Learning Engineering Models with the Minimum Description Length Principle*

R. Bharat Rao and Stephen C-Y. Lu
Knowledge-Based Engineering Systems Laboratory
University of Illinois at Urbana-Champaign

Abstract

This paper discusses discovery of mathematical models from engineering data sets. KEDS, a *Knowledge-based Equation Discovery System*, identifies several potentially overlapping regions in the problem space, each associated with an equation of different complexity and accuracy. The *minimum description length* principle, together with the KEDS algorithm, is used to guide the partitioning of the problem space. The KEDS-MDL algorithm has been tested on discovering models for predicting the performance efficiencies of an internal combustion engine.

Introduction

It is well known that to achieve the objective of best classifying unseen data, it is not always best to construct a perfect model, which predicts every single data point in the training set without any error. Greater accuracy in the prediction of new data is often achieved by using an imperfect, smaller model [Breiman *et al.*, 1984], rather than one that may be over sensitive to statistical irregularities and noise in the training set. While cross-validation techniques may help to avoid over-fitting, they are fairly expensive. The *minimum description length* (MDL) principle [Rissanen, 1985] is an elegant and powerful theory, that balances model complexity with model error. In combination with KEDS, the *Knowledge-based Equation Discovery System* [Rao and Lu, 1992; Rao *et al.*, 1991], the KEDS-MDL algorithm is able to discover accurate and comprehensible models in engineering domains.

Engineering phenomena are often *non-homogeneous*, namely, the relationship between the domain variables varies across the problem space. However, many discovery techniques make the underlying assumption that the phenomenon being modeled is homogeneous, and are unsuitable for engineering problems. KEDS, in addition to being a model-driven empirical discovery system, can also be viewed as a *conceptual clustering* system, which partitions the data into regions based upon the mathematical relationships that it discovers between the domain variables. The intertwining of the discovery and partitioning phases enables KEDS to overcome many of the problems involved in learning relationships from engineering data.

The output of KEDS is a single region R_i, which is associated with a mathematical equation f_i that describes the behavior of the variables within the region. In earlier research [Rao and Lu, 1992], KEDS was invoked multiple times until all available resources were exhausted to produce a collection of overlapping regions. These regions were then combined to obtain a model that covered the entire problem space. This approach was very wasteful of resources. It is more efficient to run KEDS for a limited time, select a single region-equation pair from the available candidates, run KEDS on the remainder of the problem space, and so on. However, while a number of metrics can be used to select a (R_i, f_i) pair, many metrics (for example, accuracy or comprehensibility) are unsatisfactory because the result of evaluating a candidate (R_i, f_i) pair provides a local, rather than a global measure of how well the candidate models the entire data set. The chosen metric should be able to select between two (R_i, f_i) pairs, where the regions are of different sizes and cover distinct (potentially overlapping) regions within the problem space, and the equations have different complexity and accuracy. The MDL principle is an ideal metric for discriminating between alternate candidates. The selection of the next region in the problem space is made globally by choosing the candidate (from those created by running KEDS for a limited period of time) that minimizes the description length of the entire data set, rather than selecting a candidate that optimizes a local metric (for example, selecting the most accurate (R, f) pair). As empirically demonstrated in this paper, the models discovered by using MDL as an evaluation metric with KEDS, out-

*This research was partially supported by a research contract from the Digital Equipment Corporation. For further information please contact the first author at Beckman Institute for Advanced Science and Technology, 405 N. Mathews, Urbana, IL 61801, *bharat@cs.uiuc.edu*.

