(1989)

Tcheng, D. K., Lambert, B. L., Lu, S., & Rendell, L. (1991). AIMS: An adaptive interactive modelling system for supporting engineering decision-making. In L. Birnbaum and G. Collins (Eds.), *Machine Learning: Proceedings of the eighth international workshop* (pp. 645-649). San Mateo, CA: Morgan Kaufmann.

AIMS: An Adaptive Interactive Modeling System for Supporting Engineering Decision Making

David K. Tcheng
Computer Science Dept.
University of Illinois
at Urbana-Champaign

Bruce L. Lambert
Speech Communication Dept.
University of Illinois
at Urbana-Champaign

Stephen C-Y. Lu
MIE and CS Dept.
University of Illinois
at Urbana-Champaign

Larry A. Rendell
Computer Science Dept.
University of Illinois
at Urbana-Champaign

Abstract

Engineering decision making is a two-phase process involving model formation and model utilization. Simulation models have been useful for decision support but, can be too slow for interactive applications. An integrated methodology is presented for inducing faster simulation models and using these models to support interactive design. In AIMS, both model formation and model utilization are controlled by a multiple objective optimizer, ISO. This knowledge processing framework is applied to combustion engine design.

1 Introduction

At present, there exist simulators for many domains. In engineering design, simulators are often mathematical models based on the physics of the process or product being simulated. Such systems have important advantages. They can accurately predict the performance of potential designs without the expense of building a prototype. However, they require many expert man-hours to construct. Also, they may function too slowly to support interactive decision making. A more fundamental problem with simulators is that they are built to "analyze" a given design. This is useful during the verification of a final design. Early in the design process, it would be more useful to have a tool to help "synthesize" candidate designs based on the design objectives.

The Adaptive Interactive Modeling System (AIMS) attempts to provide a tool which (1) reduces the amount of time required to construct simulation models by inducing them from a database of real-world examples, (2) increases the speed of existing simulators by using the original simulation model to generate examples and inducing a faster simulation model from these examples, and (3) uses the original or induced simulation model to find Pareto-Optimal designs which tradeoff the designers' competing objectives.

This paper is developed in three main sections. First the engineering design process is characterized as a multiple objective optimization problem. The second section describes the AIMS knowledge processing framework. Included is a discussion of the two main components of AIMS, the induction toolbox and the optimizer. Section three reports results of several experiments that have been conducted in the area of combustion engine design. The final section draws conclusions and sets out goals for the future.

2 Design as Optimization

The design engineer's task is best characterized as an optimization problem. The designer's goal is to search a space of design parameters to find a point that maximizes some criteria. Take the internal combustion engine, for example, as a target product for a given design team. Here, the decision space might include: stroke to bore ratio, ignition timing, fuel intake rate, compression ratio, engine speed, number of cylinders, and displacement. The objective space might consist of: horse power, fuel efficiency, reliability, ease of assembly, noise, weight, vibration, emission levels, and cost. The problem is solved when the designer discovers a point which best meets the competing objectives.

We want a decision support system that takes as input the specification of a decision space and a set of user objectives and outputs decisions that maximize the objectives. Two factors complicate this problem. First, the function that maps design parameters to any one objective may be complex, probabilistic, discontinuous, many-peaked, etc. Elsewhere, we have argued that such poorly behaved functions can be learned by using an integrated approach to induction [Tcheng, Lambert, & Lu, 1989; Tcheng, et al., 1989; Lu & Tcheng, 1990]. Second, the presence of multiple objectives complicates optimization. AIMS addresses both of these difficulties.

In creating a system to support this kind of complex decision making, the AI researcher must decide how to approach the implied multiple objective optimization problem. The literature on multiple objective decision making is diverse (see [Buchanan, 1986; Chankong & Haimes, 1983; Hwang, C. L., Paidy, S. R., Yoon, K. & Masud, 1980; Hwang & Masud, 1979]), and we make

no attempt at a systematic survey here. Instead, we enumerate two of the most common approaches, briefly touching on the advantages and disadvantages of each.

2.1 Using Single Objective Optimizers

Perhaps the easiest way to deal with the complexity introduced by multiple objectives is to weight each objective in proportion to its importance and collapse them into a single super-objective function as in Equation (1):

(1) Os= w1O1 + w2O2 + w3O3 + . . . + wnOn

where Os is the super-objective function, the wi's are weights, and the Oi's are objective functions whose domains are defined over the decision space. By reducing the dimensionality of the problem to a single, scalar-valued global criterion of optimality, the analyst can proceed to apply any of a variety of well understood single objective optimization methods (e.g., hill-climbing, genetic search, simulated annealing, response surface fitting, etc.) to get a solution. This is a powerful advantage that has led many AI researchers to choose this approach to multiple objective decision making.

