

Selecting the Best Units in a Fleet: Performance Prediction from Equipment Peers

Anil Varma, Kareem S. Aggour, Piero P. Bonissone

GE Global Research
One Research Circle
Niskayuna, NY 12309
{varma, aggour, bonissone}@research.ge.com

Abstract. We focus on the problem of selecting the few vehicles in a fleet that are expected to last the longest without failure. The prediction of each vehicle's remaining life is based on the aggregation of estimates from 'peer' units, i.e. units with similar design, maintenance, and utilization characteristics. Peers are analogous to neighbors in Case-Based Reasoning, except that the states of the peer units are constantly changing with time and usage. We use an evolutionary learning framework to update the similarity criteria for peer identification. Results indicate that learning from peers is a robust and promising approach for the usually data-poor domain of equipment prognostics. The results also highlight the need for model maintenance to keep such a reasoning system vital over time.

1. INTRODUCTION

The problem of selecting the best units from a fleet of equipment occurs in many military and commercial applications. For example, given a specific mission profile, a commander may have to decide which five armored vehicles to deploy in order to minimize the chance of a breakdown. In the commercial world, rail operators often need to make decisions on which locomotives to use in a train traveling from coast to coast with time sensitive shipments. Asset selection for complex electromechanical equipment is often driven by heuristics and/or expert opinions. Some 'obvious' strategies include picking the newest, the most recently serviced, or the latest model equipment.

Long-term data that allows reliability and MTBF (mean time between failure) computations at the fleet and individual unit level can also drive such decisions. However, this work was motivated by the special needs of military equipment on new platforms. In the case of a new aircraft, tank, or ship, there is simply no long-term data to assess reliability across the vast range of potential missions. Second, the usage pattern of military equipment can be described as a sequence of 'pulses'—long periods of inactivity followed by relatively short periods of intense usage. Given the possibility of very sparse

deployment history on any individual unit, how can we best assess its feasibility for a new mission in a new environment and terrain?

We present an approach where the time-to-failure prediction for each individual unit is computed by aggregating its own track record with that of a number of ‘peer’ units—units with similarities along three key dimensions: system design, patterns of utilization, and maintenance history. The notion of a ‘peer’ is close to that of a ‘neighbor’ in CBR, except that the states of the peers are constantly changing. Odometer-type variables like mileage and age increase, and discrete events like major maintenance or upgrades occur. It is reasonable to assume that after every significant mission, the peers of a target unit may change based upon changes in both the unit itself, and the fleet at large. This is in contrast to a conventional diagnostic system such as the locomotive CBR system described by Varma and Roddy (1999), where, once stored in the case base, the case description remains static.

Our results suggest that estimating unit performance from peers is a practical, robust and promising approach. Two types of experiments were conducted—retrospective estimation and prognostic estimation. In the first experiment, we explore how well the median time-to-failure for any unit can be estimated from the equivalent median of its peers. In the second experiment, for a given instant in time, we predict the time to the next failure for each unit using the history of the peers.

Because we use estimates composed from peers, constructing an effective similarity criterion for peer selection is critical (as it is for any case-based reasoning system). However, because the elements in the case base are changing with time, systematically evaluating and updating the similarity criterion for peer selection is necessary. We use an evolutionary algorithm to tune the similarity criterion, and show that an evolutionary learning framework contributes significantly to keeping the reasoning process vital.

Section 2 provides an overview of the motivation for this work, the data sources, and the experimental setup. Section 3 reviews related work and approaches. Sections 4 and 5 focus on the system design, parameter optimization, and model maintenance using the evolutionary learning framework. Section 6 presents the results, and Section 7 contains our conclusions.

2. PROBLEM FRAMEWORK

2.1 Motivation

In military deployments, commanders often have to select a subset of available units from a fleet to last the duration of a mission in a self-sustained manner. Data from the National Training Center in Fort Irwin, California indicate that M1A1 tanks and Bradley vehicles have a non-mission-capable rate of 27% → 46% over a 7-day mission duration (‘pulse’) for half and full tempo operations respectively. The US Defense Advanced Research

Projects Agency (DARPA) approached GE Global Research to explore learning and reasoning methodologies to address this unit selection problem, characterized by sparse data over a wide variety of performance environments, making direct application of standard statistical techniques difficult. We seek to predict mission reliability by using a *collective* of equipment *peers* for a given unit, each with limited performance data.

The specific application domain is an extension of the locomotive diagnostics system described by Varma and Roddy (1999). GE Rail remotely monitors about 4000 locomotives, and uses a case base of fault logs to diagnose if any proactive maintenance is needed. The case base is populated with successful diagnoses verified by actual repair, represented as about a 3-day fault log supplying the input variables and the maintenance action code as the output variable. The cases are episodic, have definite beginning and ending times, and do not change once entered into the case base. The system has been operational for about 6 years.

