Robust Interactive Decision-Analysis (RID): Concepts, Methodology, and System Principles

Po-Young Chu
Pennsylvania State University
The Capital College
Middletown, PA 17057

Herbert Moskowitz
Purdue University
School of Management
W. Lafayette, IN 47907

Richard T. Wong
Purdue University
School of Management
W. Lafayette, IN 47907

ABSTRACT

We have proposed a novel interactive procedure for performing decision analysis, called Robust Interactive Decision-Analysis (RID), which avoids the difficult problems in measuring utility and state probability information associated with traditional decision tree analysis. Instead, the RID method permits a decision maker (DM) to voluntarily and interactively express strong (viz, sure) binary preference for actions, partial decision functions, and full decision functions, and only imprecise probability and utility function assessments. These inputs serve as operators to prune the event probability space and decision space until an optimal choice strategy is obtained. Conceptually, the operation of the RID method can be regarded as a state-space pruning system and the computer implementation of the RID methodology can be viewed as a decision support system. Having described the RID methodology and its theoretical developments, we continue the illustration of the corresponding state-space pruning system and its properties. We also discuss the system principles which we have derived from psychological theories and end-user interface studies as well as our implementation experiences with the RID method.

Introduction

Problem Statement and Motivation

Decision analysis is a powerful concept for resolving choices among actions for complex decision making problems in the face of uncertainty. It is a scientific discipline for systematic evaluation of alternatives by explicitly quantifying uncertainties and an individual Decision Maker's (DM's) utility function. Hence, decision analysis enables us to structure, to decompose and to analyze complex problems so that the chosen action is consistent with our basic values and knowledge.

Problem statement. Notwithstanding the wide applicability of decision analysis, one common obstacle to its widespread use is the negative attitude of managers toward the usefulness of attempting to quantify uncertainty when the information about the uncertain quantity is weak [4]. Applying traditional decision analysis for solving a decision making problem under uncertainty requires full knowledge of the underlying probability function and a DM's utility function. However, such information is often very difficult to extract from a DM accurately by using formal assessment procedures; therefore most real world problems generally fall short of taking advantage of the full power of this useful decision tool.

Motivations. Traditional decision analysis often fails because of the measurement problems associated with it. Subjective probability and utility information is costly to obtain from DMs and is often biased, thus rendering decision tree analysis ineffective [9], [10], [27]. Moreover, the optimal strategy in a decision analysis problem is generally robust with respect to the degree of preciseness of the DM's beliefs (state probability function) and tastes (utility function); viz, only fuzzy information of this nature is necessary to achieve optimality. Why then collect such precise information, when it may be unnecessary, to arrive at an optimal choice strategy?

To avoid these cited measurement problems inherent in the traditional explicit procedure, we propose an interactive approach which relaxes its stringent information requirements by solving a traditional decision tree problem implicitly. This approach is based upon the premise that, in general, a DM can only consistently specify some imprecise knowledge about state probabilities or state probability function and can only consistently articulate some strong preferences about pairs of possible actions available. Thus, we incorporate a DM's imprecise state probability (ordinal or interval) knowledge and a DM's "strong" preferences sequentially into an interactive procedure we call Robust Interactive Decision Analysis (RID), which repeatedly prunes the set of possible decision functions until a single "optimal" or efficient decision function remains.

Specifically, three major reasons have motivated us to use a less strict probability measurement in our proposed interactive methodology. First, it would appear to be a much easier task for a DM to consistently rank the state probabilities or give interval probability estimates than to give precise point probability estimate. Such imprecise state probability knowledge is useful in searching for good decision functions [18]. Second, inaccurate point state probability estimates obtained may lead to a non-optimal solution in the traditional explicit method. If fuzzy state probabilities are reliable and sufficient to search for solutions, it is not necessary to use the unreliable point probability estimates. Third, in group decision making, it is generally true that individuals may not agree on the point probability estimates but will agree on the imprecise (ordinal or interval) probability estimates. As a result, we can use the imprecise probability knowledge to generate a set of commonly accepted solutions to reconcile differences.