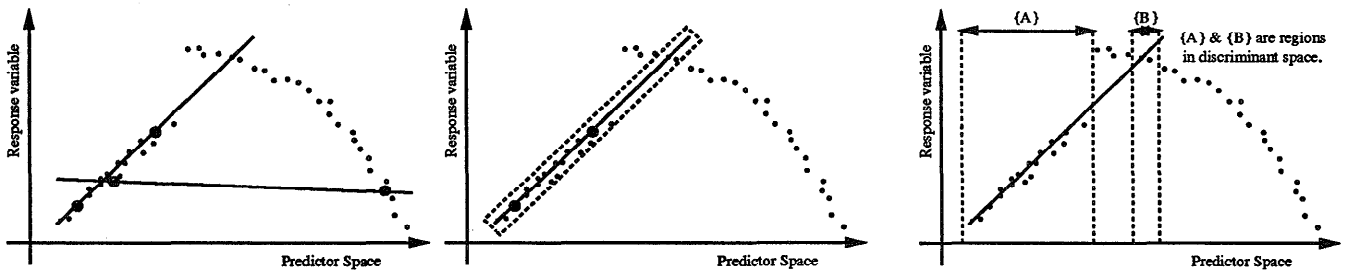

Figure 1: KEDS: (a) Sampling the data set, (b) Computing covers, & (c) Partitioning the problem space

perform those created by KEDS with other metrics. The KEDS-MDL algorithm presented in this paper is a multi-dimensional extension of the MDL surface reconstruction algorithm presented in [Pednault, 1989] (which applied to functions of a single variable).

A variety of tools have been applied to form engineering models (see [Finger and Dixon, 1989]). Statistical techniques like CART [Breiman *et al.*, 1984] and MARS [Friedman, 1991], and machine learning techniques like ID3 [Quinlan, 1986] and PLS [Rendell, 1983], may be characterized as split-and-fit systems, which first partition (split) the problem space and then successively model (fit) each region. KEDS is a fit-and-split system which builds up hypotheses in a bottom-up fashion and partitions the problem space to find the enclosing region. Techniques like those in [Kadie, 1990] can be used to combine multiple overlapping hypotheses. Some traditional empirical discovery systems [Langley *et al.*, 1987; Falkenhainer and Michalski, 1986] perform well when the equations that best describe the data have relatively small integer coefficients and the problem space does not have to be partitioned into several regions. AIMS (Adaptive and Interactive Modeling System [Lu and Tcheng, 1991]), integrates machine learning and optimization techniques to aid engineering decision making. Conceptual clustering programs [Stepp, 1984; Fisher, 1985; Cheeseman *et al.*, 1988] produce partitions superior in comprehensibility to those discovered via numerical clusterings. However, the mathematical relationships that exist between attributes of the events are not used to partition the events. The MDL principle has been used to infer decision trees [Quinlan and Rivest, 1989], for image processing [Pednault, 1989; Pentland, 1989; Leclerc, 1989], and for learning concepts from relational data [Derthick, 1991].

The KEDS System

Given : A training data set with n data points, $\mathbf{E} = \{e_1, e_2, \ldots, e_n\}$. Each data point e has a *response* (or dependent) attribute y, and P *predictor* (or independent) attributes, $\vec{x} = x_1, x_2, \ldots, x_P$. All of y and x_i are real-valued and continuous variables (although nominal variables are acceptable as well).

Goal : From $\mathbf{E}$, build a model $\mathcal{Q}$ that predicts the value of y from any given set of values of $\vec{x}$: $\hat{y}_i = \mathcal{Q}(\vec{x}_i)$.

In addition to the above data set, KEDS is provided with some generalized knowledge about the relationships that we expect to find in engineering domains. KEDS generates models of the form $\hat{y} = F(\vec{x})$, where F is a model chosen from a collection of parameterized polynomial models provided to KEDS. For the experiments in this paper we have considered five different families of parameterized models: $y = a$, $y = ax + b$, $y = ax^2 + bx + c$, $y = ax_1 + bx_2 + c$, and $y = ax_1 + bx_2 + cx_3 + d$, where the x's are the nominal parameters of F (the names of the real-valued predictor variables x_i that are assigned to F), and $a\,b\,\ldots$ are the real-valued parameters (coefficients discovered by KEDS). The equation templates provided to KEDS correspond to the basis functions used in statistical methods like linear regression. Since the domain can be non-homogeneous, the underlying function that generated the data is assumed to be of the form $\{\mathrm{R}_1 \Rightarrow y = F_1(x) + \mu(\sigma_1),\ \mathrm{R}_2 \Rightarrow y = F_2(x) + \mu(\sigma_2)\ \ldots\}$ where $\mu(\sigma)$ represents a 0-mean Gaussian process, and R_i is a region in predictor space.

