Chapter 15

INDUSTRIAL STRENGTH GENETIC PROGRAMMING

Empirical Modeling and Symbolic Regression via GP: Integrated Methodologies, Best Practices, Lessons Learned

Mark Kotanchek,[1] Guido Smits,[2] and Arthur Kordon[3]
[1]*Dow Chemical, Midland, MI USA* [2]*Dow Benelux, Terneuzen, NV* [3]*Dow Chemical, Freeport, TX USA*

Abstract Since the mid-1990's, symbolic regression via genetic programming (GP) has become a core component of a multi-disciplinary approach to empirical modeling at Dow Chemical. Herein we review the role of symbolic regression within an integrated empirical modeling methodology, discuss symbolic regression system design issues, best practices and lessons learned from industrial application, and present future directions for research and application

Keywords: Genetic Programming, Empirical Modeling, Symbolic Regression, Support Vector Machines, Neural Networks

In theory, there is no difference between theory and practice. In practice, there is. – *Jan L.A. van de Snepscheut*

1. Introduction

Our objective is to summarize our experience in industrial application of genetic programming to empirical modeling and to transfer key learnings with respect to real-world application. Hence, there are three themes which are explored in the following pages: (1) the role of GP in empirical modeling, (2) symbolic regression system design and (3) open issues and research suggestions.

Our interest in symbolic regression is motivated by the industrial need to quickly and efficiently convert multivariate data sets and data streams into actionable insight and knowledge. Although we use a variety of

technologies in our analysis methodology, GP offers some unique capabilities which are complementary to those of neural networks, support vector machines, linear statistics, etc. Hence, we explore the framework of industrial empirical modeling, practical issues, and how modeling success may be best achieved. In general, success is achieved by using a combination of tools and techniques and analysis discipline.

We have developed two symbolic regression environments in support of our modeling efforts; the second theme explores the system design objectives, issues, components, experiences, and lessons learned in the development and exploitation of those environments. Some tricks-of-the-trade for achieving efficiency, accuracy and simplicity – from the human as well as genetic programming perspective – are presented.

Although symbolic regression via GP has proven to be a valuable tool, there remain open issues which are worthy of further research to improve the speed, robustness and effectiveness of GP and empirical modeling. To illustrate, truth is an evasive term in the realm of empirical modeling. Hence, we often assemble ensembles of models and use their concurrence, or lack thereof, as an indicator of model confidence. The validity of this strategy depends upon the ensemble containing diverse models with comparable complexities. Hence, we have a fundamental need to characterize both *diversity* and *complexity* in a population of models. Unfortunately, quantifying either of these is a difficult task. Another research area of potentially large impact is the conversion of hard problems into easy ones – either via integration with other techniques, incorporation of heuristics and *a priori* knowledge or identification of metavariables to simplify the solution discovery.

2. Industrial Application of GP to Empirical Modeling

In this section we review the importance of empirical modeling to industry, the unique capabilities of symbolic regression via GP for empirical modeling, and how an integrated methodology exploits the strengths of GP and complementary analysis techniques.

2.1 Empirical Modeling

In an industrial setting, the objective of empirical modeling is to convert data to knowledge which can then be transformed into money. As such, it tends to be goal-driven rather than the exploratory data analysis as is typically the case with data mining. In other words, we typically have a response (or responses) which we seek to model as a prelude to understanding underlying mechanisms or system optimization. Reality,

Table 15.1. Industrial applications of empirical models

Application	*Comment*
Research Acceleration	Understanding variable relationships to each other as well as to the response of interest accelerates research due to providing cues to underlying physical mechanisms or areas of parameter space to explore. These models facilitate exploration of highly multivariate problems. [Kordon02]
Inferential Sensors	Also known as soft sensors, these empirical models combine one or more parameters to infer a system state (response) which may be difficult to measure directly. Predictive models can give advance warning of impending product quality issues or other operational issues. [Kordon01]
Emulators	System optimization can be difficult for industrial systems due to the cost and difficulty of acquiring data points. An empirical model may be used as a system emulator to provide insight as a prelude to more focused experimental validation. Additionally, an emulator may be used as a lower-fidelity approximation of a first-principles model for coarse optimization or online deployment. [Mercure01]
Variable Transforms	Identifying combinations of variables which have a physical meaning can lead to simpler and more robust system models. These metavariables can also provide insight into underlying mechanisms or convert a nonlinear problem into a linear one.[Kordon02; Castillo02]
Nonlinear Design-of-Experiments	In environments where data points are expensive to collect, the ability to understand the underlying system behavior while simultaneously facilitating focusing data collection in critical regions. An example of this might include design of model discrimination experiments to distinguish between competing models of chemical kinetics. [Castillo02]

