On Optimizing Anomaly Detection Rules for Gas Turbine Health Monitoring

Weizhong Yan[1], Lijie Yu[2], Jim Sherbahn[3], and Umang Brahmakshatriya[4],

[1,4]*General Electric Global Research Center, Niskayuna, NY, 12309, USA*

yan@ge.com
brahmaks@ge.com

[2,3]*General Electric Power and Water Engineering, Atlanta, GA, 30339, USA*

Lijie.yu@ge.com
James.sherbahn@ge.com

ABSTRACT

Gas turbine health monitoring is a critical process in preventing costly unplanned maintenance and secondary damage. To monitor gas turbine health, control signals are typically collected and analyzed using anomaly detection rules and models to assess failure likelihood based on observed data patterns. An analytic designer will often deal with rule optimization tasks in order to maximize failure detection and reduce false alarms. Manual tradeoff analysis is typically time consuming and suboptimal. In this paper, we attempt to address this issue by introducing a strategy for automatic and efficient rule optimization. By focusing on optimizing rule parameters while keeping rule structure intact, we maximize the rule performance by integrating domain knowledge with data driven optimization techniques. Realizing that automated rule tuning can be computationally expensive and infeasible to complete in reasonable time, we will leverage our recently-developed scalable learning framework - iScale that allows for automatically distributing rule tuning tasks to a large number of cloud computers, which not only dramatically speeds up tuning process, but also enables us to handle big size of historical data for tuning. We also explore different search methods to make rule tuning more efficient and effective and finally demonstrate our rule optimization strategy by a real-world application.

1. INTRODUCTION

Today thousands of GE manufactured gas turbines are serving customers worldwide for a wide variety of industrial applications. Most customers are adopting a contractual service agreement (CSA) with GE and rely on GE's OEM expertise for actively monitoring turbine health, i.e., to proactively detect anomalies and prevent costly unplanned maintenance. Aiming for more accurate and robust detection of incipient faults as early as possible, over the years we at GE have developed and fielded a spectrum of advanced analytics models (both rule-based and data-driven models as well).

Pure data-driven modeling techniques work well if sufficient labeled data are available. However in real-world applications like in gas turbine monitoring, obtaining sufficient labeled data is labor-intensive, if ever possible. In particular, true positive cases might be sparse or noisy. Using small set of labeled data for data-driven modeling may cause model over-fitting or ill-formed model representation. In addition, pure data model may not have explicit knowledge structure or explainable reasoning logic that engineers prefer, which often hinders user acceptance of the model.

Consequently, for gas turbine health monitoring applications, rule-based models are still dominantly used. In fact, most PHM systems make use of diagnostic rules in one form or another. For the sake of clarity, consider one of the simplest forms, which has the following form.

$$\text{IF } (\ T < X\ ) \text{ THEN } (\ \text{STATE} = A\ )$$

What this rule basically says is that if a parameter (X) has a value that exceeds a limit (T), the system is in a specific state (A). Generally the state identified reflects a particular, degraded state. It is worth stressing that real-world rules in gas turbine monitoring are typically much more complex, not only consisting of a large number of such simple-form rules, but also having complex-form rules.

Traditionally decision rules and their associated rule constants are determined by domain or engineering experts

based on their understanding of the physical system. The rules are continuously refined until each rule produces acceptable detection accuracy. Such manual refinement not only is labor-intensive, but also often fails to find true optimal values.

To address the above-mentioned challenge, in this paper we take a different approach, that is, to maintain the existing rule knowledge forms, but leverage machine learning based optimization platform to improve rule performance. By maintaining the expert rule forms we ensure correct physics understanding is maintained in rule logic. Tunable parameters, or knobs, are selected to optimize rule performance using training and testing data sets. The derived analytics benefits from both domain knowledge capturing as well as data driven optimization. The impact of having such a capability is significant in that instead of requiring detailed "face time" of an expensive engineer, an analyst could use this process to learn from labeled data and the tuning could be done in an automated and even online manner.

To that end a generic rule optimization platform has been developed, which is independent of specific rules or rule platforms. It allows a user to create a rule tuning job through a web-based configuration interface. A user has the flexibility to choose among unit level or fleet wise optimized rule.

In the remainder of this paper, we will first present the architecture of cloud-based machine learning system - iScale, followed by a discussion of available optimization method available. A case study will then be presented using iScale to perform a specific rule tuning job, and then the conclusions summary will be provided in the last section.

