

SOFT-CBR: A Self-Optimizing Fuzzy Tool for Case-Based Reasoning

Kareem S. Aggour, Marc Pavese, Piero P. Bonissone, William E. Cheetham

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

Abstract. A generic Case-Based Reasoning tool has been designed, implemented, and successfully used in two distinct applications. SOFT-CBR can be applied to a wide range of decision problems, independent of the underlying input case data and output decision space. The tool supplements the traditional case base paradigm by incorporating Fuzzy Logic concepts in a flexible, extensible component-based architecture. An Evolutionary Algorithm has also been incorporated into SOFT-CBR to facilitate the optimization and maintenance of the system. SOFT-CBR relies on simple XML files for configuration, enabling its widespread use beyond the software development community. SOFT-CBR has been used in an automated insurance underwriting system and a gas turbine diagnosis system.

1. INTRODUCTION

Case-Based Reasoning (CBR) is a popular technique that relies on past cases with known solutions to make decisions on new cases (Aamodt & Plaza 1994). Fuzzy Logic (FL), a superset of Boolean logic that supports the concept of partial truth, is an approximate reasoning technique used in situations where information is imprecise or incomplete (Zadeh 1965). Evolutionary Algorithms (EAs) are stochastic search techniques that can perform optimization without relying on gradient information or becoming trapped in local minima (Goldberg 1989). The trade-off in using EAs for optimization is that their robust global search, which can also be applied to discrete landscapes, cannot guarantee an optimal solution, but only regions of good solutions. Soft Computing (SC) is an approach to computing which parallels the ability of the human mind to reason and learn in an environment of uncertainty and imprecision. SC combines knowledge, techniques, and methodologies from Fuzzy Logic, Neural Networks, Probabilistic Reasoning, and Evolutionary Algorithms to create intelligent systems (Bonissone et al 1999). Our goal was to combine CBR and SC techniques into a generic, Self-Optimizing Fuzzy Tool for Case-Based Reasoning (SOFT-CBR) capable of handling a wide variety of problems in which an existing case base would be used to build solutions to new cases.

General Electric (GE) requires automated decision tools such as SOFT-CBR to bring consistency, accuracy, and speed to decision-making processes throughout the company. This fits with GE's 6σ Quality initiative, since we often observe that variability and inconsistency in decision-making is the root cause of many process defects. GE has applied both CBR and SC in various applications (Cheetham 1997, Bonissone & Cheetham 1998, and Bonissone et al 2002), and opportunities for further use are growing throughout the company. As we move forward, it is critical that researchers be able to invest their time researching these opportunities rather than rewriting code. It was therefore our intention to combine the technologies described above in a flexible tool capable of being applied to a wide range of problem domains. SOFT-CBR is designed to be extensible so that if new functionality is required it can be easily incorporated into the system without having to rewrite or recompile existing components. In addition, EA technology is incorporated to facilitate system maintenance by searching for a set of parameters to make the CBR output as accurate as possible. The EA optimization is configurable like the rest of the tool to handle domain-specific problems.

Section 2 provides background information, including a review of some past CBR work relevant to this paper. Major design choices and system architecture are described in Section 3. Section 4 provides a more detailed view of the design, including a discussion of major SOFT-CBR components such as the EA. The successful use of SOFT-CBR in two applications, an Automated Insurance Underwriting system and a Gas Turbine Diagnosis system, is described in Section 5.

2. BACKGROUND & PRIOR ART

2.1 Brief Summary of Fuzzy Case-Based Reasoning Systems

The use of Fuzzy Logic in CBR systems goes back to the early 90's, when researchers began to use attributes with fuzzy values and a fuzzy pattern matcher for case retrieval (Plaza & Lopez de Mantaras 1990, Dubois & Prade 1992). In fact, FL techniques have proven to be very useful in addressing many other problems typical of CBR systems. For example, FL can be used in case representation to provide a characterization of imprecise and uncertain information. It can also be used for case adaptation through the concept of gradual rules (Dubois & Prade 1992). FL is enabled within SOFT-CBR through:

- **Case Representation:** Approximate or incomplete knowledge of case attributes can be represented by fuzzy intervals or sets, which in turn can be associated with linguistic terms stored as text.
- **Case Retrieval:** A concept of "neighborhood" or partial match has been implemented for numeric attributes. Non-numeric attributes (such as fuzzy linguistic terms) can either be handled by adjusting the distance calculation or by extending the current components.

- Case Similarity: Distance calculation is highly customizable. A fuzzy similarity based on the Generalized Bell function exists. Alternative fuzzy similarity measures can also be coded and used.

A brief description of some existing Fuzzy CBR systems follows. The list is by no means complete. For a more detailed review, see (Bonissone & Lopez de Mantaras 1998).

