcome set which best matches the decision maker's preferences. We call this problem the "*optimal*" problem. The other one is to find a set of solutions with ranking order, which can be called the "*ranking*" problem.

In this paper, we survey various approaches which can be used to solve the multi-attribute decision problems. Generally speaking, these methods can be divided into four categories: The constraint satisfaction problem (CSP) framework, the multi-attribute utility theory, the reasoning method based on CP-network, and the method based on heuristic decision strategies. This paper is organized as follows: Section 2 introduces the classical-CSP framework and its extension forms by using soft constraints: soft-CSPs. In section 3, we study the method of solving multi-attribute decision problem based on multi-attribute utility theory. The reasoning method CP-network which based on conditional ceteris paribus preference statements is explored in section 4. Section 5 investigates the heuristic decision strategies which are adopted by human beings when solving multi-attribute decision problems. Finally a summary of discussion will be given.

2. Framework of Constraint Satisfaction Problems (CSPs)

Constraint satisfaction problems (CSPs)[23, 40] have been widely used in AI research area for many years to solve different real-life problems ranging from map coloring, vision, robotics, VLSI design, etc. It provides a natural way of representing problems for the user needs only to state the constraints of the problem to be modeled. Once the constraints are specified, some effective searching algorithms can be adopted to find the optimal solution. In this section we first introduce the general concepts of classical-CSP and the related searching algorithms, and then study its extension, soft-CSP, and discuss how to use the soft-CSP to solve the multi-attribute decision problem.

2.1 Definition

Typically a constraint satisfaction problem (CSP) is defined by a triple $< \mathbf{X}, \mathbf{D}, \mathbf{C} >$:

- a set of variables $\mathbf{X} = \{x_1, ..., x_n\}$[1];
- a set of domain values $\mathbf{D} = \{D_1, ..., D_n\}$: where each $D_i (1 \le i \le n)$ is a set of possible values for the variable x_i;
- a set of constraints $\mathbf{C} = \{C_1, ..., C_m\}$ where each $C_j (1 \le j \le m)$ is a constraint function on a subset of variables $\mathbf{X}$ to restrict the values they can take.

If a CSP only has constraints on either unary or binary variables, it is called *binary CSP*. It is possible to convert the CSP with *n*-ary constraints to another equivalent binary CSP [33]. So without losing generality, we usually concentrate on binary CSP for simplification.

A solution of a CSP is a set of value assignment $\mathbf{v} = \{x_1 = v_1, ..., x_n = v_n\}$ (in short as $\mathbf{v} = \{v_1, ..., v_n\}$) satisfying 1) $v_i \in D_i (1 \le i \le n)$; 2) all constraints in $\mathbf{C}$ are satisfied. If a CSP has a solution, we say that it is satisfied.

As we can see from the above definition of CSP, the constraints are crisp since they are either allowed or violated. Usually this kind of constraints are called hard constraints (or crisp constraints), and the CSP with hard constraints only is called classical CSP in order to distinguish from its extension introduced in section 2.3.

2.2 Solving Classical CSPs

When a classical CSP has been defined, a number of different approaches have been developed for solving this problem [19]. In this subsection we survey some algorithms which can be used to solve classical CSPs as defined above.

One simple approach of solving CSPs is the *generate-and-test* paradigm. In this paradigm, each possible combination of the variables is systematically generated and then tested to see if it satisfies all the constraints. The first combination that satisfies all the constraints is the solution. The number of combinations considered by this method is the size of the Cartesian product of all the variable domains. It is easy to see that the complexity of the generate-and-test algorithm is exponential. When the CSP is complex, this algorithm is impracticable due to the overwhelming computational complexity.

A more efficient method for solving CSPs is the *backtracking algorithm*. In this method, variables are instantiated sequentially. As soon as all the variables relevant to a constraint are instantiated, the validity of the constraint is checked. If a partial instantiation violates any of the constraints, backtracking is performed to the most recently instantiated variable that still has alternatives available. The backtracking method essentially performs a depth-first search of the space of potential CSP solutions.

Backtracking is strictly better than the generate-and-test method since whenever a partial instantiation violates a constraint, backtracking is able to eliminate a subspace from the Cartesian product of all variable domains. However, the run-time complexity of backtracking algorithm for most nontrivial problems is still exponential. One of the reasons for this poor performance is that the backtracking paradigm suffers from *thrashing*(search in different parts of the space keeps failing for the same reason) [13]. Thrashing can be

[1] When we use CSP to represent the multi-attribute decision problem (MADP), the attribute of MADP is conceptually equivalent to the variable of CSP, so here we abuse the same symbol **X** to denote both of them.

avoided by *intelligent backtracking*; that is, by a scheme in which backtracking is done directly to the variable that caused the failure. Another drawback of the standard backtracking algorithm is that it has to perform redundant work. This drawback can be eliminated by the method of *dependency-directed backtracking* [38]. Other search algorithms for classical CSPs include: *Forward Checking, Partial Lookahead, Full Lookahead,* and *Really Full Lookahead.* They are introduced and their performances are compared in [26].

2.3 Soft CSPs

The solution of a classical CSP needs to satisfy all the hard constraints. If we compare the definition of classical CSP and MADP, we can see that the main difference between them is that the MADP has a set of preferences some of which can be violated when finding the optimal solution. If we convert the preferences in MADP into hard constraints directly and solve it in the framework of classical CSP, most likely we couldn't get a feasible solution because of over-constraint. Luckily people have extended the classical CSPs to soft CSPs, in which not all the given constraints need to be satisfied when finding the optimal solution. In this subsection, we recall several different kinds of soft CSPs and study their relationships.