Similar reasons justify our use of strong preferences to obviate a precise utility measurement. First, a DM's preference over alternatives can provide useful information to differentiate among competing solutions. Second, a DM's strong preferences can be very effective in generating efficient solutions. As strong preferences help characterize the underlying state probabilities, we can quickly reduce the size of the efficient solution set by sequentially pruning inefficient solutions implied by the characterized state probabilities. Third, a strict procedure of measuring a DM's utility function directly may lead to a nonoptimal solution.
An Overview of Theoretical Developments

We briefly introduce the major theoretical developments which the RID method has integrated, namely, (1) Raiffa's normal form of decision analysis [19], (2) Moskowitz [13] and Moskowitz and Wallenius' [14] algorithms, and (3) State-space pruning modeling.

**Normal form of decision analysis.** The normal form of decision analysis [19] is the analytical framework used in the RID method. While both approaches, the extensive and normal forms of analysis, are theoretically equivalent, they are computationally different. The extensive form of analysis is the conventional approach used in the classroom and in practice. However, the extensive form of analysis requires an initial and complete explicit measurement of a DM's probability and utility function to be incorporated directly and immediately into the analysis. In contrast, the normal form of analysis incorporates subjectivity (the assessed probability and utility functions) later into the analysis, after the nondominated (efficient) decision functions have been identified.

**Recursion algorithms.** Moskowitz [13] developed a recursion algorithm for determining the set of nondominated decision functions to reduce the computational difficulty associated with the normal form of analysis. To further reduce the size of the remaining efficient set, Moskowitz and Wallenius [14] augmented the original recursion algorithm with the DM's strong binary preferences at each stage of recursion. As the number of efficient decision functions under consideration is reduced progressively, the solution space will ultimately converge to a unique optimal decision function if the DM can continue giving strong preferences over presented binary alternative comparisons.

While the augmented preference algorithm always prunes some inefficient solutions, namely the one less preferred and the ones concluded to be inefficient by statistical dominance, the number of possible binary alternative comparisons under consideration will increase exponentially. A unique optimal total decision function will consist of only one efficient partial decision function for each experimental outcome. If the number of actions is K and the number of experimental outcomes is M, then the number of possible total decision functions under consideration is K^M.

In the RID method, to overcome this observed computational problem, we have incorporated a decomposition concept which partitions the decision problem into several subproblems according to different experimental outcomes. The RID method as a state-space pruning system. Conceptually, the operation of the RID methodology can be viewed as a state-space pruning system which has incorporated a decomposition concept with the original Moskowitz and Wallenius preference recursion algorithm. States, which describe the current condition of the problem during analysis [2],[16], represent the status (efficient or inefficient) of all possible decision functions for the problem. Operators, which for our context correspond to pruning inefficient decision functions, are a means to transform the problem from one state to another until a final goal state is achieved and the problem is solved. Consequently, by modeling the RID procedure as a state-space search problem, we can expedite the solution search and minimize a DM's cognitive load.

Moreover, we can regard the computer implementation of the RID methodology as a decision support system for a DM. The RID approach is ideally suited for operationalization as an interactive computer decision support system where the user (DM) is queried for some imprecise probability estimates and iteratively asked to give strong preference choices. To design an effective user-oriented interactive system, we have closely followed several important system principles which have been derived from psychological theories and end-user interface studies.

**Concepts and Methodology**

We propose a novel interactive procedure for performing decision analysis, RID, which permits a DM to interactively express strong binary preferences for partial decision functions and only imprecise state probability and utility function assessments. Operationally, these inputs serve as operators, along with the others, to prune the decision space until an optimal choice strategy is obtained.

We first describe the RID method with a flowchart and an example. After having given a detailed definition for each constituent of the state-space pruning system, we continue the illustration of two important perspectives of the RID method, namely, (1) the corresponding state-space pruning system to represent and to operate the RID method and (2) the decomposition and the aggregation concepts to improve the search efficiency of the corresponding state-space system.