2. SCALABLE LEARNING FRAMEWORK - ISCALE

Creating solutions for analytically hard problems is presently a time- and cost-intensive process. This is largely due to the fact that the design of advanced analytic solutions is largely manual, requiring involvement of one or more analysts. These analysts apply their specific knowledge and expertise within a given area of analytic problem-solving to create an acceptable solution. This process has resulted in a major bottleneck in the company's ability to create advanced analytic solutions rapidly. Aiming at tackling this bottleneck, we at GE have been developing a cloud-enabled analytics framework (called iScale) [Yan et al (2011)].

iScale is primarily designed to be a distributed computing environment for creating, refining, deploying and maintaining analytic solutions. As shown in Figure 1, the core of the framework consists of several key components, including the job manager, the resource manager, the job scheduler, the executer, and the optimizer. These components serve as an "orchestrator" among users, compute machines and algorithms. Specifically it takes user's inputs (data and performance requirements, etc.), picks a subset of algorithms in the library that are most relevant to the problem, intelligently distributes the search tasks to different computer resources, and outputs the best combination of models and associated model parameters that maximally meet user specified performance requirements. The framework provides a web-service that can be accessed from a laptop computer and other mobile devices as well. The framework also maximally leverages

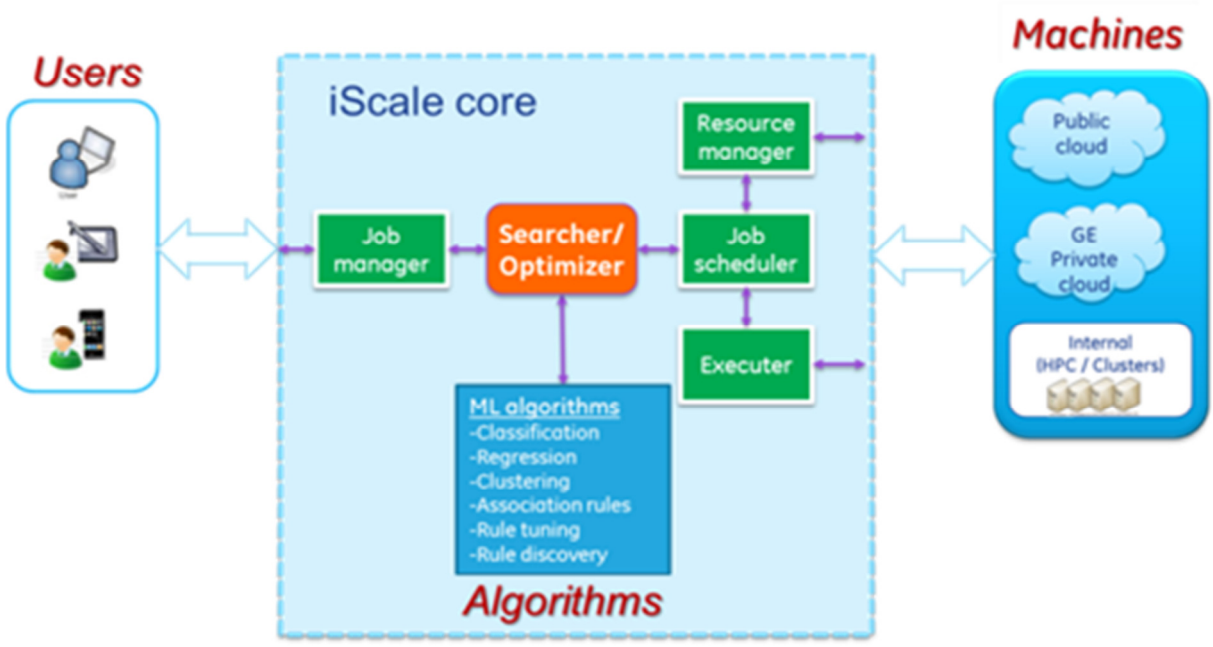

Figure 1: iScale Framework

heterogeneous compute machines (cloud machines and internal HPC/clusters) and is flexible in integrating different algorithms written in different languages, e.g., C/C++, Java, and R. More importantly, it is highly scalable, that is, it is capable of handling large size of data by leveraging distributed computing technology.