The KEDS algorithm consists of two-phases: *discovery* and *partitioning*. The partitioning is model driven and is based upon the relationships that are discovered from the data, while the discovery process is restricted within the boundaries of the regions created by the partitioning. In the initial discovery phase, an equation template F is chosen and the data set $\mathbf{E}$ is sampled to determine the coefficients of a candidate equation, f. This is illustrated in Figure 1(a) for the template $y = ax + b$, where two separate candidate equations are created (based on two distinct instantiations of the initial discovery phase). Note that the horizontal axis in Figure 1 is *multi-dimensional*, representing the P-dimensional space of predictor variables.

In the partitioning phase, the deviation Δy_i between the actual and predicted value for each data point e_i, is converted into a positivity score via a *score* function. This is illustrated in Figure 1(b) for a very simple score function, where all data points with deviation $\Delta y_i \leq \epsilon$ are assumed to be positive examples of $f$with a score of 1, and all other data points are negative examples with a score of 0. The training data set $\mathbf{E}$ is partitioned over the space of predictor variables to isolate regions

1. For a template F with k unknowns, repeat Step 2 below, $N(F,k,\delta)$ (see Equation 1) times.
2. Initialize S to k random samples from the data set $\mathbf{E}$.
3. Discovery: determine the k coefficients of f from S.
4. Partitioning: (a) For all examples in $\mathbf{E}$, compute positivity-score$(f, e_i) = \exp(-(\Delta y_i)^2/2\sigma^2)$.
   (b) Partition $\mathbf{E}$ to return regions Rwith high score.
   (c) *for all* R_i (subject to m): set S to events $e \in R_i$.
5. Return to Step 3 and refine f, until f ceases to improve.

Figure 2: The KEDS algorithm

with a low error (i.e., high score), as in Figure 1(c). The score for a region is the average of the scores of the data points in the region. The discovery phase is then invoked in each region to *refine* the candidate equation. Since we are using polynomial equations, we can perform multi-variable regression over the events in the region to improve the values of the coefficients. The KEDS algorithm is summarized in Figure 2.

KEDS uses three parameters to control its search for formulae. The accuracy parameter, is a score function that converts the deviation into a positivity score. The score function creates a transformation of the data set, wherein the score associated with each data point measures the likelihood that the data point is correctly predicted by the candidate equation. The simple score function shown in Figure 1(b) was used in [Rao *et al.*, 1991]. The score function used here, $\exp(-(\Delta y_i)^2/2\sigma^2)$, is the normalized probability density function for a normal distribution (see Figure 2).

The size parameter, m, is the minimum fraction of the events in the data set that must be described by a equation. For the two regions A and B found in Figure 1(c), region B is rejected because it covers fewer events than permitted by m. The cover of the formula f is modified to (the set of events in) region A. Note that the regions A and B are hypercubes in the P-dimensional predictor space. The choice of PLS [Rendell, 1983] as the partitioning algorithm imposes the constraint that these regions be defined by conjunctive concepts over the discriminant attributes.

