ISMAUT Tools: A Software Tool Kit for Rational Tradeoffs Among Conflicting Objectives

Joseph G. D'Ambrosio

Abstract

One primary task of engineering design is resolving the conflicting objectives that are inherent in the design process. As a result, design automation tools must provide designers with the ability to represent and apply the preferential knowledge required to resolve conflicting objectives. This paper describes a decision-theoretic approach for resolving conflicting objectives and a software took kit, ISMAUT Tools, that enables access to this approach. ISMAUT Tools includes a library of primitive routines that can be linked into other applications, and a text-based interface providing stand-alone decision making capabilities.

1.0 Introduction

One primary task of engineering design is resolution of the conflicting objectives that are inherent in the design process. These conflicts arise due to the physical relations among objectives and resource limitations. Improvements in one objective are often only achieved at the expense of other objectives. For example, the objective to minimize cost is usually at odds with the objective to maximize performance. Engineering design tools that automate portions of the design process must provide mechanisms to capture the preferential knowledge required to make the necessary tradeoffs.

The most common approach to solving this problem is to represent preferences with a weighted evaluation function. For each objective, one or more measurable attributes are identified that characterize the objective. For example, in the design of a microprocessor-based system, the objective to improve performance might be characterized by the MIPS rating of the processor. A weighted evaluation function combines attribute levels into a single objective function by using a weighted sum of attribute levels:

$$z = \sum_k w_k r_k$$

where r_k is the level of attribute k and w_k is a weight describing the relative importance of the attribute versus the others. MBESDSD (Wu et al., 1990) and MICON (Birmingham et al., 1992) are two examples of electronic system design tools that apply a weighted evaluation function to resolve conflicting objectives. Goal programming (Murty, 1983), a popular mathematical programming technique, is also based on a weighted evaluation function.

The main limitation of a weighted evaluation function is the difficulty in correlating weight levels to the solutions produced. Without a clear understanding of the details of a design tool, it is difficult for a designer to predict the results for a given weight assignment. In addition, design results may be sensitive to small changes in weight values, making it difficult for a designer to establish a correlation between results and weight assignments. Thus, the process for assigning levels to weights does not guarantee the optimal design alternative will be identified.

Formal models of preferences (Thurston, 1991; Sykes and White, 1991) have shown promise in resolving conflicting design objectives. A formal model, e.g., a value (utility) function, in contrast to *ad hoc* techniques, provides a well-defined basis for predicting the quality of the results achieved. This property is highly desirable, since many engineering-design problems are so large that the desirability of the solution produced is difficult, if not impossible, to determine.

Thurston (1991) demonstrates the application of multiattribute utility theory for selecting the best design alternative. The utility of each design alternative is evaluated, and the one that maximize utility is chosen. Evaluation based on utility analysis requires detailed quantitative analysis, which may be difficult if a design problem has a high degree of imprecision, such as during the early stages of the design process. Wood *et al.* (1990) note the difficulties involved in using utility analysis when most design information is incomplete, and Wood & Antonsson (1989) describe an alternative approach based on fuzzy analysis, which allows evaluation based on qualitative information. Thurston and Carnahan (1992) counter this argument, stating that utility analysis is superior in all but the earliest design stages.

In this paper, I describe a software tool kit, ISMAUT Tools, which is based on imprecise multiattribute utility theory. In addition to the benefits of a formal model, imprecise utility theory facilitates decision making even when given incomplete information about the problem to be solved. This is important since it is impossible to enumerate all feasible alternatives for most complex design problem.

In section 2.0 of this report I provide an overview of multiattribute utility theory. Section 3.0 describes ISMAUT Tools, and Section 4.0 presents test results. I conclude with a summary and discussion in Section 5.0.

2.0 Multiattribute Utility Theory

Many design problems are characterized by multiple conflicting attributes, and in many cases a multiattribute value function can represent the preference structure needed to solve the problem. Two conditions must be met for a multiattribute value function to exist. The first condition is a monotonicity condition, which states if the value of one attribute improves while there is no loss in value for other attributes, preference must increase. The second is a continuity condition, which states if a_i £ a_j £ a_k[1], then there must be a unique point where the decision maker is indifferent between the increase from a_k to a_j and the increase from a_j to a_i.

One drawback of representing preferences by a multiattribute value function is the amount of work required to construct the function, which is due to the multi-dimensionality of the problem. In many cases, the amount of work required can be reduced by decomposing the value function into subsets of attributes that are independent of the others. Krantz et al. (1971) show that if each attribute is *preferentially independent* of its complement, the value function can be decomposed such that $v = f(v_1, v_2, \ldots v_k)$, where v_k is the attribute value function for attribute k. An attribute k is preferentially independent if the weak order specified by v_k is independent of the level of other attributes. A physical relationship may exist between preferentially independent attributes; it is only required that the preference order for attribute levels be independent of the others. In the case of all attributes being preferentially independent, attribute value functions can be constructed independently of other attributes, resulting in a significant reduction in the work required to construct the overall function.

