Chapter 9

A CO-EVOLUTIONARY FUZZY SYSTEM FOR RESERVOIR WELL LOGS INTERPRETATION

Tina Yu[1] and Dave Wilkinson[2]
[1]*Memorial University of Newfoundland, St. John's, NL A1B 3X5, Canada,* [2]*Chevron Energy Technology Company, San Ramon, CA 94583, USA*

Abstract Well log data are routinely used for stratigraphic interpretation of the earth's subsurface. This paper investigates using a co-evolutionary fuzzy system to generate a well log interpreter that can automatically process well log data and interpret reservoir permeability. The methodology consists of 3 steps: 1) transform well log data into fuzzy symbols which maintain the character of the original log curves; 2) apply a co-evolutionary fuzzy system to generate a fuzzy rule set that classifies permeability ranges; 3) use the fuzzy rule set to interpret well logs and infer the permeability ranges. We present the developed techniques and test them on well log data collected from oil fields in offshore West Africa. The generated fuzzy rules give sensible interpretation. This result is encouraging in two respects. It indicates that the developed well log transformation method preserves the information required for reservoir properties interpretation. It also suggests that the developed co-evolutionary fuzzy system can be applied to generate well log interpreters for other reservoir properties, such as lithology.

Keywords: reservoir modeling and characterization, fuzzy logic, co-operative co-evolution, time series, well logs interpretation, genetic programming.

1. INTRODUCTION

In reservoir characterization, well log data are frequently used to interpret physical rock properties such as lithology, porosity, pore geometry, depositional facies and permeability. These properties are keys to the understanding of an oil reservoir and can help determining hydrocarbon reserves and reservoir producibility. Based on the information, decisions of where to complete a well, how to stimulate a field, and where to drill next, can be made to maximize profit and minimize risk.

Well log data, ranging from conventional logs, such as spontaneous potential, gamma ray, and resistivity, to more advanced logging technology, such as Nuclear Magnetic Resonance (NMR) logs, are sequence of curves indicating the properties of layers within the earth's subsurface. Figure 9-1 gives an example of gamma ray, neutron and spontaneous potential (SP) logs. The interpreted lithology is listed on the left-hand side.

Well log interpretation is a time-consuming process, since many different types of logs from many different wells need to be processed simultaneously. This paper investigates using a co-evolutionary fuzzy system to generate a well-log interpreter that can process well log data and interpret reservoir permeability automatically. The developed methodology has 3 steps:

- Transform well log data into fuzzy symbols which maintain the character of the original log curves.

- Apply a co-evolutionary fuzzy system to generate a fuzzy rule set that classifies permeability ranges.

- Use the fuzzy rule set to interpret well logs and infer the permeability ranges.

Similar to time series, well logs are sequential data, which are indexed by the depth under earth's surface where the data were collected. To interpret earth properties, similar consecutive log data can be grouped into blocks, since rock properties formation is frequently developed in layers. By examining blocked wells logs across the same depth, geologists are able to detect earth properties at that particular layer.

In this research, we developed a computer system to carry out the well log blocking process. Additionally, the numerical data are transformed into fuzzy symbols (Yu and Wilkinson, 2007). Fuzzy symbol representation has advantages over its numerical counter-part in that it is easier for computers to manipulate and to carry out the interpretation task. Meanwhile, because fuzzy symbols have no precise boundaries, they allows efficient interpretation under the uncertainty embedded in the data sets.

The second step of the process uses a co-evolutionary fuzzy system to extract fuzzy rule patterns in the transformed well logs fuzzy symbols to distinguish different permeability ranges. Since permeability can be divided into more than one ranges (3 in this study), the evolutionary system maintains multiple populations, each of which evolves rules that classify one permeability range from others. These populations co-evolve to produce a combined fuzzy rule set that can classify all possible permeability ranges. Once completed, this fuzzy rule set can be used to interpret permeability of other wells with similar geological characteristics.

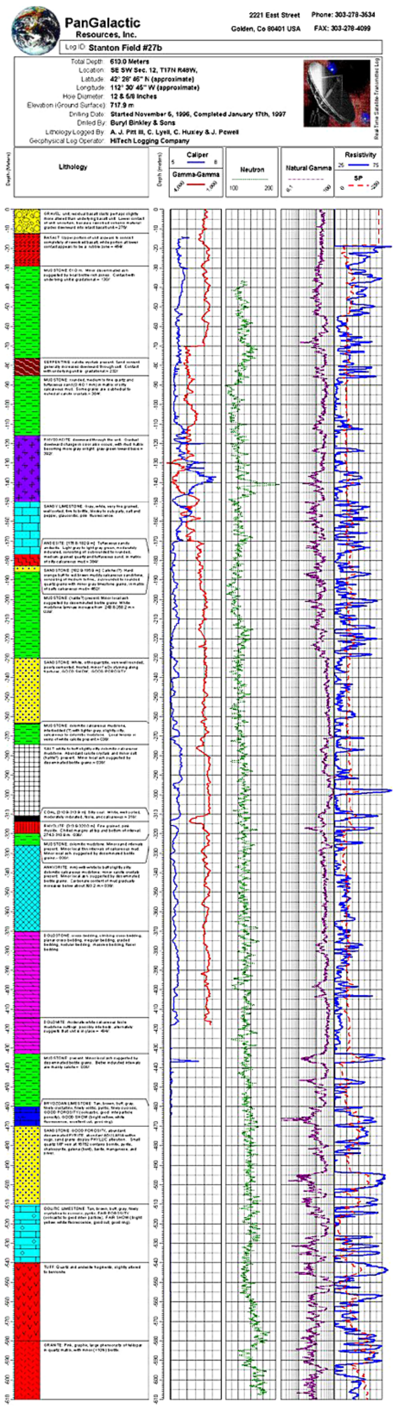

Figure 9-1. An example of gamma ray, neutron and spontaneous potential logs. The interpreted lithology is listed on the left-hand side.