however, often requires that empirical modeling effort includes an exploratory aspect to identify key parameters and variable relationships. Empirical modeling may have a first-principles aspect if *a priori* information provides models which should be fitted to the available data.

2.1.1 Industrial Applications of Empirical Models. Some of the industrial applications of empirical models are captured in Table 15.1. In general, we seek either system insight or the ability to more effectively and efficiently monitor, control or optimize a system

2.1.2 Requirements for a Successful Model. Achieving an empirical model may be accomplished using a number of technologies

in addition to symbolic regression: linear statistics, neural networks, support vector machines, particle swarm optimizers, genetic algorithms, GMDH, fuzzy rules, etc. A quality model must have aspects of:

credibility	the model matches the observed behavior
robustness	the model accuracy is able to withstand minor changes in the targeted system
extrapolation	closely related to robustness, this is the ability to predict outside the training range (within reason)
self-assessment	the model is able to assess the quality of its output after deployment
interpretability	humans are able to explore and qualitatively validate the "reasonableness" of the model"
cost effectiveness	in addition to development cost, the total cost of ownership includes ease-of-use and effective maintenance and support

Model quality is not a static notion; rather, in addition to raw accuracy, it must consider the cost and timeliness of the development process as well as performance and maintainability during its lifetime.

2.1.3 The Role of Symbolic Regression. As we shall see when we review the integrated methodology, the primary analysis techniques used are linear statistics, neural networks, support vector machines and symbolic regression via genetic programming. Each of these foundation technologies have their strengths and weaknesses which are complementary [Kotanchek02]. In general, symbolic regression is more computationally intensive and produces less accurate models than the other nonlinear techniques. In compensation, it has unique benefits which include those summarized in Table 15.2.

2.2 Integrated Methodology

Combining engineering discipline and rigor with nonlinear and classical data analysis techniques leads to efficient and reliable empirical model development. The components of the methodology are summarized in Table 15.3. As indicated, the primary nonlinear modeling tools used are: neural networks, support vector machines and symbolic regression via genetic programming. Neural networks are universal approximators and are, therefore, capable of modeling any system. Of course, they can also model noise very well so caution must be used in their development since it is often difficult to interpret the developed models due to their black box nature.

Support vector machines (SVMs) [Jordaan02; Hastie01] are a recent introduction to the empirical modeling toolbox. The basic idea is that a subset of the data samples — the support vectors — will capture

Table 15.2. The unique benefits of symbolic regression to empirical modeling include:

Benefit	*Discussion*
Physical Insight & Understanding	Examining the models returned by a symbolic regression may be interpreted by an expert with first-principles insight to identify underlying mechanisms or to suggest new outside-the-box approaches.
Visualization	Visualizing highly multivariate spaces is inherently difficult. A symbolic representation is, effectively, a visual representation which summarizes the interaction between variables. Of course, the model may also be converted into graphical form if desired.
Simple Summary Models	Evaluating developed symbolic models may require fewer computing resources and complexity than those derived by their nonlinear brethren such as neural networks or support vector machines. This can facilitate online model implementation in manufacturing plants.
Identify Transforms & Metasensors	Natural combinations of variables may appear as part of the symbolic regression model building. These (typically nonlinear) synthetic metavariables can be used to capture variable dependencies as well as facilitate data transforms so that classical linear statistics may be applied and metrics such as lack-of-fit used for model validation. Additionally, incorporating these metavariables may increase the efficiency and accuracy of additional symbolic regressions by transforming the problem into a different space which is easier for model development.
Variable Selection	Often more variables are available to the empirical modeling than are warranted by the underlying physics. GP can identify these significant variables since they will, typically, appear in the more fit models. Focusing the subsequent modeling on these variables can increase the efficiency of the modeling effort.
Variable Sensitivity	Once a model (or models) have been developed, they can be numerically or symbolically differentiated to understand the sensitivity to parameter changes. This can be useful in applications such as robust design.