**The RID Method**

Figure 1 flowcharts the basic steps of the RID method. We elicit some imprecise (ordinal or interval) probability estimates and strong preferences and incorporate them into an interactive procedure which repeatedly prunes the set of possible decision functions until a single 'optimal' or efficient decision function remains.

During the analysis, the DM is prompted to evaluate a multidimensional vector (payoff vector or weighted version of it) corresponding to an action (or a preferred set of actions) where each element corresponds to the payoff (or weighted payoff) under a particular state of nature. Consider the example shown in Table 1: there are three actions (invest in stocks, bonds or a money market), three states of nature (a good, flat, or bad economy) and three experimental outcomes (POOR, FUZZY or ROSY outlook). The payoff vector (50,20,0) represents the values received in a good, flat, and bad economy respectively by investing in stocks when no experimental information is available. By comparing binary vectors weighted by the fuzzy state probabilities in the DM's mind, the DM may express a strong preference for one vector over another.

<table>
<thead>
<tr>
<th>Table 1. Hypothetical Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action-State Payoff Matrix</strong></td>
</tr>
<tr>
<td>θ</td>
</tr>
<tr>
<td>Action</td>
</tr>
<tr>
<td>stock</td>
</tr>
<tr>
<td>bond</td>
</tr>
<tr>
<td>mkt</td>
</tr>
</tbody>
</table>

256
A State-Space Pruning System

We can view the operation of the RID method as a state-space pruning system. States, which describe the current condition of the problem during analysis [2],[16], represent the status (efficient or inefficient) of all possible decision functions for the problem. Operators, which correspond to pruning inefficient decision functions, are means to transform the problem from one state to another until a final goal state is achieved and the problem is solved. For our purpose a goal state corresponds to the situation where only a single decision function remains efficient.

There are three distinct phases in the RID method where this state-space pruning is applied. In phase 1 (prior analysis) we perform the analysis without any experimental information, that is, without any conditioning event; in phase 2 (posterior analysis) there is experimental information corresponding to an experimental outcome z as the conditioning event; in phase 3 (pre-posterior analysis) the conditioning event is a vector of distinct experimental outcomes ℤ=(zi1,...,zin). In phases 2 and 3 we must perform the pruning analysis for each possible conditioning event (for every relevant experimental outcome or set of outcomes). Thus, phases 2 and 3 can be viewed as a natural decomposition of the problem according to each experimental outcome or relevant set of outcomes.

States. At the start of the analysis, all partial/total decision functions are assumed efficient and these conditions form the initial state [6]. The system searches for a possible goal state by using applicable operators to prune inefficient solutions. Since there is only one initial state of the problem (there is only one M goal states, a forward chaining (data driven) approach is recommended for better search efficiency [16]. Let E represent the current set of efficient solutions and P represent the set of feasible probability functions defined over the feasible region bounded by the constraint set S.

Operators. For all three phases there are three pruning mechanisms used: vector dominance, preference dominance, and statistical dominance. The pruning mechanisms all perform in essentially the same way. Vector dominance involves testing whether one vector dominates another vector. The strong preference test involves querying the DM as to whether he/she has a strong preference of one vector over another. The expected value of a vector corresponds to weighting the vector elements by each corresponding state probability. The statistical dominance test involves determining whether the expected value of one vector is always greater than or equal to the expected value of another vector for any feasible set of values for the state probabilities. The following is a detailed definition of each operator.

Vector Dominance:
Given two vectors X=(x1, x2, ...,xn) and Y=(y1, y2, ...,yn), X is vector dominant over Y if and only if all xi’s are greater than or equal to yi with at least one strict inequality [28],[15],[14].

Preference Dominance:
Given two vectors X=(x1, x2, ...,xn) and Y=(y1, y2, ...,yn), X is preference dominant over Y if and only if the DM states that X is "strongly" preferred to Y [14].