We have tested the developed method on well log data collected from oil fields in offshore West Africa and the results are very encouraging. Based on this initial study, we are currently applying the system to develop a reservoir lithology interpreter, which requires a more sophisticated co-evolutionary model to interpret 5 different types of lithology. In this case, 4 populations are co-evolved together to accomplish this task.

We organize the paper as follows. Section 2 presents the methodology to transform well log data into fuzzy symbols. Information about the testing well log data and the transformed results are given in Section 3. Section 4 introduces the co-evolutionary fuzzy system developed to generate fuzzy rules. After that, the experimental setup for fuzzy rule generation is given in Section 5. In Section 6, we report the experimental results. Analysis and discussion are then provided in Section 7. Finally, Section 8 concludes the paper.

2. WELL LOG TRANSFORMATION

The fuzzy symbolic representation is an approximation of well logs data that maintains the trend in the original data. The transformation process has four steps: 1) segmentation of the numerical well log data; 2) determining the number of segments; 3) symbol assignment; and 4) symbol fuzzification. These steps are explained in the following sub-sections.

2.1 Segmentation

Well log segmentation involves partitioning log data into segments and using the mean value of the data points falling within the segment to represent the original data. In order to accurately represent the original data, each segment is allowed to have arbitrary length. In this way, areas where data points have low variation will be represented by a single segment while areas where data points have high variation will have many segments.

The segmentation process starts by having one data point in each segment. That is the number of segments is the same as the number of original data points. Step-by-step, neighboring segments (data points) are gradually combined to reduce the number of segments. This process stops when the number of segments reaches the predetermined number.

At each step, the segments whose merging will lead to the least increase in $error$ are combined. The $error$ of each segment is defined as:

$$error_a = \sum_{i=1}^{n} (d_i - \mu_a)^2$$

where n is the number of data points in segment a, μ_a is the mean of segment a, d_i is the ith data point in segment a.

This approach is similar to the Adaptive Piecewise Constant Approximation proposed by (Keogh et al., 2001) and SAX (Lin et al., 2003). However, our

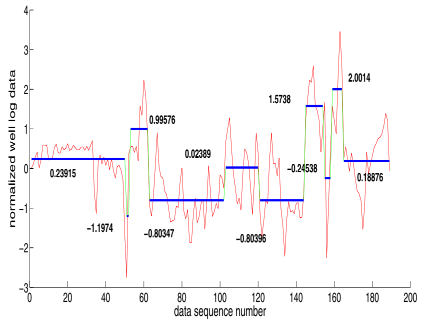

Figure 9-2. 10 segments.

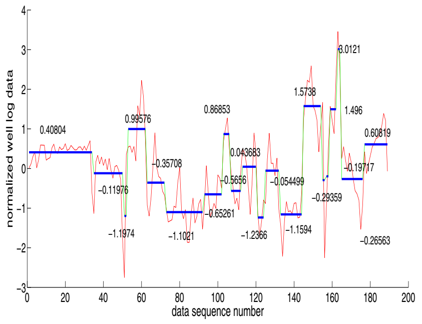

Figure 9-3. 20 segments.

method has an extra component that dynamically determines the number of segments (see Section 2.2). Another similar work using a different approach to determine the number of segments is reported in (Abonyi et al., 2005).

Figure 9-2 is an example of a well log with 189 data points, which are partitioned into 10 segments. The same data are partitioned into 20 segments in Figure 9-3. The average value of the data points within each segment is used to represent the original data.

2.2 Number of Segments

Although a larger number of segments capture the data trend better, it is also more difficult to interpret. Ideally, we want to use the smallest possible number of segments to capture the trend of the log data. Unfortunately, these two objectives are in conflict: the total $error$ of all segments monotonically increases as the number of segments decreases (see Figure 9-4). We therefore devised a compromised solution where a penalty is paid for increasing the number of segments. The new $error$ criterion is now defined as the previous total $error$ *plus* the number of segments:

$$f = N + \sum_{i=1}^{N} error_i \quad \text{where } N \text{ is the number of segment.}$$

During the segmentation process, the above f function is evaluated at each step when 2 segments were combined. As long as this value f is decreasing, the system continues to merge segments. Once f starts increasing, it indicates that farther reducing the number of segments will sacrifice log character, hence the segmentation process terminates. For the 189 data points in Figure 9-2, the final number of segments is 50 (see Figure 9-5).

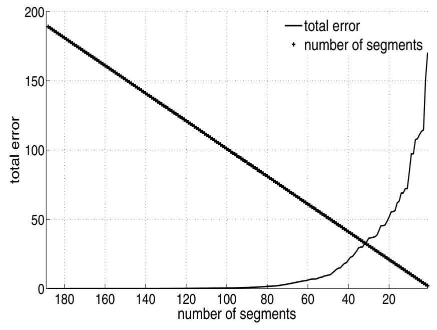

Figure 9-4. number of segments vs. total error.

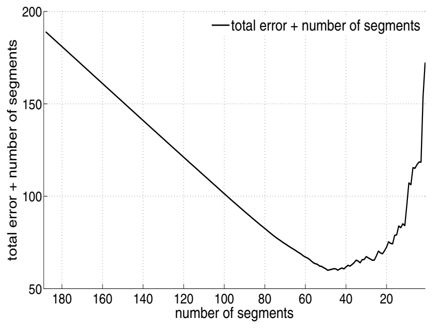

Figure 9-5. a compromised solution.