The confidence parameter, δ, controls the number of times $N(F)$, that the KEDS algorithm is called for a given template F. If all k data points are chosen from a region associated with an equation, experiments with synthetic data sets with added Gaussian noise have shown that KEDS quickly converges to the correct equation and region. KEDS find regions that cover m fraction of the events with a (high) probability $(1-\delta)$. If a given template F has k unknown coefficients, then it follows that the total number of times $N(F)$, that the KEDS algorithm needs to be instantiated for F (Step 1 in Figure 2), is:

$$N(F) \geq \frac{\log \delta}{\log(1-m^k)} \qquad (1)$$

Applying the MDL Principle

According to the Minimum Description Length principle [Rissanen, 1983], the theory that best accounts for a collection of observations $\mathbf{E} = \{e_1, e_2, \ldots, e_n\}$ is the one that yields the shortest description [Rissanen, 1985; Barron, 1984]. Therefore, the best theory to induce from a set of data will be the one that minimizes the sum of the length of the theory and the length of the data when encoded using the theory as a predictor for the data. Let $\mathcal{Q}$ be a model created from the region-equation pairs discovered by KEDS. Then the model that will be the best predictor of yet unseen data will be the model $\mathcal{Q}$, that minimizes the length:

$$\mathcal{L}(\mathcal{Q}, \mathbf{E}) = \mathcal{L}(\mathcal{Q}) + \mathcal{L}(\mathbf{E}|\mathcal{Q}) \qquad (2)$$

where $\mathcal{L}(\mathcal{Q})$ is the number of bits required to encode the model $\mathcal{Q}$, and $\mathcal{L}(\mathbf{E}|\mathcal{Q})$ is the number of bits needed to encode the *difference* between the training data set $\mathbf{E}$ and the values predicted for the events in $\mathbf{E}$ by $\mathcal{Q}$. The $\mathcal{L}(\mathcal{Q})$ term in Equation 2 corresponds to a complexity penalty (increasingly complex models will require a greater number of bits to encode) and the $\mathcal{L}(\mathbf{E}|\mathcal{Q})$ term corresponds to an error penalty. These two terms create a balance between modeling the data accurately and overfitting the data.

There are different encoding techniques that can be used to encode the total code length $\mathcal{L}(\mathcal{Q}, \mathbf{E})$. It is important that these coding techniques be efficient. An inefficient method of coding the model will penalize larger models too heavily, and will result in the selection of smaller models with large error. Similarly, an inefficient method for coding the difference term $\mathcal{L}(\mathbf{E}|\mathcal{Q})$, will result in complex, overly large models being selected.

Encoding the Model – $\mathcal{L}(\mathcal{Q})$ Consider the simple case where the model is composed of exactly one equation f (i.e., $\mathcal{Q} = f$). As we are working with a fixed family of models, such that the description of the family does not depend upon $\mathbf{E}$ or the parameters used in the models, $\mathcal{L}(f)$ is the cost of encoding the family F of f, plus the cost of encoding the values of the parameters in f. For the experiments in this paper, we have considered five families of models (Section 2). Therefore, the cost of encoding F is $\log 5$ bits (all logarithms in this paper are base 2). If F takes w predictor variables and all the P predictor variables can be assigned to F with equal probability, then encoding the nominal parameters of f costs $\log \binom{B}{w}$ bits. Each of the u real-valued coefficients of f requires $\frac{1}{2}\log n$ bits [Rissanen, 1983] to encode. An additional $\frac{1}{2}\log n$ bits are needed to encode σ^2, the value of the variance used later in Equation 5 to calculate the difference term.

$$\mathcal{L}(f) = \log 5 + \log \binom{B}{w} + \frac{(u+1)}{2}\log n \qquad (3)$$

As the domain is non-homogeneous, $\mathcal{Q}$ is a piecewise-continuous model that is a collection of

region-equation pairs. If $\mathcal{Q}$ is divided into r regions $[\mathrm{R}_1, \ldots \mathrm{R}_r]$ and a unique equation f_j is associated with each region R_j, then the cost of encoding $\mathcal{Q}$ is

$$\begin{aligned}\mathcal{L}(\mathcal{Q}) &= \mathcal{L}\left(\sum_{j=1}^{r}(\mathrm{R}_j, f_j)\right) = \lambda(r) + \sum_{j=1}^{r}[\mathcal{L}(\mathrm{R}_j, f_j)] \\ &= \lambda(r) + \sum_{j=1}^{r}[\mathcal{L}(\mathrm{R}_j) + \mathcal{L}(f_j)] \qquad (4)\end{aligned}$$