Statistical Dominance:
Given two vectors X=(x1, x2, ...,xn) and Y=(y1, y2, ...,yn), and the probability function P=(p1, p2, ...,pn), X statistically domi-
nates Y iff \((X - Y)P \geq 0\) for all feasible probability functions \(P\) with at least one strict inequality. A statistical dominance of \(X\) over \(Y\) implies that the expected value of \(X\) is always greater than or equal to the expected value of \(Y\) for all feasible probability functions. Therefore \(Y\) is an inefficient (dominated) solution.

**How the Operators Work.** Since the vector dominance and the preference dominance operators eliminate inefficient solutions in a straightforward manner, only the statistical dominance operator needs more illustration.

**Sources of Statistical Dominance.**

Two sources of statistical dominance are: a DM's strong preferences and a DM's probability inputs. Each of the DM's strong preferences impute one linear constraint under the assumption of a linear utility function or known nonlinear utility functions. Without the loss of generality, all examples are illustrated under the linear utility function. For example: if \(X\) is preference dominant over \(Y\), then \((X - Y)P > 0\). As a result, the competing strategy \(Y\) is eliminated due to the DM's preference. Moreover, the newly added imputed linear constraint may lead to the additional pruning of inefficient strategies as the new constraint set further characterizes the underlying state probability feasible solution region.

Similar to strong preference inputs, all DM's probability inputs can be converted into linear constraints. For example: if there are three states of nature, let the probability for each state to be possibly realized be labeled as \(P_1\), \(P_2\), and \(P_3\), and suppose the DM states that state \(1\) is more likely to occur than the other two states, and state \(2\) is more likely to occur than state \(3\). We can convert such probability knowledge into the following linear constraints:

\[
P_1 - P_2 \geq 0
\]
\[
P_2 - P_3 \geq 0
\]

in addition to the default constraint \(P_1 + P_2 + P_3 = 1\).

A DM's interval state probability estimates also can be converted into linear constraints. For example, if the DM states that the probability of state \(1\) is between 0.2 and 0.7, corresponding, we add the following linear constraints to the current constraint set \(S\) to restrict the values of the underlying state probabilities.

\[
0.7 \geq P_1 \geq 0.2
\]

As the analysis progresses and more linear "cuts" are added to the constraint set \(S\), the feasible region defined by \(S\) will become smaller. The reduced feasible region can be used to further check the solution efficiency via statistical dominance.

The intermediate feedback, the resultant current efficient set, is presented to the DM and more binary preference comparisons will be constructed. With more imputed constraints by strong preferences, the solution space will ultimately converge to a single "optimal" or efficient decision function. For further specific details of the statistical dominance checking process, see Chu [6].

**Decomposition and Aggregation.**

From the previous example, we see the original decision problem has been naturally decomposed into three subproblems according to each experimental outcome. The most important characteristics of these naturally decomposed subproblems is that a strong preference input from any particular subproblem may prune other inefficient decision functions in other subproblems simultaneously. Since a DM's strong preferences characterize the same underlying state probabilities, we can use the characterization of the state probabilities to repeatedly prune all inefficient decision functions in all subproblems simultaneously via statistical dominance.

Hence, we can first solve each subproblem individually before adopting the preference recursion algorithm suggested by Moskowitz and Wallenius [14] to avoid the problem of the exponential increase of the number of possible binary preference comparisons. Consequently, it is more efficient computationally as well as cognitively than the original vector and preference recursion algorithms.

The concept of "divide and conquer" has been used extensively in the artificial intelligence domain in search of solutions [2],[16]. The commonly shared probability space of competing alternatives and the proposed interactive sequential decision making process improve the computational efficiency of our state-space pruning system approach in search of efficient solutions.