The memory organization of the ARC system (Plaza & Lopez de Mantaras 1990) is a hierarchy of classes and cases. Each class is represented by a fuzzy prototype describing the features common to most of the cases belonging to the class. The retrieval step consists of selecting the most promising classes by means of a fuzzy pattern-matching algorithm. Next, cases are selected based on the similarities and differences between the classes and the cases. Finally, near misses are used to avoid repeating past failures.

The CAREFUL system (Jaczynski & Trousse 1994) focuses on the first two steps of a case-based reasoner: case and problem representation and case retrieval. The representation is based on a hierarchy of fuzzy classes. Fuzzy sets represent imprecise values of the attributes of the cases. The retrieval process proceeds in two steps. First, the problem specification and case filtering, which guides the operator in specifying the problem and identifies potentially interesting cases, and second, the selection that chooses the nearest cases. In the second step, each value of a problem attribute represents a fuzzy constraint and the process selects those cases that better satisfy the constraints according to a weighted fuzzy pattern matching technique.

In PROFIT (Bonissone & Cheetham 1998), a fuzzy CBR system was developed to estimate residential property values for real estate transactions. The system enhances CBR techniques with fuzzy predicates expressing preferences in determining similarities between subject and comparable properties. These similarities guide the selection and aggregation process, leading to the final property value estimate. PROFIT has been successfully tested on thousands of real estate transactions.

2.2 Brief Summary of Evolutionary Algorithms to Tune CBR Systems

The proper integration of SC technologies such as fuzzy logic and EAs has resulted in the improvement of many reasoning systems (Bonissone et al 1999). There is a small body of literature dealing with the tuning of CBR parameters using EAs, however. While some authors have combined or integrated EA and CBR systems (Inazumi et al 1999, Kuriyama et al 1998, Job et al 1999), few researchers have used EAs as design tools for developing and maintaining a CBR system. Ishii et al (1998) proposed an EA-based method for learning feature weights in a similarity function. Jarmulak et al (2000) proposed the use of evolutionary algorithms to determine the relevance of case features and to find optimal retrieval parameters. They consider this approach to be a self-optimizing CBR retrieval because the optimization is performed using the data in the case base.

Bonissone et al (2002) described the use of evolutionary algorithms for automating the tuning and maintenance of two fuzzy decision systems: a rule-based and a case-based

classifier for insurance underwriting. A configurable multi-stage mutation-based EA tunes the decision thresholds and internal parameters of fuzzy rule-based and case-based systems that decide the risk categories of insurance applications. The encouraging results of that work motivated the implementation of SOFT-CBR.

2.3 Other CBR Tools

In the following, we cite some of the interesting CBR tools known to us. Citations with URLs are provided in the references section. CBR*Tools from INRIA is a generic software library that requires some programming to produce a CBR system. The CBR Shell from AIAI is developed in Java and uses an EA for weight learning. CASPIAN, from the University of Wales, Aberystwyth, relies on a custom case language configuration file and is developed in C. Finally, orange from empolis GmbH is a comprehensive and extensible CBR tool. Its architecture seems very close in spirit to SOFT-CBR. We believe SOFT-CBR has useful features that together are not covered in any one tool above. First, it incorporates an EA for parameter optimization. Second, it performs dynamic component invocation through XML configuration files without requiring code rewrites. Third, it has a simple, open architecture that allows integration of new functionality with zero impact to the other components.

3. SYSTEM ARCHITECTURE

SOFT-CBR is written in Java, is platform independent, and has been executed on UNIX and Windows operating systems. A simple, flexible component-based architecture has been adopted using Object-Oriented design paradigms. The components are logically and functionally distinct and each implement a major step in the CBR process:

1. **Neighbor Retrieval** (“**Retrieve**” component): Compares a probe case against the case base of past cases and retrieves any cases that are similar to the probe case.
2. **Distance Calculation** (“**Distance**” component): Calculates a distance between each retrieved case and the probe to quantify the similarity between the two cases.
3. **Decision Making** (“**Decide**” component): Determines a solution for the probe case using the known solutions for the retrieved cases and their distances to the probe case.

A fourth major component, “**CaseBase**”, is used within the Retrieve component to perform case base access. An abstract interface for each component has been defined, and a library of components is provided in the “core” SOFT-CBR, providing a significant amount of well-tested, generally applicable functionality. These components are likely to be useful for many applications, but SOFT-CBR is intended to be useful even in situations that are not covered by pre-existing components. New functionality can be added (and existing components can be extended) to cover more complex situations. If and when

additional functionality is required, a software developer can easily program new components that obey the appropriate Java interface definitions.