Fuzzy CSPs. Fuzzy CSPs (FCSPs) [10, 35]extend the crisp constraints of classical CSPs by the *fuzzy constraints.* A Fuzzy constraint is a mapping from the direct product of the finite domain of the variables referred by the constraint to the [0,1] interval. For a fuzzy constraint c, suppose a fuzzy constraint c associated with t variables $\hat{x}_1,...,\hat{x}_t$, the value $c(\hat{v}_1,...,\hat{v}_t)$ denotes the degree of satisfaction (or preference level) of the constraint c by the value set $(\hat{v}_1,...,\hat{v}_t)$, If $c(\hat{v}_1,...,\hat{v}_t)=1$ we say that $(\hat{v}_1,...,\hat{v}_t)$ fully satisfies c, while if $c(\hat{v}_1,...,\hat{v}_t)=0$, we say that $(\hat{v}_1,...,\hat{v}_t)$ fully violates c. A CSP with fuzzy constraints is called fuzzy CSP. The solution of a fuzzy CSP is then defined as the set of n-tuples of values which have the maximal value. And the value associated to each n-tuple is obtained by minimizing the values of all its sub tuples.
Fuzzy CSPs is a very significant extension of classical CSPs. In fact, it can be used to model partial constraint satisfaction[12]. We have already known that finding a solution to a classical CSP is a NP-complete task, hence, finding the best solution of FCSP is at least NP-hard. The FCSP can be solved in a similar way as classical-CSP: we can define a threshold of satisfaction and turn all fuzzy constraints into hard constraints, then find the optimal solution for the newly transformed CSP. The threshold can be adjusted iteratively to ensure that transformed CSP is solvable. The efficiency of this method can be further improved by maintaining a set of possible thresholds dynamically during search and pick the one that just allows a solution.

Probabilistic CSPs. Probabilistic CSPs (PCSPs)[11] have been introduced to model those situations where each constraint c has a certain independent probability $p(c)$ to be part of the given real problem. This allows one to reason about problems which are only partially known. Let $\mathbf{v}$ be a n-tuple value set on the domain **D**, and a t-tuple $(\hat{v}_1,...,\hat{v}_t)$ be the subset of $\mathbf{v}$ and the value of the t variables $\hat{x}_1,...,\hat{x}_t$ that constraint c associated with, if $(\hat{v}_1,...,\hat{v}_t)$ is allowed by c, the probability of $(\hat{v}_1,...,\hat{v}_t)$ being part of the solution of the real problem is 1; if $(\hat{v}_1,...,\hat{v}_t)$ violates c, the probability of $(\hat{v}_1,...,\hat{v}_t)$ being part of the solution of the real problem is $1-p(c)$. Considering all the constraints that the n-tuple $\mathbf{v}$ violates, we can see that the probability of n-tuple $\mathbf{v}$ being a solution to the real problem is $\prod_{\text{all } c \text{ that } v \text{ violates}} (1-p(c))$.
The aim of solving PCSPs is to get the n-tuple with the maximal probability.
The main difference between PCSPs and FCSPs lies in the fact that PCSPs contain crisp constraints with probability levels, while FCSPs contain non-crisp constraints. Another difference between them is the definition of the associated value to the constraints. In PCSPs, each constraint has a fixed probability value, independent with the value set it associated with. While in FCSPs, each constraint may have different degree of satisfaction on different value set. From this point of view, the FCSPs have more parameters than PCSPs. Moreover, as we already have seen, the criteria for choosing the optimal solutions are different. In

fact, it is possible to model PCSPs by using a transformation which is similar to that proposed in [8] to model prioritized constraints via fuzzy constraints in the FCSP framework.

Weighted CSPs. Weighted CSPs (WCSPs) [20, 21] allow one to model optimization problems where the goal is to minimize the total cost (time, space, number of resources, etc) of the proposed solution. In WCSPs, there is a cost function for each constraint, and the total cost of a n-tuple value is defined by summing up the costs of each constraint with the corresponding subtuple values. Thus the aim is to find the n-tuples with minimal total cost as the optimal solution.
Usually WCSPs can be solved by *Branch and Bound algorithm* [22, 31]. Branch and Bound is a well-known algorithm for solving NP-hard combinatorial optimization problems. The algorithm is started by considering the root problem (the original problem with the complete feasible region), the lower-bounding and upper-bounding procedures are applied to the root problem. If the upper bounds match, then an optimal solution has been found and the procedure terminates. Otherwise, the root problem is divided into two or more subproblems and the algorithm is applied to each subproblem recursively. At each step, if the lower bound for a node exceeds the best known feasible solution, no globally optimal solution can exist in the subspace of the feasible region represented by the node. Therefore, the node can be removed from consideration. The search proceeds until all nodes have been solved or pruned, or a certain pre-defined stop threshold is met.
The main difference between the WCSPs and FCSPs is that WCSPs contain the cost functions ranging with $[0,+\infty)$, while FCSPs contain Fuzzy functions ranging with [0, 1]. And the meanings of the two soft-constraints are also different.

2.4 The Semiring-based CSP Framework

Bistareli etc [3] introduced a semiring-based CSP framework which can describe both classical and soft CSPs. In this framework, A semiring is a tuple $(A,+,\times,0,1)$ such that: A is a set and $0,1\in A$; $+$ is a closed, commutative, and associative operation on A and 0 is its unit element; $\times$ is a closed, associative, multiplicative operation on A and 1 is its unit element and 0 is its absorbing element. Moreover, $\times$ distributes over $+$. A c-semiring is a semiring such that + is idempotent, $\times$ is commutative, and 1 is the absorbing element of +.
Both the classical CSPs and the different type of soft CSPs can be seen as instances of the semiring CSP framework. More precisely, each of such CSPs corresponds to the choice of a specific constraint system (and thus a semiring). The classical CSPs are Semiring-CSPs over the semiring $S_{CSP}=(\{false,true\},\vee(or),\wedge(and),false,true)$, which means that there are just two preferences (*false* or *true*), that the preference of a solution is the logic *and* of the preferences of their subtuples in the constraints, and that *true* is better than *false*. Fuzzy CSPs are the Semring-CSPs over the semiring $S_{FCSP}=([0,1],max,min,0,1)$, which means that the preferences are over [0,1], and that we want to maximize the minimum preference over all the constraints. Similarly, the semiring corresponding to a PCSP is $S_{PCSP}=([0,1],max,\times,0,1)$, and the WCSPs can be represented by the semiring $S_{WCSP}=(R^{+},\min,+,+\infty,0)$.