However, Zadeh [1963] has argued that early in the decision making process people do not typically know the trade-offs between objectives well enough to construct a good a priori weighting for the super-objective function. For instance, it might be the case that for an insignificant increase in cost per engine, fuel efficiency could be improved significantly. If cost of materials is weighted too highly in proportion to fuel efficiency, such a decision would never be returned by the optimizer. In short, the bias introduced by a certain set of weights may effectively hide decisions that the decision maker would have found most preferable if the true relationship between the competing objectives were known.

2.2 Using Multiple Objective Optimizers

Multiple objective optimizers avoid this weighting problem by producing a set of Pareto optimal (i.e., Pareto optimal, efficient, non-inferior) solutions. This set of solutions contains all optimal solutions with respect to any weighting of the objectives. Rigorous mathematical definitions of the Pareto optimal solution set are available in the literature. An intuitive sketch of the concept is provided below.

For a given multiple objective problem, a decision is Pareto optimal if and only if there exists no other valid decision that improves one objective score without degrading any other objective scores [Chankong & Haimes, 1983]. Take combustion engine design for example. For any Pareto optimal point, there may exist a point with higher horsepower or a point with higher fuel efficiency, but there are no points with both higher horsepower and higher fuel efficiency.

In this section, we have argued that a multiple objective design problems is best solved by using a multiple objective optimizer instead of collapsing all objectives into a single super-objective function. The next section describes the AIMS architecture, our attempt to combine an integrated approach to induction with optimization methods that deal with the full complexity of multiple objectives.

3 The AIMS Architecture

The Adaptive Interactive Modeling System (AIMS) is a prototype knowledge processing framework designed to support engineering decision making in the realm of product and process design (see Figure 1). The system has two main components: a toolbox of induction algorithms and a multiple objective optimizer. The induction toolbox (called CRL for the "competitive relation learner"), the optimizer (called ISO for the "induce and select optimizer"), and the interaction between the two systems have all been described and empirically evaluated elsewhere [Lu & Tcheng, 1990; Tcheng, Lambert, Lu, & Rendell, 1989; Tcheng, et al.,1989]. Only a general overview is provided here.

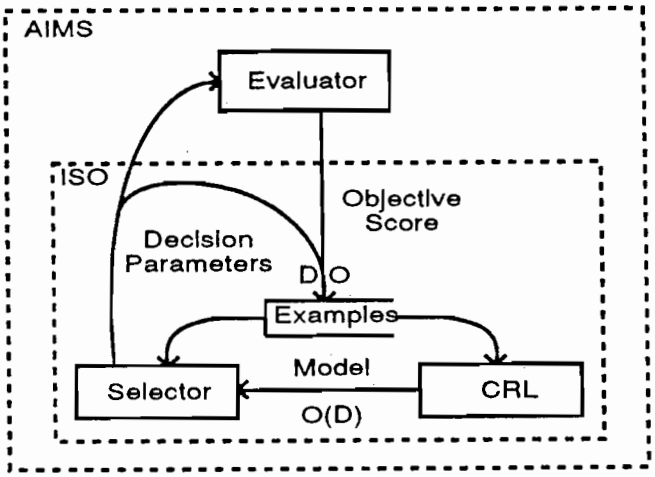

Figure 1: The AIMS knowledge processing framework.

3.1 CRL: The Induction Toolbox

Many induction algorithms exist, each designed to learn a particular kind of function. In both model formation and model utilization, one may encounter poorly behaved objective functions not easily learned by conventional techniques. CRL was designed to fit these poorly behaved functions. The basic algorithm is a generalization of traditional recursive splitting algorithms [Tcheng, Lambert, Lu, & Rendell, 1989; Tcheng, et al., 1989;]. Such algorithms solve problems by recursively splitting up input space and fitting each region with an appropriate model. Traditional recursive splitters (e.g., ID3 [Quinlan, 1986], PLS1 [Rendell, 1983]), and CART [Breiman, et al. 1984] use a single decomposition (splitting) and one learning (fitting) strategy for forming decision trees. CRL, in contrast,

manages the competition between multiple decomposition strategies and multiple learning strategies, and can produce models with hybrid representations. Currently implemented in CRL are several learning strategies (mean, mode, exemplar, neural net, and regression) and several decomposition strategies (distance-based, population-based, and arbitrary hyperplane).