where $\lambda(r)$ is the number of bits require to encode the integer r. According to the Elias-Cover-Rissanen prior for integers, the number of bits needed to encode r is $\log *(r+1) + c$ [Rissanen, 1983]. However, this encoding is optimal only for relatively large integers, and is inefficient for the small number of regions that occur in $\mathcal{Q}$. It is more efficient to use a fixed number of bits to encode r. Note that when comparing two models, the $\lambda(r)$ term cancels out and can be dropped from Equation 4.

A region R is a hypercube in the space of the predictor variables $\vec{x}$, and is defined by a collection of I intervals on some, none, or all of the P predictor variables. Instead of using the Elias-Cover-Rissanen prior to encode I, it is more efficient (for small P) to use P bits to indicate the presence or absence of each of the predictor variables. An interval over the variable x_j can be written as $[\mathrm{lo}_j < x_j < \mathrm{hi}_j]$. While it is possible to encode both the interval end points lo_j and hi_j, we note that for an arbitrary interval either lo_j or hi_j could be the boundary value for variable x_j. We can take advantage of this and significantly reduce the code length for intervals that begin or end at a boundary value, while increasing it slightly for other intervals. To encode the three different possibilities (i.e., either lo_j or hi_j is a boundary value or neither is) costs $\log 3$ bits. If L is the number of bits to encode an interval end-point that is not a boundary value, an additional L or $2L$ bits are needed to encode the interval completely. Encoding the interval end-points with full precision costs $\log n$ bits, but this is obviously too costly as we only use $\frac{1}{2}\log n$ bits to encode the parameters for f. Encoding with reduced precision is used to get a smaller cost for L.

Encoding the Difference Term – $\mathcal{L}(\mathbf{E}\,|\mathcal{Q})$ Again, consider the case when $\mathcal{Q}$ is composed of a single function f. To encode the data points in $\mathbf{E}$, the difference between the actual value of the response variable y and the value $\hat{y}$ predicted by the model $\mathcal{Q}$ is analyzed statistically by viewing the difference as a random process. This random process along with the model $\mathcal{Q}$ induces a probability distribution p on the data points, where p is the *maximum likelihood* function. From the Shannon coding measure in information theory [Gallager, 1968], we can encode the data points using a number of bits equal to the negative log likelihood $(-\log p(\mathbf{E}))$. When $\mathcal{Q}$ is a collection of regions, Equation 5 (below) is applied to each region.

$$\mathcal{L}(\mathbf{E}\,|\mathcal{Q}) = \frac{n}{2}\log(2\pi\sigma^2) + \log e \sum_{e_i \in \mathbf{E}} \frac{\Delta y_i^2}{2\sigma^2} \qquad (5)$$

Encoding the Exceptions An individual (R,f) pair can be encoded by Equations 4 and 5. Our goal is to evaluate the entire model, and yet somehow select equations one at a time rather than having to evaluate the model only after it is completely formed. We do this by dividing $\mathcal{Q}$ into two models: Q_{eqn}, consisting of all the equations chosen in $\mathcal{Q}$, and Q_{exc}, consisting of all the exceptions in $\mathcal{Q}$ (i.e., the events in $\mathbf{E}$ that are not covered by the equations in Q_{eqn}). Q_{exc} is encoded by fitting the exceptions to an averaging model (the mean value of the exceptions). The code length of the averaging model is computed using Equation 4, while Equation 5 is used to calculate the difference term for the exceptions. The code length $\mathcal{L}(Q_{exc})$ is a measure of the randomness of the exceptions.