A critical component of iScale involves automating a significant portion of the currently manual process involved in problem formulation, data preparation, model selection, model tuning, domain knowledge integration and ensemble creation. The larger aspiration of iScale is to move analytic development activity from a one-off, largely manual process to a one-click, largely automated process. This is expected to significantly increase the rate at, and the ease with, which a significantly larger employee population in the company will create analytic solutions. It will help to make the creation of analytics an increasingly pervasive activity across the entire company. iScale will be the ecosystem in which analytic modules are born, sustained and continuously improved. A by-product of having such an ecosystem is it can bring to bear a large number of diverse analytic approaches to a single problem, thereby increasing the likelihood of finding a solution of very high quality. As an ecosystem, it is also expected to release next generation innovation that is evident in other similar analogues in the market today, like the Apple AppStore. By harnessing the virtually infinite and elastic compute power of the cloud, iScale is able to conduct a comprehensive and iterative search for the optimal analytic solution to a problem from across a diverse array of applicable approaches. Thus iScale is well suited for rule tuning as well.

3. OPTIMIZATION METHODS

Rule tuning is considered as an optimization problem where design space is defined by the tunable variables and objective function is defined by the rule performance metrics, i.e., probability of detection (POD) and false alarm volume (FAV). The rule tuning optimization problem has an important feature, that is, its objective values are available, but the derivative of the objective function is not computable. Another feature associated with rule tuning optimization is that the objective function evaluation is computationally expensive. These two features call for derivative-free (also called zero-order based) optimization methods [Rao (2009)] for rule tuning. Also, since multiple objectives (performance metrics) are involved in rule tuning, rule tuning is characterized as a multi-objective optimization (MOO) problem [Marler & Arora (2004)].

In literature there are many different derivative-free optimization methods [Conn et al (1997)]. In this paper we employ two different optimization methods, grid search (GS) and differential evolution (DE), for rule tuning. While both GS and DE fall to the category of global optimization methods [Rao (2009)], GS is a deterministic optimization method and DE, on the other hand, is a stochastic (also called heuristic or meta-heuristic) optimization method.

GS performs optimization as follows: dividing the n-dimensional design space into a n-dimensional grid, evaluating the objective function at all of these grid points, and picking the grid point that gives the minimal (or maximum) objective function value. Grid search is a simple global optimization method and can be easily distributed to many computer nodes to speed up the search process. The main drawback is that it suffers from the *curse of dimensionality*, i.e., the number of objective function evaluations grows exponentially with the number of design parameters.

- Initialization
- Evaluation
- Repeat the followings
  - *Mutation*
  - *Crossover*
  - *Selection*
  - *Evaluation*
- Until stopping criteria are met

Figure 2 – General procedure for differential evolution

DE is a simple, but powerful at the same time, population-based, stochastic optimization method [Storn & Price (1997)]. Like other population-based optimization methods (e.g., GA, PSO), DE optimization follows the general procedure as shown in Figure 2. Essentially, an initial population of solutions is randomly generated and evaluated; and those solutions are improved upon by applying mutation, crossover, and selection operators until a stopping criterion is met.

Compared to other EA optimization methods, DE optimization has several advantages, including fast convergence, having fewer control parameters, and ease in programming. As a result, DE has been used to solve a wide range of real world optimization problems [Das & Suganthan (2011)].

For multi-objective optimization problems, since there rarely exists a single solution that optimizes all objectives simultaneously, the optimum is given by a set of solutions known as the Pareto optimal set. The elements in this set are said to be non-dominated since none of them is better than the others in terms of all objectives. Using DE for MOO problems involves changes to its operators, mutation, crossover, and selection. Many different design strategies have been proposed. For details, refer to [Xue, et al (2003) and Reyes-Sierra & Coello (2006)].

Figure 3 illustrates the flow diagram of using DE for optimizing rules. In the optimization process shown in the

diagram, the step where rule engine execution based on a specified parameter set and the data is the most computational expensive one. That is where iScale helps by distributing the rule engine execution tasks to many compute nodes so that rule evaluation can be performed simultaneously. In the diagram, GS and DE differ in that GS is a one-pass operation while DE involves many iterative steps until convergence condition is met. Another difference between GS and DE is that the parameter set is pre-defined grid in GS while in DE the initial parameter set is randomly generated.