The most desirable form of a decomposable value function is an additive value function. An additive value function is a weighted sum of attribute values, such that the value of an alternative with k attributes is given by:

$$v = \sum_k w_k \, v_k(r_k) \qquad \textbf{(1)}$$

where w_k is the tradeoff weight for attribute k, and $v_k(r_k)$ is the value produced by the value function of attribute k. Keeney and Raiffa (1976) describe the process for determining the attribute value functions and tradeoff

---

[1] a_i £ a_j means that a_i is preferred to a_j.

weights. This form of the multiattribute value function requires that the attributes be *mutually, preferentially independent.* A set of attributes is mutually, preferentially independent if every subset of these attributes is preferentially independent of its complementary set. It can also be shown that in general if each pair of attributes is preferentially independent of its complement, then Eqn. 1 is valid (Gorman, 1968). Unlike standard value functions, the additive value function is unique up to a positive linear transform.

2.1 Imprecise Value Functions

In addition to decomposing a multi-attribute value function, the amount of quantitative analysis required to construct the function can be reduced by using an imprecise value function. Imprecisely Specified Multi-Attribute Utility Theory (ISMAUT) (White et al., 1984) creates a partial order based on preference relationships among a subset of alternatives. ISMAUT uses a weighted sum of attribute values. Thus, the value of alternative a_i is given by Eqn. 1. A preference statement of the form a_i £ a_j implies an inequality in the space of possible weights according to the following relation:

$$\sum_k w_k \left[v_k\left(r_{ik}\right) - v_k\left(r_{jk}\right) \right] \geq 0 \qquad \textbf{(2)}$$

According to this interpretation, the statement that a_i is preferred to a_j means that the tradeoff weights are such that the total weighted value of a_i is at least as great as that of a_j. Direct pair-wise preferences among attributes also translate to inequalities, such as:

$$w_{power} \geq w_{dollars}$$

Furthermore, all weights must be positive and their sum must be unity:

$$\forall k,\ w_k \geq 0 \qquad \textbf{(3)}$$

$$\sum_k w_k = 1 \qquad \textbf{(4)}$$

All these inequalities confine the weight space to a subspace, *W*', that satisfies the inequalities. Thus, from pair-wise preference statements, ISMAUT determines ranges of attribute weights consistent with designer's preferences.

The imprecise value function *v* can order pairs other than those specified by the designer: a_i is preferred to a_j if, for every possible vector of weights $<w_1, w_2, ..., w_k>$ within *W*', the value of a_i is greater than the value of a_j, *i.e.*,

$$\text{Min} \sum_k w_k \left[v_k\left(r_{ik}\right) - v_k\left(r_{jk}\right) \right] \geq 0, \qquad w_k \in W'$$

This relation can be tested for every pair of alternatives that the designer has not already stated a preference. Thus, the preferences specified by a designer create a partial order over *all* design alternatives, and this partial order can identify the nondominated set of design alternatives. The following example illustrates how implicit preference relations can be derived.

Alternative	$v_1(r_1)$	$v_2\ (r_2)$	$v_3(r_3)$	**Preference Order**
$\mathbf{a_1}$	0.75	1.0	0.4	2
$\mathbf{a_2}$	0.5	0.0	0.8	1 (highest)
$\mathbf{a_3}$	0.0	1.0	1.0	4 (lowest)
$\mathbf{a_4}$	1.0	0.0	0.0	3

Table 1: Feasible alternatives for an example design problem.

Assume that a_1 and a_2 are a random sample of feasible alternatives (see Table 1). Since Table 1 specifies a_2 is preferred to a_1, $v(a_2) - v(a_1) \geq 0$[2]. Substituting into Eqn 2,

$$[v_1(a_2) - v_1(a_1)]\ w_1 + [v_2(a_2) - v_2(a_1)]\ w_2 + [v_3(a_2) - v_3(a_1)]\ w_3 \geq 0$$

and using the attribute values given in Table 1 yields:

$$[0.5 - 0.75]\ w_1 + [0.0 - 1.0]\ w_2 + [0.8 - 0.4]\ w_3 \geq 0$$

$$-0.25\ w_1 + -1.0\ w_2 + 0.4\ w_3 \geq 0$$

This constraint and the constraints given by Eqn. 3 and Eqn. 4 define the imprecise value function.