In working with DARPA's steering committee, we determined that the extensive remote monitoring and diagnostics data accumulated by GE Rail could prove an excellent surrogate for conducting experiments for the selection phase of mission support.

<u>Data Category</u>	<u>Source</u>
1. Design & Configuration Information	GE Rail
2. Maintenance Information	
- Fault Codes	EOA Service
- Recommendations	GE Rail
- Repairs	GE Rail / Railroads
3. Utilization Information	Railroads

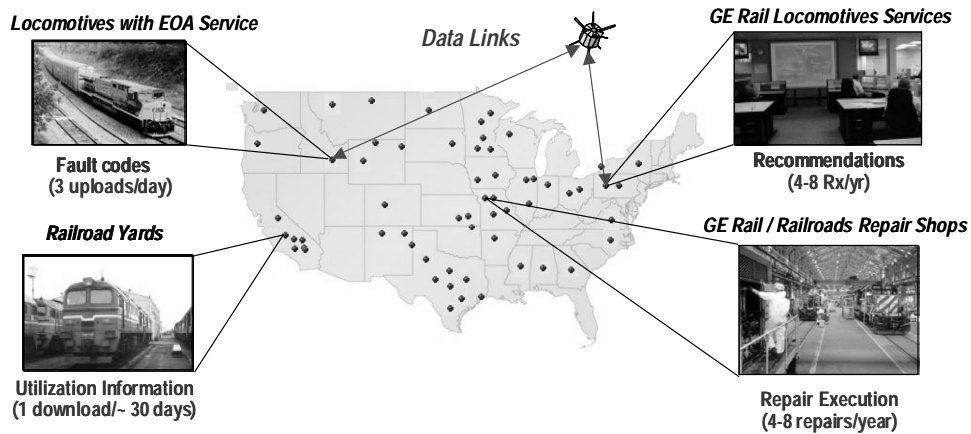


Fig. 1. Locomotive Design, Maintenance, and Utilization Data Sources

2.2 Data Sources

As shown in Figure 1, three distinct types of data are available for each locomotive:

1. **Design and Configuration:** This data was obtained from GE Rail as the original equipment manufacturer. This includes information about the locomotive model, in-service date, upgrades, options and configuration items.
2. **Maintenance:** This information was obtained from the GE Expert on Alert™ (EOA) center in Erie, Pennsylvania. Data from each locomotive is uploaded to the center three times a day. The EOA diagnostic tools analyze this data and if a problem is identified, a workflow case is created for review by a monitoring engineer. The expert can then issue a Red, Yellow or White recommendation (Rx) to the railroad, or choose to wait. A Red recommendation implies serious problems that need to be addressed in the next 3-5 days. A Yellow recommendation should be addressed in the next 7-14 days, and a White recommendation is usually informational with no impending failure expected.

Once a Red or Yellow recommendation is delivered, the actual repair is carried out soon afterwards. The measure of ‘time to failure’ from any given point in time is the time until the next repair.

Table 1 shows a sequence of Red recommendations issued on a particular unit. The Rx Close Date indicates the date of repair. Each entry in column 5 (days from previous Red or Yellow repair) indicates the length of time that the unit operated without a failure.

Table 1. Maintenance Recommendation Information

SERIAL NO	RECOMMENDATION DELIVERY DATE	URGENCY	RX CLOSE DATE	DAYS from previous RED or YELLOW repair
1001	7/28/2002 18:14	R	7/29/2002 21:39	12.72
1001	7/16/2002 23:03	R	7/17/2002 4:20	14.01
1001	7/2/2002 7:47	R	7/3/2002 4:01	15.78
1001	6/13/2002 9:25	R	6/14/2002 20:38	6.94

3. **Utilization:** Each locomotive records parameters on-board that are cumulative in nature, and consequently monotonic. Examples of these include, age, mileage, total megawatt hours developed, total hours moving, and total hours idle. About every 30 days, these are downloaded by the railroad and stored. Additional parameters can be computed from these values. Table 2 shows columns indicating the cumulative hours spent by a unit in any one of 8 ‘notches’ or gear positions. These, divided by total operating time, provide what percentage of time the unit spends in lower vs. higher gear positions—an approximation to an operating profile.