Table 15.3. The components of the integrated empirical modeling methodology

Component	*Discussion*
Problem & Success Definition	Although often neglected, *defining* success is *the* key ingredient to achieving success. This involves understanding the system to avoid context-free analysis (which tends to lead to confidently wrong answers) as well as specifying the requisite levels of model accuracy and robustness.
Data Review & Preparation	There is no substitute for understanding the input data and its quality. Bad data points should be removed and appropriate data transforms and combinations performed to synthesize metavariables. Data should be partitioned into test and training sets. If system changes have occurred, the data should be partitioned to avoid mode mixing. Often, data samples should also be downsampled to reflect the actual information content (this avoids redundant data points skewing the evaluation). Regression-oriented support vector machines are especially appropriate for downsampling data as well as identifying outliers.
Select Variables & Metavariables	Often many more variables are available than required. Downselecting variables to the critical parameters helps to focus the models onto the underlying physics rather than system noise. Additionally, many modeling techniques make an implicit assumption that variables are uncorrelated; violating that assumption can invalidate the resulting models. Genetic programming is useful for identifying possible metavariables. Although GP can be used for variable downselection, neural networks are generally more computationally efficient.
Build Models	Many techniques are available for model building These include model fitting simple mathematical models (e.g., polynomials) or first-principles models as well as nonlinear data-driven techniques. Support vector machines generally produce the most robust models, neural networks the most accurate, and symbolic regression the most insightful and simplest to deploy. The metavariables identified via GP can enable model linearization which is generally preferable from a robustness perspective.
Analyze & Validate Models	Any empirical model should be viewed with suspicion until proven valid. The possibility of overfitting must be explored as well as the physical reasonableness of results. If necessary, additional data partitioning and data cleansing should be done and the modeling exercise repeated.
Select & Combine Models	The best model may not be the most accurate depending upon the definitions of success. For example, simplicity of implementation or model parsimony with "good enough" performance may be preferred. Stacking or boosting models [Hastie01] can result in improved performance as well as an indication of operation in unknown regions of parameter space. However, using the consensus of models as an indicator of model validity makes an implicit assumption of independent constituent models which may be difficult to verify.
Exploit Models	In an industrial setting, the modeling effort is not a success unless the models are being exploited either by providing system insight, enabling optimization or deployed in an operational system.

the response function of the data if we locate appropriate kernels (e.g, radial basis functions or polynomials) on the support vectors. SVMs have a reasonably rigorous theoretical foundation founded in statistical learning theory and the theory specifically addresses small data sets. The benefit is SVMs can recognize sets which are oversampled with respect to their information content (or specified complexity). This also leads to the truly spectacular capability of SVMs — their ability to perform nonlinear outlier detection. They may also be used for downselection in the sample space to ensure that the input to other modeling tools are balanced in terms of their information content.

GP [Jacob01; Banzhaf98] offers an escape from the black box modeling techniques and provides a link between the empirical and the fundamental. Although there are continuing theoretical debates regarding the mechanisms of effective function evolution, this is not a major issue from a practical perspective since effective application has been demonstrated. GP is attractive for a number of reasons including the natural selection of the most important inputs, relatively parsimonious analytical expressions which are evolved and the associated ease of implementation and the lack of *a priori* model assumptions.

As a final note, irrespective of modeling technique, correlation (quality of fit of a model) does not imply causality. Causality always needs to be verified independently.

3. System Design

In this section we first discuss some of the key design issues for a symbolic regression environment. Following that, the components which have been implemented into our *Mathematica* [Kotanchek03] and MATLAB implementations are reviewed and key features discussed. Finally, we close with a review of tricks-of-the-trade in the practical use of the system capabilities.