From the imprecise value function, it can be determined that a_2 is preferred to a_4. For a_2 to be preferred to a_4, it must be true that:

$$\text{Min } v(a_2) - v(a_4) \geq 0$$

Solving the following linear program gives a positive value for Min $v(a_2) - v(a_4)$:

$$\begin{aligned}
&\text{Minimize} \quad z = v(a_2) - \mathrm{v}(a_4) \\
&\text{Subject to:} \\
&\quad -0.25\ w_1 + -1.0\ w_2 + 0.4\ w_3 \geq 0 \\
&\quad \forall k,\ w_k \geq 0 \\
&\quad \sum_k w_k = 1
\end{aligned}$$

Thus, it can be concluded that a_2 is preferred to a_4 without a direct preference statement. ISMAUT provides the means to partially order a set of alternatives using a small set of preference statements, and thus identify a set of nondominated alternatives guaranteed to contain the optimal one.

---

[2] $v(a_i)$ is the value of alternative i.

3.0 ISMAUT Tools

I have developed a software package, ISMAUT Tools, to support decision making based on imprecise value functions. ISMAUT Tools provides a library of routines for defining attributes, alternatives, and preferences, and for performing dominance checks between alternatives. In addition to the library, a text-based interface has been developed. The text-based interface allows ISMAUT Tools to be used in isolation as a decision support tool, and the library of routines provides other applications the ability to perform dominance checks based on an imprecise value function.

In this section, I will describe a basic decision-making algorithm that can be implemented through the interaction between a decision maker and the text-based interface, or through the interaction of design automation tool and the ISMAUT Tools library.

3.1 A Basic Decision-Making Algorithm

Figure 1 specifies a basic decision-making algorithm supported by ISMAUT Tools. The first step is to define the attributes that will be used to evaluate the decision alternatives. Defining an attribute requires specification of a name (e.g., cost or power), and stating whether it is a more-is-better or less-is-better attribute. Next, a list of decision alternatives must be defined. Each decision alternative is described by a name and list of attribute levels. Table 2 provides a sample list of alternatives.

```
Basic decision maker() {
   Define attributes;
   Define list of alternatives;
   Identify Pareto-optimal set;
   Define attribute value functions
   Calculate attribute values;
   Specify preferences among a subset of alternatives;
   Identify nondominated set;
   Return (nondominated set);
}
```

Figure 1: A basic decision-making algorithm

Name	Cost ($)	Speed (MHz)	Performance (Relative to PPC 604)	Power (Watts)
PPC 604	550	130	1.0	10
Intel P6	1000	133	1.1	14
68040	100	40	0.2	6
MegaWaste 860	1100	133	1.05	14

Table 2: Example list of alternatives with attribute levels.

Once the alternatives are defined, the Pareto-optimal set can be identified. A design alternative a_i is Pareto-optimal if there does not exist another alternative a_j with all attribute levels equal to or better than that of a_i.

Reducing the set of alternatives to the Pareto-optimal set is desirable since this polynomial-time operation can eliminate dominated alternatives, thus reducing the number of linear programs that ISMAUT Tools must solve to identify the nondominated set of alternatives. ISMAUT Tools applies a simple pair-wise comparison algorithm ($\Theta(n^2k)$, where n is the number of alternatives and k is the number of attributes) to identify the Pareto-optimal set. Note that for the set of alternatives given in Table 2, the last alternative would be eliminated since all attribute levels of the Intel P6 are better or equal: the Intel P6 cost less, has the same speed and power consumption, and performs better.

At this point attribute values (utilities) would be found for each attribute of each remaining alternative. To accomplish this, attribute value functions must be defined for each attribute. ISMAUT Tools supports both linear and piecewise linear attribute value functions. A linear attribute value function maps an attribute into a *more-is-better* real number, ranging from 0.0 to 1.0, where 0.0 is the worst value and 1.0 is the best. For example, the cost (cost is a *less-is-better* attribute) of $100 has the value of 1.0, while a cost of $1000 has value 0.0 (see Figure 2). Table 3 shows the attribute values for the Pareto-optimal set of alternatives. A piecewise linear function also maps an attribute into a more-is-better real number between 0.0 and 1.0, but it can also model effects such as diminishing returns (Thurston, 1991).