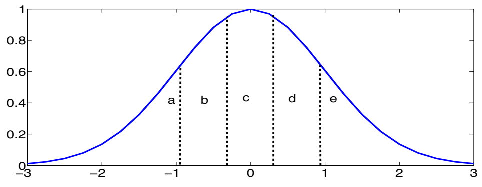

Figure 9-6. Using 4 breakpoints to produce 5 symbols with equal probability.

2.3 Symbol Assignment

Segmented well logs are represented as a set of numerical values, $\overline{WL} = \overline{s_1}, \overline{s_2}, \overline{s_3} \ldots$, where $\overline{s_i}$ is the mean value of the data within the ith segment. This numerical representation is farther simplified using symbols. Unlike numerical values, which are continuous, symbols are discrete and bounded. This makes it easy for any subsequent computer interpretation scheme.

While converting the numerical values into symbols, it is desirable to produce symbols with equal-probability (Apostolico et al., 2002). This is easily achieved since normalized sequence data have a Gaussian distribution (Larsen and Marx, 1986). We therefore applied z-transform to normalize the data and then determined the breakpoints that would produce n equal-sized areas under the Gaussian curve, where n is the number of symbols. Figure 9-6 gives the four breakpoints -0.84, -0.25, 0.25 and 0.84 that produce 5 symbols, a, b, c, d, e, with equal probability. If only 3 symbols (a, b and c) are used, the breakpoints are -0.43 and 0.43.

Once the number of symbols, hence the breakpoints have been decided, we assign symbols to each segment of the well logs in the following manner: All segments have mean values that are below the smallest breakpoint are mapped to the symbol a; all segments have mean values that are greater than or equal to the smallest breakpoint and less than the second smallest breakpoint are mapped

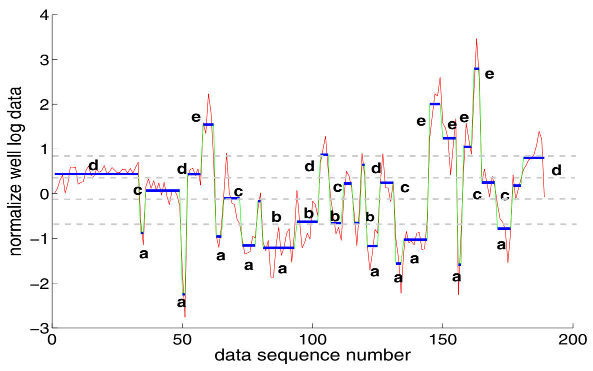

Figure 9-7. A well log transformed using 5 symbols.

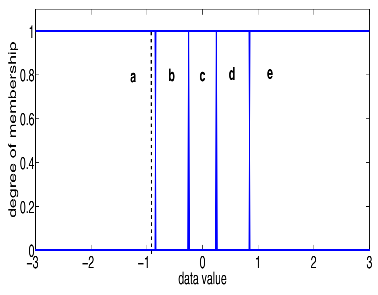

Figure 9-8. A segment with mean -0.9 is transformed as a crisp symbol a.

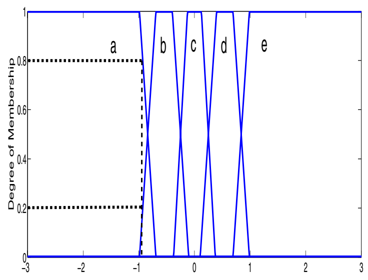

Figure 9-9. A segment with mean -0.9 is transformed as fuzzy symbol a (80%) and b (20%).

to the symbol b and so on. Figure 9-7 gives a well log that is transformed using 5 symbols.

2.4 Symbol Fuzzification

While some segments are clearly within the boundary of a particular symbol region, others may not have such clear cut. For example, in Figure 9-7, there are 3 segments that lie on the borderline of regions a and b. A crisp symbol, either a or b, does not represent its true value. In contrast, fuzzy symbols use membership function to express the segment can be interpreted as symbol a and b with some possibility. As an example, with the crisp symbol approach, a segment with mean -0.9 is assigned with symbol a with 100% possibility (see Figure 9-8). Using fuzzy symbols designed by trapezoidal-shaped membership functions, the segment is assigned with symbol a with 80% possibility and symbol b with 20% possibility (see Figure 9-9). Fuzzy symbol representation is more expressive in this case.

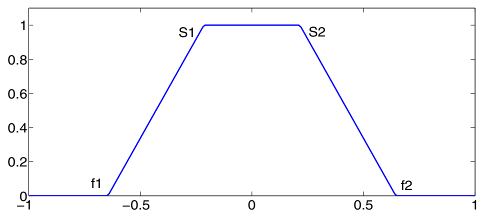

Figure 9-10. The 4 parameters, f1, f2, s1, s2, that define a trapezoidal-shaped membership function.

In fuzzy logic, a membership function (MF) defines how each point in the input space is mapped into a membership value (or degree of membership) between 0 and 1. The input space consists of all possible input values. In our case, z-normalized well log data have open-ended boundaries with mean 0. When 5 symbols are used to represent a well-log, 5 membership functions are defined, one for each of the 5 symbols.

To design a trapezoidal-shaped membership function, 4 parameters are required: f_1 and f_2 are used to locate the 'feet' of the trapezoid and s_1 and s_2 are used to locate the 'shoulders' (see Figure 9-10). These four parameters are designed in the following way.

Let c_1 and c_2 be the breakpoints that define a symbol x and $c_2 > c_1$:

$$d = \frac{c_2 - c_1}{4}$$
$$f_1 = c_1 - d; s_1 = c_1 + d; s_2 = c_2 - d; f_2 = c_2 + d$$

There are two exceptions: symbol a has f_1 = c_1 and symbol e has f_2 = c_2. Table 9-1 gives the four parameters used to design the membership functions for each symbol.