The current implementations are currently restricted to evolving sample-oriented models (i.e., no recursive functions or operations across the entire data space). The typical symbolic regression will operate on a data set of 3–10 variables and 20–3000 data points. (Recall that we often use neural net and SVM-based preprocessing to reduce the size of the input data set.) However, we have used GP to perform sensitivity analysis of systems with up to 100 variables.

3.1 Implementation & Design Issues

Better. Faster. Cheaper. Those are essentially the objectives of any industrial design effort. In the context of symbolic regression system design, these translate into:

- speed of model development and selection,
- robust parsimonious models, and
- human efficiency of model development and selection.

Achieving these objectives requires a system level approach to provide an environment efficient for both the computer and the user. Since garbage in results in garbage out, input data quality assessment tools must be provided to assist in the data set design process. The initiation and execution of the evolutionary process must also not be labor intensive. From the computer standpoint, the evolutionary process must be computationally viable either through efficient code or via parallel execution. For maximum efficiency the symbolic regression should be automatous to minimize demands on the modeler. The evolved solutions also must balance complexity and performance. Finally, given the plethora of solutions which will typically be evolved, tools must be provided for the user to sort through the entities (candidate expressions), evaluate them, extract insights, and facilitate model deployment.

3.2 System Components

On the order of 200 functions have been defined as part of the *Mathematica* symbolic regression and data analysis environments; hence, of necessity, we will only cover the highlights of the design functionality.

3.2.1 Define Success: Fitness & Parsimony Penalty Functions. The first step in the symbolic regression is to define a fitness function to characterize the quality of the fit of the evolved expression to the available data. Generally, we opt for an absolute correlation metric since a translation and scale coefficient can be calculated in post processing — thereby simplifying the search problem by eliminating the need to evolve those terms. There are multiple aspects of fitness to consider when evaluating an entity. The raw (or absolute) fitness characterizes the ability of the expression to predict the targeted response behavior.

We generally prefer parsimonious (simple) functions. A parsimony adjustment which penalizes complexity also prevents explosive growth of function complexity due to the inclusion of introns. Although we continue to explore alternate strategies, the default, which is generally

appropriate, is to use a multiplicative penalty function which penalizes the raw fitness according to the number of nodes in the genome. A sigmoid function,

$$ParsimonyFunction \rightarrow 1 - \frac{1}{1 + e^{-\frac{\text{nodes}[genome]-a}{b}}}$$

is the default adjustment to the raw fitness. Adjusting the sigmoid coefficients, a and b, allows tweaking of permissible complexity and complexity sensitivity.

3.2.2 Initial Population Creation. Functions need to be provided to define entities as a starting point for the evolutionary process. These can either be created randomly from the genetic code components or drawn from a pool of previously evolved entities. Additionally, functions must be provided to seed the populations with *a priori* expressions; hence, the ability to inverse map from defined mathematical expressions (the phenotypes) into their genetic codes (genotypes) is critical — at least to give the user the feeling of control!

3.2.3 Genetics. Although not strictly required, our implementation of genetic programming distinguishes between the genetic code abstraction (genotype) and the mathematical expression (phenotype) upon which the raw fitness is calculated. This abstraction leads to considerable flexibility in the application of the symbolic regression.

The Genetic Code: Building Blocks for Evolution. The genetic code for symbolic regression requires three components:

- *Function Patterns*: codings for mathematical and logical operators
- *Variables*: the independent parameters in the data matrix
- *Constants*: constant integers and reals (our implementation synthesizes random constants upon demand)

The problem is to assemble these into a structure which describes the observed behavior. We can simplify the evolution process by including function patterns appropriate for the problem at hand as well as downselecting the variables to an appropriate subset. In addition to controlling the maximum arity (number of slots or parameters) of functions like Plus[ ] or Times[ ] which can handle an arbitrary number of arguments, we also control the complexity and structure by weighting the likelihood of any given building block (or class of building block) being selected. To illustrate, the default function patterns are:

$$\text{FunctionPatterns} \rightarrow \{\Sigma[\_\_], \mathbb{S}[\_, \_], \Pi[\_\_], \mathbb{D}[\_, \_], \mathbb{P}[\_, \_]\}$$

which, when converted into the phenotype, leads to summation, subtraction, multiplication, division, and power, respectively. Summation and multiplication are arbitrary arity functions (denoted by two underscores) whereas the others are explicitly limited to two (possibly compound) slots. The default is to automatically synthesize variable names $(x_1, x_2, \ldots)$ from a user-defined symbol or explicit symbols, as desired.