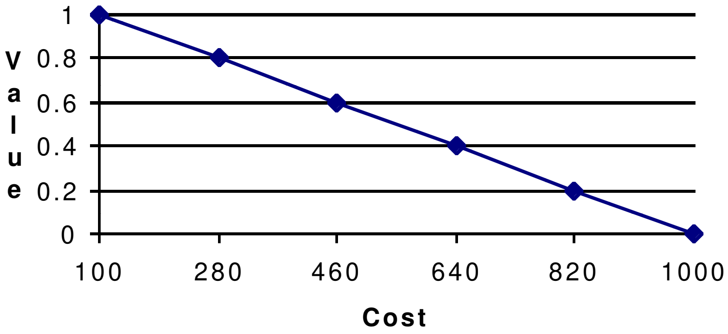

Figure 2: Attribute value function for cost.

Name	v_{cost}	v_{speed}	$v_{Performance}$	v_{Power}
PPC 604	0.5	0.97	0.89	0.5
Intel P6	0.0	1.0	1.0	0.0
68040	1.0	0.0	0.0	1.0

Table 3: Pareto-optimal set with attribute values.

At this point, pair-wise preferences can be specified among decision alternatives. For example, assume that for the alternatives given in Table 3, Intel P6 £ PPC 604. Given this pair-wise preference, ISMAUT Tools dominance checks can identify that Intel P6 £ 68040, and thus Intel P6 is the best choice. Performing dominance checks requires execution of $\Theta(n^2)$ linear programs, where n is the number of alternatives. Typically, ISMAUT Tools can solve several linear programs within one second on a Unix workstation.

4.0 Current Results

In this section I review test results related to the incorporation of ISMAUT Tools into two design tools. By linking in ISMAUT Tools, both design programs can evaluate designs using an imprecise value function. To perform an evaluation, these design tools must convert alternatives from internal representations into ISMAUT Tools data structures, execute ISMAUT Tools dominance checks, and convert the returned ISMAUT Tools data structures back into an internal representation. As describe in the following, test results indicate that the overhead introduced by ISMAUT Tools is offset by the quality of results produced.

4.1 DesignMaker

DesignMaker is a design optimization tool for exploring design topologies (architectures) and component selections. Given a set of functions (e.g., CPU, RAM, and ROM) to implement, DesignMaker explores combinations of components from a library to find the subset of components that results in the optimal design. DesignMaker is based on a combinatorial optimization algorithm, similar to those discussed by Haworth and Birmingham (1993) and D'Ambrosio and Birmingham (1995), and has been applied to several design domains, including hardware/software codesign (D'Ambrosio and Hu, 1994).

Design evaluation in DesignMaker may be based on one of two methods: Pareto optimality or imprecise value function (ISMAUT Tools). When evaluation is based on Pareto optimality, DesignMaker identifies the Pareto-optimal set of designs, which is guaranteed to contain the true global optimal design. When evaluation is based on an imprecise value function, DesignMaker identifies the set of nondominated designs based on the revealed preferences of a designer. The nondominated set is a subset of the Pareto-optimal set, and is also guaranteed to contain the global optimal design.

To evaluate the performance of an imprecise value function against generation of the Pareto-optimal set, a suite of test cases based on a computer design problem was developed. The function specification for each test case is identical, but the number of parts (domain size) available to implement the functions ranges from 29 to 463. The imprecise value function was created from linear attribute value functions and ranking a random sample of 10 feasible alternatives. The only difference among runs is how evaluations were performed (Pareto preference or imprecise value function). The results are given in Table 4.

One important finding from this suite of tests is that for problems of any reasonable size, solving the problem with an imprecise value function requires less time to find a solution set. Each dominance check based on designer's preferences requires the execution of a linear program. For trivial problems, the extra time required to solve the linear programs may result in DesignMaker executing longer. For problems of any significance, the extra pruning capability provided by the imprecise value function dramatically reduces execution time.

A second finding is that the size of solution set is drastically reduced by evaluation based on an imprecise value function. For the test case J90, Pareto preference produces 3495 Pareto-optimal solutions, while the imprecise value function produces only 4. DesignMaker could not solve any of the Pareto-optimal cases with more than 90 parts, since the memory required to maintain all of the solutions was too large.

	J29	**J63**	**J77**	**J90**	**J109**	**J194**	**J465**
# Feasible Alternatives	10^3	10^{11}	10^{12}	10^{14}	10^{16}	10^{21}	10^{28}
Pareto Preference							
# Solutions	28	379	1698	3495	NA	NA	NA
Time (sec.)	0.7	23.4	310.6	1303.4	NA	NA	NA
Imprecise Value Function							
# Solutions	2	2	4	4	4	24	28
Time (sec.)	1.2	2.8	4.4	5.1	5.5	39.3	55.2

Table 4: Comparison of dominance checks based on Pareto optimality and ISMAUT Tools.