Once the 4 parameters are decided, the membership function f is defined as follows:

$$f(x, f_1, f_2, s_1, s_2) = \begin{cases} 0 & \text{if } x \leq f_1, \\ \frac{x-f_1}{s_1-f_1} & \text{if } f_1 \leq x \leq s_1. \\ 1 & \text{if } s_1 \leq x \leq s_2, \\ \frac{f_2-x}{f_2-s_2} & \text{if } s_2 \leq x \leq f_2. \\ 0 & \text{if } f_2 \leq x, \end{cases}$$

Using the described fuzzy symbol scheme, the 10 segments lying between the two symbol regions in Figure 9-7 were mapped into fuzzy symbols shown in Figure 9-11.

Table 9-1. Parameters used to design the trapezoidal-shaped membership function for each symbol.

data	symbol	f_1	s_1	s_2	f_2
well-log	a	-3	-3	-0.9875	-0.6925
	b	-0.9875	-0.6925	-0.3975	-0.1025
	c	-0.375	-0.125	0.125	0.375
	d	0.1025	0.3975	0.6925	0.9875
	e	0.6925	0.9875	3	3
perm	a	-3	-3	-0.645	-0.215
	b	-0.645	-0.215	0.215	0.645
	c	0.215	0.645	3	3

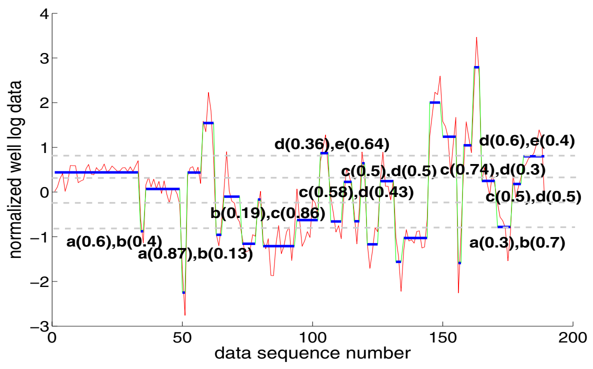

Figure 9-11. A well log represented with fuzzy symbols.

In most cases, a reservoir well has multiple logs. To carry out the described transformation process, a reference log is first selected for segmentation. The result is then used to segment the other logs in the same well. After that, fuzzy symbols are assigned to each segmented data.

3. WELL LOG DATA

We tested the developed transformation method on 2 sets of well log data collected from an offshore West Africa field. The first set is from Well A and contains 227 data points while the second set is from Well B and contains 113 data points. Each well has 3 different logs: PHI (porosity), $RhoB$ (density) and DT (sonic log). Additionally, V-shale (Volume of shale) information has been calculated previously (Yu et al., 2003). The core permeability data are available and will be used to test the evolved fuzzy rules.

Since permeability is the interpreted target, it is chosen as the reference log to perform segmentation described in Sections 2.1 and 2.2. For symbol assignment, permeability has 3 possible symbols, a, b and c, representing low, $medium$ and $high$ permeability. The 3 well logs and V-shale, however, have 5 possible symbols, a, b, c, d, e. This allows the evolved fuzzy rules to have a finer granularity in interpreting well log data. Figures 9-12, 9-13, 9-14, 9-15 and 9-16 give the transformed logs in Well A. The resulting transformations give sensible blocking and resemble the original log curves reasonably well. Due to space constraint, the results of Well B, which have a similar pattern, are not shown here.

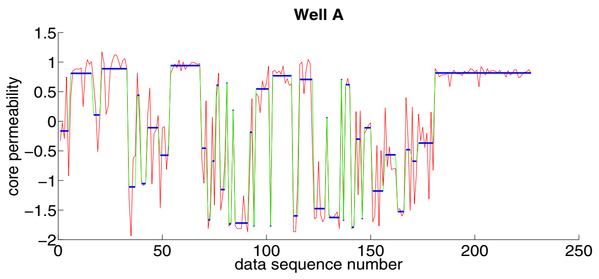

Figure 9-12. The transformed core permeability (k).

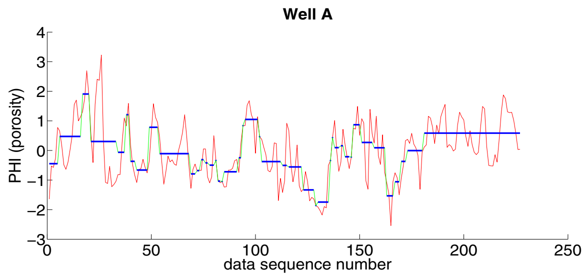

Figure 9-13. The transformed PHI log.

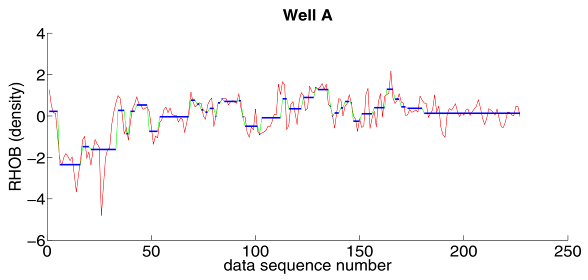

Figure 9-14. The transformed RHOB log.

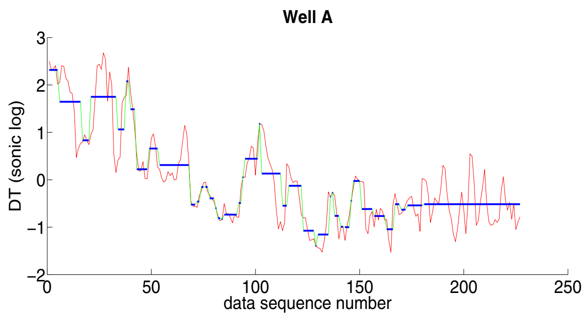

Figure 9-15. The transformed DT log.

