

An Evolutionary Process for Designing and Maintaining a Fuzzy Instance-based Model (FIM)

Piero P. Bonissone, Anil Varma, Kareem S. Aggour
General Electric Global Research Center
Schenectady, NY 12309, USA
E-mail: {bonissone, varma, aggour}@research.ge.com

Abstract—We illustrate the typical life cycle of a fuzzy knowledge-based model, from its development, testing, optimization, and deployment, through the maintenance of its knowledge base. After a brief review of related work on a static problem, the underwriting of insurance applications, we focus on the design, implementation, and updating of a Fuzzy Instance-based Model (FIM) for prediction and classification in a dynamic environment, illustrated by a fleet selection problem. After formalizing the FIM, we describe its evolutionary-based design process and evaluate its performance. Within this example of asset selection, we show the accuracy of the evolved FIM's and their performance loss when we did not update them every six months. Finally, we advocate the use of evolutionary search intertwined with local search to further improve model life cycle.

I. INTRODUCTION

As discussed in [1-2], the success of *hybrid soft computing (SC)* systems has been driven by the synergy derived from the integration of their constituent technologies. This integration provides complementary reasoning and search methods that enable us to combine domain knowledge with empirical data. The domain knowledge is typically a combination of first principles and empirical knowledge, which is often incomplete and sometimes erroneous. The available data are typically a collection of input-output measurements representing instances of the system's behavior, and are generally incomplete and noisy. Soft computing is a flexible framework in which we can find a broad spectrum of design choices to perform the integration of knowledge and data to develop flexible computing tools and solve complex problems.

A. Need to Support Model Maintenance

In real-world applications, before deploying a model in a production environment we must address the model's entire life cycle, from design and implementation, to validation, tuning, production testing, use, monitoring and maintenance. Specifically, we need to keep the model vital (e.g., non-obsolete) and adaptable. There are two main reasons to focus on maintenance. Over the life cycle of the model, maintenance costs are the most expensive (as software maintenance is the most expensive component). Second, when dealing with mission-critical software we need to guarantee continuous operations or at least rapid recovery from failures to avoid lost revenue and other business costs. Given the role played by model maintenance, we cannot afford for it to be an after-

thought, but rather it should be an integral part of the design process. For efficiency we also want to minimize the amount of human intervention required [3-5]. In this paper we limit our discussion to supervised learning with long time-scale requirements for model updates (measured in months).

B. Paper Structure

In Section II we describe typical soft computing models, some key issues in the model life cycles, and common search methods that could be used to generate the models. After reviewing the design of a static discrete classifier for insurance underwriting, we focus on the design of a model for prediction and classification in a dynamic problem. We define a set of experiments in this context, formalize the description of a Fuzzy Instance-based Model (FIM), and explain its evolutionary design. Finally, we analyze the results of the experiments, compare the evolved FIM models with the static ones, and provide insights for future work.

II. MODELS, FUZZY MODELS, AND MODEL LIFE CYCLE

A. Model Generation

A model is characterized by its *representation* (structure and parameters) and its associated *reasoning mechanism*, which is usually related to the representation. The generation (and updating) of a model requires a *search* method to define the model's representation and to characterize its reasoning mechanism.

In the development of classification models we usually face a design trade-off between *accuracy* and *interpretability* [6]. When using SC techniques, the equation "**model = structure + parameters + search**" acquires a new connotation, as we can leverage a much richer repertoire to represent the structure, tune the parameters, and iterate over this process [2]. Such a repertoire enables us to choose among different trade-offs between the model's accuracy and interpretability. For instance, one approach aimed at maintaining the model's transparency usually starts with *knowledge-derived linguistic models*, in which domain knowledge is translated into an initial structure and parameters. The model's accuracy is further improved using global or local *data-driven search methods* to tune the structure and/or parameters. An alternative approach, aimed at building more accurate models, might start with *data-driven search methods*. Then we can embed domain knowledge into the search operators to control or limit the search space, or to maintain the model's interpretability. Post-

processing approaches can also be used to extract explicit structural information from the models. The commonalities among these models are the tight integration of knowledge and data leveraged in their construction, and the loose integration of their outputs, exploited in their off-line use. This characterization of a model is summarized in Table I.

TABLE I
REPRESENTATION, REASONING MECHANISMS, AND DESIGN SEARCH METHODS FOR VARIOUS MODELING TECHNIQUES

Modelling Technique	Linear Diff. Eq. (LDE)	Bayesian Nets (BBN)	Neural Nets (NN)	Fuzzy Systems (TSK)	Instance-based
Model Structure	Order	Topology	Topology	Rule Set	Attribute Space
Model Parameters	Coefficients	Prior Prob. Cond Prob.	Biases Weights	Term Sets Scaling Factors, Coefficients	Attribute Weights & Similarity Parameter
Reasoning Mechanism	Solve Eqs: Closed Form or Approx.	Node eval. & Propagation	Node eval. & Propagation	Node eval. & Propagation	Local Model eval. & Output Aggreg.
Design Search Method	First Princ. Energy-based Method	Manual, Evolut. Alg. (EA) Expect. Max. (EM)	Manual EA Backpr. Conj. Grad.	Manual EA Back-prop ...	Manual EA ...