Table 2. Utilization Information

DOWNLOAD DATE	CUM N1 HRS	CUM N2 HRS	CUM N3 HRS	CUM N4 HRS	CUM N5 HRS	CUM N6 HRS	CUM N7 HRS	CUM N8 HRS	CUM BRAKE HRS	CUM ENGINE HRS MOVING	DELTA ENGINE HRS MOVING
5-Apr-03	361.4	148.6	136.5	102.5	91.2	72.6	46.6	254.0	274.8	1,488.13	31.8
2-Apr-03	351.6	142.6	130.8	100.4	89.5	71.3	46.1	253.5	270.5	1,456.30	43.4
28-Mar-03	340.5	138.4	127.0	97.4	86.5	68.8	44.3	247.4	262.6	1,412.90	116.5
...
28-Jul-01	259.0	113.1	97.7	79.5	72.1	55.9	36.4	176.0	196.4	1,086.24	0.0

We consolidated and scrubbed the data from GE Rail and utilization data from Union Pacific to generate a case base with 1,178 locomotives. A single data vector was associated with each unit, containing raw data such as age, mileage, and number of repairs/year as well as compound variables generated from the raw data (number of repairs per 100,000 miles, for example).

2.3 Experiment Description and Metrics

The data collected for the experiment spanned approximately two years. To simulate the military scenario of a sequence of missions, we selected three instances in time approximately six months apart, as indicated in Figure 2. We refer to these as time slices. The first time slice was on 22 May 2002. Treating this date as the present time, the case base had data on 262 units. By Slice 2, about six months later on 01 Nov 2002, the case base had data on 634 units, as several locomotives entered service or had at least two failure events to provide a time-to-failure computation. Slice 3 was set at 01 May 2003, and by this time data was available on 845 units. Once we imposed the requirement that each unit have at least two failures recorded, the number of usable units declined from 1178 to 965.

After consulting with domain experts, we restated the objective as: At the beginning of each time slice, select the best 20% of the locomotives, measured by their ability to operate the longest without requiring repair (starting at the beginning of the time slice). With this definition, the performance metric was easily computed. At Slice 1, with a fleet of 262 locomotives, a given algorithm would pick 52 locomotives that it determined to be the best. Because this was historical data, the locomotives could be ranked by how they actually performed, producing the top 52 “golden units” based on actual time to failure. The success rate was defined as the number of golden units selected by the algorithm divided by the total number of golden units. If a given algorithm, after picking 52 units, had 20 golden units in its pool, its performance would be $20/52 = 38\%$. Once the mission was over, the case base would be updated with any new fleet data, and the selection process would be repeated for the next mission.

For our experiment, the performance metric was computed once for each slice. With chronological data, Slice 2 had the benefit of seeing the aging of the Slice 1 units, the addition of some new units, as well as information on the performance of the algorithm on

Slice 1. Similarly, Slice 3 benefits from both Slice 1 and 2. We expected that with increasing information, the selection performance would improve.

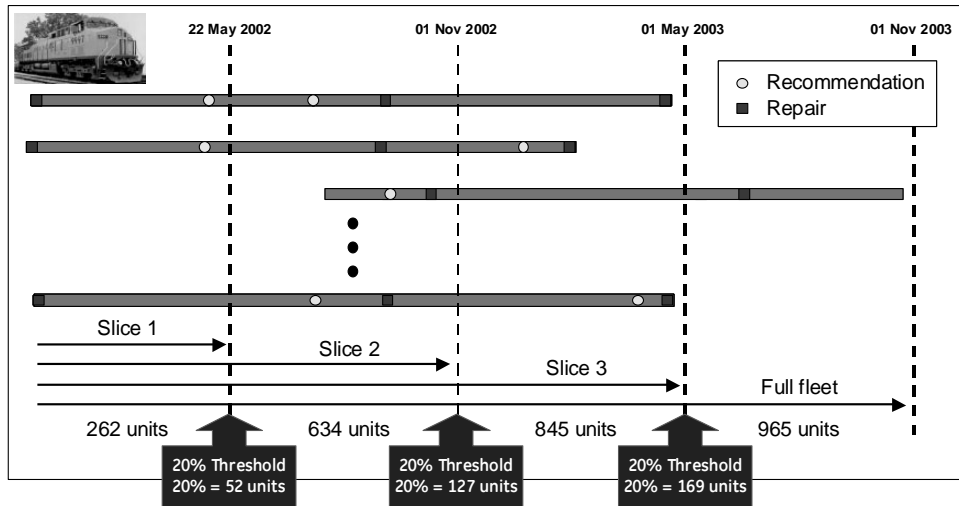


Fig. 2. Experiment Time Slices and Unit Counts

3. BACKGROUND

Three main concepts are embedded in this work. The first is the notion of estimating electromechanical equipment readiness from the performance of other similarly used and maintained units. This is especially applicable to military domains where any single unit may not get enough usage to build a long individual track record to estimate its fitness for a given mission. We address this through incremental learning from experience fragments drawn from fleet peers. This application is probably most relevant to the maintenance community, and we will not discuss it in detail in this paper.