The traditional CBR cycle as described in (Aamodt & Plaza 1994) has four processes: retrieve, reuse, revise, and retain. A new case is solved by retrieving one or more previously experienced cases, reusing those cases to produce a solution, revising that solution through simulation or test execution, and retaining the new experience by incorporating it into the existing case base. The SOFT-CBR components “Retrieve”, “Distance”, and “Case Base” together implement the retrieve process from the traditional CBR cycle. In our case, the flexible case base access requirement introduces additional complexity to the retrieve process. Neighbor retrieval is performed based on range queries defined around features in the probe case. Then, distance is refined using partial ordering induced from fuzzy membership functions. The SOFT-CBR component “Decide” implements the reuse process from the traditional CBR cycle. The core Decide components do not feature any adaptations, but they can be included in future Decide modules as required. The revise and retain processes are handled outside of SOFT-CBR.

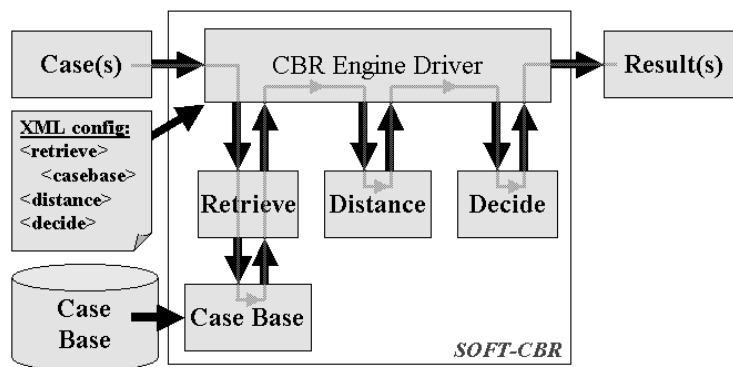


Fig. 1. Decision Process Modules

SOFT-CBR is configured using a file written in eXtensible Mark-up Language (XML). In this file one can specify what components to use for each of the functional steps. All of the parameters required to configure each component are also exposed to the user in appropriate sections of this file. Hayes and Cunningham (1999) use XML for CBR case representation, which incorporates similar technologies but with different objectives. Performing tasks such as changing the attributes used to define cases, changing the method by which distance calculations are performed, and determining what types of outputs are valid can all be done by modifying this file. Typically, no code writing or compilation is required—one does not need to be a software developer to use SOFT-CBR. A main “engine driver” in the tool is responsible for parsing the configuration file and determining what components to load dynamically at run time. Therefore, any component can be extended or replaced by another that implements the same interface. The driver

passes data from one component to the next, acting as the glue between the components and allowing them to work together to make decisions, as illustrated in Figure 1.

When a probe case is sent to the engine, the engine first passes the case to the retrieve class. Retrieve invokes a case base class that connects to a data store and obtains neighbors to the probe based on range queries around features in the probe case. The case base class obtains a set of neighbors and feeds them back through the retrieve class to the engine driver. The allowable variation of retrieved cases from the probe case is specified in the XML configuration file. The engine driver then passes the probe case and its neighbors to the distance class, which calculates the distance between the probe and each neighbor using fuzzy membership functions. The XML file specifies the relative weights of each parameter in the case base, as well as other configurable parameters used to calculate the distance. These distances are then returned to the engine. Finally, the driver passes the neighbor cases and their distances to the decide class, which reuses the solutions of the retrieved cases to generate a solution for the probe case. This solution adaptation procedure is configured in the XML file.

To build a CBR decision model with SOFT-CBR, we need a framework for connecting to different data sources (read new cases for evaluation), run the engine, and generate output (allow access to CBR results on those cases) in whatever format is desired. The same componentized, object-oriented approach as above is adopted for this purpose. Our library of input components includes taking cases from a database, flat files, or XML files. Our library of output components includes an API for programmatic access to the results, or they can be written in flat file and XML formats. As above, new components can be developed and used as needs arise, without altering any other SOFT-CBR component. We have also developed a set of “wrappers” that can be used interchangeably to integrate with SOFT-CBR. These include EJB, SOAP, and Servlet wrappers, through which the CBR can be used within any J2EE application, Web Services framework, or JSP or Servlet-based Web application, respectively. These scenarios cover a large portion of the new development work at GE, which again increases the usefulness of the tool.

4. SYSTEM FEATURES & DESIGN

4.1 Engine Initialization and Operation

Upon initialization, an XML file laid out as shown in Figure 2 is passed to the engine. This file contains three top-level elements: `<retrieve>`, `<distance>`, and `<decide>`, that configure the major components of SOFT-CBR.