B. Evolutionary Search in the Model Design Space

Regardless of the underlying model representation, it is crucial to have a systematic, robust, global search method to define the models' structure and parameters. Given a fixed model structure, using a wrapper approach, Evolutionary Algorithms (EA's) can be used to determine the inputs and pre-processors (attribute selection, weighing, and construction) as well as the parameter values required by the models. The appealing aspect of this approach is the ease of incorporating knowledge into the EA to control the search [4]. Furthermore, the same methodology can be applied after deployment to retune and or reconfigure the model, hence extending its vitality. It is also possible to intertwine global search (more robust but less efficient) with local searches such as greedy induction, gradient-based techniques, etc. in the quest to design and maintain better models. In the following sections, we will illustrate the application of EA's in the development and maintenance of optimal fuzzy classifiers.

C. The Model Maintenance Problem

Figure 1 shows the typical evolution of a model from the formulation of its underlying theory. Refined by experiments and limited by a set of axioms, we develop a computational tool with applicability conditions derived from the axioms and computational limitations. Finally we generate a model with that tool, by following a rapid prototype methodology as shown in the lower left portion of Figure 1.

Let's ask ourselves the question: "What is wrong with this picture?" The answer can be found in the inherent interactions present in the cascading process used to design and deploy

Knowledge-based Systems or Expert Systems. The *human-in-the-loop* component, although useful in developing prototypes, severely impairs their maintainability, limits the frequency, quality, and cost of updates, etc. Therefore, this architecture does not scale-up efficiently, nor does it allow for easy model maintenance. This was a common problem that impacted the life expectancy of expert systems deployed in the 80's [7].

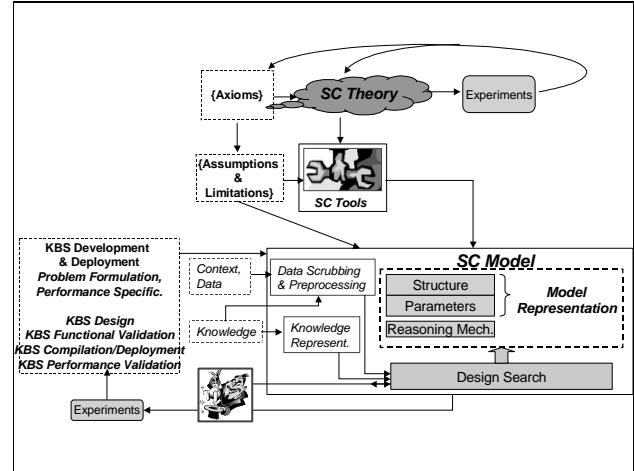


Fig. 1. Typical Model Design: Theory→Tool→SC Model Rapid Prototype

Figure 2 depicts a typical maintenance problem experienced in deploying AI expert systems that continues to be a critical issue. The maintenance of these models is essential to their long-term usefulness since, over time, the models' representation (structure and parameters) may become sub-optimal. Thus, we must be able to modify the model and reflect contextual changes without keeping the human in the design deployment cycle, as depicted in Figure 2.

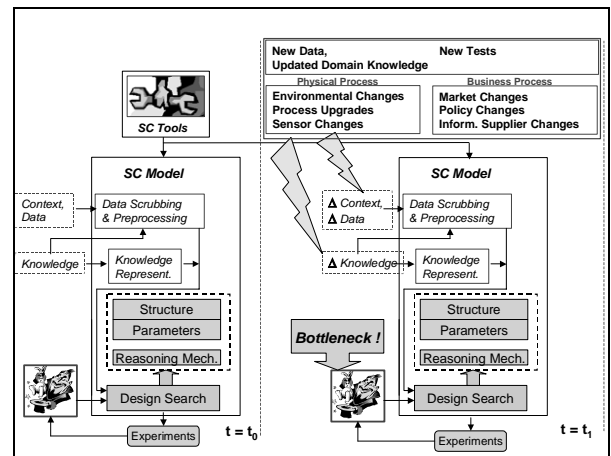


Fig. 2. The Bottleneck in the Update Process

D. Model Life Cycle Stages and Proposed Solution

Rather than handcrafting the model by manually searching and testing the model design space, we should inject as much domain knowledge as possible into evolutionary algorithms that perform these searches. In this case, we apply the

modifications reflected new data, new domain knowledge, or new policy changes to the Standard Reference Data (SRD) set that is used to derive the fitness function guiding the evolutionary search.