The evolution can also be accelerated by inclusion of proper metavariables (combinations of variables or transforms) which simplify the task of discovering a quality genome. Currently, metavariables are defined by users examining evolved structures or using physical insight; however, a natural extension is to adopt a coevolution strategy for automated definition.

In the ensuing discussion, it is important to note that any genome can be represented as a tree-based structure with terminals (constants or variables or metavariables) at the end of the branches and functions at the other nodes. The term *node* refers to any specific location in the genetic tree — function or terminal.

Complexity Control. In addition to the indirect means of complexity control by applying parsimony pressure and limiting the arity of arbitrary arity functions, we have explicit limits on the maximum allowable tree depth and leafcounts for the genome structures. These are safety regulators since, if unfettered by parsimony controls, expressions can rapidly evolve with millions of elements — which likely do not meet the engineering definition of success!

3.2.4 Phenotypes. The genetic code must be converted into its phenotype (physical representation) for fitness evaluation.. This is the representation which can be evaluated by *Mathematica* or MATLAB at each of the evaluation points and the predicted result compared to the actual observed response. As with natural selection, it is the phenotype (entity) which is judged and the underlying genetics which are propagated.

The phenotype can also learn from experience to improve its performance. We shall address this more as part of the genetic engineering discussion.

Representation. In addition to the genetic code, any given entity also has three fitness metrics: absolute, adjusted and relative. The absolute fitness is the quality independent of parsimony pressure whereas

the adjusted fitness penalizes this value based upon its complexity. The relative fitness is the quality relative to its peers in the breeding pool. Additionally, we track miscellaneous attributes as part of a personality aspect for analysis flexibility.

Evaluating Entities. Evaluating entities is a relatively straightforward process in that the genetic code is converted into an executable form and evaluated. The mapping can be straightforward, e.g., $\mathbb{SR}[x] \rightarrow \sqrt{x}$, or they can be *protected*, e.g., $\mathbb{SR}[x] \rightarrow \sqrt{|x|}$, which can handle the situation when x is negative. The advantage of protected functions is that they avoid pathologies in the phenotypes and, therefore, may result in a faster solution. The disadvantage is an increase in complexity of the evolved expressions, although not necessarily in the underlying genetic code.

Each entity is evaluated at each of the sample points and the result judged relative to the observed response behavior. This is where design of the data set comes into play relative to correspondance of the calculated vs. true quality of the correspondance. If the data set is *unbalanced* in the sense that much of the data is localized in one region of parameter space, then the evolved entities will be judged primarily on how well they fit *that* region. SVMs are especially useful to address this problem and produce *balanced* data sets which avoid this problem. Alternately, human insight, experimental design, genetic algorithms or other techniques may be used to preserve *diversity* in the data space.

3.2.5 Propagation. The purpose of a phenotype is to earn the right to propagate its genetic code into the next generation. As with nature, there are a plethora of competition models which can be used to assign breeding rights; these include:

- **Proportional** – Each entity is awarded breeding rights proportional to its contribution to the cumulative fitness of the population. This is also known as the roulette wheel model since the size of each entity's bin corresponds to it's component of the total population fitness (*Mathematica* default).

- **Rank-Based** – The breeding rights are proportional to the rank of an entity (the lowest quality entity gets a rank of one and the rank increments by one from there). (MATLAB default).

- **Elitist** – The top fraction of entities are treated as equals from a breeding rights perspective and the lower ranked entities have no chance to propagate.

Table 15.4. Diversity introduction mechanisms include:

Mechanism	*Description*
MutateSubtree	Replace an arbitrary node and all of its branches with a new subtree
MutateNode	Replace an arbitary node with a different function or terminal which is a valid replacement.
Crossover	Replace subtrees from an arbitrary node with those from another entity. Although analogous to sexual reproduction, the genetic exchange is not restricted to be pairwise.
Clone	Sometimes reproduction is asexual without an introduction of diversity
Hoist	Create a new entity from a subtree of the donor entity.