In Figure 2, the “...” within the various elements indicate that content has been left out for simplicity. Within each of these three XML elements, a `<class>` element is required to specify the Java class to use for that step. The engine driver reads each of the `<class>` elements and uses Java’s reflection API (run-time class loading) to instantiate those classes. This is how the engine can operate without knowing exactly what type of

distance calculation it is performing, for example—all the engine has to know is what class to invoke. After the `<class>` element, all additional information within the XML block is used to initialize that component. For example, any additional information in `<retrieve>` is passed to the Retrieve component that was loaded by the engine driver. No constraints are placed on the information a given component can require. This affords users of SOFT-CBR a great deal of flexibility in the implementation of new components.

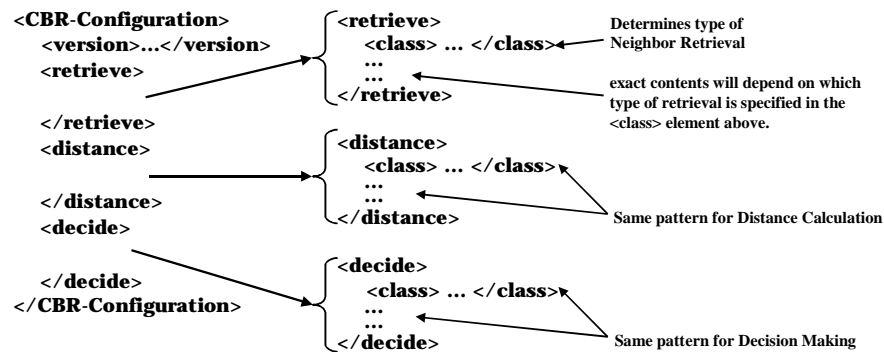


Fig. 2. Example Initialization File

4.2 Core Components

The “core” SOFT-CBR provides a library of generally useful components.

4.2.1 Neighbor Retrieval

One neighbor retrieval component exists, referred to as “Basic Retrieve.” The Basic Retrieve component does not access the case base directly, but uses a case base component to perform the data source access. The reason for using a separate component to do the case base access is flexibility. The user may wish to connect to different case base types while still using the same neighbor retrieval component. The case base component is responsible for connecting to a data source (a database or flat file, for example) and then providing neighbor cases back to the retrieve component.

4.2.2 Case Base Access

Currently, two simple but quite general case base components have been developed that allow cases to be accessed from a table in any JDBC compliant database. Individual cases must exist in the table as single rows. The table containing the case base data may also contain other fields which SOFT-CBR does not use. The case base components are responsible for constructing the query that retrieves neighbors of the probe case, and then returning those cases to the retrieve component. In the present implementation, the query

is described by ranges of values around each attribute used for retrieval. Other retrieval algorithms may also be incorporated into SOFT-CBR, as necessary. By way of the case base class, the retrieve component returns all cases that appear similar to the probe, and then the engine uses a distance calculation component to rank them. The existing components assume cases have the same representation, but sparse representation can be handled in future implementations.

4.2.3 Distance Calculation

Two distance calculation components have been implemented. They are: “Manhattan” and “Generalized Bell Function” distance.

For two points in an N-dimensional space, the Manhattan Distance is defined as the weighted sum of the absolute values of the differences between each of the dimensions. In this application, each case is a point and each attribute can be considered a dimension. Therefore, we can define the Manhattan Distance between cases $c1$ and $c2$ to be

$$dist(c1, c2) = \sum_{k=1}^n w(k) * |c1(k) - c2(k)| \quad (1)$$

The notation $c1(k)$ represents the value of attribute k for case $c1$, $c2(k)$ represents the value of attribute k for case $c2$, $w(k)$ represents the weight for attribute k , and n is the total number of attributes.

The Generalized Bell Function (GBF) Distance is used to favor values that are close together and penalize those that are far apart. Different Bell function shapes are defined around each attribute in the input case, and the distance that the neighbor case’s value falls on the Bell curve is used as the distance between those two attributes. The absolute values of the distances are then summed for all of the attributes to create the overall distance between two cases. We can define the GBF Distance to be

$$dist(c1, c2) = \sum_{k=1}^n w(k) * \left(1 - \frac{1}{1 + \left(\frac{|c1(k) - c2(k)|}{a} \right)^{2b}} \right) \quad (2)$$

Here $c1(k)$, $c2(k)$, and $w(k)$ have the same meaning as in Equation 1. The parameters a and b control the shape of the Bell function.

These distance calculation components each assume the attributes are numeric. For non-numeric attributes, these components both assign a distance of 0 to exact matches and a distance of 1 to mismatches. One can easily extend these classes to implement fuzzy similarity relations over non-numeric attributes. Both of these components also allow us to assign weights to the attributes. These weights take into account any potential unit conversion issues, as well as the fact that one attribute may be more significant than another in terms of case similarity. The distance calculation components determine how similar the neighbors are to the probe case, and the CBR engine driver then passes this data to the decision making step to make a final decision for the probe case.