2.5 Discussion

From the definition, a MADP can be looked as a CSP with a set of preferences which can be violated. The soft CSPs are quite suitable for modeling the MADPs since the preference statements in a MADP can be transformed to some soft-constraints of a soft CSP. For a given MADP, we first need to determine which kind of soft CSP is the ideal form for modeling the problem, this selection largely depends on the feature of the preferences set: for example, if we can easily get the probability of each preference statement, then we may choose probabilistic CSP as the framework; if the cost of violating each preference statement is easier to obtain, then we may use weighted CSP instead. Once the specific soft CSP framework is determined, we need to transform the preference statements into soft-constraints as required. The transform of preference statements to soft-constraints would be straightforward. Finally the optimal solution can be generated by some search algorithms.

In this paper we have assumed that the decision maker's preferences can be fully elicited before the procedure of finding optimal solution. However, research shows that actually the preferences are constructed incrementally during the interaction procedure[30]. So most likely we select the type of soft CSP first, and then elicit the preference gradually to meet the requirement of the special kind of soft CSP, till the optimal solution is found. The integration of eliciting preference and solving soft CSPs during the interaction procedure would be an interesting research topic.

3. Multi-Attribute Utility Theory

3.1 Introduction of Utility Theory

The origination of Utility Theory can be dated back to 1738 when Bernoulli proposed his explanation to the St. Petersburg paradox by the term of utility of monetary value [2]. Two centuries later it was von Neumann and Morgenstern (1944) who revived this method to solve problems they encountered in economics [42]. Later in the early 1950s, in the hands of Marschak[24] and of Herstein and Milnor[16], the Expected Utility Theory was established on the ground of a set of axioms and the von Neumann Morgenstern theorem (VNM Theorem). In this section we review briefly the general concepts in utility theory. More details can be found in [25, 36].

Let $\mathbf{O} = \{o_1, ..., o_l\}$ denote a set of outcomes of the multi-attribute decision problem, $\mathfrak{L}$ be the set of all risky prospects (also called lotteries) on the set of $\mathbf{O}$ (i.e. $\sum p_i o_i \in \mathfrak{L}$, where $p_i \in [0,1]$, and $\sum p_i = 1$), let $\succsim$ be a binary relation on $\mathfrak{L}$. We define the following 4 axioms first:

(A1) $\succsim$ is complete, i.e. either $\mathbf{x} \succsim \mathbf{y}$ or $\mathbf{y} \succsim \mathbf{x}$, for all $\mathbf{x}, \mathbf{y} \in \mathfrak{L}$;

(A2) $\succsim$ is transitive, i.e. either $\mathbf{x} \succsim \mathbf{y}$ and $\mathbf{y} \succsim \mathbf{z}$, then $\mathbf{x} \succsim \mathbf{z}$ for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathfrak{L}$;

(A3) Continuity Axiom:

if $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathfrak{L}$ such that $\mathbf{x} \succ \mathbf{y} \succ \mathbf{z}$, then there is an $\alpha, \beta \in (0,1)$ such that $\alpha\mathbf{x} + (1-\alpha)\mathbf{z} \succ \mathbf{y}$, and $\mathbf{y} \succ \beta\mathbf{x} + (1-\beta)\mathbf{z}$;

(A4) Independence Axiom:

for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathfrak{L}$ and any $\alpha \in [0,1]$, $\mathbf{x} \succsim \mathbf{y}$ if and only if $\alpha\mathbf{x} + (1-\alpha)\mathbf{z} \succsim \alpha\mathbf{y} + (1-\alpha)\mathbf{z}$.

Then the VNM Theorem proved the existence of utility function theoretically provided that the relation $\succsim$ satisfies the axioms (A1)-(A4):

VNM Theorem: Let $\mathfrak{L}$ be a convex subset of a linear space, and let $\succsim$ be a binary relation on $\mathfrak{L}$, then $\succsim$ satisfies (A1), (A2), (A3) and (A4) If and only if there is a real-valued function $u : \mathfrak{L} \to \Re$ such that:

(a) u represents $\succsim$, i.e. $\forall \mathbf{x}, \mathbf{y} \in \mathfrak{L}, x \succsim \mathbf{y} \Leftrightarrow u(\mathbf{x}) \geq u(\mathbf{y})$

(b) u is affine, i.e. $\forall \mathbf{x}, \mathbf{y} \in \mathfrak{L}$ and $\forall \alpha \in (0,1), u(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) = \alpha u(\mathbf{x}) + (1-\alpha)u(\mathbf{y})$

The function u is called the *utility function*, and the affinity feature of the utility function can be generalized to the case of more than two outcomes that the following equation is valid:

$$u(\sum p_i o_i) = \sum p_i u(o_i)$$

The left side of the equation is the utility of a lottery, while the right side is the sum of the utility of all the outcomes that the lottery involves. This equation shows that the utility of a lottery (uncertainty involved) can be calculated by the sum of the utility of a set of outcomes (certainty only). The utility function also reflects the decision maker's attitude towards risk: If the decision maker is risk averse, the utility function is in a concave shape, and if the attitude is risk prone, the utility function must be in a convex form.

3.2 Multi-Attribute Utility Theory

Keeney and Raiffa [18] extended the utility theory to the case of multi-attributes. Multi-attribute utility theory is concerned with the valuation of the consequences or outcomes of a decision maker's action. Following the definition of MADP, we use $\mathbf{X} = \{x_1, ..., x_n\}$ to denote the attributes, D_i to denote the set of values of an attribute x_i can have $(1 \le i \le n)$, and $\mathbf{D}$ to denote the space of all possible outcomes (i.e. the Carthesian product $D_1 \times D_2 \times \cdots \times D_n$). In the following discussion, we often use bold lowercase letter $\mathbf{x}$ to denote an outcome (i.e. a value vector in $\mathbf{D}$).

For a decision problem that each action has a deterministic outcome, the decision maker needs only to express preferences among outcomes. The preference relation $\succsim$ on the certainty outcomes can be captured by an order-preserving, real-valued value function as defined below:

Value Function: A function v, which associates a real number $v(\mathbf{x})$ to each point $\mathbf{x}$ in the outcome space, is said to be a value function representing the decision maker's preference structure provided that, for all $\mathbf{x}', \mathbf{x}'' \in \mathbf{D}$, $\mathbf{x}' \sim \mathbf{x}'' \Leftrightarrow v(\mathbf{x}') = v(\mathbf{x}'')$ and $\mathbf{x}' \succ \mathbf{x}'' \Leftrightarrow v(\mathbf{x}') > v(\mathbf{x}'')$, then the optimal problem of the multi-attribute decision problem can be put into the format of the standard optimization problem: Find an outcome $\mathbf{x}$ in $\mathbf{D}$ to maximize $v(\mathbf{x})$.