- **Tournament** – Subsets of the population are placed into pools for competition with the winners achieving propagation rights. The intent here is, again, to preserve the diversity in the population.

Our preference is to run many independent evolutions; hence, premature convergence and loss of genetic diversity isn't a major problem. We have also used mixtures of selection strategies.

Successful evolution requires only two things: a fitness-based pressure and a means to introduce diversity into the population. If we recall that the genome is represented as a tree, possible operators to introduce this diversity include those summarized in Table 15.4.

Although analogous to cloning, we also have found the presence of a Methuselah to be effective wherein the best entity is given a new lease on life and propagates unchanged into the next generation. This leads to a more continuous fitness behavior during an evolution which simplifies termination criteria and related issues. The others in a generation are killed off and replaced by their progeny.

3.2.6 Genetic Engineering: Optimizing for Performance & Simplicity. When Darwin proposed the theory of evolution based in survival of the fittest, there was not yet an awareness of genetics and their role in evolution. One alternate theory was proposed by Lamarck wherein the experiences collected by an entity would be passed on to the next generation. Thus, a giraffe stretching its neck reaching for leaves would result in progeny with longer necks. Although Lamarckian genetics are not realistic in natural systems, the implicit genetic engineering is easily achieved in symbolic regression via genetic programming. To achieve this, we need inverse mappings from phenotypes to genotypes

Table 15.5. Genetic engineering techniques include:

Operation	*Description*
Remove Introns	Remove nonfunctional genetic code to produce a simplified genome
Simplify	Evaluate the phenotype expression and simplify it using mathematical rules. This is relatively easy in an environment such as *Mathematica* which provides support for such symbolic manipulations. The simplifed form is then converted into it genome equivalent and becomes the new genetic code of the entity.
Align	Scale and translate the entity to maximize the alignment to the observed response behavior. This linear operation is generally fairly efficient computationally.
Optimize	Optimize the constants in the phenotype to maximize alignment to the response behavior. Note that this is typically a nonlinear optimization problem and relatively time consuming.
Optimize Structure	Prune the structure to eliminate branches and leaves which do not provide a benefit in terms of increased phenotype fitness. Alternately, prune out selected variables.
Local Optimize	Randomly select a constant in the genome and optimize its value.

as well as tools to manipulate and optimize both phenotypes and genotypes. The basic classes of genetic engineering which we have found to be effective include those listed in Table 15.5.

With the exception of the local optimization which is relatively efficient and occasionally included as a propagation operator, the other genetic engineering operations are performed offline on the best entities resulting from a round of evolution.

3.2.7 Working with Populations: Isolated and Cascaded Evolutions. Although, in principle, it should be possible to evolve high-quality expressions in a single monolithic evolution, maintaining sufficient genetic diversity and avoiding premature convergence on suboptimal solutions has proven to be very difficult in our experience. A more robust strategy has been to spawn many isolated and relatively small populations which evolve in parallel. The best entities out of each of these *worlds* are placed in a pool and randomly selected as the seed population for a subsequent round of evolution of many parallel worlds. This cascaded evolution model has proven to be very effective and robust.

The term *efficient* in genetic programming is a relative term since GP is, typically, very time and compute resource intensive. Towards that end, eliminating unecessary variables from the modeling exercise simplifies the task of high quality expression discovery. Although we generally have pruned the number of variables using a neural network based sensitivity analysis as part of the data processing, a good practice is to examine the presence and frequency of variables in the best evolved structures to ascertain whether additional variable elimination is possible. Additionally, identification of metavariables from the evolution results can help to convert a hard analysis problem into a much easier one.

Given the investment in time and computation in the evolutionary process, a viable system design needs to include tools for archiving evolutionary results, retrieving those results, and merging populations. Our utility functions in this space also provide for automatic file naming which facilitates analysis project organization and execution and avoids inadvertent file overwrites.