4.2.4 Decision Making

Two decision-making components have been implemented. They are: “Single Nearest Neighbor” (SNN) and “Weighted Mode” decide. Both of these components are for discrete decision problems. The SNN Decide component takes the solution from the single closest neighbor and uses it as the decision. The Weighted Mode Decide component builds a weighted histogram of possible solutions based on the known solutions from the nearest neighbors and their distances to the input case. The weighting of the neighbor case solutions is based on the distance of the neighbor to the probe—the closer the neighbor, the larger the weight its solution is assigned. Once the histogram is complete, this component assigns the decision that has the most weight. If no neighbors are retrieved, a default value specified in the configuration file is used. This default value represents “no decision” from the engine.

4.3 Maintenance & Optimization with Evolutionary Algorithms

Maintenance of a CBR system is critical to its long-term usefulness since, over time, the configuration of the system may become sub-optimal. It is therefore critical to have the ability to optimize the configuration in a convenient manner. The implementation of an Evolutionary Algorithm directly into SOFT-CBR greatly simplifies this task.

EAs (Goldberg 1989, Holland 1994) define an optimization paradigm based on the theory of evolution and natural selection. Figure 3 visualizes how the EA and CBR interact in SOFT-CBR. Our 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. Any numerical value in the SOFT-CBR configuration (in any of the four major components) can be tuned, so the size of the chromosome depends on the number of parameters being tuned. Example tunable parameters include the range of each parameter used to retrieve neighbor cases and the relative weights associated with each parameter used for distance calculation. One of the most important aspects of the SOFT-CBR EA is its ability to dynamically determine the structure of the chromosomes. One can specify directly in the configuration file what attributes should be tuned and within what range for each attribute. This enables one to dynamically define the search space over which the EA will optimize. Any combination of SOFT-CBR parameters may be tuned while others remain static.

Since a chromosome defines a complete configuration of the CBR, 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 decoding process to allow them to initialize a CBR on the right. The CBR then goes through a round of leave-one-out testing. We can determine the quality of that CBR instance (the “fitness” of the chromosome) by analyzing the results.

A fitness function f is used to give a quantitative representation of the quality of the output. We define a generic fitness function for discrete classification problems using two matrices. The first matrix M_{TCT} is a confusion matrix that contains frequencies of correct

and incorrect classifications for all possible combinations of the benchmark and model decisions (see Figure 4 for an example). For instance, cell (i,j) contains the frequency of class i being misclassified as class j . Clearly, all the cells on the main diagonal represent correct classifications. This matrix is often used in character recognition problems, and is a valuable tool for visualizing the accuracy and coverage of the CBR. The second matrix P_{TXT} is a reward/penalty matrix and is of equal dimension as M_{TXT} . Here, cell (i,j) represents the penalty for misclassifying a class i as a class j . Rewards are assigned to elements on the main diagonal.

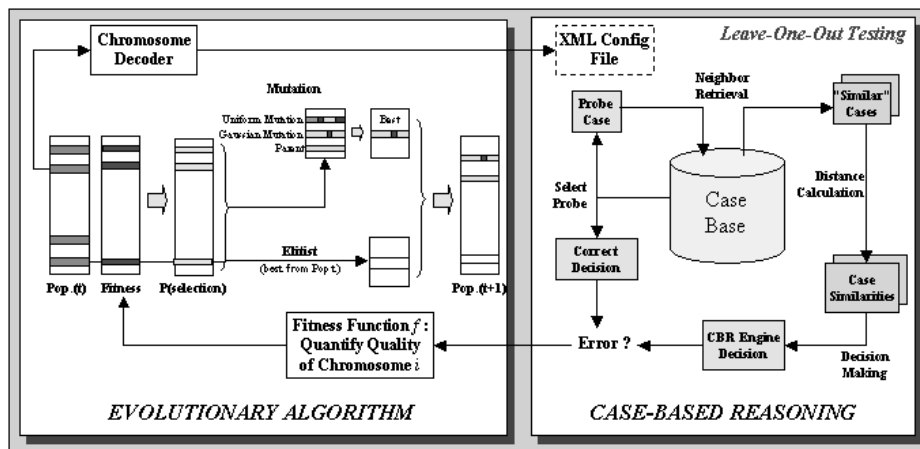


Fig. 3. EA and CBR Interaction

The fitness function f takes the confusion matrix M_{TXT} (produced from a run of leave-one-out testing of the CBR configured with chromosome c_i) and combines it with the matrix P_{TXT} using the formula in Equation 3 to produce a single value.

$$f(c_i) = \sum_{j=1}^T \sum_{k=1}^T M(j,k) * P(j,k) \quad (3)$$

When tuning is invoked in SOFT-CBR, a second configuration file is specified. This file includes the entire set of configuration information required to initialize the tuning process, including the number of generations to run for the EA and how many chromosomes to include in the population. The fitness/penalty matrix is also specified directly and flexibly in this configuration file. Finally, through this file one can also optimize the engine on only a portion of the case base. This is valuable when running a complete iteration of leave-one-out testing is time consuming.