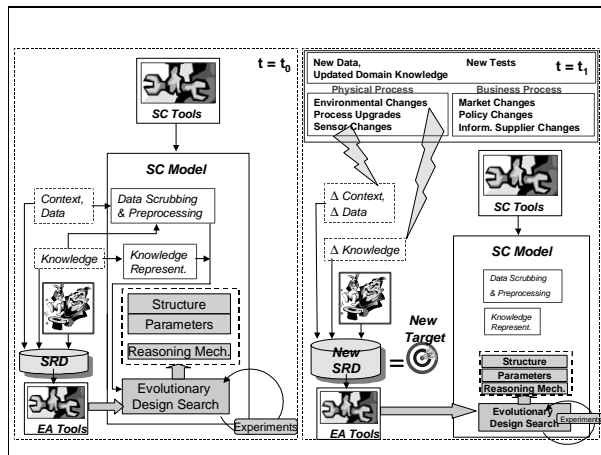


Fig. 3. Proposed Maintenance Solution for Supervised Learning Models

III. RELATED WORK

A. Evolutionary Design of Fuzzy Rule-based Reasoning Systems

We have developed and deployed a robust method for automating the tuning and maintenance of fuzzy decision-making systems. A Fuzzy Rule-based Classifier (FRC) was developed for insurance underwriting, a complex decision-making task that is traditionally performed by trained individuals. An underwriter must evaluate each insurance application in terms of its potential risk for generating a claim, such as mortality in the case of term life insurance. An application is compared against standards adopted by the insurance company, which are derived from actuarial principles related to mortality. Based on this comparison, the application is classified into one of the risk categories available for the type of insurance requested. The estimated risk, in conjunction with other factors such as gender, age, and policy face value, determine the appropriate price (premium) for the insurance policy. We abstracted the underwriting (UW) process as a discrete classifier that maps an input vector \bar{X} (containing discrete, continuous, and attribute variables that represents the applicant's relevant medical and demographic information), into a discrete decision space \bar{Y} (containing an ordered list of rate classes). A configurable multi-stage mutation-based evolutionary algorithm tunes the decision thresholds and internal parameters of the fuzzy rule-based system that places the insurance applications into risk categories. The tuneable parameters have a critical impact on the coverage and accuracy of the decision-making, and a reliable method to optimally tune these parameters is critical to the accuracy and maintainability of these systems. This application, described in [8-9], exemplifies a classification problem in an almost static environment since UW policies do not vary quickly other time.

IV. EVOLUTIONARY DESIGN FOR FUZZY INSTANCE-BASED MODELS (FIM'S)

A. The Fleet Selection Problem

We analyze the task of selecting the most reliable units within a fleet of locomotives and formulate it as a prediction and classification problem. The focus of our work is mission reliability. This can be broadly defined as: given a mission of duration X days, what percentage of units assigned to that mission are able to complete the mission without a critical failure. This application exemplifies a prediction and classification problem in a dynamic, complex environment. Locomotives, tanks, and aircraft vary considerably across different phases of their life cycle. Assets that are identical at the time of manufacture 'evolve' into somewhat unique systems based on their usage and maintenance history. Utilizing these assets efficiently requires a) the ability to create a model characterizing their expected performance, and b) keeping the model updated as the behavior of the underlying assets change.

1) *Data Sources and Data Segmentation*: To train our models and validate our experiments we used maintenance and utilization data collected from four sources: 1) Locomotive Design & Engineering Data from GE Rail; 2) Locomotive Diagnostics Data from GE Rail EOA™ remote monitoring and diagnostics service; 3) Locomotive Maintenance Data from Repair Shops; and 4) Locomotive Utilization data from a selected railroad.

We wanted to determine how environmental, operational, and maintenance changes could affect our experiments and whether our learning techniques could adapt to such changes. Furthermore, we needed to assess how incremental data acquisition could improve our learning techniques' performance. Thus, we decided to create three data slices, on *May 22, 2002, November 1, 2002, and May 1, 2003*. The size of the fleet increased over time (from 262 to 634 and 845 units, respectively), as new units were placed in service and as we started collecting more utilization and maintenance data from existing units. As a result, the number in the top 20% of performers increased with the size of the fleet to 52, 127, and 169 units, respectively. These three data slices were used to determine the *target* units for our selection. For each data slice we constructed three sets of targets, i.e., ground truth for the experiments: 1) the *best 20% past performers*; 2) the *best 52 past performers*; 3) the *best 20% future performers*. We identified the first two target sets by sorting the units in decreasing order using the *median time-between-failure* of past operational durations, until we identified the best 20% or the best 52 units. We identified the last target set by sorting the units in decreasing order using the duration of the *first period of operation* after the data slice.

2) *Performance Metrics and Baselines*: The primary metric was the ability of a classifier to select the best N units for any given time slice (i.e., its precision). We investigated two approaches to define the top ' N ' units. *Fixed Percentage Approach*: The classifier selects the top 20% of the units. The actual number increased with fleet size as we progressed from slice 1 to slice 3. *Fixed Number Approach*: The classifier selects a constant number of units. We used 20% of the first slice, i.e. 52 units. As the size of the fleet increased the

selection task became more difficult, as 52 units represented 8% and 6% of the fleet size in slices 2 and 3.

We used two baselines to measure the increase in capability provided by the peer-based algorithms. *Random*: The first baseline measured the expected performance if selection of the best ‘N’ units was done randomly. This obviously represented a worst-case scenario. *Heuristics*: The second baseline calculated the best performance achieved by using a single (or multiple) heuristic to rank the fleet and pick the top N. In this approach, the best performance achieved from any one of all the heuristics was used as the baseline.