Estimating the Variance The nearest neighbor variance σ_{nn}^2, is calculated by using the nearest neighbor of a point in a normalized predictor variable space as the best estimate $\hat{y}_i$. The true variance of $\mathbf{E}$ is assumed to be $\sigma^2 = \sigma_{nn}^2/2$ [Stone, 1977; Cover, 1968]. Note that σ must be estimated separately for each region (the only assumption made is that the variance is constant within a region). Encoding the value of σ accounts for the extra $\frac{1}{2}\log n$ bits in Equation 3.

Once a region has been modeled by an equation f, the deviation from f itself can be used to calculate σ. Then the cost of encoding the difference term for a region (from Equation 5) reduces to $\mathcal{L}(\mathrm{R}_i|f_i) = n_i \log(\sqrt{2\pi e}\sigma)$. This seems counter-intuitive at first; for low values of σ, the code length $\mathcal{L}(\mathrm{R}_i|f_i)$ is negative. However, the σ^2 term in Equation 5 should actually be a σ^2/Γ^2 term, where Γ is the precision used in representing the data. Obviously $\Gamma < \sigma$ and $\log(\sigma/\Gamma)$ will always be positive. Note that when comparing two candidate equations, the $\log(1/\gamma)$ term cancels out, and therefore, can be ignored. The value for the code length $\mathcal{L}(\mathbf{E}\,|\mathcal{Q})$ is simply a relative rather than an absolute value.

The KEDS-MDL Algorithm The KEDS-MDL algorithm (summarized in Figure 3) calls KEDS iteratively and uses MDL as an evaluation metric to select the best candidate at the end of each loop. The total available resources, $N(F)$ for each template F are calculated via Equation 1, and are divided among I iterations. At the end of each iteration the candidate that minimizes the description length $\mathcal{L}(\mathcal{Q})$ of the data set, is added to Q_{eqn}, and the data points within the region are removed from Q_{exc}. This continues until no exceptions remain or the description length cannot be reduced. For the purposes of this paper, the available resources $N(F)$ are divided equally among the I iterations.

Predictor Variable		Min	Max	Response Variable	
CR	Compression Ratio	7.0	10.0	η_{gr}	Gross Indicated Thermal Efficiency
PI	Inlet Pressure (bars)	0.4	1.0	η_{net}	Net Indicated Thermal Efficiency
TI	Inlet Temperature (K)	270	310	η_{vol}	Volumetric Efficiency
Φ	Fuel/Air Equivalence Ratio	0.7	1.3		

Table 1: Predictor & Response variables in OTTONET (IC engine simulator)

1. $Q_{eqn} = \{\}$, $Q_{exc} = \{\mathbf{E}\}$, determine $N_i(F)$, $i \leq I$.
2. For each F, call KEDS $N_i(F)$ times.
3. For each R, f: $Q_{exc} = Q_{exc} - \mathrm{R}$, $Q_{eqn} = Q_{eqn} + f$. Compute $\mathcal{L}(\mathcal{Q})$.
4. Select R, f that minimizes $\mathcal{L}(\mathcal{Q})$.
5. Return to (2) unless $Q_{exc} = \{\}$ or I iterations done.

Figure 3: The KEDS-MDL algorithm

Experimental Results

Ottonet [Assanis, 1990], a simulator for an internal combustion engine, was used to provide the data for KEDS. The predictor (input) and response (output) variables for the internal combustion engine domain are shown in Table 1. The input variables were randomly varied over the ranges shown in Table 1, to generate 300 events, 50 of which were randomly set aside to create a test set. The remaining 250 events were used as a training data set for the two sets of experiments described below.

Experiment I: In the first series of experiments, the parameters were set at $m = 0.3$ and $\delta = 0.1$. Three separate experiments were run. (a) KEDS was run in a breadth first fashion and the results were combined at the end to produce a model. (b) KEDS-MDL was run with the same resources as in I.a. (c) KEDS-MDL was run again with the available resources *reduced* by a factor of 10. In the KEDS-MDL experiments (I.b and I.c), the available resources were divided equally among four iterations (i.e., $N_i(F) = N(F)/4, i \leq 4$).