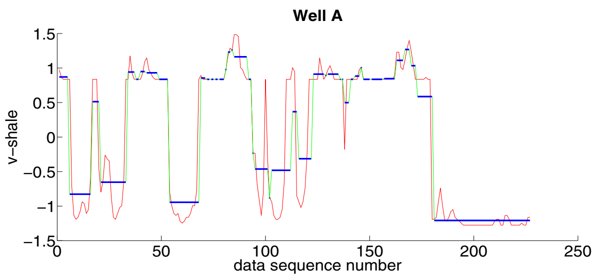

Figure 9-16. The transformed V-shale data.

After the transformation process, all logs in Well A have 43 segments and all logs in Well B have 15 segments. Among the 43 permeability segmentations in Well A, 22 are low-permeability (symbol a), 9 are medium-permeability (symbol b) and 12 are high-permeability (symbol c). The number of low, medium and high permeability segments in Well B is 6, 2 and 7 respectively.

4. CO-EVOLUTIONARY FUZZY SYSTEM

Using the transformed well log data, we applied a co-evolutionary fuzzy system to identify rule patterns that can interpret well logs having high, medium or low permeability. The interpretation task is decomposed into two sub-tasks: the first one separates one permeability range data from the rest of the data and the second one distinguishes another permeability range data from the others. By combining the two sub-solutions using an if-then-else construct, the final solution is able to determine whether a well log segment has either high, medium or low permeability.

We adopted a co-operative co-evolution approach to address these two sub-problems (Potter and Jong, 1994; Potter and Jong, 2000). In this approach, two populations are maintained, each of which is evolved toward one of the two

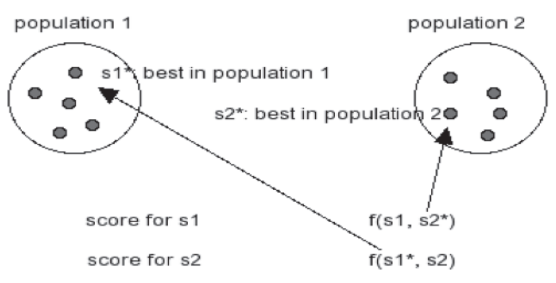

Figure 9-17. The co-operative co-evolution model.

different sub-goals. However, to encourage their co-operation to evolve the best overall permeability interpreter, the fitness of an evolved rule is determined by how well it collaborates with the rules evolved in the other population. In terms of implementation, a rule from one population is combined with the $best$ rule in the other population and the performance of this combined rule-set defines the fitness of the rule in the current population. Figure 9-17 illustrates the described co-evolution mechanism.

There are other works using this co-evolutionary model to evolve fuzzy rules. For example, fuzzy co-co (Pena-Reyes and Sipper, 2001) maintains two populations: one evolves membership functions and the other evolves fuzzy rules.

4.1 Fuzzy Rule Generation

The co-evolutionary system is implemented in a genetic programming(GP) system called PolyGP (Yu, 2001), which has a type system to perform type checking during rules evolution. In this way, the evolved rules (genotype and phenotype) are always type checked prior to fitness evaluation. There are other methods to evolve type-correct solutions (Yu and Bentley, 1998). For example, (Bentley, 2000) mapped type-incorrect fuzzy rules to correct ones using a repair method.

Table 9-2 gives the functions and terminals with their type signatures for the GP system to evolve type-correct fuzzy rules. The 3 well logs (*PHI*,*RHOB*, *DT*) and v-shale have a vector type of 5 values, each of which specifies the degree of membership to the 5 symbols a, b, c, d and e. For example, a segment with mean value 0.9 has a vector values [0.8, 0.2, 0, 0, 0]. The function *is-a*, *is-b*, *is-c*, *is-d* and *is-e* take a vector as argument and returns the degree of membership belongs to symbol a, b, c, d and e respectively. For example, *is-a*[0.8, 0.2, 0, 0, 0] = 0.8. Three fuzzy operators used to construct fuzzy rules are and, or and not: $and(x, y) = min(x, y)$, $or(x, y) = max(x, y)$, $not(x) = 1 - x$. Figure 9-18 gives an evolved fuzzy rule example.

Table 9-2. Function and Terminal Sets

Function Terminal	Type
is-a	[float,float,float,float,float]→float
is-b	[float,float,float,float,float]→float
is-c	[float,float,float,float,float]→float
is-d	[float,float,float,float,float]→float
is-e	[float,float,float,float,float]→float
and	float→float→float
not	float→float
or	float→float→float
PHI	[float,float,float,float,float]
RHOB	[float,float,float,float,float]
DT	[float,float,float,float,float]
v-shale	[float,float,float,float,float]

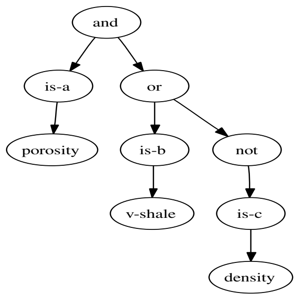

Figure 9-18. An evolved fuzzy rule example.

To work with this fuzzy rule tree representation, we employed four genetic operators in this study: homologous crossover, *and*-crossover, *or*-crossover and mutation. Homologous crossover selects common location in both parent trees to carry out the crossover operation. The *and*-crossover combines two parent rules into one rule using the *and* operator. The *or*-crossover combines two parent rules into one rule using the *or* operator. The mutation operation can perform sub-tree, function and terminal mutations, depending on the selected mutation location.

4.2 Fuzzy Rules Evaluation

After evaluation, a fuzzy rule produces a numerical value between 0 and 1. This value indicates the degree of membership the data belongs to the classified permeability. We uses a simple defuzzification mechanism to interpret the result:

if the degree of membership is greater than or equal to 0.5, the data belongs to the classified range.