As shown in Figure 3, our EA uses mutation (randomly permuting parameters of a single chromosome) to produce new individuals in the population. The more fit chromosomes in generation t will be more likely to be selected for this 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 so that it can be used as the CBR's new configuration.

5. APPLICATIONS

SOFT-CBR has been successfully used in two distinct applications: Automated Insurance Underwriting for GE Financial Assurance (GEFA), and Gas Turbine Diagnosis for GE Power Systems (GEPS). These two applications are vastly different in the data they use and the decisions they make, but were both implemented by writing XML configuration files for the same underlying software—SOFT-CBR. The rapid and successful development of these two applications is a direct testimony to the flexibility and extensibility of SOFT-CBR.

5.1 Automated Insurance Underwriting

Insurance underwriting is a complex decision-making task traditionally performed by trained individuals. An underwriter must evaluate each insurance application in terms of its risk of generating a claim. A given insurance application is compared against standards adopted by the insurance company, which are based on actuarial principles relating to the insured risk (e.g., mortality in the case of life insurance). The application is then classified into one of the risk categories available for the type of insurance requested. These risk categories are discrete “bins” into which the underwriter places applicants with similar risk profiles. The assigned risk category, in conjunction with other factors such as gender and age, will then determine the appropriate premium that the applicant must pay for an insurance policy. Therefore, we may consider the underwriting process as a discrete classification problem that maps an input vector of applicant data into the decision space of rate categories. Our goal was to automate as high a fraction of these underwriting decisions as possible using AI and Soft Computing techniques. A major constraint faced was the need for interpretability of decisions—no black box models could be used due to consumer communication considerations.

Specializing to life insurance, there is a natural dichotomy in applicants—those who have medical impairments (such as hypertension or diabetes) and those who do not (who are “clean”). Clean case underwriting is relatively simple and we have been able to represent it by a compact set of fuzzy logic rules (Bonissone et al 2002). Impaired underwriting is more difficult, as the applicant's medical data is more complex. Underwriters thus use more judgment and experience in these cases. Therefore, rather than create an enormous and un-maintainable fuzzy rule base, we turned to CBR to handle the impaired cases. However, to validate the use of CBR for insurance underwriting, we

first applied both FL and CBR systems to the same clean case data set. This has an added benefit as the CBR model can be used to guide auditing and quality assurance of the FL model. For example, cases for which the two models give different results might be suspect and therefore subject to more frequent audits. In addition, this provided an opportunity to compare a CBR model with other modeling techniques on the same data.

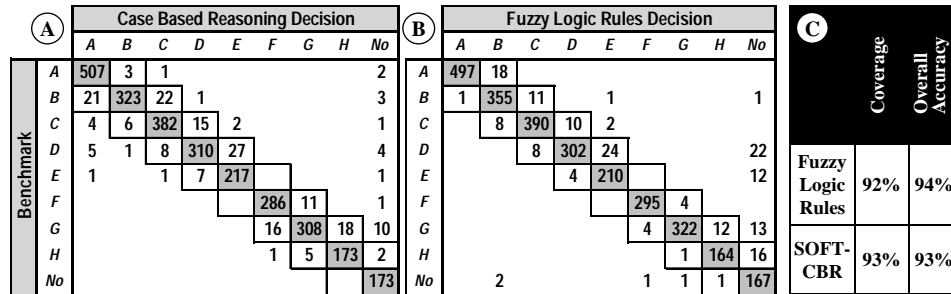


Fig. 4. Confusion Matrices for Clean Case Insurance Underwriting

To develop the underwriting models, we created a “standard reference” dataset of approximately 3000 cases taken from a stratified random sample of the historical clean case population. Each of these cases received a rate category decision when it was originally underwritten. However, to reduce variability in these decisions, a team of experienced underwriters reviewed the cases blind to the original underwriter’s decisions and determined the “standard reference” decisions. These cases were then used to create and optimize both the FL and CBR models.