When there is uncertainty involved in the decision problem, the outcomes of the decisions are characterized by probabilities. To differentiate between certain and uncertain outcomes, we call uncertain outcomes as *prospects (or lotteries)*, and hereafter we use *outcomes* to refer to outcomes in certainty case only. The utility function defined in section 3.1 can be used to capture the preference relation on the lotteries. A utility function is a value function, but a value function is not necessarily a utility function. In the case that only certainty involves, the utility function and value function are interchangeable.

Another two important concepts in MAUT are Preference Independence (PI) and Utility Independence (UI). PI concerns preference for outcomes in certainty case, while UI concerns preferences for lotteries that do involve uncertainty:

Preference Independence: Attribute set $\mathbf{Y}$, where $\mathbf{Y} \subset \mathbf{X}$, is preferentially independent of its complement $\overline{\mathbf{Y}}$ if the preference order of consequences involving only changes in the levels in $\mathbf{Y}$ does not depend on the levels at which attributes in $\overline{\mathbf{Y}}$ are held fixed.

Utility Independence: Attribute set $\mathbf{Y}$, where $\mathbf{Y} \subset \mathbf{X}$, is utility independent of its complement $\overline{\mathbf{Y}}$ if the preference order of *lotteries* involving only changes in the levels in $\mathbf{Y}$ does not depend on the levels at which attributes in $\overline{\mathbf{Y}}$ are held fixed. From the definition we can see that UI is a restrict condition stronger than PI, and in the certainty case, UI is the same as PI.

While MAUT can be used to handle problems in both case of certainty and uncertainty, the multi-attribute decision problem we discussed in this paper only concerns the case of certainty, where the outcomes are deterministic and the decision is mainly a tradeoff process. In the following discussion we only focus on the certainty case. For more discussion of MAUT on the case of uncertainty, please see Keeney and Raiffa's book [18].

As we mentioned earlier in this section, the optimal problem in the multi-attribute decision problem can be solved easily as a maximization problem on value function $v(\mathbf{x})$ which represents the preference structure of the decision maker, provided that the value function $v(\mathbf{x})$ is given. So the critical point of solving multi-attribute decision problem is to determine the value function. To do this, we first analyze the preferential dependency between attributes according to the decision maker's preferences structure. If a certain relation is held, we can determine the general form of the value function according to the theorems in MAUT, and then estimate the parameters in the value function. Let's take the case of mutual preference independence as an example to illustrate the procedure of assessing the value function.

Mutual Preferential Independence (MPI): The attributes $\mathbf{X} = \{x_1, ..., x_n\}$ are mutually preferentially independent if every subset $\mathbf{Y}$ of $\mathbf{X}$ is preferentially independent of its complementary set.

If the mutual preference independence is held, then it can be proved that the form of the value function is additive by the following theorem:

Theorem of additive value function: Given attributes $\mathbf{X} = \{x_1, ..., x_n\}, n \geq 3$, an additive value function $v(x_1, ..., x_n) = \sum_{i=1}^{n} \lambda_i v_i(x_i)$ (where v and v_i are scaled from zero to one, and $\sum_{i=1}^{n} \lambda_i = 1, \lambda_i > 0$) exists if and only if the attributes are mutually preferentially independent.

After the form of the value function is determined, we then need to estimate the unknown parameters in this specific form. For each attribute, we need to assess the component value function $v_i(x_i)$ and the component scale constant λ_i . Before introducing the method of evaluating the component value function $v_i(x_i)$, we define the two terms of *differentially value-equivalent* and *midvalue*:

Differentially value-equivalent: for each attribute x, The value pair (a, b) is said to be differentially value-equivalent to the pair (c, d) where $a < b$ and $c < d$, if whenever we are willing to go from b to a for a given increase of some other attributes $\mathbf{Y}$, we would be just willing to go from d to c for the same increase in $\mathbf{Y}$.

Midvalue: For any value interval [a, b] of the attribute x, its midvalue point c is such that the pairs (a, c) and (c, b) are differentially value-equivalent.

For each component value function $v_i(x_i)$, we set $v_i(x_i^{best}) = 1$, and $v_i(x_i^{worst}) = 0$, and determine the midvalue of $[x_i^{worst}, x_i^{best}]$ (denoted as $x_i^{0.5}$). From the definition, we have $v_i(x_i^{0.5}) = 0.5$. Then we continue to find the midvalues of $[x_i^{worst}, x_i^{0.5}]$ and $[x_i^{0.5}, x_i^{best}]$, and so on. With enough midvalues been evaluated, the final form of the component value function $v_i(x_i)$ can be approximated.

To estimate each component scale constant λ_i, we first choose one attribute as the baseline (say x_1), and for each $x_i (i = 2, ..., n)$, we then estimate how much decrease of x_i can be compensated by a unit increase of x_1, keeping other attributes the same value. This preferential indifference of two outcomes can be formalized as an equation containing variable of λ_i and λ_1 only. Finally we can get the value of each λ_i by solving all the equations we get and the constraint $\sum_{i=1}^{n} \lambda_i = 1, \lambda_i > 0$.

MPI is a very strong condition that it is often not applicable in real-world situations. A weaker condition is that only each single attribute $x_i \in \mathbf{X}$ is preferentially independent to its complementary attributes. In this case it can be proved that the value function would be in a multi-linear form instead:

$$v(x) = \sum_{\varnothing \neq \mathbf{Y} \subset \mathbf{X}}^{n} k_{\mathbf{Y}} \prod_{x_i \in \mathbf{Y}} v_i(x_i)$$

Where $v_i(x_i)$ are component value functions, and $k_{\mathbf{Y}}$ ($\varnothing \neq Y \subset X$) are scaling coefficients. Ha and Haddawy proposed an algorithm based on polyhedral cone to find the optimal solution in this situation [14][2].