B. Fuzzy Instance-based Models (FIM's)

Instance-Based Reasoning (IBR) relies on a collection of previously experienced data stored in their raw representation. Unlike Case-Based Reasoning (CBR), they do not need to be refined, abstracted and organized as cases. Like CBR, IBR is an analogical approach to reasoning since it relies upon finding previous instances of *similar* problems and uses them to create an ensemble of local models. Hence the definition of similarity plays a critical role in the performance of IBR's. Typically, similarity will be a dynamic concept and will change over the use of the IBR. Therefore, it is important to apply learning methodologies to define and adapt it. Furthermore, the concept of similarity is not crisply defined, creating the need to allow for some degree of vagueness in its evaluation. We addressed this issue by evolving the design of a similarity function in conjunction with the design of the attribute space in which the similarity was evaluated. Specifically, we used the following four steps: **1) Retrieval** of similar instances from the database (DB); **2) Evaluation of similarity measures** between the probe and the retrieved instances; **3) Creation of local models** using the most similar instances (weighted by their similarity measures); **4) Aggregation of outputs** of local models to probe. A description of the FIM model and some preliminary results were described in [13]. We will summarize the four steps and analyze a complete set of recent results.

1) Retrieval: We look for all instances in the DB that have a behavior *similar* to the probe. These instances are the potential peers of the probe. The peers and probe can be seen as points in an n -dimensional feature space. For instance, let us assume that a probe Q is characterized by an n -dimensional vector of feature values \bar{X}_Q , and $O(Q)=[D_{1,Q}, D_{2,Q}, \dots, D_{k(Q),Q}]$ the history of its operational availability durations:

$$Q = [\bar{X}_Q; O(Q)] = [x_{1,Q}, \dots, x_{n,Q}; D_{1,Q}, \dots, D_{k(Q),Q}] \quad (1)$$

Each other unit u_i in the fleet has a similar characterization:

$$u_j = [\bar{X}_j; O(u_j)] = [x_{1,j}, x_{2,j}, \dots, x_{n,j}; D_{1,j}, D_{2,j}, \dots, D_{k(j),j}] \quad (2)$$

For each dimension i we define a *Truncated Generalized Bell Function*, $TGBF_i(x_i; a_i, b_i, c_i)$, centered at the value of the probe c_i , which represents the degree of similarity along that dimension. Specifically:

$$TGBF_i(x_i; a_i, b_i, c_i) = \begin{cases} \left[1 + \left| \frac{x_i - c_i}{a_i} \right|^{2b_i} \right]^{-1} & \text{if } \left[1 + \left| \frac{x_i - c_i}{a_i} \right|^{2b_i} \right]^{-1} > \varepsilon \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where ε is the truncation parameter, e.g. $\varepsilon = 10^{-5}$.

Since the parameters c_i in each $TGBF_i$ are determined by the values of the probe, each $TGBF_i$ has only two free parameters, a_i and b_i , to control its spread and curvature. In a coarse retrieval step, we extract an instance in the DB if all of its features are within the *support* of the $TGBF_i$'s. Then we formalize the retrieval step. $P(Q)$, the set of potential peers of Q, is composed of all units within a range from the value of Q: $P(Q) = \{u_j, j = 1, \dots, m \mid u_j \in N(Q)\}$ where $N(Q)$ is the neighborhood of Q, defined by the constraint $\|x_{i,Q} - x_{i,j}\| < R_i$ for all potential attributes i for which the corresponding weight is non-zero. R_i is half of the support of the $TGBF_i$, centered on the probe's coordinate $x_{i,Q}$.

2) Similarity Evaluation: Each $TGBF_i$ is a membership function representing the partial degree of satisfaction of constraint $A_i(x_i)$. Thus, it represents the *closeness* of the instance around the probe value for that particular attribute. For a given peer P_j , we evaluate the function $S_{i,j} = TGBF_i(x_{i,j}; a_i, b_i, x_{i,Q})$ along each potential attribute i . The values (a_i, b_i) are design choices manually initialized, and later refined by the EA's. Since we want the most similar instances to be the closest to the probe along *all* n attributes, we use a similarity measure defined as the intersection of the constraint-satisfaction values. Furthermore, to represent the different relevance that each criterion should have in the evaluation of similarity, we attach a weight w_i to each attribute A_i . Therefore, we extend the notion of a similarity measure between P_j and the probe Q as a weighted minimum operator:

$$S_j = \text{Min}_{i=1}^n \{ \text{Max}[(1 - w_i), S_{j,i}] \} \quad (4)$$

$$= \text{Min}_{i=1}^n \{ \text{Max}[(1 - w_i), TGBF_i(x_{i,j}; a_i, b_i, x_{i,Q})] \}$$

where $w_i \in [0,1]$. The set of values for the weights $\{w_i\}$ and parameters $\{(a_i, b_i)\}$ are critical design choices that impact the proper selection of peers. In this section we assume a manual setting of these values. In the next subsection, we will explain their selection using evolutionary search.