**System Principles, Theories and Implementations.**

We have viewed the computer implementation of the RID methodology as a decision support system which should effectively present problem information to assist the DM (user) in interacting with the system. According to Hansen [11], just as a composer follows a set of harmonic principles to write music, the system designer must follow some set of principles, which Hansen called user engineering principles, in order to design a user-oriented interactive system. Hansen's first principle in the design of an interactive system is "know the user." The interface must be designed to suit the needs and abilities of the individual user. In order to design an acceptable user interface, the psychology of the system user should be carefully considered. Some interface design issues are directly related to user psychology and are independent of the background or experience of the user. Shneiderman [21],[322] has outlined many important design criteria for software development. Rubinstein and Hersh [20] have provided many useful guidelines for designing computer systems for people. In addition, through our conceptual and implementation work with the RID method, we also have evolved some system principles in order to insure effective and informative interactions with the user. In this section, we will discuss these design principles, related theoretical bases, and the corresponding RID system features.

**Facilitate the DM's Cognitive Process by Reducing the Cognitive Load.**

Miller [12] has identified the limited capacity people have for absorbing information. People can rapidly recognize approximately seven "chunks" of information at a time and hold them in Short-Term Memory for about fifteen to thirty seconds. Similarly, in the Model Human Processor [5], two important memories are proposed in the cognitive system: a Working Memory to hold the intermediate products of thinking and the Long-Term Memory to store knowledge in terms of symbols, called chunks, for future use. Consequently, probably the most severe constraint on human cognitive processes arise from the limited capacity of Working Memory and the need to keep working data within this limit. Human inability to perform quantitative operations has been attributed to too many intermediate states needed to be retained in Short-Term Memory.

Shneiderman [22] indicates that information overloading will occur when the interface presents the user with too many distinct items of information. If more than seven distinct items are presented to the user at the same time, they cannot all be retained in Short-Term Memory by the user and some are likely to be forgotten. In designing an interface, therefore,
information should be presented in chunks so that the user has time to assimilate that information. Minimizing conceptual load and avoiding clutter are also suggested by Rubinstein and Hersh [20] in designing computer systems for people.

In studying the human cognitive process, Sommerville [23] suggests that people have an inherent desire for closure--because of Short-Term Memory limitations. Because of this desire for closure, user interfaces are best organized as a sequence of short operations rather than a single large, complex task, even though this may result in more interactions with the system.

Information studies reported by Shneiderman [22] suggest that users prefer short operations in sequence rather than a single, more complicated, operation. Smaller tasks generally require less Working Memory for their performance than do larger tasks, which is one reason why a user confronted with a large task will break it into smaller tasks, which have been called unit tasks [5]. In addition, Brown and Newman [3] suggest that system complexity--multiple and simultaneous processes--is one of the important factors that affect the intelligibility of information systems.

In designing the RID system, we have taken these theoretical issues into account and implemented them accordingly in terms of the following system features.

**Decompose problem solving into three phases of analysis and simplify the preference procedure via binary vector comparisons.** The decomposition concept applied to the RID system can be perceived at both macro and micro levels. At the macro level, we have proposed the decompositional concept applied to the RID system. At the micro level, we further simplify the procedure of structuring the DM's strong preferences via binary vector comparison. By decomposing the user's interactions into a series of choices where each decision involves comparing only two vectors, the cognitive requirements to perform any particular decision are quite modest even though the overall decision problem may be very large and complex.

**Use menu selection rather than entry.** To minimize memorization in using a system, Hansen [11] has suggested the user select the appropriate item from a list displayed by the computer rather than type a character string or operation name. The RID system basically is menu driven. To interact with the system, the user simply selects the desired option from hierarchical menus to solve traditional decision tree problems progressively. With respect to the menu-based design, Rubinstein and Hersh [20] have suggested that menus be kept short and labeled clearly. We have developed our menus accordingly.

**Interact with the user via graphics.** We believe graphs and charts are the best way of rapidly communicating the major trends or interactions contained in information such as a data set or a summary of the system status. Mental resource limitations--narrow bandwidth of Short-Term and perceptual memory--are cited as the "structural features" which affect human decision performance [5].

Graphic systems allow the information stored and processed by the computer to be displayed in such a way that the user can gain an overall impression of it. This type of overview is needed by the RID system user to assimilate binary vector comparisons easily and to learn the current status of problem solving as it progresses. Graphics which automatically aggregate data into higher-level units can avoid the problem of presenting too many detailed data on the same screen. The presentation of too many distinct items to the user at the same time has been cited as the reason for information overloading [22].