The second concept of interest is the idea of viewing an equipment fleet as a case base whose cases evolve with time, with extended time histories describing their state. The notion of a 'case' in a case-based system is often episodic in nature. A case traditionally captures a set of attributes that define an episode of interest at a particular instant in time—whether a customer call, meal plan, or a failure requiring diagnosis, with an associated outcome. The episode, once captured as a case, is itself not expected to change (though its relevance calculation may be weighed by the age of the case).

We believe the notion of peers and peer-based-reasoning is a useful specialization of the general CBR approach when dealing with a case base of complex equipment. Representing and reasoning with time-extended cases has been discussed by a variety of

authors in different domains. The notion that the state of each case in a case base can be a function of time was referred to as ‘Continuous Case-Based Reasoning’ and identified as distinct from discrete, symbolic representations by Ram and Santamaria (1997). They describe Continuous CBR in the context of a driving task, where problem solving is incremental due to limited prior knowledge, and continuous adaptation and learning is essential to incorporate new experiences. An important issue raised by the authors is case representation—whether the entire experience to date is a single case, or if there should be a criterion that defines the scope of a single case. In our application, a major maintenance or overhaul could define such a splitting criterion. Jaczynski (1997) focuses on the retrieval aspects of cases that reflect time-extended situations. He distinguishes between sampled numeric series and event-based time series. In our application, each unit is associated with a sequence of maintenance events, and to that extent meets the second definition. While we have not made an attempt to characterize the series of maintenance events in a richer context, this is a logical extension of our work.

Schlaefler, Schröter, et al. (2001) and Fritsche, Schlaefler, et al. (2002) describe using CBR on a series of medical tests for kidney transplant recipients. Their data is described as a “series of infrequent measurements at irregular intervals.” They utilize dynamic time warping (Berndt and Clifford 1996) to normalize multiple series for similarity computation and retrieval. In our application, events in a locomotive’s history are not represented as a time series to be used for similarity matching. We use the ‘odometer’ approach, where the entire history is summarized in a state vector. We feel that this is a limitation, since the type and sequence of maintenance events, upgrades, and missions are likely to have a significant impact on readiness.

The third concept of interest is the importance of model maintenance, including both adaptation and optimization, and its special importance in peer-based reasoning. As time passes, the attributes of the object under consideration change, and so do its peers. This is in addition to the expected turnover in the case base as units are decommissioned and new units enter service. The learning task here is to update the ‘similarity metric’ or the criterion for peer selection.

Research in the CBR community has focused extensively on case based maintenance. Leake and Wilson (1998), in their review of CBR maintenance dimensions, point out that the indexing scheme is an integral part of the case base along with the cases themselves. Smyth (1998) describes a strategy for case deletion with minimal impact on performance—a pruning approach. Zhang and Yang (1998) also stress index maintenance to keep a CBR system current. They propose an iterative approach to weight refinement. An evolutionary algorithm handles the corresponding function in our application. Leake and Wilson (1999) explicitly address the situation when changing tasks and environment could render part of the case base obsolete or invalid. They identify problem-solution regularity as a basic premise of CBR and advocate monitoring performance over time to spot a decline in this measure. In our application, unit to peer *irregularity* over time is almost a given, and the analogous issue becomes how often to update the ‘peer selection’ or similarity metric to maintain performance while incorporating new knowledge. In our

experiments, similarity metric updates occur at each of the three time slices. Results presented later highlight the need for continual adaptation and parameter optimization.

4. SYSTEM IMPLEMENTATION

Each case in the case base represents a distinct locomotive with a number of features associated, including age, mileage, and parameters related to maintenance and repair history. Neighbor retrieval is based on these features, resulting in peers that are of similar design and usage. Each locomotive record also contains a record of its own pulse durations between repairs, i.e., how long the train was in a useful state. These pulses represent the availability durations of the locomotives, and so the peer's pulses are used to predict the remaining availability of the probe.

The CBR system was implemented using SOFT-CBR: a Self-Optimizing Fuzzy Tool for Case-Based Reasoning (Aggour, Pavese, et al. 2003). SOFT-CBR is an extensible, component-based tool with a number of pre-existing modules to implement a CBR system in Java. SOFT-CBR significantly reduced the implementation and testing cycles for these experiments, as it provided large portions of the functionality pre-built and pre-tested.