With such a large space of alternatives, selecting the best inductive bias is time consuming. Therefore, it is desirable to have a method for inductive bias selection in large bias spaces. Our solution has been to parameterize CRL's bias space and use multiple objective optimization techniques to search it. The next section describes ISO, the optimizer used to accomplish this task.

3.2 ISO: A Multiple Objective Optimizer

Optimization is a process of selecting the best decisions from a pre-defined space of alternatives. Input to a multiple objective optimizer is a decision space and a set of objective functions. The output of the optimizer is a set of Pareto optimal decisions. Optimization in ISO is an iterative process involving three steps: selection, evaluation, and induction (see Figure 1). The process begins with the random selection of the first decision (D). The decision is evaluated by applying the objective functions to candidate decision. The decision is associated with its objective scores (O) and the resulting input/output pair is added to the database of examples available to the inducer. The inducer, CRL, takes these examples as input and outputs a model mapping decisions to objectives. In subsequent iterations, selection is influenced by two competing considerations: novelty and performance. Novelty biases selection towards unexplored regions of decision space. Performance biases selection towards regions of decision space deemed optimal by the current induced model. Obviously the weighting between novelty and performance is an important bias to ISO. In the following experimental results, both performance and novelty were weighted equally.

4 Experimental Results

In this section, we focus on using AIMS to support the design of internal combustion engines. The experiments can be divided into two broad classes: those dealing with model formation and those dealing with model utilization. In the model formation phase, we demonstrate how AIMS can produce a Pareto optimal set of models with respect to multiple model formation objectives. In the model utilization phase, we show how AIMS can be used to generate a Pareto optimal set of design decisions with respect to multiple engine design objectives. Before we go into the details of model formation and utilization, we describe the physics-based simulator and how it was used to construct examples for learning.

4.1 The Simulator

An analytical simulator for a turbo-compounded diesel engine [Assanis & Heywood 1986] is used in these experiments. Although the simulator has been verified to give accurate predictions of engine performance, it is quite slow and takes about 150-200 CPU seconds for each simulation on a DECstation 3100 (with 24 MB of physical memory). This evaluation speed is unacceptable for real time analysis and synthesis.

4.2 Example Generation

The decision space in this experiment had 7 dimensions and consisted of stroke to bore ratio (STBRAT), injection timing (TINJ), fuel mass injected per cycle (FMIN), compression ratio (CR), engine speed (SPEED), number of cylinders (NCYL), and volumetric capacity (VOL). The performance of each engine design was measured by the simulator in terms of two performance variables brake horse power (BHP) and brake specific fuel consumption (BSFC). The dimensions of the decision space were chosen by the simulator designers and cover a range of possible engine designs and are shown in Table 1. These ranges were used to generate 1000 examples by randomly selecting design within this space based on a uniform distribution. The ranges for the performance parameters are estimated by finding the minimum an maximum performance values over all the examples and are shown in Table 2.

Table 1: The decision parameters.

Parameter	Units	Minimum	Maximum
STBRAT		0.8	1.2
TINJ	degrees	320	335
FMIN	g/cycle	0.1	0.4
CR		13	17
SPEED	rpm	1000	2400
NCYL		2	8
VOL	liters	5	15

Table 2: The performance parameters.

Parameter	Units	Minimum	Maximum
BHP	KW	5	615
BSFC	g/KW-hr	187	5904

4.3 Model Formation

For this experiment, we compared three different induction algorithms implemented by CRL: simple recursive splitting (SRS), linear recursive splitting (LRS) and backpropagation (BPG). SRS is similar to PLS [Rendell, 1983] and CART [Breiman, et al. 1984]. LRS is like SRS but uses linear regression to form function within each input space region created. Utgoff's perceptron tree learning algorithm [Utgoff, 1988] is similar

to LRS but uses a perceptron algorithm at the nodes instead of linear regression. BPG is an extension of the backpropagation algorithm given in [McClelland & Rumelhart, 1987].

