Evolving Finite State Transducers to Interpret Deepwater Reservoir Depositional Environments

Tina Yu, Dave Wilkinson, Julian Clark and Morgan Sullivan

Abstract— **Predicting oil recovery efficiency of deepwater reservoirs is a challenging task. One approach to characterize and predict the producibility of a reservoir is by analyzing its depositional information. In a deposition-based stratigraphic interpretation framework, one critical step is the identification and labeling of the stratigraphic components in the reservoir according to their depositional information. This interpretation process is labor intensive and can produce different results depending on the stratigrapher who performs the analysis. To relieve stratigrapher's workload and to produce more consistent results, this research developed a methodology to automate this process using various computational intelligent techniques. Using a well log data set, we demonstrated that the developed methodology and the designed workflow can produce finite state transducer models that interpret deepwater reservoir depositional environments adequately.**

I. Introduction

THE petroleum industry is increasingly moving exploration into the deep-water realm to meet the growing demand for oil and gas. Deepwater reservoir projects involve a large amount of financial investment; consequently, business decisions need to be made based on a clear understanding of the producibility of the reservoirs.

One key geologic characteristics that has strong impact on a reservoir's oil recovery efficiency is its depositional environment: the area in which and physical conditions under which sediments are deposited. These include sediment source and supply, depositional processes such as deposition by wind, water or ice, and location and climate, such as desert, swamp or river. Most deepwater reservoirs are deposited in a wide range of depositional systems, and occur at a variety of temporal and spatial scales. Prediction of net-to-gross, continuity, architecture, and quality of these reservoirs therefore requires integration of seismic, well-log, and core data with appropriate subsurface and outcrop analogs [6].

When conducting such a comparative analysis, it is critical that similar stratigraphic components are compared to one another. The first step in the analysis, therefore, is to identify and label each of the stratigraphic components according to their depositional information (see Section II). This interpretation process is labor intensive and can produce different results depending on the stratigrapher who performs the analysis. With the goal of relieving stratigrapher's workload and to produce more consistent results, this research developed a methodology to automate this process using various computational intelligent techniques. In particular, we treat the problem as a language interpretation problem. The task is to translate a sequence of symbols from one language (reservoir well log data) into another language (the depositional labels).

Finite state transducers (FSTs) techniques have been widely applied to human language translation. Examples are text-to-speech pronunciation modeling [1] and the parsing of web page [2]. In this research, we applied an evolutionary computation system to generate FSTs that translate a sequence of reservoir well log data into a sequence of depositional labels. We also conducted a case study on a deepwater reservoir data set using the developed techniques. The results are comparable to the depositional labels produced by the stratigrapher on this data set. As the project is at the concept-proofing stage, only one data set was provided to model the process and design the workflow. Encouraged by the initial success, we will continue the project and acquire new data sets from other deepwater reservoir fields to test the developed methods.

The paper is organized as follows. Section II explains a stratigraphic interpretation framework using depositional information, which this research is based on. The developed methodology and the designed workflow are then presented in Section III. In Section IV, the data set used in our case study is described. Following the designed workflow step by step, we give the work conducted and their results in Section V, VI and VII. In Section V, well log data were transformed into fuzzy symbols. In Section VI, five classifiers to distinguish 5 depositional types were generated using a co-evolutionary system. Section VII reports the results using the produced fuzzy symbols and classifiers from the previous sections to generate FSTs. With a brief outline of our future work, we conclude the paper in Section VIII.

II. A Stratigraphic Interpretation Framework

One systematic approach to identify and label stratigraphic components of deepwater reservoirs is by describing them within a hierarchical framework that is based solely on the physical attributes of the strata and is independent of thickness and time. In this framework, the fundamental building block of this hierarchical classification is an *element*, defined as the cross-sectional characterization of the *volume of sediment deposited* within a single cycle of deposition and bounded by an avulsion or abandonment. With this classification scheme, individual *elements* exhibit a predictable change from axis to margin in grain size, litho-facies type and

Tina Yu is with the Department of Computer Science, Memorial University of Newfoundland, St. John's NL A1B 3X5, Canada (email: tinayu@cs.mun.ca).