The models were used to predict the attribute values of the test events (the test events were not part of the data set seen by KEDS). The models were evaluated in terms of the predictive error and the model creation time (in seconds on a DECStation 5000). The results are summarized in Table 2. Note that even with the same available resources, KEDS-MDL (Experiment I.b) took less time than in Experiment I.a. This is because the limitation of available resources refers to the maximum number of times $N(F)$ that the KEDS algorithm may be called. On the surface, although the KEDS algorithm appears to be independent of the number of data points n, the discovery and partitioning steps (Steps 3 and 4 in Table 2) depend heavily on n. As KEDS-MDL discovers regions and adds them to Q_{eqn}, n decreases on each iteration.

What is also very interesting is that even when provided with limited resources (one-tenth that available in I.a and I.b), KEDS-MDL learned models that were extremely competitive with those that had been learned using full resources. This indicates that the MDL metric, is effective even with extremely limited resources.

Experiment II: These series of experiments were designed to compare the performance of different metrics for model discovery. Using the same limited amount of resources available in I.c above, a second series of experiments were run with the following four metrics for model selection: (a) MDL (identical to I.c), (b) *accuracy*, the most accurate equation was chosen, (c) *size*, the region that covered the maximum number of events was chosen, and (d) *goodness* (= accuracy * size). In each experiment the appropriate metric was used in place of MDL in Steps 3 and 4 in Figure 3, to choose a region-equation pair in each iteration.

The generated models were evaluated for predictive error for all the response variables. The results are presented in Table 3. As can be seen KEDS-MDL outperformed the KEDS algorithm using other metrics. Below is the model created for the response variable η_{vol} in Experiment IIc (using KEDS-MDL with limited resources).

[CR > 7.2] [PI > 0.573] $\Longrightarrow$
$\eta_{vol} = 0.63$ CR + 18.78 PI + 75.3
ELSE [PI < 0.573] $\Longrightarrow$
$\eta_{vol} = 1.88$ CR + 50.03 PI - 1.68 Φ + 48.88

Conclusions

In this paper, we have defined an encoding schema for the models discovered by KEDS, and demonstrated that MDL can be used as an evaluation metric to efficiently acquire models from complex non-homogeneous domains. In the future we intend to apply KEDS-MDL to other engineering domains, such as modeling the delay of a VLSI circuit. KEDS will be enhanced so as to consider domain knowledge other than equation templates (for example, analyzing the topology of a VLSI circuit to determine various possible critical paths). The KEDS-MDL algorithm was motivated by our overall goal of developing a methodology to support engineering decision making. Under this methodology, called *inverse engineering*, the models discovered by KEDS-MDL will be used to directly support synthesis activities in engineering decision-making. This will greatly reduce design iterations and take advantage of the design expertise already captured in computer-based analysis tools.

Var	(a) Breadth-first		(b) KEDS-MDL		(c) KEDS-MDL:(1/10)	
	Error	RunTime (s)	Error	RunTime (s)	Error	RunTime (s)
η_{gr}	0.00639	4701	0.00312	1411	0.00685	89
η_{net}	0.00765	1956	0.00724	572	0.00838	287
η_{vol}	0.00511	5151	0.00365	1222	0.00395	168

Table 2: Experiment I: Comparing KEDS (breadth-first) with KEDS-MDL

Metric	Error: η_{gr}	Error: η_{net}	Error: η_{vol}
(a) MDL	0.007	0.008	0.004
(b) Accuracy	0.025	0.044	0.042
(c) Size	0.007	0.054	0.026
(d) Goodness	0.007	0.049	0.022

Table 3: Expt II: Predictive Error for different Metrics

Acknowledgments

We are grateful to Jorma Rissanen for his comments and suggestions. Our thanks also go to Dennis Assanis for the OTTONET data, to Robert Stepp and Bradley Whitehall for help while developing KEDS, and to Edwin Pednault for introducing us to minimum length encoding.