[2] In Ha and Haddawy's paper, the discussions were based on the utility function and utility independence for both certainty case and uncertainty case. In the case of certainty, utility function is equal to value function, and utility independence is equal to preference independence.

3.3 Reasoning with Partial Preference Order

In the above of this paper we hold an underlying assumption that the decision maker's preferences could be fully elicited: given any two outcomes $\mathbf{x}$ and $\mathbf{y}$, the decision maker can tell either $\mathbf{x} \succsim \mathbf{y}$ or $\mathbf{y} \succsim \mathbf{x}$. This assumes that all the outcomes can be ordered totally by the preference information. However, this is usually not true in the real case: comparing a pair of multi-attribute outcomes (usually tradeoff has to be made) is not easy due to the decision maker's limited computational capacity. And what's more, most decisions need to be made within a limited time range. Only part of user's preference is elicited with severe time constraints. When only part preferences are elicited, we can only get the partial preference order on the outcome set.

Ha and Haddawy proposed the method of reasoning with partial preference order based on a similarity measure method called *probabilistic distance*[15]. In this method, first a set of user preference is studied and stored in the database. When a new active user comes, the similarity between the active user and each user in the database is computed. Since the active user only has partial preference information, the similarity is measured by probabilistic distance. Then the most similar user is selected and the sampled linear extension of the active user with the most similarity to that user is chosen as the complete representation of the active user's preferences. Ha etc also acclaimed that, by the comparison experiments on the Decision-Theoretic Video Advisor (DIVA) system[27], this method based on probabilistic distance outperformed the GroupLens collaborative filtering algorithm [32] in terms of both precision and recall.

3.4 Discussion

Utility Theory and MAUT have been widely used in solving decision problems in economics especially for those involving uncertainty and risk. Given the utility function, the decision maker's preferences will be totally determined, and the optimal solution of the decision problem is very easy to be selected (the outcome with the maximal utility). We can also rank the outcomes according to their utilities easily.
When using MAUT to solve a multi-attribute decision problem which only involves certainty, the main task is to assess the value function according to the decision maker's preferences. As we have mentioned earlier about the procedure, we first analyze the preference independency between attributes, then several theorems can be used to determine the general form of the value function according to the relationship between attributes. Finally we assess the parameters in the value function by the preferences information.

One limitation of MAUT is that if the relation of preferential independency doesn't hold between attributes of a given MADP (preferential independency is a strong condition compare to the conditional preferential independency which will be introduced in the next section), we can't use the MAUT framework because the form of value function can't be determined. Another disadvantage is that requires the decision maker's preferences being fully elicited before making decision, this doesn't fit the nature of preferential elicitation.

4. CP-network

Both the CSP framework and the MAUT framework require the decision maker to provide sufficient preferential information to the system before a final decision can be drawn. For instance if we want to solve a multi-attribute problem by fuzzy CSPs, the decision maker first needs to specify his constraints (either hard or soft), and then the fuzzy value for each constraint with a certain value set of variables. The interaction complexity is quite heavy.
Boutilier etc. proposed a graphical representation of preferences that reflects conditional dependence and independence of preference statements under a *ceteris paribus* (all else being equal) interpretation: CP-network [5-7] . The CP-network is based on the concept of *conditional preferential independence:* Let **Y**, **Z**, and **W** be nonempty sets that partition **X** (the set of all attributes), **Y** and **Z** are conditionally preferentially independent given an assignment **w** to **W** if and only if, for all $\mathbf{y}_1, \mathbf{y}_2, \mathbf{z}_1, \mathbf{z}_2$ (here $\mathbf{y}_1, \mathbf{y}_2$ are two values of **Y**, $\mathbf{z}_1, \mathbf{z}_2$ are two values of **Z**). we have $(\mathbf{y}_1, \mathbf{z}_1, \mathbf{w}) \succsim (\mathbf{y}_2, \mathbf{z}_1, \mathbf{w}) \Leftrightarrow (\mathbf{y}_1, \mathbf{z}_2, \mathbf{w}) \succsim (\mathbf{y}_2, \mathbf{z}_2, \mathbf{w})$ (denoted as $CPI(\mathbf{Y}, \mathbf{w}, \mathbf{Z})$). If for all $\mathbf{w} \in \mathbf{W}$ we have $CPI(\mathbf{Y}, \mathbf{w}, \mathbf{Z})$, then **Y** and **Z** are *CPI* given **W** (Denoted as $CPI(\mathbf{Y}, \mathbf{W}, \mathbf{Z})$).

To construct the CP-network of a multi-attribute decision problem, for each attribute x, the decision maker is asked to specify a set of parent attributes $Pa(x)$ that can affect her preferences over the values of x. That is, given a particular value assignment to $Pa(x)$, the decision maker should be able to determine a preference ordering for the values of x, all other things being equal. With this information, we are able to create the graph of the CP-network in which each node x has $Pa(x)$ as its immediate predecessors. Then the decision maker is asked to explicitly specify her preferences over the values of x for each assignment to $Pa(x)$. This conditional preference ranking over the values of x is captured by a conditional preference table (CPT) which annotates the node x in the CP-Network. Formally, the CP-network is defined as below: A CP-network over attributes $\mathbf{X} = \{x_1, ..., x_n\}$ is a directed graph $\mathbf{G}$ over $x_1, ..., x_n$ whose nodes are annotated with conditional preference tables $CPT(x_i)$ for each $x_i \in \mathbf{X}$. Each conditional preference table $CPT(x_i)$ associates to a total order $\succ^i_{\mathbf{u}}$ with each instantiation $\mathbf{u}$ of x_i's parents $Pa(x_i) = \mathbf{u}$.

The following simple example illustrates the form of CP-network. Suppose a MADP has only two attributes x_1 and x_2, where x_1 is a parent of x_2 and x_1 has no parents. Attribute x_1 has two values: a and $\bar{a}$, x_2 has two values: b and $\bar{b}$, Assume the following conditional preferences:

$$a \succ \bar{a}; \quad a: b \succ \bar{b}; \quad \bar{a}: \bar{b} \succ b$$

With the above information, the CP-network would be constructed as figure 1:

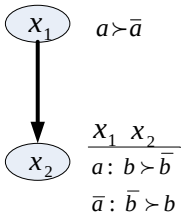

Figure 1: the CP-network

In this example, the conditional preferences information is surprisingly sufficient to totally order the outcomes of the multi-attribute decision problem: $ab \succ a\bar{b} \succ \bar{a}\bar{b} \succ \bar{a}b$ [3].