Dave Wilkinson, Julian Clark and Morgan Sullivan are with Chevron Energy Technology Company, San Ramon CA 94583, U.S.A. (email: {davidawilkinson}@chevron.com).

978-1-4244-1823-7/08/$25.00

architectural style. Meanwhile, since avulsion, which is the lateral shifting of a channel or lobe, controls the distribution of these characteristics, *elements* can be used to understand the distribution of reservoir and non-reservoir facies.

Two or more *elements* of similar grain size, litho-facies and architectural style form a *complex*. Elements within a complex are genetically related and exhibit a predictable organization and depositional trend. A *complex set* is comprised of either individual complexes of different architectural style and/or complexes of similar architectural style that exhibit depositional trends independent of one another. The description of deepwater sand-bodies utilizing this hierarchical approach provides a powerful methodology to directly compare similar stratigraphic components and greatly improve reservoir characterization and the prediction of productivity.

The *elements* that are of particular interest are *channel*-related as they are the areas where hydrocarbon (oil and gas) deposit. For a finer characterization of a reservoir, we subdivide channel-elements into *channel-axis*, *channel off-axis*, and *channel-margin* associations. Channel-axis deposits (A) are dominated by highly-amalgamated, massive sandstones deposited by high-concentration turbidity currents and exhibit a sharp-based, blocky *gamma ray* response. The channel off-axis association (OA) typically displays weakly blocky to moderately serrated *gamma ray* log character and is composed of stacked, semi- to non-amalgamated, massive to planar-stratified sandstones and inter-laminated shales. The channel-margin deposits (M) contain a variety of litho-facies and are characterized by a hetero-lithic mixture of high- and low-concentration turbiditic sandstones interbedded with thick shales exhibiting a serrated, and generally high *gamma ray* log response.

Two other element types that are non-channel and need to be identified and separated from channel elements are *overbank* and *mass transport complex*. Overbank deposits (OB) are dominated by shale and interbedded with thin sandstones which display an irregular character, lacking a distinct *gamma ray* log trend. Mass transport complexes (MTC) consist of aggregated components dominated by mass transport. Mass wasting of basin margins and the influx of large quantities of resedimented material may occur at any time as a basin fills. Depending on their source, these complexes can either be very muddy or very sandy, but all tend to be internally chaotic. Due to the lithologic variability of MTC, it is difficult to uniformity character their log response, but commonly they display an irregular, chaotic character with an elevated *gamma ray* response.

According to the described classification scheme, the five depositional types (A, OA, M, OB and MTC) have different response to gamma ray. It is natural to use gamma ray (GR) log data as an indictor to classify the 5 types of deposition.

III. Methodology and Workflow

To automate this process of using GR log data to classify 5 different depositional types, we take the machine learning approach: using GR log data and the depositional labels assigned by a stratigrapher to train a model that can produce the same results as that produced by the stratigrapher. Figure 1 gives the series of steps developed to achieve that goal.

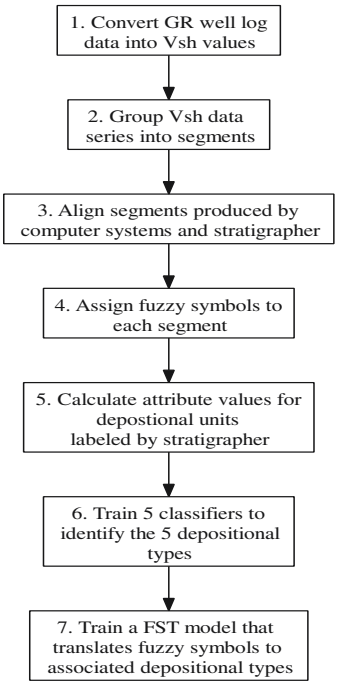

Fig. 1. Workflow to automate the interpretation of reservoir depositional environments.