Each algorithm compared had one or more control parameters which were optimized by ISO. The model formation objectives were predictive accuracy, model formation time, and model evaluation time. To evaluate an inductive bias, 10 model formation problems were formed by randomly selecting n training examples (n = 20, 40, 80, 160, or 320) and 320 testing examples from the original set of 1000 examples. The standard error of each model produced was measured and all errors were averaged to calculate the predictive accuracy which was defined as (1 - (average standard error / range))*100. Model formation time and evaluation time were measured in terms of seconds and milli-seconds respectively.

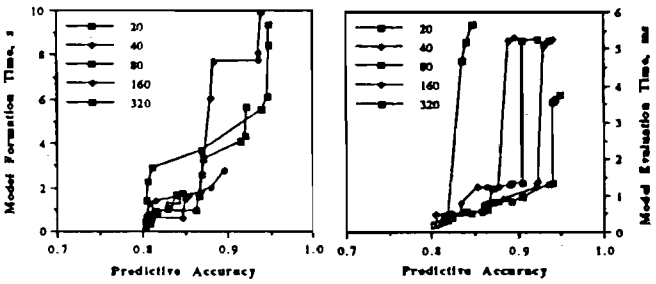

Figure 2: Accuracy vs. model formation and evaluation time by example set size.

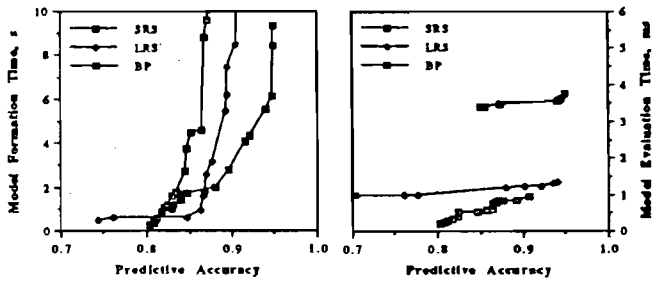

Figure 3: Accuracy vs. model formation and evaluation time by algorithm.

Figure 2 shows how the size of the dominated region increases as more examples are used to form the models. In the case of model formation time (2a) fewer examples should be used if faster formation times are required but, in the case of model evaluation time (2b), more examples are a always better. Figure 3 shows the Pareto optimal biases for each induction algorithm compared. From these Pareto optimal sets, AIMS can now select the best inductive bias based on a model formation or model evaluation time constraints.

4.4 Model Utilization

In this phase, the engine designer uses models produced by AIMS in conjunction with ISO. For this experiment, 800 training examples were used for model formation which resulted in BHP and BSFC models with an accuracy of 96% and 97% respectively. These models were then used by ISO to construct a Pareto optimal set of engine designs with respect to BHP and BSFC. Figure 4a shows how these two objectives trade off. More power can only be attained at the expense of fuel efficiency and at about 550 KW, the tradeoff becomes severe. Figure 4b shows how individual engine design parameters such as STBRAT and TIMING relate to BHP for Pareto optimal designs. If more power is required, both STBRAT and TIMING must be increased.

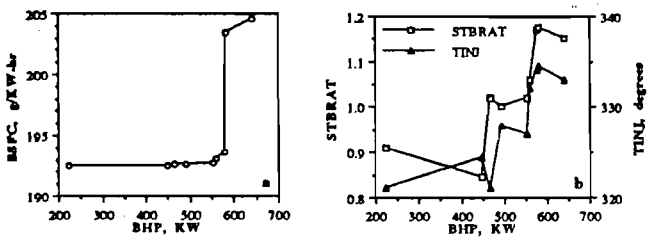

Figure 4: Tradeoffs between the design objectives BHP and BSFC, and the relationship between BHP and design parameters STBRAT and TINJ.

Table 3: Final decision parameters.

Parameter	Units	Value
STBRAT		1.17
TINJ	degrees	332
FMIN	g/cycle	0.217
CR		14.32
SPEED	rpm	2352
NCYL		6
VOL	liters	12.67

Table 4: Final performance parameters.

Parameter	Units	Predicted	Actual
BHP	KW	475.2	454.5
BSFC	g/KW/-hr	192.5	202.5

Typically, the engine designer iteratively converges on an engine design by using AIMS to discover relationships between the design objectives and Pareto-optimal engine designs, restricting decision space or change the objectives, and repeating the process until a single point in decision space has been chosen. This point is then verified with the original engine simulator. Tables 3 and 4 illustrate an example of a design constructed with this procedure. For this problem the designer wanted an engine with a power of about 450 KW with the least number of cylinders. Note that

the predicted performance of the induced model was optimistic, but within the estimated error the models. For better predictive accuracy, more examples must be used and, based on our initial experiments, we think the methodology will scale up nicely. For more details concerning applications of AIMS to engine design see [Lu, et al. 1991].