SOFT-CBR is configured using an eXtensible Mark-up Language (XML) file. Changing the parameters in the file can change the attributes used to define cases, the method by which similarities are calculated, and determine what types of outputs are valid. Optimizing the engine requires the optimization of a set (or subset) of the parameters in this configuration file. The SOFT-CBR modules used are described below.

4.1 Retrieve

An Oracle database contains the complete case base, with individual cases occurring as single rows in a table. In SOFT-CBR, neighbors are first retrieved, and then a similarity score is calculated between the probe and each neighbor. The probe refers to the unit for which we are trying to predict the remaining life.

Neighbor Retrieval

A SOFT-CBR case base component is responsible for constructing an SQL database query to retrieve peers of the probe case. Each case in the case base is represented by an array of N distinct features, creating an N -dimensional feature space. To retrieve neighbors, a range query is constructed around each numerical feature, defining an N -dimensional hyperrectangle in the feature space. A single support value s_i is defined for each dimension i . Neighbors are retrieved if and only if each of their features x_i fall within the support of the probe's features p_i , such that $p_i - s_i \leq x_i \leq p_i + s_i \forall i=1, \dots, N$. The case base component returns all cases that appear similar to the probe (fall within this hyperrectangle), and then the engine uses a similarity calculation component to rank them.

Similarity Calculation

A Truncated Generalized Bell Function (TGBF) (Jang 1993) along each dimension is a fuzzy membership function that produces a score representing the degree of similarity of that feature. For each dimension i , a separate $TGBF_i(x_i; a_i, b_i, p_i)$ function exists centered at the feature values of the probe p_i , as shown in Equation 1. Here, ε is a truncation parameter, e.g. $\varepsilon = 10^{-5}$.

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

Feature value x_i is determined from the peer and value p_i is from the probe, so each $TGBF_i$ has only two free parameters, a_i and b_i . This function was selected because it affords these two degrees of freedom, enabling the control of both the spread and curvature of the fuzzy membership function.

The most similar peers should be the closest to the probe along all N dimensions, so a similarity measure defined as the intersection of the individual $TGBF_i$ values is used. Further, to represent the different relevance that each criterion should have in the evaluation of similarity, a weight w_i is attached to each feature. The similarity measure $S(p, x)$ between probe p and neighbor x becomes a weighted minimum operator, as shown in Equation 2, where weights $w_i \in [0, 1]$.

$$S(p, x) = \min_{i=1}^N \{ \max[(1 - w_i), TGBF_i(x_i; a_i, b_i, p_i)] \} \quad (2)$$

The set of values for the supports s_i , weights w_i , and parameters a_i and b_i are design choices that impact the proper selection and ranking of peers. Each value is initially chosen by hand and then optimized using an evolutionary algorithm.

4.2 Reuse

Once identified, the peers are used to make a prediction of the remaining operational availability of the probe locomotive. Each pulse begins with the return of a locomotive to service, and ends with an event that renders the locomotive temporarily unavailable (a breakdown or scheduled or unscheduled maintenance). The SOFT-CBR reuse component uses the peer's historical pulses to first estimate the next pulse of each peer. These estimates are then aggregated to predict the remaining availability of the probe. If no neighbors are retrieved, then a default value specified in the configuration file is used. This default value represents "no decision" from the engine.

Each neighbor x has m_x historical pulses, which can be represented in a vector $H_x = [P_{1,x}, P_{2,x}, \dots, P_{m,x}]$. For each neighbor, the goal is to determine the duration of the next pulse $P_{m+1,x}$. There was not enough historical data to generate reliable local regressions, so simpler models were experimented with, such as averages and medians. It was found

that the most reliable way of generating the next pulse $P_{m+1,x}$ from the pulse vector H_x was to use an exponential average that gives more relevance to the most recent information. The exponential average function can be found in Equation 3, where weight $\alpha \in [0.5,1]$. Also critical to the performance of the model is the choice of the value of α .

$$\begin{aligned} P_{m+1,x} &= \bar{P}_{m,x} = \alpha P_{m,x} + (1-\alpha)\bar{P}_{m-1,x} \\ &= \alpha \sum_{i=2}^m (1-\alpha)^{m-i} P_{i,x} + (1-\alpha)^{m-1} P_{1,x} \end{aligned} \quad (3)$$

These individual predictions $P_{m+1,x}$ are aggregated to make a prediction $P_{m+1,p}$ of the remaining availability of the peer. A weighted average of the individual neighbor predictions is calculated, using the similarities of the peers as weights. Equation 4 shows the specific weighted average calculation.

$$P_{m+1,p} = \frac{\sum_{x=1}^n S(p,x) \times P_{m+1,x}}{\sum_{x=1}^n S(p,x)} \quad (4)$$

5. OPTIMIZATION

The CBR had a number of parameters that required tuning to identify an optimal combination of values. Using the Evolutionary Algorithm (EA) already implemented in SOFT-CBR greatly simplified the task of optimizing the CBR's parameters. EA's (Goldberg 1989; Holland 1992) define an optimization paradigm based on the theory of evolution and natural selection.