3.2.8 Reviewing Entities. Machine intelligence should not be confused with intelligence. At this stage of the technology curve, humans must be involved in reviewing evolutionary results, validating performance and selecting models for deployment and exploitation. From a system design standpoint, the human interaction must be efficient and insightful. Towards that end, we provide a number of tools in our systems to facilitate that interaction; these include those summarized in Table 15.6.

3.3 Tricks of the Trade

Although our symbolic regression environments are actively being used in empirical analysis projects, they are also the subjects of continued research and development. Despite the success of the application efforts, it would also be hubris to believe that the choices of evolutionary operators, population sizes, population operators, selection strategies, complexity controls, etc. are optimal. However, they are good enough — which is a sufficient level of performance from an evolutionary (and engineering) viewpoint. That said, key features of the symbolic regression system designs are:

- **Multiple Worlds** — many smaller evolutions minimizes issues around maintaining genetic diversity

Table 15.6. Tools to review evolution results include:

Tool	*Description*
Genome Tree Plot	Treeform graphic of the entity genetic structure
Entity Selection Table	Tablular view of top entity expressions, their fitnesses (raw, absolute, and relative), and included variables
Response Surface Plot	Response surface or curve over a specified or automatic region of parameter space. Unvaried variables may either be specified or automatically calculated from the data matrix.
Entity Evaluation Plot	The predicted vs. observed response. Ideally, all points should fall on the diagonal
Entity Residual Plot	Residual (error between predicted and observed) vs. sample number
Variable Prediction Plot	Actual vs. predicted values as a function of specified variable
Entity Regression Report	Summary statistics relative to the quality of fit of the entity to the observed behavior.

- **Cascaded Evolutions** — cascading evolutions using the best entities from prior rounds initializes the subsequent rounds with genetically diverse but high quality entities
- **Intron Removal** — removing the genetic noise helps to ensure that the key genetic building blocks are propagated
- **Entity Optimization** — optimizing the entity structure or coefficients in post-processing improves the quality of the models
- **Absolute Correlation** — using absolute correlation as the fitness metric eliminates the need to evolve scaling and translation coefficients
- **Protected Functions** — this speeds up the discovery of fit models, albeit at the cost of increased phenotype complexity and difficulty in post-processing simplification and optimization
- **Variable Downselection** — trimming the number of variables to those which most drive the response behavior helps to reduce the search space and accelerates the model discovery process
- **Metavariables** — the right synthetic variables also reduce search space and accelerate the discovery process
- **Evolution, Evaluation & Review Tools** — ease of use and analysis is critical for success since humans provide the contextual insight and real-world validation

4. Open Issues & Research Topics

Although the industrial value of symbolic regression and genetic programming have been established, the optimal strategies and parameters settings remain a topic of ongoing research. Additionally, there are deeper conceptual and theoretical explorations which remain as significant issues to fully capture the value of GP in industry. Some of these are discussed in the following sections.

4.1 How to control complexity and characterize smoothness?

The quality of an entity is determined by how well it describes the observed response behavior *at the sample points*. The open question is the behavior between the sample points and in the surrounding region. To some extent, we attempt to control this via a parsimony pressure which penalizes complexity; however, that does not necessarily eliminate pathologies and singularieties which may not be physically reasonable. Occam's razor implies that we want "smooth" behavior unless we have information to the contrary. The question is, "How do we *efficiently* recognize and characterize the *overall* and *local* smoothness (complexity) over parameter space?" The ability to answer this question should lead to more robust evolved expressions.

4.2 How to blend heuristics and prior knowledge?

Currently we have three basic strategies for including *a priori* insight and knowledge into a symbolic regression: (1) choice of function set and variables, (2) inclusion of metavariables, and (3) seeding the evolution with likely expressions. Improving the inclusion of prior knowledge should lead to greater physical interpretability of evolved expressions and, presumably, more robustness and more efficient and faster evolution.

4.3 How do we identify metavariables?

In the ideal case we could identify transforms which would convert the nonlinear analysis problem which would be tractable via linear techniques. Currently, identification of metavariables and transforms is done by "eyeballing" the evolved fit solutions. There has got to be a better way! Coevolution of metavariables is one possible approach; however, we could also use a pattern analysis post-processing of the best solutions.