Given a CP-network structure which specifies the decision maker's preferences over outcome space, two kinds of useful queries can be answered. One is outcome comparison query – preferential comparison between a pair of outcomes. Intuitively from the above example, a chain of "flipping feature values" can be used to show that one outcome is better than another. Also, we can see that the parent preferences have higher priority than the child preferences, violations are worse the higher up they are in the network. We can construct a sequence of increasingly preferred outcomes using only valid conditional independence relations represented in the CP-network by *flipping* values of attributes. If we want to compare a pair of outcomes o_1 and o_2, we can start from outcome o_1, changing the value of a "higher priority" attribute (higher in the CP-network) to its preferred value, even if this introduces a new preference violation for some lower priority attribute (a descendent in the CP-network). This *flipping* operation is repeated until either the outcome o_2 is reached or no more attribute in outcome o_1 can be flipped. If outcome o_2 is reached, we say that o_2 is preferred to o_1. More formally, the flipping sequences can be searched through the *improving search tree* or *worsening search tree.* If an outcome *o* is not preferred by any other outcomes, we say that *o* is a non-dominated outcome.

[3] Though in this simple example the outcomes can be totally ordered by the CP-network, more complicated examples show that only part of outcomes can be ordered by the CPI statements. For instance, we cannot compare two (or more) lower level violations to violation of a single ancestor constraint.

Another useful query is outcome optimization query – determining the set of non-dominated feasible outcomes. Some search algorithms can be helpful for determining the non-dominated outcome set. One possible search method is a straightforward, depth-first, branch-and-bound style algorithm[22, 31]. The algorithm proceeds by assigning values to attributes in a depth-first fashion, using a variable ordering that is consistent with the ordering constraints imposed by CP-arcs (i.e., no child can be assigned before its parents). Suppose initially x is an attribute without parent nodes in the CP-network with the assigned value a, the set of constraints passed on to the next search node can be reduced: the CP-arcs that emanate from x can be removed in all subsequent search steps. This can result in disconnected fragments of the CP-network, and each of which can be optimized independently given $x = a$. During this procedure there is some pruning information that can take place in the search tree. Suppose that the attribute x has two values a and b, and a is preferred than b, if assignment $x = b$ satisfies an equal or smaller set of constraints than was satisfied by $x = a$, then we do not continue to search under $x = b$: any feasible outcome given involving $x = b$ is dominated by some feasible outcome involving $x = a$.
When the non-dominated set is determined, if it contains only one outcome, then this outcome is the optimal solution for the multi-attribute decision problem. Otherwise the decision maker needs to select the most preferred outcome from the non-dominated set.

The CP-network has the advantage of representing the decision maker's preferences effectively by the conditional preference statements which is nature to be captured. The conditional preferential independence is a weaker condition than preferential independence, thus it is applicable to much wider situations than the methods based on preferential independence introduced in section 4. However, being a qualitative method, the CP-network cannot represent quantitative utility information. Boutilier etc further extended the CP-network to UCP-network by adding quantitative utility information to the conditional preference table of each attribute [4].

5. Heuristic Decision Making Strategies

Till now the methods we discussed above for solving multi-attribute decision problems are based on the machine side because generally they require such heavy computational load that human beings seldom could afford. It would be meaningful if we take a look on human side to see how human beings make decisions. In fact, research from psychology area has shown that individuals usually adopt various heuristic strategies to solve MADPs whenever they meet[17, 28, 29]. In this section we first outline some of the heuristic decision making strategies, and then we discuss the potential of solving MADPs by these heuristic strategies.

The Equal Weight (EQW) heuristic. This processing strategy examines all attribute values for each outcome. The decision making is simplified by ignoring information about the relative importance (probability) of each attribute. An overall value for each outcome is obtained by simply summing the values for each attribute of that outcome.
The Elimination by Aspects (EBA) heuristic. This strategy begins by determining the most important attribute. Then, the cutoff value for that attribute is retrieved, and all alternatives with values for that attribute below the cutoff are eliminated. The process continues with the second most important attributes, then the third, and so on, until only one outcome remains. This strategy is first described by Tverskey [41].
*The Majority of Confirming Dimensions (MCD***)** *heuristic.* Described by Russo and Dosher [34], the MCD strategy involves processing pairs of outcomes. The values for each of the two outcomes are compared on each attribute, and the outcome with a majority of winning (better) attribute values is selected. The retained outcome is then compared with the next outcome among the set of outcomes. The process of pairwise comparison repeats until all outcomes have been evaluated and the final winning outcome has been identified.
The Satisficing (SAT) heuristic. Satisficing is one of the oldest heuristics identified in the decision making literature[37]. With this strategy, outcomes are considered one at a time, in the order they occur in the set. Each attribute of an outcome is compared to a predefined cutoff level, which often known as *aspiration level*. If any attribute value is below the cutoff value, that alternative is rejected. The first outcome which passes the cutoffs for all attributes is chosen, so a choice can be made before all outcomes have been

evaluated. In the case where no outcome passes all the cutoffs, the cutoff can be relaxed and the process repeated, or an outcome can be randomly selected.
The Lexicographic (LEX) heuristic. For this strategy, the most important attribute is determined, the values of all the outcomes on that attribute are examined, and the outcome with the best value on that attributes is selected. If two outcomes have tied values, the second most important attribute is examined. And so on, until the tie is broken. Sometimes the LEX strategy includes the notion of a just-noticeable difference (JND). If several outcomes are within a JND of the best outcome on the most important attribute, they are considered to be tied. This version of the LEX strategy is sometimes called *lexicographic-semiorder (LEXISEMI)*.The potential advantage of the LEXSEMI rule is that it ensures that an option is marginally better on the most important attribute but much worse on other attributes will not necessarily be selected.
The Frequency of good and bad features (FRQ) heuristic. Alba and Marmorstein[1] suggest that decision makers may evaluate or choose outcomes based simply upon counts of the good or bad features the outcomes possess. To implement this heuristic, the decision maker needs to develop cutoffs for specifying good and bad features, and then counts the number of such features. This heuristic could be viewed as the application of a voting rule to multi-attribute choice, where the attributes can be viewed as voters.