3) Creation of Local Models: The idea of avoiding pre-constructed models and creating a local model when needed can be traced back to memory-based approaches [10-11] and lazy-learning [12]. Within the scope of this paper, we will focus on the creation of local predictive models used to forecast each unit's remaining life. First, we use each local model to generate an estimated value of the predicted variable. Then, we use an aggregation mechanism based on the similarities of the peers to determine the final output.

For instance, let us assume that for a given probe Q we have retrieved m peers, $P_j(Q)$, $j=1, \dots, m$. Each peer $P_j(Q)$ has a similarity measure S_j with the probe. Furthermore, each peer P_j has a track record of operational availability between failures $O(P_j) = [D_{1,j}, D_{2,j}, \dots, D_{k(j),j}]$. Each peer $P_j(Q)$ will have $k(j)$ availability pulses in its track history. For each peer P_j , the goal is to determine the duration of the *next* availability duration $D_{k(j)+1,j}$. Then we want to combine the prediction of all the peers $\{D_{k(j)+1,j}\}$ ($j=1, \dots, m$) to estimate the availability duration for the probe Q. Since there were not enough historical data to reliably generate local regressions, we experimented with simpler models such as averages and

medians. We determined that the most reliable way of generating the next availability duration $D_{k(j)+1,j}$ from the operational availability vector $O(P_j)=[D_{1,j}, D_{2,j}, \dots, D_{k(j),j}]$ was to use an exponential average that gives more relevance to the most recent information, namely:

$$\begin{aligned} D_{k(j)+1,j} &= \bar{D}_{k(j),j} = \alpha \times D_{k(j),j} + (1-\alpha) \times \bar{D}_{k(j)-1,j} \\ &= (1-\alpha)^{k(j)-1} D_{1,j} + \sum_{i=2}^{k(j)} (1-\alpha)^{k(j)-i} \times \alpha \times D_{i,j} \end{aligned} \quad (5)$$

Again, critical to the performance of this model is the choice of the value of α and its best value will be determined by evolutionary search.

4) *Aggregation Mechanism*: We need to combine the individual predictions $\{D_{k(j)+1,j} \mid j=1, \dots, m\}$ of the peers $P_i(Q)$ to generate the prediction of the **next** availability duration, $D_{Next,Q}$ for the probe Q. To this end, we compute the weighted average of the peers' individual predictions using their normalized similarity to the probe as a weight, namely:

$$D_{Next,Q} = \frac{\sum_{j=1}^m S_j \times D_{k(j)+1,j}}{\sum_{j=1}^m S_j} \quad (6)$$

Given the critical design roles of the weights $\{w_i\}$, the parameters $\{(a_i, b_i)\}$, and the exponent α , it was necessary to create a methodology that could generate their best values according to our metric, i.e., classification precision.

C. Evolutionary Design

After testing several manual peer-based models, we decided to use evolutionary searching to develop and maintain the fuzzy instance based classifier. In the wrapper methodology detailed in [8], we defined the use of Evolutionary Algorithms to tune the parameters of a classifier used to underwrite insurance applications. In this application, we extend evolutionary search beyond parametric tuning to include structural search, via attribute selection and weighting [14].

The EA's are composed of a population of individuals ("chromosomes"), each of which contains a vector of elements that represent distinct tuneable parameters within the FIM configuration. Examples of tuneable parameters include the range of each parameter used to retrieve neighbor instances and the relative parameter weights used for similarity calculation. The EA's used two types of mutation operators (Gaussian and uniform), and no crossover. Its population (with 100 individuals) was evolved over 200 generations.

Each chromosome defines an instance of the attribute space used by the associated classifier by specifying a vector of weights $[w_1, w_2, \dots, w_n]$. If $w_i \in \{0,1\}$, we perform *attribute selection*, i.e., we select a crisp subset from the universe of potential attributes. If $w_i \in [0,1]$, we perform *attribute weighting*, i.e., we define a fuzzy subset from the universe of potential attributes

$$[w_1 w_2 \dots w_n][(a_1, b_1), (a_2, b_2), \dots, (a_n, b_n)][\alpha] \quad (7)$$

where $w_i \in [0,1]$ for attribute weighting and

$w_i \in \{0,1\}$ for attribute selection

n = Cardinality of universe of features U , $|U| = n$

$d = \sum_i^n w_i$ (fuzzy) cardinality of selected features

(a_i, b_i) = Parameters for GBF_i

α = Parameter for Exponential Average

The first part of the chromosome, containing the weights vector $[w_1, w_2, \dots, w_n]$, defines the attribute space (the FIM structure) and the relevance of each attribute in evaluating similarity. The second part of the chromosome, containing the vector of pairs $[(a_1, b_1), \dots, (a_i, b_i), \dots, (a_n, b_n)]$ defines the parameter for retrieval and similarity evaluation. The last part of the chromosome, containing the parameter α , defines the forgetting factor for the local models. The fitness function is computed using a *wrapper* approach [13-14]. For each chromosome, represented by (7), we instantiate its corresponding FIM. Following a *leave-one-out* approach, we use the FIM to predict the expected life of the probe unit following the four steps described in the previous subsection. We repeat this process for all units in the fleet and sort them in decreasing order, using their predicted duration $D_{Next,Q}$. We then select the top 20%. The fitness function of the chromosome is the precision of the classification, $TP/(TP+FP)$.