To assign a fitness to the evaluated fuzzy rule, the rule is first combined with the *best* rule in the other population using the following template:

if rule-1 $\geq$ *0.5*
then high-permeability
else if rule-2 $\geq$ *0.5*
then low-permeability
else medium-permeability.

where *rule-1* is a rule from the first population and *rule-2* is a rule from the second population. If the evaluated rule is from the first population, the $best$ rule from the second population is used to complete the template. If the evaluated rule is from the second population, the $best$ rule from the first population is used to complete the template. This combined if-then-else rule is then tested on the training data and the interpretation results are compared with the transformed permeability. If the if-then-else rule gives the correct interpretation, it is a hit. The percentage of the hit among the training data is the fitness of the evaluated rule. To promote shorter and more readable rules to be evolved, rules with length more than 100 nodes are penalized. Also, the $best$ rule in each population is updated at the beginning of every generation, so that a good rule can be immediately used to combine with rules in the other population and impact evolution.

5. EXPERIMENTAL SETUP

Both Well A and B have a greater number of high and low permeability data segments than medium-permeability data segments. We therefore used one population to evolve rules that separate high-permeability data segments and the other population to evolve rules that distinguish low-permeability data segments. In this way, both populations have a balanced number of positive and negative samples, which is important to train robust classifiers.

We used Well A data to train the fuzzy rules. The final best rule was then tested on Well B data. The crossover rates used are as follows: 20% for homologous crossover, 10% for and-crossover and 10% for or-crossover. Mutation rate is 50%. When no genetic operation is executed, an identical copy of one parent is copied over to the next generation.

The selection scheme is a tournament with size 2. We set the population size as 100 to run for 1,000 generations, where at each generation, the population is 100% replaced by the offspring except one copy of the elite (the best) which is kept and carried over to the new generation.

By combining two rules from two populations, it is sufficient to classify three permeability ranges. However, the order of their combination can effect the classification accuracy. This is because *rule-1* is evaluated first, according to the rule template. Once *rule-1* makes a wrong interpretation, *rule-2* can not correct it. Consequently, *rule-1* has a stronger impact than *rule-2* on the performance of the combined rule.

To achieve a better interpretation accuracy, it is desirable to have the rule which has better accuracy be *rule-1*. Unfortunately, we do not know in advance which of the two rules will give better accuracy. We therefore made two sets of experimental runs. In the first set, *rule-1* is the rule that identifies high-permeability data segments. In the second set, *rule-1* is the rule that classifies low-permeability data segments. Fifty runs were made for each of the two setups and their final best rules were collected for evaluation.

6. RESULTS

Figure 9-19 gives the results of the two sets of runs. As shown, the fuzzy rules which classify high-permeability first produce better results: the average fitness of the best rule from the 50 runs is 0.83 on training data and 0.61 on testing data. The rules that first identify low-permeability data have average fitness of 0.77 on training data and 0.6 on testing data. In both cases, there is a big gap between the fitness on training data and the fitness on testing data. There can be a couple of explanations. First, the two wells have very different geology.

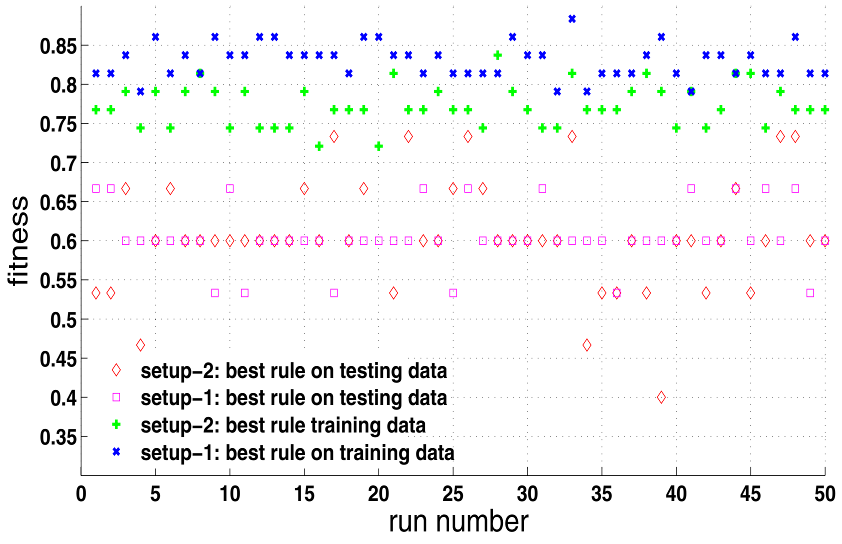

Figure 9-19. Results of the two sets of runs.

The fuzzy rules trained based on log data from Well A therefore do not work as well on Well B. Another explanation is that Well B has a smaller number (15) of data points. Consequently, even a small number of mis-classification (1 or 2) will have strong impact on the classification accuracy. The accuracy measure of Well B, therefore, is not sufficient to reflect the fuzzy rules' performance.

To give a more detailed analysis of the performance of the fuzzy rules, we selected the rule that had the best fitness (0.76) on testing data and plotted its permeability interpretation on training data (Well A) and on testing data (Well B). The results are given in Figure 9-20 and Figure 9-21.

As shown, the fuzzy rule gives permeability interpretations which are very close to the transformed target permeability in both wells. In Well A, 8 out of the 43 segments were mis-classified; all of them have medium-permeability and the fuzzy rule mis-classified them as either low-permeability or high-permeability. The degree of 'mistake' is not too serious.

For Well B, the fuzzy rule mis-classified 4 out of the 15 segments. Among them, 1 segment can be fuzzily interpreted as either medium or high permeability according to the core permeability. The segmentation method transformed it as c (high permeability) while the fuzzy rule interpreted it as medium permeability. Once this segment is excluded, the number of mis-classification on Well B becomes 3 and the classification accuracy improves to 0.8, which is close to the accuracy on Well A (0.81). Based on this detailed analysis, the fuzzy rule gives a reasonably accurate permeability interpretation for both Well A and Well C. This is a very encouraging result.