An EA is composed of a population of individuals ("chromosomes"), each of which contains a vector of elements that represent distinct tunable parameters within the CBR configuration. For our system, given N dimensions in the universe of features, the chromosome vector c can be found in Equation 5.

$$\begin{aligned} c &= [s_1, s_2, \dots, s_N; w_1, w_2, \dots, w_N; (a_1, b_1), (a_2, b_2), \dots, (a_N, b_N); \alpha] \\ \text{where } s_i &= \text{retrieval support parameters} \\ w_i \in [0,1] &= \text{feature weights} \\ (a_i, b_i) &= TGBF_i \text{ parameters} \\ \alpha &= \text{exponential average weight} \end{aligned} \quad (5)$$

Figure 3 visualizes how the EA and CBR interact in SOFT-CBR. A chromosome defines a complete configuration of the CBR, so an instance of the CBR can be initialized for each chromosome, as shown in Figure 3. On the left-hand side there is a population $P(t)$ of chromosomes c_i , each of which go through a decoder to allow them to initialize a CBR on the right. The CBR then goes through a round of leave-one-out testing.

The EA maintains a population of 30 individuals evolved over 200 generations. Two types of mutation (randomly permuting parameters of a single chromosome) are used to produce new individuals in the population pool: Gaussian and uniform. The more fit chromosomes in generation t will be more likely to be selected for mutation and pass their genetic material to the next generation $t+1$. Similarly, the less fit solutions will be culled from the population. At the conclusion of the EA's execution, the single best chromosome is written to the SOFT-CBR configuration file as the new CBR configuration.

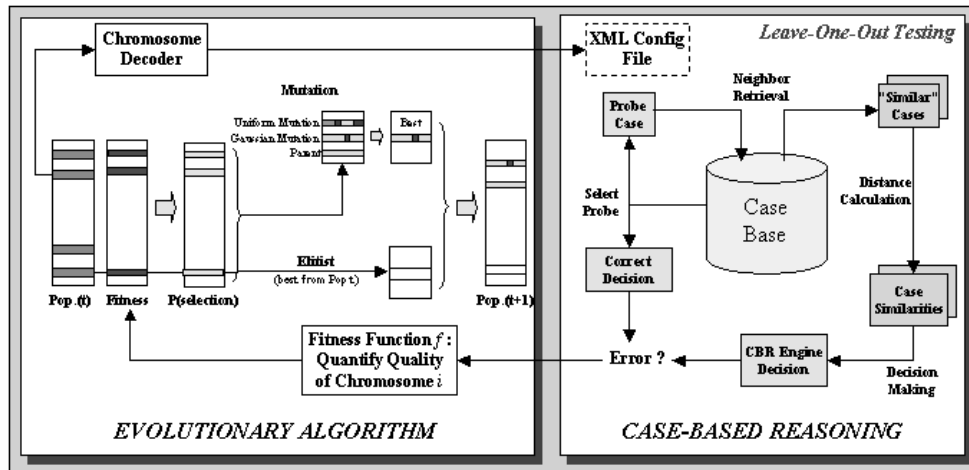


Fig. 3. EA and CBR Interaction

The quality of the CBR instance (the “fitness” of the chromosome) is determined by analyzing the results of the leave-one-out testing. A fitness function f is used to give a quantitative representation of the quality of the output. The objective of the experiment is to identify the top 20% of the locomotive population based on remaining availability, so the fitness function is defined by ordering each test case in descending order by the prediction $P_{m+1,p}$ of their remaining pulses. The top 20% are then selected, and this predicted top 20% is compared to the actual top 20%—the golden units. Equation 6 is then used to give a final fitness score $f(c)$ to the chromosome. A True Positive is simply a locomotive predicted to be in the top 20% that was in the actual top 20%. A False Positive is a locomotive falsely predicted to be in the top 20%.

$$f(c) = \frac{TP}{TP + FP}$$

where TP = count of True Positives

FP = count of False Positives

(6)

6. RESULTS AND ANALYSIS

Earlier we mentioned some simple heuristics that might be used for unit selection. We tested these by selecting the best 20% of the fleet sorted on the dimensions in Table 3. Through random selection, a sample should have 20% of its population composed of the verified golden units. It is interesting to note that the newest units by age, or those with the lowest mileage produced selections that were no better than the random selection. Having a low frequency of maintenance appeared to provide the best performance.