Besides above 6 heuristic strategies, the decision maker sometimes uses combination of strategies. Typically, combined decision strategies have an initial phase, where poor outcomes are eliminated, and then a second phase examining the remaining outcomes in more detail. For example, *The elimination-by-aspects plus weighted additive (EBA+WADD) strategy* uses an EBA process until the number of available outcomes remaining was three or fewer, and then used a weighted additive strategy (a normative strategy which will be introduce shortly after) to select among the remaining outcomes. *The elimination-by-aspects plus majority of confirming dimensions (EBA+MCD) strategy* first uses the elimination-by-aspects process to reduce the problem size, and then uses a majority of confirming dimensions heuristic to select the optimal outcome from the reduced set.

Johnson and Payne reported a Monte-Carlo simulation study on effort and accuracy of these heuristic strategies[17]. In this simulation experiment, the heuristic strategies are compared with the normative *Weighted additive (WADD) strategy*, which considers the values of each outcome on all of the relevant attributes and all of the relative importance (weights or probabilities) of the different attributes to the decision maker. Each outcome is given an evaluation value by multiplying the weight and the attribute value for each attribute and summing these weighted attribute values over all attributes. The outcome with the highest overall evaluation value is chosen as the optimal solution. Actually, the WADD decision strategy is a special case of the MAUT method with additive value function. The *Random (RAN) strategy* **(**choosing an outcome at random with no search of the available information) is used as a minimum baseline for measuring both accuracy and effort in this simulation.

The effort of each strategy in decision making is measured by the count of all elementary information processes (EIPs)[17]. Each EIP is a basic cognitive operation such as “read a value of an attribute”, “compare two values”, “add the values of two attributes”, etc. EIPs provide a common language for describing seemingly diverse decision strategies in terms of their underlying components. The relative accuracy of each strategy is calculated by that of WADD and RAN strategies: the accuracy of WADD is set to 1, and that of RAN is set to 0.

This simulation experiment shows several interesting results. First, heuristic strategies can approximate the accuracy of the normative strategy (WADD) with substantial savings in effort. A decision maker using an EQW strategy, for example, can achieve 89% of the relative accuracy with only about half the effort, in the low-dispersion, dominance-possible task environment. The lexicographic strategy can achieve 90% relative accuracy, with only about 40% of the effort in the high-dispersion task environment. Experiment results also show that heuristic strategies can be highly accurate in some environments, but no single heuristic does well across all contexts. Moreover, when under time constraints, heuristics might be even more accurate than a normative strategy such as WADD. The heuristic’s accuracy may degrade under increasing time pressure at a lower rate than WADD strategy degrades. In later experiments carried out by Payne etc[28], the results show that people shift decision strategies in response to a context change in ways that maintain accuracy without explicit outcome feedback, and under time constraints, several heuristic strategies are more accurate than a truncated normative procedure.

The heuristic strategies discussed in this section are obviously useful for individuals when they are trying to find the optimal solution of the MADP. As mentioned above, the effort of solving MADP with heuristic strategies is relative low while the accuracy is not degraded too much. The optimal solution found by heuristic strategies has the advantage of matching with the decision maker's mental model, which implies that the decision maker is easier to accept the solution.

Two problems require further study before implementing some algorithms to solve MADP based on heuristic strategies. One is the error of the decision that caused by heuristic strategies. We can see from the simulation experiments that none of the heuristic strategies can get 100% accuracy compared to the WADD rule. We need to select the right strategy so to get minimal error, and we also need to study what degree of error is acceptable for the decision maker. The other problem is the adaptive nature of heuristic strategies: people change heuristic strategies implicitly if the context changes. To solve this, we can study this phenomenon and try to make the change of heuristic strategies be predictable, or we can find several solutions by different strategies simultaneously, and then select the optimal solution among them by a certain criteria.

6. Summary

In this paper, first we formally defined the multi-attribute decision problem, and then we studied four different methods (soft-CSP framework, MAUT, CP-network, and heuristic strategies) which potentially could be used to solve the MADP, and their advantages and disadvantages are discussed respectively. It is shown that though each of the four methods is theoretically sound and complete, none of them is ideally suitable for solving the MADP problem. Here we suggest that the combination of two or more of these methods may provide better performance for the real application.

Currently few application of MADP is implemented based on these methods. It would be promising to develop a real application framework with all these methods so that their performances can be compared thoroughly at the same standard. Also, preference elicitation is another tough task for decision making in multi-attribute situation. During this paper we have assumed that the decision maker's preferences have been elicited before the decision procedure. However, usually the preferences can only be fully elicited during the interaction procedure due to the constructive nature of preferences. We need to find a better way of integrating the procedure of preference elicitation and decision making together.