In Figure 4 we present the results of comparing the standard reference decisions to (A) the CBR model, and (B) the Fuzzy Logic model. As indicated in (C), the models are of similar coverage and accuracy. This performance reflects a significant reduction in variability in comparison to the original underwriter decisions. The details of the FL model have been described in (Bonissone et al 2002). The CBR model was implemented with SOFT-CBR using the GBF distance calculation and the Weighted Mode decide method. For the CBR, automating the optimization with an EA had a significant effect on the overall accuracy. A manually-tuned prototype CBR system (created before SOFT-CBR existed) displayed only 34% global accuracy and 36% coverage, which is easily surpassed by the 90%+ accuracy and coverage shown in Figure 4.

5.2 Gas Turbine Diagnosis

GE gas turbines are used to generate electrical power for everything from cruise ships to large cities. A typical turbine costs between 20 and 40 million US dollars and provides enough electricity for a moderately sized town. It is important to keep the turbines in

proper working condition and repair them quickly when a problem occurs. Every minute that a turbine is not working is lost revenue for the power generator and a potential shortage of electricity for their customers. GE Energy Services helps its customers monitor turbines and diagnose failures. This condition monitoring is a complex decision-making task traditionally performed by trained field service technicians. In order to assist the technicians, GE is creating a set of intelligent decision aids using rule-based and case-based reasoning. SOFT-CBR has been used to create the Case-Based Reasoning tool.

		Case Based Reasoning Decision			Coverage Overall Accuracy	
		<i>Other</i>	<i>High Ex</i>	<i>Unknown</i>		
Benchmark	<i>Other</i>	220	13	4	98%	91%
	<i>High Ex</i>	8	68	3		
	<i>Unknown</i>	0	0	0		

Fig. 5. Confusion Matrix for Gas Turbine Diagnosis

GE gas turbines are operated using an embedded control system that is equipped with an On-Site-Monitor (OSM), which records the values of over 100 sensors on the turbine. Automatic condition monitoring and anomaly detection is currently done in the areas of vibration, thermal performance, and combustion. The control system is also equipped with a remote notification module, which provides immediate notification to GE technicians at a central site when anomalies are detected. The central site technicians perform initial processing, logging, and viewing of alarm data. SOFT-CBR will assist the central site technicians in diagnosing the root cause of failures.

SOFT-CBR is currently able to diagnose a subset of the possible causes for gas turbine failures, called “High Exhaust” (High Ex) failures. High Ex failures are the most common type of failure, representing about 25% of all failures. They include a set of root causes, all of which produce an unsafe temperature variation in the turbine. A case consists of a summary of the relevant sensor readings immediately preceding the failure and the verified cause of the failure. A case base that includes every turbine failure for the past year on a fleet of 500 turbines has been created. The performance of the current system is shown in Figure 5. The second row of the figure shows that 68 of the 79 “High Ex” failures in our test set were classified correctly, 8 were classified incorrectly (“Other”), and 3 were too close to call (“Unknown”). Of the 237 “Other” failures (not “High Ex”) 220 were classified correctly, 13 were incorrect, and 4 were too close to call.

6. CONCLUSIONS & FUTURE WORK

A Self-Optimizing Fuzzy Tool for Case Based Reasoning (SOFT-CBR) has been developed. It has been used successfully in two distinct applications. A key aspect of the

tool is its flexible and extensible design. A library of components is available for users to quickly configure new CBR systems. When necessary, new components can be easily integrated into SOFT-CBR and used in combination with the pre-existing ones. A second key aspect of SOFT-CBR is the incorporation of an EA optimizer. Any and all numeric parameters in any component of SOFT-CBR can be tuned to refine the accuracy and coverage of the CBR results. We hope that SOFT-CBR will gain wide acceptance in facilitating the implementation of CBR systems, allowing other developers to add new functionality to the core system. As this occurs, the class of problems treatable with pre-existing SOFT-CBR components will grow.

Future work includes automating the design of the case base and developing a module that calculates the confidence in a SOFT-CBR result. Insurance underwriting exemplifies an application in which legal and compliance constraints require the system to exhibit reasoning transparency while using a specific set of input variables. These regulations constrain the design of the CBR system and limit the number and type of features that can be used to index a case. Therefore, any improvement in the CBR performance (measured in term of coverage and accuracy) can only be obtained by parametric tuning. However, there are other applications in which these regulations are not present. In such cases, EAs can be used for structural selection and tuning, such as feature determination and variable transformations, in addition to parametric tuning. We intend to explore this approach in the use of EAs to improve upon the design of CBR's for other GE applications. Since the simultaneous optimization of CBR structures and parameters might affect the EAs search convergence, we plan to use domain knowledge to create an initial library of features, a subset of which will then be selected by the EA. When using other reasoning techniques whose structures are defined by network topologies, we will use efficient representation of those structures, such as grammatical encoding, to further address the EA convergence problem. A confidence module will be used to calculate a predicted error in the result of SOFT-CBR. If the predicted error is low, the result does not need to be checked by a human and can be automated (e.g., automatically approving a life insurance policy or notifying a turbine operator of the root cause of a failure). For low confidence decisions, a human could review the result before taking any action.