4.2 Evolutionary Codesign (EvoC)

Similar to DesignMaker, EvoC (D'Ambrosio *et al*., 1995) is another design optimization program for identifying a subset of components to implement a given set of functions. However, EvoC is based on a genetic/evolutionary algorithm, which finds good solutions for problems that cannot be solved optimally. During each generation (iteration), genetic/evolutionary algorithms use a fitness function to evaluate alternatives and identify which alternatives will survive for the next generation. Instead of implementing the fitness function as a weighted evaluation, as is common practice, EvoC accesses the ISMAUT Tools library to evaluate alternatives based on an imprecise value function.

Although a detailed evaluation of the impact of applying an imprecise value function has not been performed, test results do indicate that incorporation of ISMAUT tools improves the quality solutions identified. This implies that for a given iteration, the extra time required to determine the fitness of an alternative is offset by the quality of alternatives that survive for the next iteration.

5.0 Summary and Discussion

In this report, I have described a prototype decision support package, ISMAUT Tools. ISMAUT Tools provides a library of routines which can be linked into engineering design automation programs to facilitate decision-theoretic design evaluation. In addition to the library of routines, ISMAUT Tools includes a text-based interface, allowing ISMAUT Tools to be used as a stand-alone decision-making tool. By providing a strong theoretical foundation for design evaluation, complex problems can be solved in a rational manner, thus ensuring the best (optimal) design alternatives will be identified. Design evaluation is based on Imprecisely Specified Multi-Attribute Utility Theory (ISMAUT), which in contrast to standard utility theory, does not require complete enumeration of all design alternatives. This is important since most design problems are too complex to allow complete enumeration of all feasible alternatives.

At this point, ISMAUT Tools has been integrated into two design tools. Test results have shown improved performance when design evaluation is based on ISMAUT. The overhead introduced by performing evaluations using ISMAUT Tools is acceptable for all but the smallest design problems.

Acknowledgments

I would like to give special thanks to Dr. Xiaobo (Sharon) Hu and Dr. Garrison W. Greenwood of Western Michigan University for integrating ISMAUT Tools into the EvoC program. They performed all necessary code development and ran all experiments.

References

Birmingham W.P., Gupta A.P., and Siewiorek D.P. (1992). *Automating the Design of Computer Systems*, Jones and Bartlett, Boston.

D'Ambrosio, J.G., and Birmingham, W.P. (1995). Preference-directed design, *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AI EDAM)*, Vol. 9, pp. 219-230.

D'Ambrosio, J.G., and Hu, X. (1994). Configuration-level hardware/software partitioning for real-time embedded systems, Presented at *The Third International Workshop on Hardware-Software Codesign*, Sept.

D'Ambrosio, J.G., Hu, X., and Greenwood, G. (1995). An evolutionary approach to configuration-level hardware/software partitioning, *GM Research and Development Center Publication R&D*-8465, Dec.

Gorman, W.M. (1968). The structure of utility functions, *Review of Economic Studies*, Vol. 35, pp. 367-390.

Haworth, M.S., and Birmingham, W.P. (1993). Towards optimal system-level design, *Proceedings of 30th Design Automation Conference*, June, pp. 434-438.

Keeney, R.L., and Raiffa, H. (1976). *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, Wiley and Sons, New York.

Krantz, D.H., Luce, R.D., Suppes, P., and Tversky, A. (1971). *Foundations of Measurement*, Academic, New York, Vol. 1.

Murty, K. G. (1983). *Linear Programming*. John Wiley & Sons, New York.

Sykes, E.A., and White, C.C. (1991). Multiobjective intelligent computer-aided design, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 6, November/December, pp. 1498-1511.

Thurston, D.L. (1991). A formal method for subjective design evaluation with multiple attributes, *Research in Engineering Design*, Vol. 3, pp. 105-122.

Thurston, D.L., and Carnahan, J.V. (1992). Fuzzy ratings and utility analysis in preliminary design evaluation of multiple attributes. *Journal of Mechanical Design* 114, Dec., pp. 648-658.

White, C.C., *et al.* (1984). A model of multi-attribute decision making and trade-off weight determination under uncertainty, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-14, No. 2, March/April, pp. 223-229.

Wood, K.L., and Antonsson, E.K. (1989). Computations with imprecise parameters in engineering design: background and theory, *Journal of Mechanisms, Transmissions and Automation in Design*, Vol. 111, Dec., pp. 616-625.

Wood, K.L., Antonsson, E.K., and Beck, J.L. (1990). Representing imprecision in engineering design: comparing fuzzy analysis and probability calculus, *Research in Engineering Design*, Vol. 1., pp. 187-203.

Wu, J., *et al.* (1990). A model-based expert system for digital system design, *IEEE Design and Test of Computers*.