5 Conclusions and Future Research

The dynamic complexity of engineering decision making puts enormous demands on designers' knowledge. In many cases, relationships among design parameters are simply too complex to be understood without computational assistance. We have described a knowledge processing framework designed to support decision making in this sort of complex and changing environment. The AIMS framework combines machine learning and optimization techniques to transform knowledge embedded in physics-based simulators into more usable representations. We have argued for a two phase approach to knowledge processing in this area. The model formation phase seeks to build a Pareto optimal set of models. The model utilization phase then uses one of those models to generate a Pareto optimal set of decisions for the decision maker to choose from. We have continually emphasized the advantages of multiple objective optimization techniques, especially as a safeguard against incomplete knowledge of performance trade-offs.

Our main goal for the future is to complete and document a prototype system that can be distributed to our colleagues tackling real world design problems. Only through this sort of feedback can we validate and improve our ideas in the area where it matters most.

Acknowledgements

This research was sponsored in part by the National Science Foundation (DMC-8657116). Special thanks to Dennis Assanis for his insight into engine design and support in using the combustion engine simulator.

References

Assanis, D. N., & J. B. Heywood. (1986). Development and use of a computer simulation of the turbocompounded diesel system for engine performance and component heat transfer studies. SAE Technical Paper Series No. 860329. Reprinted from *The Adiabatic Engine: Global Developments* (pp. 95-120). Warrendale, PA: Society for Automotive Engineers.

Breiman, L., Friedman, J., Olshen, R.A., and Stone, C. J. (1984). *Classification and Regression Trees.* Wadsworth International Group, Belmont, CA.

Buchanan, T. (1986). Multiple objective mathematical programming: A review. *New Zealand Operational Research,* Vol. 14, No. 1, (pp. 1-27).

Chankong, V. & Haimes, Y. Y. (1983). Optimization-based methods for multiobjective decision- making: An overview. *Large Scale Systems*, Vol. 5, (pp. 1-33).

Hwang, C. L. & Masud, A. S. M. (1979). *Multiple objective decision making methods and applications: A state-of-the-art survey.* New York: Springer Verlag.

Hwang, C. L., Paidy, S. R., Yoon, K. & Masud, A. S. M. (1980). Mathematical programming with multiple objectives: A tutorial. *Computing and Operations Research*, Vol 7, (pp. 5- 31).

Lu, S. C-Y. & Tcheng, D. (1990). Building layered models to support engineering decision making: A machine learning approach. *Journal of Engineering for Industry*, ASME Transactions, Vol. 113, No. 1, (pp. 1-9).

Lu, S. C-Y., Yerramareddy, S., Tcheng, D. & Assanis, D. N. (1991). A machine learning approach to model formation and utilization for engineering design. Submitted to *IEEE Transactions Special Issue on Machine Learning.*

McClelland, J. & Rumelhart, D. (1987). *Explorations in parallel distributed processing: A handbook of models, programs and exercises*, MIT Press, Cambridge, MA.

Quinlan, R. (1986). Induction of decision trees. *Machine Learning*, Vol. 1, No. 1, (pp. 81-106).

Rendell, L. (1983). A new-basis for state-space learning systems and a successful implementation. *Artificial Intelligence*, Vol. 20, No. 4, (pp. 369-392).

Tcheng, D., Lambert, B., Lu, S. C-Y. & Rendell, L. (1989). Building robust learning systems by combining induction and optimization. In *Proceedings IJCAI 89*, San Mateo, CA: Morgan Kaufmann.

Tcheng, D., Lambert, B., & Lu, S. C-Y. (1989). Generalized recursive splitting algorithms for learning hybrid concepts. In *Proceedings Sixth Annual Workshop on Machine Learning.* San Mateo, CA: Morgan Kaufmann.

Utgoff, P. (1988), Perceptron trees: A case study in hybrid concept representation, in *Proceedings of the American Association for Artificial Intelligence*, Morgan Kaufmann, San Mateo, CA, pp. 601-606.

Zadeh, L. A. (1963), Optimality and Non-Scalar-Valued Performance Criteria, *IEEE Transactions on Automatic Control*, Vol. AC-8, pp. 59-60.