Table 3. Single Heuristic Classification Results

Single Heuristic	% of Correctly Classified Units
Lowest Mileage	17%
Newest Units	18%
Random	20%
Highest Energy (MWHRS) Generated	24%
Highest Miles / Hours Moving	26%
Highest Percentage Hours Moving	29%
Lowest Percentage of: Subsystem 10 Failures	38%
Lowest Ratio: Recommendations / Age [Rx/yr]	49%

We next used Weka (Witten and Frank 2000), a freely available data-mining software suite, to perform k-Nearest Neighbor retrieval over each time slice with leave-one-out testing. Some parameters like the aggregation function and number of neighbors to retrieve were manually tuned through trial and error. As the prediction variable, we used the median time to failure for each unit, and the best 20% were defined as those with the best medians. This was a first step to see how well peers can approximate an individual units' retrospective performance. The average performance achieved on a 10-fold cross-validation run was reported. The results are shown in Figure 4.

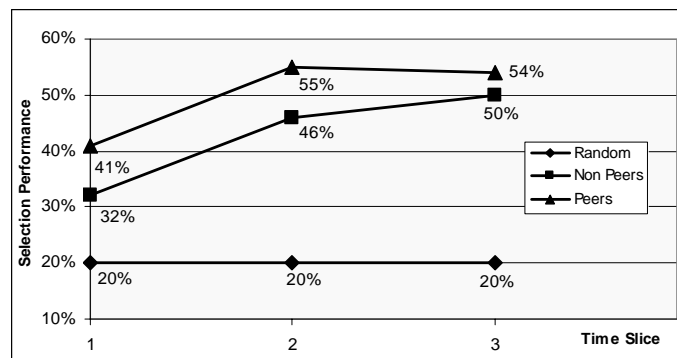


Fig. 4. Results of 1st Set of Experiments (Manual Tuning)

Using peers consistently outperformed the best available non peer-based single heuristic at each time slice. Another aspect of interest was that the number of peers required for the best estimates appeared to stabilize at ~1% of the total fleet size. These promising results led us to develop a CBR system with the ability to evolve the best parameter settings for peer retrieval and aggregation. This produced varying gains in performance, as shown in Table 4, in the column ‘Evolved Peers’.

Table 4. Results With Evolutionary Tuning of Peer Selection Parameters

Selection Performance				
Slice	Evolved Peers	Peers	Non-Peer	Random
1	48.1%	41%	32%	20%
2	55.6%	55%	46%	20%
3	60.4%	54%	50%	20%

After these experiments we approached our final objective—predicting the best units for the next mission. The retrospective median used earlier was replaced by the time-to-failure for each unit immediately after each slice. This resulted in a true prognostic experiment. Based on the DARPA steering committee’s suggestions, we changed the selection metric from a percentage to a fixed number: 52 units. In Slices 1, 2, and 3 we now try to pick the best 52 units, representing 20%, 8% and 6% of the fleet, respectively.

The most difficult experiment (Slice 3) also produced the most encouraging results. Using the best single heuristic (Non-Peer) resulted in a performance plateau around 37% (Table 5). Using a peer-based approach and evolving the selection criterion for each slice, we were able to correctly identify 63% of the units that lasted the longest. This is superior to the non-peer and random selection performance, as shown in Table 5.

Table 5. Selecting the Best Performers for the Next Pulse

Selection Performance			
Slice	Evolved Peers	Non-Peer	Random
1	48.1%	32%	20%
2	55.8%	37%	8%
3	63.5%	37%	6%

Finally, we studied the impact of not updating the peer similarity model after optimizing it for Slice 1. As shown in Figure 5, by Slice 3, the performance deteriorated to near random. This reinforces the importance of not only investing effort in the right representation and reasoning model, but also into the ability to maintain it over time.