REFERENCES

- Aamodt, A. and Plaza, E. 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, *Artificial Intelligence Communications*, vol. 7, no. 1, pp 39-59
- Bonissone, P. and Cheetham, W. 1998. Fuzzy Case-Based Reasoning for Residential Property Valuation, *Handbook on Fuzzy Computing* (G 15.1), Oxford University Press
- Bonissone, P., Chen, Y.-T., Goebel, K. and Khedkar, P. S. 1999. Hybrid Soft Computing Systems: Industrial and Commercial Applications, *Proceedings of the IEEE*, vol. 87, no. 9, pp 1641-1667
- Bonissone, P. and Lopez de Mantaras, R. 1998. Fuzzy Case-based Reasoning Systems, *Handbook on Fuzzy Computing* (F 4.3), Oxford University Press

- Bonissone, P., Subbu, R. and Aggour, K.S. 2002. Evolutionary Optimization of Fuzzy Decision Systems for Automated Insurance Underwriting, *Proceedings of the 2002 IEEE International Conference on Fuzzy Systems*, Honolulu, Hawaii, USA, vol. 2, pp 1003-1008
- CASPIAN, University of Wales, Aberystwyth, Wales, http://www.aber.ac.uk/compsci/Research/mbsg/cbrprojects/getting_caspian.shtml
- CBR Shell, Artificial Intelligence Applications Institute (AIAI), University of Edinburgh, Scotland, <http://www.aiai.ed.ac.uk/project/cbr/>
- CBR*Tools, Institut National de Recherche en Informatique et en Automatique (INRIA), France, http://www-sop.inria.fr/aid/cbrtools/manual-eng/doc_web/Abstract98.html
- Cheetham, W. 1997. Case-Based Reasoning for Color Matching, *Lecture Notes in Artificial Intelligence*, Springer Verlag, vol. 1266
- Dubois, D. and Prade, H. 1992. Gradual Inference Rules in Approximate Reasoning, *Information Science*, vol. 61, pp 103-122
- Goldberg, D.E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Massachusetts
- Hayes, C. and Cunningham, P., 1999. Shaping a CBR View with XML, *Proceedings of the 3rd International Conference on Case-Based Reasoning, Lecture Notes in Artificial Intelligence*, Springer Verlag, vol. 1650, pp 468-481
- Holland, J.H. 1994. *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, 3rd edition*, The MIT Press, Cambridge, Massachusetts
- Inazumi, H., Suzuki, K. and Kusumoto, K. 1999. A New Scheme of Case-Based Decision Support Systems by Using DEA and GA Techniques, *Proceedings, IEEE International Conference on Systems, Man, and Cybernetics*, Tokyo, Japan, vol. 3, pp 1036-1041
- Ishii, N. and Yong Wang 1998. Learning Feature Weights for Similarity Using Genetic Algorithms, *Proceedings, IEEE International Joint Symposia on Intelligence and Systems*, Rockville, MD, USA, pp 27-33
- Jaczynski, M. and Trousse, B. 1994. Fuzzy Logic for the Retrieval Step of a Case-Based Reasoner, *Proceedings Second European Workshop on Case-Based Reasoning*, pp 313-322
- Jarmulak, J., Craw, S. and Rowe, R. 2000. Self-Optimising CBR Retrieval, *Proceedings, 12th IEEE International Conference on Tools with Artificial Intelligence*, Vancouver, BC, Canada, pp 376-383
- Job, D., Shankaraman, V. and Miller, J. 1999. Combining CBR and GA for Designing FPGAs, *Proceedings, Third International Conference on Computational Intelligence and Multimedia Applications*, New Delhi, India, pp 133-137
- Kuriyama, K., Terano, T. and Numao, M. 1998. Authoring Support by Interactive Genetic Algorithm and Case Based Retrieval, *Proceedings, Second International Conference on Knowledge-Based Intelligent Electronic Systems*, Adelaide, SA, Australia, vol. 1, pp 390-395
- Orege, Empolis GmbH, <http://www.empolis.com/>
- Plaza, E. and Lopez de Mantaras, R. 1990. A Case-Based Apprentice that Learns from Fuzzy Examples, *Methodologies for Intelligent Systems, 5th edition*, Ras, Zemankova and Emrich, Elsevier, pp 420-427
- Zadeh, L. 1965. Fuzzy Sets, *Information and Control*, vol. 8, pp 338-353