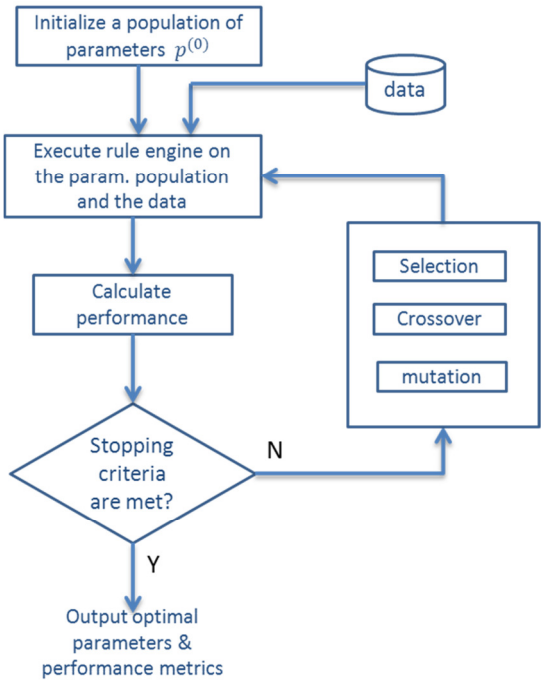

Figure 3: Flow diagram for rule optimization

4. RULE TUNING CASE STUDY

In this section we provide a use case study about performing a specific rule tuning job using iScale.

We will first give a brief introduction to the rule to be tuned. We then provide details of rule tuning process (the critical steps and data sets, etc.). At the end of the section, rule tuning results will be discussed.

4.1. The Vibration Rule for Turbine Vibration Monitoring

The rule concerned in this paper is the gas turbine vibration rule. Most gas turbines are equipped with proximity probes and seismic sensors located on the bearing housings. They provide key indicators of gas turbine hot section system integrity, including bearing damage, rotor imbalance, sudden mass loss, etc. Figure 4 shows an example of proximity sensor signals with step shift resulted after turbine blade migration due to lock wire failure. Event occurred shortly after unit was restarted and reaching high load, when multiple sensors had a step shift with increased vibration level.

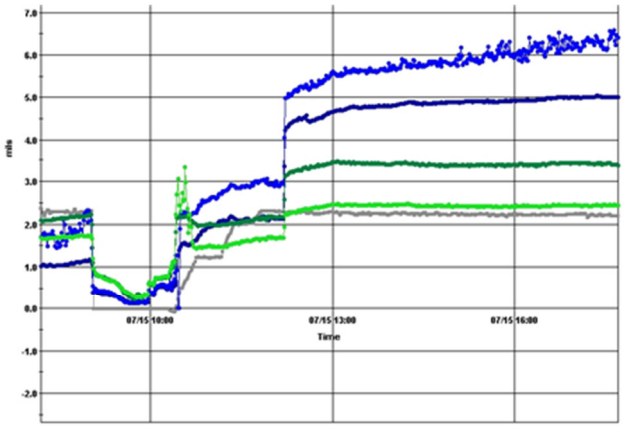

Figure 4 - Vibration Step Change Due to Turbine Blade Migration

The vibration rule is developed to examine probe signals to detect trend or step change. Once significant change is detected, statistical test is performed to assess confidence of the change. Both seismic and proximity probes will be analyzed and fused to improve alarm confidence.

Vibration signal alone may not be sufficient to determine system condition. Variation caused by operating condition change such as load shift or turbine speed change may also cause similar vibration signal shift. To separate true HGP failure from operation status change, more logic is added in the vibration to establish enabling criteria and stability criteria.

4.2. Vibration Rule Tuning Process

Within iScale platform, performing a rule tuning job is standardized into a straightforward process as shown in Figure 5. These can be carried out during the initial analytics design and development phase, or during the life cycle management for rule improvement. On the high level there are four steps from end to end.

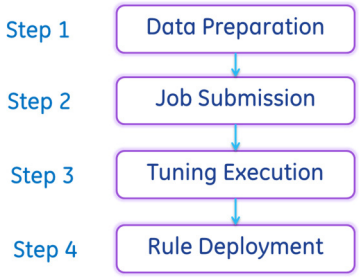

Figure 5 - iScale Rule Tuning Process

4.2.1. Data Preparation

As in any data driven modeling work, high quality data with representative feature set is a key for gaining predictive capability of the underlying analytics. Data preparation is certainly important for rule optimization.

Three different types of data set are required for rule tuning. First is historical event ground truth data. Since we focus on anomaly detection rules, these include both abnormal units and normal units, which are referred as POD cases and FAR cases, respectively.