7. References

1. Alba, J.W. and Marmorstein, H. The Effects of Frequency Knowledge on Consumer Decision Making. *Journal of Consumer Research*, *14*. 14-25.
2. Bernoulli, D. Exposition of a New Theory on the Measurement of Risk (Original 1738). *Econometrica*, *22*. 23-36.
3. Bistareli, S., Fargier, H., Montanari, U., Rossi, F., Schiex, T. and Verfaillie, G. Semiring-based CSPs and Valued CSPs: Basic Properties and Comparison. *CONSTRAINTS: An international journal*, *4* (3).
4. Boutilier, C., Bacchus, F. and Brafman, R., UCP-Networks: A Directed Graphical Representation of Conditional Utilities. in *Proceedings of the Seventeenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, (Seattle, 2001), 56-64.
5. Boutilier, C., Brafman, R., Domshlak, C., Hoos, H.H. and Poole, D. CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. *Journal of AI Research*, *21*. 135-191.
6. Boutilier, C., Brafman, R., Geib, C. and Poole, D., A Constraint-Based Approach to Preference Elicitation and Decision Making. in *AAAI Spring Symposium on Qualitative Decision Theory*, (1997).
7. Boutilier, C., Brafman, R., Hoos, H.H. and Poole, D., Reasoning with Conditional Ceteris Paribus Preference Statements. in *Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, (Stockholm, Sweden, 1999).
8. Dubois, D., Fargier, H. and Prade, H., The Calculus of Fuzzy Restrictions as a Basis for Flexible Constraint Satisfaction. in *Proceedings of IEEE International Conference on Fuzzy Systems*, (1993).
9. Faltings, B., Pu, P., Torrens, M. and Viappiani, P., Design Example-Critiquing Interaction. in *Proceedings of the International Conference on Intelligent User Interface(IUI-2004)*, (Funchal, Madeira, Portugal, 2004), ACM Press, P22-29.
10. Fargier, H., Hang, J. and Schiex, T., Selecting Preferred Solutions in Fuzzy Constraint Satisfaction problems. in *Proceedings of the First European Congress on Fuzzy and Intelligent Technologies*, (1993).
11. Fargier, H. and Lang, J., Uncertainty in Constraint Satisfaction Problems: a Probabilistic Approach. in *Proceedings of European Conference on Symbolic and Qualitative Approaches to Reasoning and Uncertainty*, (1993), 97–104.

12. Freuder, E.C. and Wallace, R.J. Partial Constraint Satisfaction. *Journal of Artificial Intelligence, 58* (1-3). 21-70.
13. Gaschnig, J. Performance Measurement and Analysis of Certain Search Algorithms, Carnegie Mellon University, Pittsburgh, 1979.
14. Ha, V. and Haddawy, P., A Hybrid Approach to Reasoning with Partial Preference Models. in *15th Conference in Uncertainty in Artificial Intelligence (UAI '99)*, (Stockholm, Sweden, 1999), 263-270.
15. Ha, V. and Haddawy, P. Similarity of Personal Preferences: Theoretical Foundations and Empirical Analysis. *Artificial Intelligence Journal, 126* (2). 149-173.
16. Herstein, I.N. and Milnor, J. An Axiomatic Approach to Measurable Utility. *Econometrica, 21*. 291-297.
17. Johnson, E.J. and Payne, J.W. Effort and Accuracy in Choice. *Management Science, 31*. 395-414.
18. Keeney, R.L. and Raiffa, H. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Cambridge University Press, 1993.
19. Kumar, V. Algorithms for Constraint Satisfaction Problems: A Survey. *AI magazine, 13* (1). 32-44.
20. Larrosa, J., Node and Arc Consistency in Weighted CSP. in *Eighteenth National Conference on Artificial Intelligence*, (Edmonton, Alberta, Canada, 2002), American Association for Artificial Intelligence, 48-53.
21. Larrosa, J. and Schiex, T., In the Quest of the Best Form of Local Consistency for Weighted CSP. in *Proceedings of 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, (Acapulco, Mexico, 2003).
22. Lawler, E.L. and Wood, D.E. Branch-and-bound methods: a survey. *Operational Research, 14*. 699-719.
23. Mackworth, A.K. Constraint Satisfaction. in Shapiro, S.C. ed. *Encyclopedia of Artificial Intelligence*, Springer Verlag, 1988, 205-211.
24. Marschak, J. Rational Behavior, Uncertain Prospects, and Measurable Utility. *Econometrica, 18*. 111-141.
25. Mongin, P. Expected Utility Theory. in Davis, J.B., Hands, D.W. and Maki, U. eds. *The Handbook of Economic Methodology*, Edward Elgar, 1998, 342-350.
26. Nadel, B.A. Tree Search and Arc Consistency in Constraint-Satisfaction Algorithms. in Kanal, L.N. and Kumar, V. eds. *Search in Artifical Intelligence*, Springer-Verlag, London, 1988, 287-342.
27. Nguyen, H. and Haddawy, P., The Decision-Theoretic Interactive Video Advisor. in *The Fifteenth Conference on Uncertainty in Artificial Intelligence*, (1999).
28. Payne, J.W. and Bettman, J.R. Adaptive Strategy Selection in Decision Making. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 14* (3). 534-552.
29. Payne, J.W., Bettman, J.R. and Johnson, E.J. *The Adaptive Decision maker*. Cambridge University Press, 1993.
30. Pu, P., Faltings, B. and Torrens, M., User-Involved Preferences Elicitation. in *workshop notes, workshop on Configuration, the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, (2003).
31. Reingold, E.M., Nievergelt, J. and Deo, N. *Combinatiorial Algorithm: Theory and Practice*. Prentice-Hall, Englewood Cliffs, NJ, USA,, 1977.
32. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J., GroupLens: An Open Architecture for Collaborative Filtering of Netnews. in *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, (Chapel Hill, NC, U.S.A, 1994), 175-186.
33. Rossi, F., Petrie, C. and Dhar, V., On the Equivalence of Constraint Satisfaction problems. in *Proceedings of European Conference on Artificial Intelligence (ECAI90)*, (Stockholm, 1990).
34. Russo, J.E. and Dosher, B.A. Strategies for multiattribute binary choice. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 9*. 676-696.
35. Ruttkay, Z., Fuzzy Constraint Satisfaction. in *Proceedings of the Third IEEE International Conference on Fuzzy Systems*, (Orlando, 1994), 1263-1268.
36. Schoemaker, P. The Expected Utility Model: Its Variants, Purposes, Evidence and Limitations. *Journal of Economic Literature, 20* (2). 529-563.
37. Simon, H.A. A Behavioral Model of Rational Choice. *The Quarterly Journal of Economics, 69*. 99-118.
38. Stallman, R. and Sussman, G.J. Forward Reasoning and Dependency-Directed Backtracking. *Artificial Intelligence Journal, 9* (2). 135-196.
39. Torrens, M. Intelligent Scalable Electronic Catalogs, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, 2002.
40. Tsang, E. *Foundations of Constraint Satisfaction*. Academic Press, London, UK, 1993.
41. Tversky, A. Elimination by Aspects: A theory of Choice. *Psychological Review, 79*. 281-299.
42. von Neumann, J. and Morgenstern, O. *The Theory of Games and Economic Behavior*. Princeton University Press, Princeton, 1944.