V. RESULTS: DYNAMIC VERSUS STATIC MODELS

Successfully deployed intelligent systems must remain valid and accurate over time, while compensating for drifts and accounting for contextual changes that might otherwise render their knowledge-base stale or obsolete. In this study, we performed three sets of experiments with dynamic and static models. The dynamic models are *fresher* models, re-developed at each time slice by using the methodology described in the previous section. The static models were developed at time slice 1 and applied, unchanged, at time slices 2 and 3. This comparison shows the benefit of automated model updating.

A. Experiments

1) *First Experiment (Top 20% Past Performers)*: We predict the best 20% of the fleet, based on their past performance. In this case a random selection would yield 20%.

2) *Second Experiment Set (Top 52 units Past Performers)*: Since the size of the fleet at each start-up time was different, we repeated the same experiments keeping the number of units constant (52 units) over the three start-up times instead of keeping a constant top 20%. Thus the random selection baseline changed from [20%-20%-20%] to [20%-8%-6%], i.e., 52/262=20%; 52/634=8%; 52/845=6%. Also, given the superior performance of the evolved peers over the manually constructed peers, we decided to use the optimized peer design instead of the manual design in all subsequent experiments.

3) *Third Experiment Set (Top 20% Future Performers)*: In the previous two experiments the targets for selection were the best-known units in the fleet based on past performance. Now we perform prediction and target the selection of the units with the best next-pulse duration. In this case a random selection would yield 20%.

B. Performance of Dynamic Models

For each time slice, we used the EA's to generate an optimized weighted subset of attributes, search parameters and forgetting

factor, to define the peers of each unit. We used the evolved peer approach to run three experiments.

1) *First Experiment: Top 20% Past Performers:*

Time slice 1 (fleet size = 262 units; top 20% = 52 units): Evolved Peers outperformed both manually-designed peers and heuristics/fleet-based approach: **48% vs. 41% vs. 32%**.

Time slice 2 (fleet size = 634 units; top 20% = 127 units): Evolved Peers outperformed both manually-designed peers and heuristics/fleet-based approach: **56% vs. 55% vs. 46%**.

Time slice 3 (fleet size = 845 units; top 20% = 169 units): Evolved Peers outperformed both manually-designed peers and heuristics/fleet-based approach: **60% vs. 54% vs. 50%**.

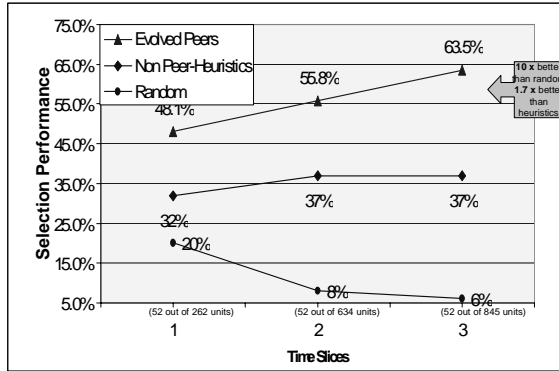


Fig. 4. Dynamic Models 2nd Experiment Set: (52 Units; Past Performers)

2) *Second Experiment Set: Top 52 units Past Performers:*

Time slice 1 (fleet size = 262 units; top 52 units = 20%): Evolved Peers outperformed heuristic/fleet-based approach: **48.1% vs. 32% (vs. 20% random selection)**.

Time slice 2 (fleet size = 634 units; top 52 units = 8%): Evolved Peers outperformed heuristic/fleet-based approach: **55.8% vs. 37% (vs. 8% random selection)**.

Time slice 3 (fleet size = 845 units; top 52 units = 6%): Evolved Peers outperformed heuristic/fleet-based approach: **63.5% vs. 37% (vs. 6% random selection)**.

3) *Third Experiment Set: Top 20% Future Performers:*

Time slice 1 (fleet size = 262 units; top 20% = 52 units): Evolved Peers outperformed both heuristic/fleet and unit's own history-based approach: **43% vs. 32% vs. 27%**.

Time slice 2 (fleet size = 634 units; top 20% = 127 units): Evolved Peers outperformed both heuristic/fleet and unit's own history-based approach: **42.2% vs. 42% vs. 34%**.

Time slice 3 (fleet size = 845 units; top 20% = 169 units): Evolved Peers outperformed both heuristic/fleet and unit's own history-based approach: **55% vs. 36% vs. 34%**.

The last two experiments' results are shown in Figure 4 and 5.

C. *Performance of Static Models*

1) *First Experiment Set: Top 20% Past Performers:* The models developed in time slice 1 exhibited poor precision when applied to the remaining two time slices: **48.1% → 48.41% → 47.92%**. As a comparison, the dynamic models had: **48.1% → 55.60% → 60.40%** precision.