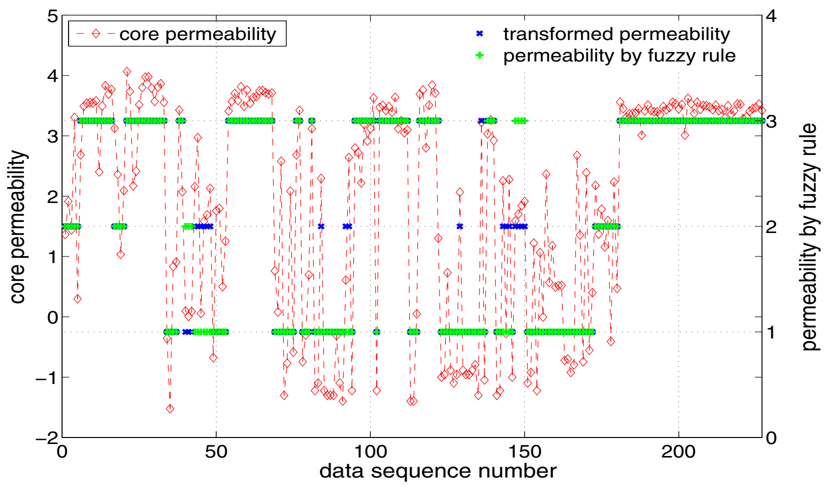

Figure 9-20. Well A permeability.

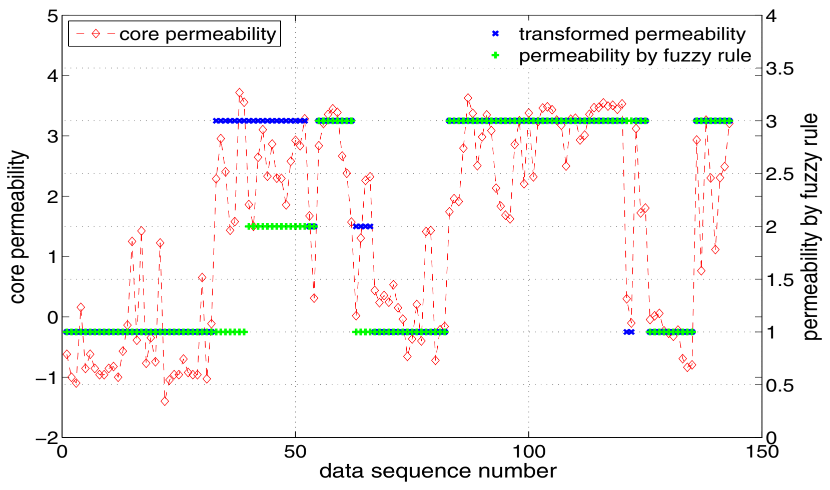

Figure 9-21. Well B permeability.

7. ANALYSIS AND DISCUSSION

To understand why rules that classify high-permeability segments first have produced better results, we calculated the average population fitness and the fitness of the best solution for all runs. The averages of the 50 runs for each set of experiments are plotted in Figure 9-22 and Figure 9-23.

When the first population is used to evolve rules that classify low-permeability segments and the second population is used to evolve rules that classify high-permeability data segments, Figure 9-22 shows that the co-evolution pressure is biased toward the second population. Average fitness of the first population is consistently lower than that of the second population. Using the worse of the two rules (the one from the first population) as *rule-1* to interpret permeability has impaired the overall interpretation accuracy.

This bias, however, does not appear in the other experiment where the rules that classify high-permeability were used as *rule-1* to interpret permeability. As shown in Figure 9-23, both populations co-evolve together with comparable average fitness. This is a healthy co-evolutionary dynamics which has produced combined if-then-else rule that give more accurate permeability interpretations than that by the other experiment.

In both sets of experimental runs, the two populations improved very quickly at the first 200 generations. After that, the improvement is not very visible. This pattern also appears in the fitness improvement of the best combined overall permeability interpreter, although to a lesser extend. One possible reason is

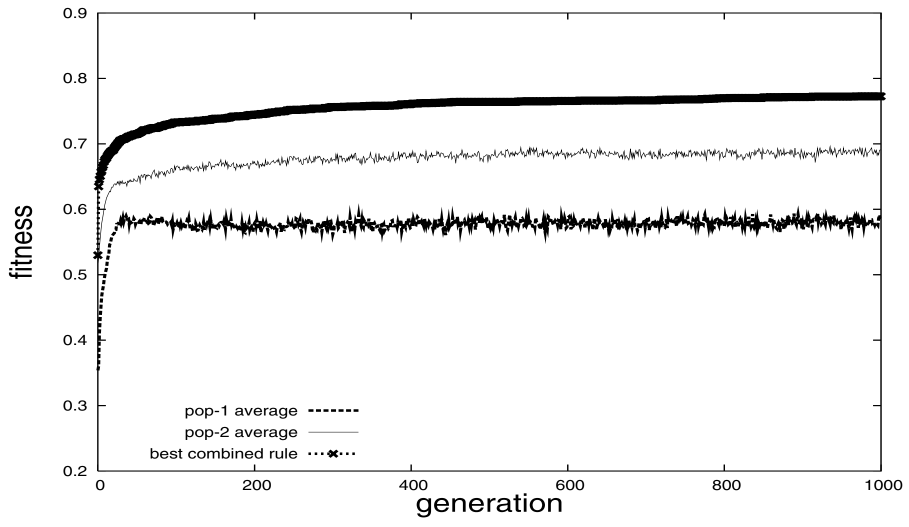

Figure 9-22. Experimental results for runs where population 1 evolves rules to identify low-permeability data.