Secondly, time series of rule input data of each historical case are extracted from the data historian, which will be used in rule evaluation. One minute resolution is used for all vibration case data. A week or so before and after POD events are prepared, and five months of FAR data is used in the vibration case study.

Thirdly, tunable rule parameters are identified and their valid ranges are also specified. In the vibration rule tuning case, there are thirty or so parameters can be adjusted outside the rule. Some values are set based on unit configuration or material properties. Among which four high sensitivity parameters are selected as the tuning target. For intellectual property protection, we are not allowed to give details of the four parameters. Valid ranges of each tunable parameter are also specified in the template, which defines the optimization search space.

4.2.2. Job Submission

iScale rule tuning is provided as a web service. A user can login to the web interface to create and monitor a tuning job. To create a new job, the need to provide information of the rule platform, rule executable package, training and testing data set, and evaluation criteria. The main interface screen is shown in Figure 6.

iScale itself is rule platform independent. However, platform specific rule wrapper and rule analysis engine will be required to execute a rule tuning within iScale to perform evaluation. The vibration rule is implemented based on CCAP platform, a GE in-house developed platform originally developed for US NAVY, specialized in plant equipment monitoring and diagnostics.

Performance evaluation criteria are also defined during rule tuning job creation, which can be multiple objectives. For anomaly detection rules, it is typically to maximize probability of detection (POD) and minimize false alarm volume (FAV). For vibration rule, an additional criterion is added to maximize rule enabling coverage. A weighting mechanism can be established to merge different criteria into an overall objective function.

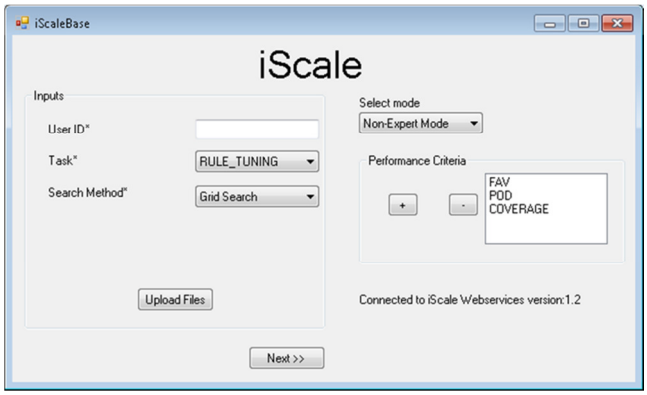

Figure 6 - iScale Rule Tuning Interface

4.2.3. Tuning Execution

Once a rule tuning job is submitted by user, iScale will create a unique job ID to trace a specific task. It will also manage job schedule, data file transfer, resource allocation, and configure rule execution and evaluation. Rule configuration search space will be traversed based on the optimization method selected by user. The execution of search is performed on a number of compute nodes simultaneously.

Execution time varies depending on the data set size, search space size, and number of available nodes. A user may login to the iScale job status checking screen to monitor job execution progress, modify or abort submitted job.

After search completed, iScale will aggregate results from different computing node to rank and summarize final result. Based on user configuration, optimization can be performed in either fleet level or unit level. Both summary result and detailed raw result are maintained for user review.

4.2.4. Rule Deployment

The final result provided to user includes the optimized rule configuration parameters and the corresponding evaluation criteria metrics. iScale can be configured as a fully integrated component within analytics development system. The integration is fairly simple since only the optimized rule configuration parameters are required to be deployed. After user review, the validated rule configuration information can be directly deployed to production system.

4.3. Rule Tuning Results

The data: 6-month of historical data for 18 turbines of the 7FA fleet were retrieved from our database and used for demonstrating iScale rule tuning. The data has a sample rate of once per minute, which leads to approximately 259k (6*30*24*60) data points per turbine unit. Out of the 18 turbine units, eight of them are considered as POD cases, i.e., have valid events at some point of the 6-month period. The rest of the 10 turbines units have no valid events.

The search grid for grid search: As discussed in Section 4.2.1, four rule parameters were selected as the tunable parameters. Due to IP issue, we are not allowed to give details of the specific tunable parameters. Here we designate the four tunable parameters as v_1, v_2, v_3, and v_4. We use 3 levels for each of the parameters and we arrive in grid with 3^4=81 grid points (or DOE experiments).