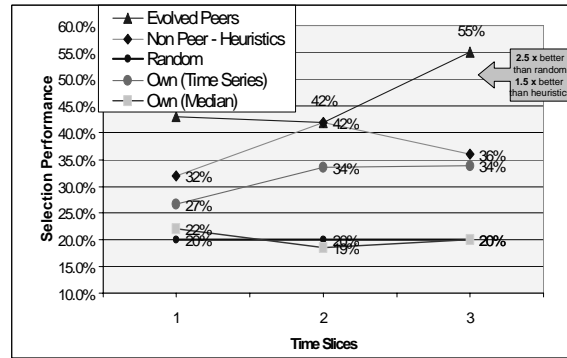


Fig. 5. Dynamic Models 3rd Set of Experiments (20%; Future Performers)

2) *Second Experiment Set: Top 52 units Past Performers:*

Again, the time slice 1 models applied to the remaining two slices were inferior to the dynamic model: **48.1% → 50.00% → 46.15%** compared with the refreshed, dynamic models: **48.1% → 55.80% → 63.50%**.

3) *Third Experiment Set: Top 20% Future Performers:*

This was the most difficult experiment and the original models showed significant deterioration over time: **42.85% → 25.86% → 24.81%**. In contrast, the dynamic models exhibited more robust precision: **42.85% → 42.24% → 54.88%**.

VI. CONCLUSIONS AND FUTURE WORK

A. Manually-Designed versus Evolved FIM's

The peers designed by the Evolutionary Algorithms provided the best accuracy overall:

–60.3% = over 3 x random, 1.2 x heuristics on selection for top 20% (*based on past performance*)

–63.5% = over 10 x random, 1.7 x heuristics on selection for top 52 units (*based on past performance*)

–55.0% = over 2.5 x random, 1.5 x heuristics on selection for top 20% (*based on future performance*).

TABLE II
WEIGHTS AND SEARCH PARAMETERS FOR TIME-SLICE 3

Index	Feature	Weight	a	Range (a)	b	Range (b)
1	RY_Rec/Yr	9.11	5.6	[0-6]	3.73	[0.5-5]
2	RY_Rec/100K_Miles	8.81	5.3	[0-8]	2.57	[0.5-5]
3	RY_Rec/100K_Engine_Hrs	7.35	35.9	[20-45]	2.67	[0.5-5]
4	RY_Rec_Count	5.69	8.5	[2.5-12]	3.37	[0.5-5]
5	RY_Rec/100K_Eng_Hrs_Move	4.08	10.1	[5-30]	2.75	[0.5-5]
6	Tot_Rec_Count	1.33	18.7	[3-50]	3.20	[0.5-5]
7	R_Rec_Count	0.87	1.0	[0.5-8]	3.16	[0.5-5]

Dynamic criteria for peer recognition (by evolving attribute weighting) demonstrated the ability to adapt to changing operational and maintenance environments. The evolution of the search parameters $\{(a_i, b_i)\}$ provided an additional performance improvement, reducing the number of null queries while improving the overall precision of the classifier. Table II shows the value of the weights and search parameter settings for time-slice 3, while Figure 6 illustrates the evolution of the weights over the three time slices.

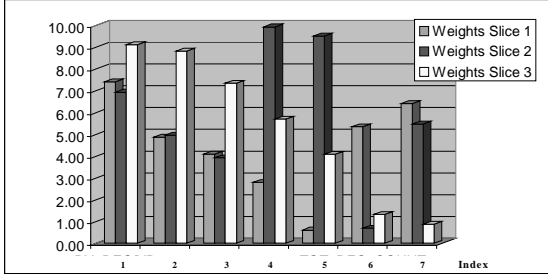


Fig. 6. Evolution of weights for the top seven variables over the three time slices (Index number described in Table II)

B. The Benefits of Updating the Models

The dynamic models performed dramatically better than the original models, which showed minor performance deterioration in experiment 2 and extreme performance drop in experiment 3. The updating of the models was achieved with minimal cost, as we automated the design search required to redesigning the more recent models.

C. Future Extensions

These experiments have shown the applicability of evolutionary algorithms, used in a *wrapper* approach, to select the best attributes for *representing peers* and to define similarity measures for *identifying the most similar peers* for a given unit. By evolving the models over different time slices, we have also shown our ability to dynamically adapt the neighborhoods of peers using incremental operational and maintenance data. In future work, we could extend the structural design of the attribute space (for the definition of peers). By using genetic programming in lieu of evolutionary algorithms, we could extend *attribute selection* and *weighting to attribute construction*. We could also improve the fitness function to trade off classifier accuracy and confidence by adding a measure of representation parsimony and finding Pareto fronts for different tradeoffs.

We could extend our approach by generating more complex local models for prediction. For instance, by using equation (5) in (2), we could summarize the history of each point by its exponential average, using the value of α determined by the EA's. The retrieved m points would now be:

$$u_j = [\bar{X}_j; y_j] = [x_{1,j}, x_{2,j}, \dots, x_{n,j}; y_j]$$

where $y_j = \bar{D}_{k(j),j} = \alpha \times D_{k(j),j} + (1 - \alpha) \times \bar{D}_{k(j)-1,j}$.