References

Assanis, D.N. 1990. OTTONET. Technical Report, Department of Mechanical and Industrial Engineering, University of Illinois, Urbana, IL.

Barron, A.R. 1984. Predicted Squared Error: A criterion for automatic model selection. In Farlow, S.J. (ed), *Self Organizing Methods in Modeling.* Marcel-Dekker. 87–103.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and Regression Trees.* Wadsworth.

Cheeseman, P. and others, 1988. Bayesian classification. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, St. Paul, MN. 607–611.

Cover, T.M. 1968. Estimation by the nearest neighbor rule. *IEEE Trans. on Information Theory* IT-14:50–55.

Derthick, M. 1991. A minimal encoding approach to feature discovery. In *Proceedings of the Ninth National Conference on Artificial Intelligence.* 565–571.

Falkenhainer, B.C. and Michalski, R.S. 1986. Integrating qualitative discovery: The ABACUS system. *Machine Learning* 1(4):367–401.

Finger, S. and Dixon, J.R. 1989. A Review of Research in Mechanical Engineering Design. Part I: Descriptive, Prescriptive, and Computer-Based Models of Design processes. *Research in Engineering Design* 1(1):51–67.

Fisher, D.H. 1985. A proposed method of conceptual clustering for structured and decomposable objects. In *Proceedings of the Second International Workshop on Machine Learning.* 38–40.

Friedman, J.H. 1991. Multivariate Adaptive Regression Splines. *Annals of Statistics.*

Gallager, R.G. 1968. *Information Theory and Reliable Communication.* John Wiley & Sons.

Kadie, C.M. 1990. Conceptual Set Covering: Improving fit-and-split algorithms. In *Proc. of the Seventh International Conference on Machine Learning*, Austin. 40–48.

Langley, P.; Simon, H.A.; Bradshaw, G.L.; and Zytkow, J.M. 1987. *Scientific Discovery: Computational Explorations of the Creative Processes.* MIT Press.

Leclerc, Y.G. 1989. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision* 3(1):73–102.

Lu, S. C-Y. and Tcheng, D. K. 1991. Building layered models to support engineering decision making: A machine learning approach. *Journal of Engineering for Industry, ASME Transactions* 113(1):1–9.

Pednault, E.P.D. 1989. Some experiments in applying inductive inference principles to surface reconstruction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.* 1603–1609.

Pentland, A. 1989. Part segmentation for object recognition. *Neural Computation* 1:82–91.

Quinlan, J.R. and Rivest, R.L. 1989. Inferring decision trees using the minimum description length principle. *Information and Computation* 80:227–248.

Quinlan, J.R. 1986. Induction of decision trees. *Machine Learning* 1(1):81–106.

Rao, R. B. and Lu, S. C-Y. 1992. KEDS: A Knowledge-based Equation Discovery System for engineering problems. In *Proceedings of the Eighth IEEE Conference on Artificial Intelligence for Applications.* 211–217.

Rao, R. B.; Lu, S.C-Y.; and Stepp, R.E. 1991. Knowledge-based equation discovery in engineering domains. In *Proceedings of the Eighth International Workshop on Machine Learning.* 630–634.

Rendell, L. A. 1983. A new basis for state-space learning systems and a successful implementation. *Artificial Intelligence* 20(4):369–392.

Rissanen, J. 1983. A universal prior for integers and estimation by minimum description length. *Annals of Statistics* 11(4):416–431.

Rissanen, J. 1985. Minimum description length principle. In Kotz, S. and Johnson, N.L., editors 1985, *Encyclopedia of Statistical Sciences*, Vol.5. John Wiley & Sons. 523–532.

Stepp, R.E. 1984. *Conjunctive Conceptual Clustering: A Methodology and Experimentation.* Ph.D. Dissertation, Department of Computer Science, University of Illinois, Urbana, IL.

Stone, C.J. 1977. Consistent nonparametric regression (with discussion). *The Annals of Statistics* 5:595–645.