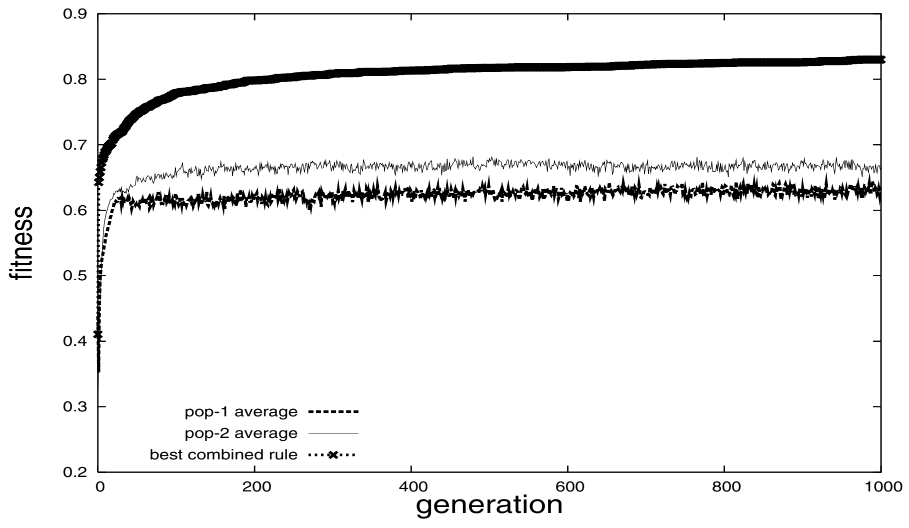

Figure 9-23. Experimental results for runs where population 1 evolves rules to identify high-permeability data.

that the *best* solution used to combine with individuals in the other population for fitness evaluation is updated *every* generation. Such a greedy approach may have reduced the population diversity necessary for continuous evolution. In our future work, we plan to investigate using a less frequent updating scheme so that the two populations only occasionally communicate with each other. This asynchronous version of co-evolution model not only allows each population to have a slower and more stable evolution pace but also is suited for a parallel implementation in which each population is evolved on a separate processor. Such parallel implementation is important for the efficient processing of a large number of well logs simultaneously.

8. CONCLUSIONS

Well log interpretation is a routine, but time-consuming task in oil companies. With the increasing global energy demand, it is a natural trend to seek computerized well log interpretation techniques to provide results more efficiently. In this work, we have devised a co-evolutionary fuzzy system to generate a well log interpreter to automatically process well log data and interpret reservoir permeability. The initial testing results show that the generated fuzzy rules give a sensible permeability interpretation. Although the result is preliminary, it provides initial evidence of the potential of the developed method. We plan to continue the work by extending the system in two areas:

- the capability to evolve fuzzy rule interpreters for other reservoir properties, such as lithology.

- a less frequent rule updating scheme between the two populations, hence the possibility of parallel implementation of the co-evolutionary fuzzy system.

Acknowledgments

I would like to thank Julian Squires for porting the PolyGP system from Haskell to Java. The data set is provided by Chevron Energy Technology Company.

References

Abonyi, J., Feil, B., Nemeth, S., and Arva, P. (2005). Modified gath-geva clustering for fuzzy segmentation of multivariate time-series. *Fuzzy Sets and Systems*, 149:39–56.

Apostolico, A., Bock, M. E., and Lonardi, S. (2002). Monotony of surprise and large-scale quest for unusual words. In *Proceedings of the 6th International Conference on Research in Computational Molecular Biology*, pages 22–31.

Bentley, Peter J. (2000). “Evolutionary, my dear watson” investigating committee-based evolution of fuzzy rules for the detection of suspicious insurance claims. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 702–709. Morgan Kaufmann.

Keogh, Eamonn, Chakrabarti, Kaushik, Mehrotra, Sharad, and Pazzani, Michael (2001). Locally adaptive dimensionality reduction for indexing large time series databases. In *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 151–162.

Larsen, R. J. and Marx, M. L. (1986). *An Introduction to Mathematical Statistics and Its Applications,2nd Edition*. Prentice Hall, Englewood.

Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*.

Pena-Reyes, Carlos Andres and Sipper, Moshe (2001). Fuzzy coco: A cooperative coevolutionary approach to fuzzy modeling. *IEEE Transactions on Fuzzy Systems*, 9(5):727–737.

Potter, Mitchell A. and Jong, Kenneth A. De (1994). A cooperative coevolutionary approach to function optimization. In *Parallel Problem Solving from Nature – PPSN III*, pages 249–257, Berlin. Springer.

Potter, Mitchell A. and Jong, Kenneth A. De (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29.

Yu, Tina (2001). Hierachical processing for evolving recursive and modular programs using higher order functions and lambda abstractions. *Genetic Programming and Evolvable Machines*, 2(4):345–380.

Yu, Tina and Bentley, Peter (1998). Methods to evolve legal phenotypes. In *Parallel Problem Solving from Nature – PPSN V*, pages 280–291, Berlin. Springer.

Yu, Tina and Wilkinson, Dave (2007). A fuzzy symbolic representation for intelligent reservoir well log interpretation. In “Hybrid Intelligent Systems using Soft Computing” of the Series on Computational Intelligence, Springer Verlag Edited by, O. Castillo, P. Melin, W. Pedrycz, and J. Kacprzyk.

Yu, Tina, Wilkinson, Dave, and Xie, Deyi (2003). A hybrid GP-fuzzy approach for reservoir characterization. In Riolo, Rick L. and Worzel, Bill, editors, *Genetic Programming Theory and Practise*, chapter 17, pages 271–290. Kluwer.