First, GR data (whose values range between 0 and 150 API gamma ray units) are converted into *volume of shale* (Vsh) using Equation 1. The purpose of this transformation is to normalize the data (Vsh has value between 0 and 1) while maintaining the trend in the original data. This normalization is necessary for the later step of symbol assignment (Step 4). In Equation 1, GR_{min} is the minimum GR reading from the log and GR_{max} is the maximum GR reading from the log.

$$Vsh_i = \begin{cases} 0, & GR_i < GR_{min}; \\ \frac{GR_i - GR_{min}}{GR_{max} - GR_{min}}, & GR_{min} \leq GR_i \leq GR_{max}; \\ 1, & GR_i > GR_{max}. \end{cases} \quad (1)$$

Step 2 groups the Vsh data series into segments. This step mimics the blocking process (group similar consecutive GR data into a block) that a stratigrapher goes through when performing interpretation. By examining blocked GR data and their associated thickness, a stratigrapher can decide the depositional type of each block.

Segments produced by computer systems are likely to be slightly different from that blocked by humans. If the differences are in the form of small edge effects, the boundary locations are adjusted (Step 3) to make them more reasonable.

Segmented Vsh data are represented as a series of numerical values, $\overline{Vsh} = \overline{s_1}, \overline{s_2}, \overline{s_3} \ldots$, where $\overline{s_i}$ is the average of the data within the ith segment. This numerical representa-

tion is farther simplified using symbols by assigning different value range to each symbol. Depending on the value range that $\overline{s_i}$ falls into, its associated symbol is determined.

Unlike numerical values, which are continuous, symbols are discrete and bounded. This makes them easier for computer to compare and manipulate for interpretation. To enhance the expressive power of the representation, fuzzy symbols are used for segments whose values lay between the boundaries of two symbols.

Step 4 completed the preprocessing of GR well log data. The transformed fuzzy symbols are ready to be used to train FST models that translate these fuzzy symbols to depositional labels.

A FST is a model that maps strings in a source language into a string in a target language. The decision of what output symbol to produce depends on two factors: the input symbol and the current state. However, stratigraphic interpretation is more complicated. In addition to GR reading, a stratigrapher considers other factors, such as the thickness of each block and the degree of variation of neighboring blocks, to give interpretation. In other words, the output symbol is decided by a model, in addition to the input symbol and the current state. This model is a decision model that gives one of the 5 possible depositional types as its output.

We constructed the decision model using a co-evolutionary system [9]. In order to construct the model that contains similar knowledge as that used by the stratigrapher to classify 5 different depositional types, Step 5 calculates the attribute values listed in Table I for every depositional unit identified by the stratigrapher.

Each depositional unit can contain 1 or more Vsh symbols. In Table I, "symbol%" column gives the percentage of each symbol's thickness over the total thickness of the unit. The "symbol thickness" column gives the accumulated thickness of each symbol in the unit. The "symbol max" column gives the maximum thickness of each symbol in the unit. The Variation is the average *distance* of the neighboring symbols in the unit, where *distance* is defined as the number of jumps between 2 symbols. For example, the distance between symbols a and dc is 8. Variation of symbol sequence a, ba, dc is (2+6)/2=4.

TABLE I

ATTRIBUTES CALCULATED FOR EACH DEPOSITIONAL UNIT.

symbol%	symbol thickness	symbol max
a%	a_ thickness	a_max
ab%	ab_thickness	ab_max
ba%	ba_thickness	ba_max
b%	b_thickness	b_max
bc%	bc_thickness	bc_max
cb%	cb_thickness	cb_max
c%	c_thickness	c_max
cd%	cd_thickness	cd_max
dc%	dc_thickness	dc_max
d%	d_thickness	d_max
variation	total_thickness	no_segments

Step 6 uses these attribute values and their associated depositional types to train 5 classifiers.The 5 classifiers, in addition to the GR fuzzy symbol inputs and their associated thickness, are then used to train a FST as the final model (Step 7). This model takes a sequence of fuzzy symbols representing GR readings and their associated thickness as inputs. It produces a sequence of depositional labels based on the decisions made by the 5 classifiers.

To apply the trained FST model to interpret new GR log data, the GR log is first transformed into symbols following Step 1 – 4. Acting like a stratigrapher, the FST model would interpret these symbols and assign their associated depositional types.

IV. Data Set

One GR well log data set from a deepwater reservoir was provided for us to develop and test our methodology. The GR readings consist of 6,150 data points, each of which was taken at a half foot interval between 4,200 and 7,200 feet under the earth's surface. A stratigrapher has examined the data and assigned 50 depositional labels at different depth level. Follow the workflow in Figure 1, we first converted the data into Vsh values. The transformation of Vsh into fuzzy symbols is given in the following section.