By using similarity measures $S_{i,j} = TGBF_i(x_{i,j}; a_i, b_i, x_{i,Q})$ as kernel functions for each dimension i , we could compute a kernel-based regression [10]. This hybrid approach would leverage local search methods to obtain the parameters of the kernel-based regression within each EA's trial.

By evolving the models over different time slices, we have shown a role that evolutionary search can play in the maintenance of models, as EA's provide a mechanism for model updating, preventing model obsolescence, and supporting model lifecycle [3,5].

These efforts represent the *first steps* of an approach aimed at supporting the model lifecycle. Still, many issues remain unresolved. In the case of supervised learning based models, by intertwining local and global search methodologies [15] we leverage local search's efficiency while benefiting from global search's resilience to local minima. In the case of unsupervised learning, we need to understand how to integrate evolutionary search with techniques such as SOM, ICA, or PCA to re-evaluate structural changes. Finally, the time constants inherent to the physical system that we want to model will dictate whether we can use a batch, re-design approach to update the model on a weekly or monthly basis, or whether we need an online, incremental approach to modify the model at run-time.

ACKNOWLEDGEMENTS

This work was funded by DARPA, through contract CACI 621-04-S-0031.

REFERENCES

- [1] P. Bonissone, "Soft Computing: the Convergence of Emerging Reasoning Technologies", *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 1, no. 1, pp 6-18, 1977.
- [2] P. Bonissone, Y-T. Chen, K. Goebel, and P. Khedkar, "Hybrid Soft Computing Systems: Industrial and Commercial Applications", *Proceedings of the IEEE*, vol. 87, no. 9, pp 1641-1667, 1999.
- [3] P. Bonissone, "The Life Cycle of a Fuzzy Knowledge-based Classifier", *Proc. of 2003 North American Fuzzy Information Processing Society*, Chicago, IL, Piscataway, NJ: IEEE, pp. 488-494, 2003.
- [4] P. Bonissone, "Automating the Quality Assurance of an On-line Knowledge-Based Classifier By Fusing Multiple Off-line Classifiers", *Proc. IPMU 2004, Perugia (Italy)*. Rome, Italy: Universita' La Sapienza, pp. 309-316, 2004.
- [5] P. Bonissone, "Development and maintenance of fuzzy models in financial applications" in *Soft Methodology and Random Information Systems*, M. Lopez-Diaz, M. Gil, P. Grzegorzewski, O. Hryniewicz, J. Lawry, Eds. Berlin, Germany: Springer Verlag, pp. 50-66, 2004.
- [6] J. Casillas, O. Cordón, F. Herrera and L.Magdalenalena, (Eds.), "Interpretability Issues in Fuzzy Modeling, and Accuracy Improvements in Linguistic Fuzzy Modeling." *Studies in Fuzziness and Soft Computing*, Vol. 128-129, Springer-Verlag, 2003.
- [7] Wu C-H, Lee S-J and Chou, H-S, "Dependency analysis for knowledge validation in rule-based expert systems", *Proc. 10th Conference on Artificial Intelligence for Applications*, pp. 327 -333, 1994.
- [8] P. Bonissone, R. Subbu and K. Aggour "Evolutionary Optimization of Fuzzy Decision Systems for Automated Insurance Underwriting", *Proceedings of the IEEE Intern. Conference on Fuzzy Systems*, Honolulu, Hawaii, Piscataway, NJ: IEEE, pp 1003 - 1008, 2002.
- [9] A. Patterson, P. Bonissone, and M. Pavese, "Six Sigma Quality Applied Throughout the Lifecycle of and Automated Decision System", *International Journal of Quality and Reliability*, (to appear).
- [10] C.G. Atkeson, "Memory-based approaches to approximating continuous functions", in *Nonlinear Modeling and Forecasting*, M. Casdagli and S. Eubank, Eds. Harlow, UK: Addison Wesley, pp. 503-521, 1992
- [11] C.G. Atkeson, A. Moore, and S. Schaal, "Locally Weighted Learning", *Artificial Intelligence Review*, 11(1-5): 11-73, 1997.
- [12] H. Bersini, G. Bontempi, and M. Birattari, "Is readability compatible with accuracy? From neuro-fuzzy to lazy learning", *Proceedings in Artificial Intelligence 7*, C. Freksa, Ed. Berlin, Germany: Infix/Aka, pp. 10-25, 1998.
- [13] P. Bonissone and A. Varma, "Predicting the Best Units within a Fleet: Prognostic Capabilities Enabled by Peer Learning, Fuzzy Similarity, and Evolutionary Design Process" FUZZ-IEEE 2004, submitted.
- [14] A. Freitas *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, Springer-Verlag, Berlin, Germany: Springer-Verlag, 2002.
- [15] J.-M. Renders, H. Bersini, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways", *Proceedings 1st IEEE CEC*, Orlando, FL, NJ: IEEE, 1994. 312-317 vol.1.