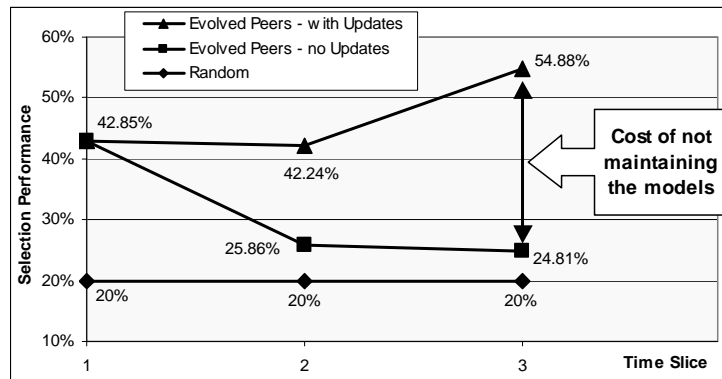


Fig. 5. Decline in Selection Performance Due to Lack of Peer Similarity Model Maintenance

7. CONCLUSIONS

For DARPA, the motivation of these experiments was to test if equipment readiness could be computed with relatively sparse deployment history. This required a specialized data set derived from actual equipment operation, one that GE was able to provide by drawing on the breadth of its transportation operations. The results show that the physical analogy of ‘equipment peers’ holds promise as a way to incrementally reason from experience. We believe this is a key contribution of this work.

From the beginning, CBR has effectively relied on analogy to position itself as a reasoning approach. The notion of a case base of ‘objects’ and ‘entities’ rather than experiences was intriguing to us. Peer groups forming and disbanding over the lifecycle of the fleet was a natural extension of the analogy. Maintenance of the criteria for peer identification was critical to this task, given the dynamic nature of the domain.

Directions for future work include experimenting with fleets with a mix of long and short track records, and better understanding how a unit’s own track record should be integrated with the estimates provided by its peers. We also plan to test this approach on data from different kinds of vehicles, including aircraft and medical imaging equipment.

8. ACKNOWLEDGEMENTS

This work was funded by DARPA, through contract CACI 621-04-S-0031. The authors acknowledge the help of Drs. Norm Sondheimer, Al Wallace, and Peter Will, members of the DARPA Steering Committee, and GE Rail who provided us with the data sets and domain knowledge that were indispensable for the model generation and validation.

REFERENCES

- Aggour, K.S., Pavese, M., Bonissone, P.P., and Cheetham, W.E. 2003. SOFT-CBR: A Self-Optimizing Fuzzy Tool for Case-Based Reasoning, *Proceedings of the 5th International Conference on Case-Based Reasoning*, Springer-Verlag, pp 5-19
- Berndt, D.J. and Clifford, J. 1996. Finding patterns in time series: A dynamic programming approach, *Advances in Knowledge Discovery and Data Mining*, American Association for Artificial Intelligence, pp 229-248
- Fritsche, L., Schlaefter, A., Budde, K., Schröter, K., and Neumayer, H.H. 2002. Recognition of Critical Situations from Time Series of Laboratory Results by Case-Based Reasoning, *Journal of the American Medical Informatics Association*, vol. 9, no. 5, pp 520-528
- Goldberg, D.E. 1989. Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc.
- Holland, J.H. 1992. Adaptation in Natural and Artificial Systems, MIT Press
- Jaczynski, M. 1997. A Framework for the Management of Past Experiences with Time-Extended Situations, *Proceedings of the 6th International Conference on Information and Knowledge Management*, ACM Press, pp 32-39
- Jang, R. 1993. ANFIS: Adaptive-Network-Based Fuzzy Inference System, *IEEE Transactions Systems, Man, and Cybernetics*, vol. 23, no. 3, pp 665-685
- Leake, D.B. and Wilson, D.C. 1998. Categorizing Case-Base Maintenance: Dimensions and Directions, *Proceedings of the 4th European Workshop on Case-Based Reasoning*, Springer-Verlag, pp 196-207
- Leake, D.B. and Wilson, D.C. 1999. When Experience is Wrong: Examining CBR for Changing Tasks and Environments, *Proceedings of the 3rd International Conference on Case-Based Reasoning*, Springer-Verlag, pp 218-232
- Ram, A. and Santamaria, J.C. 1997. Continuous Case-Based Reasoning, *Artificial Intelligence*, vol. 90, pp 25-77
- Schlaefter, A., Schröter, K., and Fritsche, L. 2001. A Case-Based Approach for the Classification of Medical Time Series, *Proceedings of the 2nd International Symposium on Medical Data Analysis*, pp 258-263
- Smyth, B. 1998. Case-Base Maintenance. *Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, vol. 2, pp 507-516
- Varma, A. and Roddy, N. 1999. ICARUS: design and deployment of a case-based reasoning system for locomotive diagnostics, *Engineering Applications of Artificial Intelligence*, vol. 12, no.6, pp 681-690
- Witten, I.H. and Frank, E. 2000. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers
- Zhang, Z. and Yang, Q. 1998. Towards Lifetime Maintenance of Case Base Indexes for Continual Case Based Reasoning. *Proceedings of the 8th International Conference on AI Methodologies, Systems and Applications*, pp 489-500