V. Transform Log Data into Fuzzy Symbols

There are three steps in this transformation process (Step 2, 3, 4). The first step is segmentation, which 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.

There are various approaches to segment data. Due to budget constraint, we only adopted one [3] and modified it to suits our project. This segmentation method 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 of segment.

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:

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

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 value in segment a.

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. We therefore devised a compromised solution where a penalty is paid for increasing the number of segments. Equation 3 gives the new $error$ criterion which is defined as the total previous error $plus$ the number of segments, where N is the number of segments.

$$f = N + \sum_{i=1}^{N} error_i \tag{3}$$

Based on the described concept, a computer system was developed to carry out the segmentation task. Computer generated segments, however, are not always aligned with the data points where stratigrapher assigns depositional labels. We adjusted the boundary locations of the depositional units to fix the edge effect. In most cases, this requires shifting the labels up or down a few data points.

The last step of data preprocessing is fuzzy symbols assignment. Segmented Vsh data are represented as a set of numerical values, $\overline{Vsh} = \overline{s_1}, \overline{s_2}, \overline{s_3} \ldots$, where $\overline{s_i}$ is the mean value of the data within the ith segment. We assign each segment with one of the symbols a, b, c, d according to the following rule:

$$symbol_i = \begin{cases} a, & \overline{s_i} < 0.3; \\ b, & 0.3 \leq \overline{s_i} < 0.5; \\ c, & 0.5 \leq \overline{s_i} \leq 0.7; \\ d, & \overline{s_i} > 0.7. \end{cases} \tag{4}$$

While some segments are clearly within the boundary of a particular symbol region, others may not be so clear. In this case, a crisp symbol does not represent its true value. In contrast, fuzzy symbols use membership function to express the segment can be interpreted as both symbols with some possibility. Fuzzy symbols are more expressive in this case.

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, Vsh data have values between 0 and 1. Since we adopted 4 symbols to represent Vsh data, 4 trapezoidal-shaped membership functions were designed, one for each of the 4 symbols. The details are given in [8].

Using the 4 trapezoidal-shaped MFs, each segment ($\overline{s_i}$) is mapped into one of the 10 symbols: a, ab, ba, b, bc, cb, c, cd, dc, d. Symbol ab indicates that the segment belongs to the region of symbol a more than the region of symbol b. If a segment belongs to the region of one symbol 100%, that symbol is used to represent the segment.

A. Results

The 6,150 data points were segmented and mapped into 62 symbols. They are shown in Figure 2. The division points of the 62 symbols are not all aligned with the positions where the stratigrapher assigned the 50 depositional labels. In the case where one symbol region has more than one depositional labels assigned to it, we divided the region into segments according to where the label position is. After the process, the total number of symbol regions is 82.

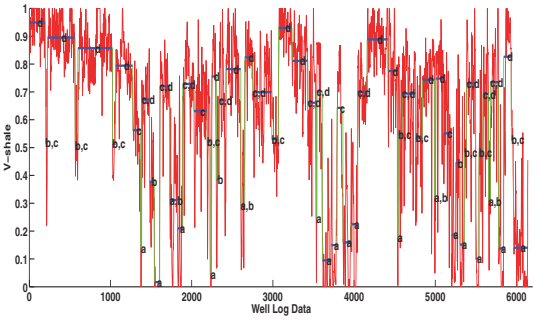

Fig. 2. Vsh data transformed into 62 fuzzy symbols.

The number of segments in each depositional unit varies, ranging from 1 to 5. Also, some segments have a larger number of data points (thicker) than others. This information is important in determining their depositional types. Step 5 calculates the attribute values listed in Table 1 for each of the 50 depositional units. The resulting data are then used to train 5 classifiers described in the following section.

VI. Co-Evolution of Classifiers

Among the 50 depositional units, 4 are A, 9 are OA, 14 are M, 19 are OB and 4 are MTC. If we train each classifier individually, it would be very difficult to obtain good results since the number of data points for one class is too small to balance against the number of data points for the 4 other classes. An alternative approach is to train the 5 classifiers simultaneously. The co-evolutionary system described in [9] was developed based on this concept.