Using graphs and charts is also one of the guidelines provided by Rubinstein and Hersh [20] for the effective presentation of information to the user. For example, to compare two vectors for preference dominance, color graphs and bar charts are used to display the information concerning the vectors. The graphic approach provides an alternative means to present information, in addition to the traditional approach of listing the numerical values of the vectors.

**Enhance a DM's Information-Processing Capabilities by Grouping Related Chunks of Information by Using Colors and Providing Feedback.**

Research has confirmed that structurally grouping related information will improve task performance significantly [8], [25]. Information that is organized is much easier to learn and to remember than information with no apparent structure. Rubinstein and Hersh [20] have suggested that the semantics of attributes be made clear and consistent in designing computer systems for people. As a result, windowing techniques have been used to group related information for novice users. Teitelman [24] indicates the technique of interacting with different windows makes it easy for the user to manipulate the windows or the contents of a particular window by a combination of keyboard inputs. The human cognitive processes of learning, including internal mental representation and chunks, may be used to explain how grouping related information leads to such improved performance.

The human cognitive processes involved in a decision (aided or unaided) are strongly dependent on the person's internal mental representation of that problem [5]. When a human learns an item, it is encoded in Long-Term Memory with possible cues for recall [3]. Grouping related information may make these chunks more easily organized and assimilated cognitively, as the unity of learning is not the textual item but the chunk [12]. If the user already has the material partly organized in chunks, then only the remaining chunks must be learned. Neal [15] has studied the human chunking phenomenon in typing data records organized into numerical, alphanumerical, and English fields. People learn to cope with complex problems by developing higher-level concepts that combine several lower-level concepts into a single chunk. Novices at any task tend to work with smaller chunks until they can cluster concepts into larger chunks.

Besides grouping related information, using color is another means to improve screen design. The advantages of using color in graphic displays for greater information density have been confirmed experimentally. In describing an experiment that used color in one of four versions of a CRT display, Tullis [25] surmised that user reactions to the color graphics display were positive.

Human-computer communications feedback has been cited as an important factor affecting the number of errors made by the system user [1]. Immediate feedback not only informs the user that an error has occurred, but also provides him/her an opportunity to make a quick correction. Corresponding to the issues cited above, we implement the following system features to enhance human information-processing capabilities:

**Display semantically associated information in the same screen via windows of graphics and numbers simultaneously.** In order to present problem information effectively so that the user can easily assimilate different chunks of information, the RID system uses different windows to cluster semantically related chunks throughout.
For example, we use three windows in the same screen simultaneously to show different perspectives of the problem. One window shows the current decomposed subproblem as a number matrix or a group of bar charts, another window compares binary vectors graphically, and the third window depicts the overall problem-solving status as a global decision tree. By viewing these three windows together, the user can see the immediate impacts of his/her strong preference on problem solving locally and globally as well. Such prompt feedbacks facilitate the DM's learning process greatly.

Use hierarchical windows accompanied by consistent colors to help the DM assimilate different problem-solving phases. Architectural plans benefit from color coding of electrical, phone, hot water, cold water, and natural gas lines. Similarly, map makers can have greater information density when color coding is used. In densely packed displays where screen space is at a premium, similar colors can be used to group related items. A consistent color pairing along with hierarchical windows are used to semantically associate all information with the solving of a decomposed subproblem. In comparing binary vectors, we also use color pairings to contrast the differences between two selected decision functions, so that the user can easily assimilate such differences in order to construct his/her strong preferences.

Color can also be used as an attention-getting device with an effect on human perception similar to traffic light switching. By changing to a different color pairing, we signal the condition of solving different subproblems. From the error rate of reading, Pace [17] alerts users to some problems with particular color pairings such as blue and red. We have avoided using those troublesome color pairs in our screen design.