The configuration for DE optimization: For DE optimization, we integrate the ECJ package (http://cs.gmu.edu/~eclab/projects/ecj/) into the iScale framework. The population size and number of generations are set to 50 and 20, respectively. Other DE parameters (e.g., mutation and crossover rates) are set to ECJ default values.

Table 1: Rule tuning results

	Original design	After tuning
design parameters	[20,0.04, 0.3,0.4]	[10,0.04, 0.5,0.4]
POD	100%	100%
FAV	255	12

Rule tuning performance results: Table 1 shows rule tuning results using grid search. For comparison purpose, also shown in Table 1 are the performance metrics (POD and FAV) for the default setting. As seen from the table, rule tuning reduces the number of false alarms from 255 to 12 for the 7FA fleet data concerned in the paper. The reduction of false alarms improves monitoring engineer's productivity and prevents unnecessary inspection or troubleshooting. We have to point out that the DE optimization is still a work in progress, and we would like to share the results in a later time.

5. CONCLUSIONS

Gas turbine health monitoring is critical in preventing costly unplanned maintenance and in reducing life-cycle costs of power plant operations. Currently a great majority of anomaly detection engines are rule-based; and the rules and their associated thresholds/constants are initially designed based on domain and engineering knowledge and manually modified based on the rules' performance in the field. To improve fault detection performance (accuracy and robustness), systematical and efficient approaches allowing for optimally determining rules (rule discovery) and their associated constants (rule tuning) are needed. This paper is our initial effort towards addressing the need. Specifically we propose a way to automatically find optimal rule constants based on historical data; that is, we attempt to address the rule tuning need. Realizing that automated rule tuning can be computationally expensive and infeasible to complete in reasonable time, in this paper we leverage our recently-developed scalable learning framework - iScale that allows for automatically distributing rule tuning tasks to a large number of cloud computers. Such distribution not only dramatically speeds up tuning process, but also enables us to handle big size of historical data for tuning. In this paper we also explore different search methods to make rule tuning more efficient and effective.

By tuning the vibration rule, a real-world gas turbine detection rule, we demonstrate that the proposed rule tuning can be effective and efficient. The iScale-enable rule optimization not only eliminates the needs for manual tweaking thus a productivity gain for rule development, but also enables fully automated rule deployment and future adaptation, reduces the overall rule life cycle maintenance cost.

In future we would like to explore other optimization methods to further improve efficiency and effectiveness of rule tuning. It is also our great interest to extend our current automated rule tuning to automated rule discovery.

NOMENCLATURE

CSA	contractual service agreement
DE	differential evolution
EA	evolutionary algorithm
FAV	false alarm volume
POD	probability of detection
MOO	multi-objective optimization

REFERENCES

Conn, K. Scheinberg, and P. L. Toint (1997), On the convergence of derivative-free methods for unconstrained optimization, in *Approximation theory and optimization*, M. D. Buhmann and A. Iserles, eds., Cambridge, 1997, Cambridge University Press, pp. 83-108.

Das, S. and Suganthan, P.N. (2011), "Differential Evolution: A Survey of the State-of-the-Art," IEEE Trans. On *Evolutionary Computation,* vol.15, no.1, pp.4-31, Feb. 2011.

Marler, R.T. and Arora, J.S. (2004), Survey of multi-objective optimization methods for engineering, Structural and Multidisciplinary Optimization, 26 (6) (2004), pp. 369–395.

Rao, S. (2009), *Engineering Optimization: Theory and Practice*. 4th Edition, John Wiley & Sons, Inc., Hoboken, New Jersey.

Reyes-Sierra, M. and Coello, C.A.C. (2006), Multi-objective Particle Swarm Optimizers: A Survey of the State-of-the-Art, International Journal of Computational Intelligence Research, Vol.2, No.3, 2006, pp.287-308.

Storn, R. and Price, K. (1997), "Differential Evolution, A Simple and efficient Heuristic Strategy for Global Optimization over Continuous Spaces", Journal of Global Optimization, Vol. 11, pp. 341-359.

Xue, F., Sanderson, A. C., and Graves, R. J. (2003), "Pareto-based multiobjective differential evolution." *Proc., 2003 Congress on Evolutionary Computation (CEC'2003)*, Vol. 2, IEEE, New York, 862–869.

Yan, W., Iyer, N., Bonissone, P. and Varma, A. (2011), "iScale – Next Generation Framework for Creating Machine Learning Models", GE Global Research Center Whitepaper 2011.