Making the metavariable selection more automated and robust should improve the speed of analysis and interpretation.

4.4 How do we detect diversity in a population?

A strategy which we have found to be effective is to *stack* (very similar to boosting) models and to use their average or median as a robust predictor and their consensus as an indicator of prediction confidence. The validity of this approach depends upon the *diversity* of the population. Obviously, stacking minor variations on the same theme will not add a great deal of robustness to the composite estimate.

To assemble a collection of diverse models we need a metric to characterize diversity. To some extent, we can evolve a diverse population by using many isolated evolutions or by evolving expressions using different sets of building block functions and variables. But, how can we be sure that we have truly achieved the desired diversity objective?

Related to the diversity issue is the ability to measure expression complexity since we would like to balance the complexity of the expressions used in the composite model.

4.5 Are we using the wrong paradigm?

A cultural metaphor (e.g., particle swarm optimization, PSO) appears to hold a significant effectiveness advantage over their genetic brethern (e.g., genetic algorithms) for optimization problems and has replaced GAs in our color additive formulation and appearance engineering systems. As with human culture, the interchange of *memes* (ideas) operates in a far faster and more effective fashion than the genetic transfer operating in parallel. This leads us to wonder whether the *genetic programming* paradigm is passé and should be replaced or augmented with *memetic programming*? The question, of course, is how to to rephrase the symbolic regression search problem from a genetic analogy into a cultural framework?

5. Summary

GP-based symbolic regression has become a key component in Dow Chemical's empirical modeling capability due to its unique ability to provide physical insight, generate computationally efficient models, and identify key variable and associated transforms and combinations. Used with other analysis techniques as part of an integrated methodology, the result is improved modeling accuracy, robustness, and development efficiency. Continued improvement of the symbolic regression foundations and their integration with other linear and nonlinear analysis techniques

continues to be a significant research effort at Dow Chemical with an expectation of a corresponding return on investment.

References

Banzhaf, W., Nordin, P., Keller, R. E., & Francone, F. D. (1998). *Genetic Programming: An Introduction*. San Francisco: Morgan Kaufmann.

Castillo, F. A., Marshall, K., Green, J., & Kordon, A. K. (2002). *Symbolic Regression in Design of Experiments: A Case Study with Linearizing Transformations*. In W. B.. Langdon, et al. (Ed.), *Proceedings of GECCO 2002* (pp. 1043–1048). New York: MorganKaufmann.

Hastie, T., Tibshirani, R. and Friedman, J. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag.

Jacob, C. (2001). *Illustrating Evolutionary Computation with Mathematica*. San Francisco: Morgan Kaufmann.

Jordaan, E. Maria. (2002). *Development of Robust Inferential Sensors: Industrial Application of Support Vector Machines for Regression*. Eindhoven: Universiteitsdrukkerij TU Eindhoven.

Kordon, A. K., Pham, H. T., Bosnyak, C. P. & Kotanchek, M. E. (2002). *Accelerating Industrial Fundamental Model Building with Symbolic Regression: A Case Study with Structure-Property Relationships*. In D. Davis & R. Roy (Ed.), *GECCO 2002: Presentations in the Evolutionary Compution in Industry Track* (pp. 111–116). New York: GECCO-2002.

Kordon, A. K. & Smits, G. F. (2001). *Soft Sensor Development using Genetic Programming*. In L. Spector, et al. (Ed.), *Proceedings of GECCO 2001* (pp. 1346–1351). NY: Morgan Kaufmann.

Kotanchek, M. E., et al. (2002). *Evolutionary Computing in Dow Chemical*. In D. Davis & R. Roy (Ed.), *GECCO 2002 Presentations in the Evolutionary Computation in Industry Track* (pp. 101–110). New York: GECCO-2002.

Kotanchek, M. E., (2003). *Industrial Strength Symbolic Regression: Evolving Empirical Models from Industrial Data. 2003 Mathematica Developers Conference* Champaign, IL : Presentation.

Mercure, P. Kip., Smits, G. F., & Kordon, A. K. (2001). *Empirical Emulators for First Principle Models. Fall 2001 AIChE Meeting*. Reno: Presentation.