Use the frame-input-prune loop to provide feedback to the user and allow a progressive learning of the problem. In all three phases of the RID method, a frame-input-prune loop is used as the fundamental operation. Namely, (1) enumerate and present all binary vector pairs to the DM, (2) the DM selects a pair of vectors to evaluate and possibly to express a strong preference, (3) the less preferred vector is pruned and statistical dominance is invoked to eliminate inefficient solutions, and (4) present the resulting preference-efficient set to the DM, and repeat the process. The user receives immediate feedback as to implications of his/her strong preferences. By maintaining such consistency, we do not allow the user to needlessly complicate and confuse the analysis by entering a set of conflicting choices. Our behavioral studies have shown that DMs are quite capable of making meaningful and consistent strong preferences [7] so this consistency restriction does not appear to be a very significant one.

System opacity has been criticized as one important factor affecting the intelligibility of an information system [3]. A user can employ backtracking to make the system more transparent and to undo previously imposed strong preferences. Therefore, a user can use a sense-making strategy and a trial-and-error approach to understand the system behavior.

Maintain consistency throughout the analysis. In the RID method, we solve decomposed subproblems sequentially by repeatedly pruning inefficient decision functions. Once a user has made a strong choice preference, all decision functions made inefficient as a result are found and eliminated. In later choices the user is not permitted a strong choice preference which would imply that a previously determined inefficient decision function is preferred to a currently efficient decision function.

Research indicated that people made inconsistent preferences systematically under experimental conditions [26]. In general, DMs would correct the inconsistent preferences once they were told. People are observed to commonly use a sequential approach in complex decision making, as a sequential decision process generally would reduce the frequency of inconsistent preferences. Therefore, we have adopted the sequential decision procedure in the RID method in agreement with human decision-making behavior.

Use backtracking to access system information and enhance system flexibility. Humans forget. In the RID system, we use a backtracking feature to augment human memory. Hansen [11] suggests that accessing to system information can minimize memorization in using a system.

By backtracking, the user can see the previous strong preference inputs and their pruning consequences. This system feature will not only help the user to logically associate the strong preferences with their impacts, but also provide the user with a desirable flexibility to correct mistakes made earlier.

This backtracking facility allows the user to take back strong choice preferences and is therefore very useful for addressing "what-if" types of sensitivity analysis questions. Also, the task of interacting with the system is eased and made more comfortable, since user decisions are not irrevocable and can be easily modified.

Optimize Operations by Using a DM's Heuristics and by Implementing the RID System in C-Language

The implications of optimizing operation in the RID system are twofold: first, we want to maximize the pruning power by sequentially applying different operators to the problem so that we can find the optimal decision function efficiently; second, we want a short response time in interacting with the system. In order to achieve these two objectives, we implement the following system features:

Implement decomposition concepts and use a DM's imprecise probability knowledge to expedite pruning operations. In the methodology section, we discussed the potential improved pruning efficiency of augmenting Moskowitz's [13] original recursion algorithm with the decomposition concept. Furthermore, computational results have shown that the strong preference choices and the ordinal probability inputs are very powerful in reducing the set of decision functions [6].

Reduce the response time by implementing the RID system in C language. Longer response times may lead to less desirable task performance. The computer system's response time is the number of seconds it takes from the moment a user initiates an activity until the computer begins to present results on the screen or printer. Long studied delays of approximately 1 to .5 seconds in the time for a keystroke to produce a character on an impact printer. He found that both unskilled and skilled typists worked more slowly and made more errors with longer response times than shorter response times. Our
first prototype RID system was built in GURU, which is an integrated software system including graphics, spreadsheet, rule based subsystems, and a higher-level language capability. GURU is an interpreter environment; the response time is therefore somewhat slow. Our second implementation was in C language, which is very well suited to our purposes, although it has no built-in capabilities like those of the GURU system.

References
6. Chu, P., "Robust interactive decision-analysis (RID)," Ph.D. Dissertation, Krannert Graduate School of Management, Purdue University, W. Lafayette, IN, December 1987.