In this co-evolutionary system, a classifier is represented as a rule tree (see Figure 6). Five populations, each for one of the five classifiers, are evolved independently and simultaneously. To encourage the co-operation of the 5 populations to evolve the best team, the fitness of an evolved individual rule is determined by how well it collaborates with the rules evolved in the 4 other populations to perform the classification task.

In terms of implementation, a rule from one population is combined with the $best$ rules in the 4 other populations using an if-then-else template, such as the following:

if (*OA-rule* is evaluated to be True)
  then *OA*
else if (*A-rule* is evaluated to be True)
  then *A*
else if (*MTC-rule* is evaluated to be True)
  then *MTC*
else if (*OB-rule* is evaluated to be True)
  then *OB*
else *M*.

The performance of this combined team defines the fitness of the rule in the current population.

At generation 0 when no $best$ rule is known, a rule is randomly selected from each population to represent the $best$ rule for that population. After that, the $best$ rules is updated at the end of every generation, so that a good rule can be immediately used to combine with rules in the other population and impact evolution. This co-operative co-evolution model was proposed by [5]. Figure 3 illustrates this co-evolution mechanism using 2 populations.

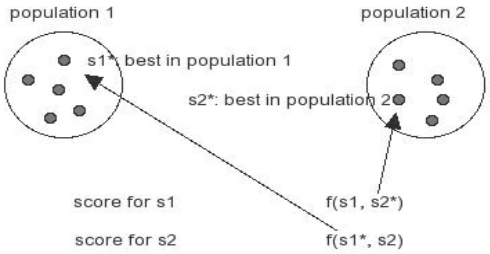

Fig. 3. The co-operative co-evolution model.

With 5 rules in each team, there are 5! ways to combine them using the if-then-else template. Since rules applied earlier in the if-then-else template are weighted more than rules that are applied later in the classification decision, it is important to acquire a suitable rule sequence for a team to achieve good classification results.

To learn the best rule order strategy, we used a simple hill-climbing approach. At the beginning of each experimental run, a random order of rule sequence is generated (e.g. OA, A, MTC, OB, M). This order is used to combine rules for fitness evaluation at generation 0. At the end of generation 0, the best rule from each of the 5 populations are selected. Meanwhile, the current order sequence is mutated to obtain a new rule order sequence. If the team of the 5 best rules combined using the new order sequence gives better or equal classification accuracy than that produced by the team combined based on the old sequence, the new sequence is kept and used in the next generation for rule combination. Otherwise, the old sequence is retained and reused in the next generation.

A. Experimental Setup

The co-evolutionary model is implemented in a genetic programming system called PolyGP [7]. The function set to construct the classification rules are: and, or, nor and $nand$. They can combine any subset of the 33 attributes listed in Table 1 to form Boolean rules.

To work with the 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.

The fitness function is the hit rate of the combined if-then-else rule. When the rule gives a correct interpretation for a data point, it is a hit. The percentage of hits on the entire training data set is the fitness of the evaluated rule.

Among the 5 depositional types, A and OA are close to each other while M, OB and MTC are similar. To promote rules that produce less serious mis-classification, the distance between the target label and classified label is used as the secondary criteria for selection. Here, the 5 depositional labels are ordered as follows: A, OA, M, OB, MTC. The *distance* between two depositional labels is the number of jumps between them. For example, the distance between A and OA is 1 and the distance between A and MTC is 4. If two rules have the same hit rate, the one that gives a smaller distance error is the winner during selection. Table II lists the parameter values used to conduct 50 experimental runs.

TABLE II
GP PARAMETERS FOR EXPERIMENTAL RUNS.

parameter	value	parameter	value
tournament selection	size 2	crossover rate	30%
max tree depth	6	orXover rate	30%
population size	100	mutation rate	30%
max generation	200	andXover rate	30%

B. Results

Figure 4 gives the average fitness of the 5 co-evolved populations. It shows that all 5 populations improve consistently throughout the run of 200 generations. Among them, MTC-rule population has the highest average fitness. This is followed by the A-rule and OB-rule populations. The M-rule and OA-rule populations have the lowest average fitness.

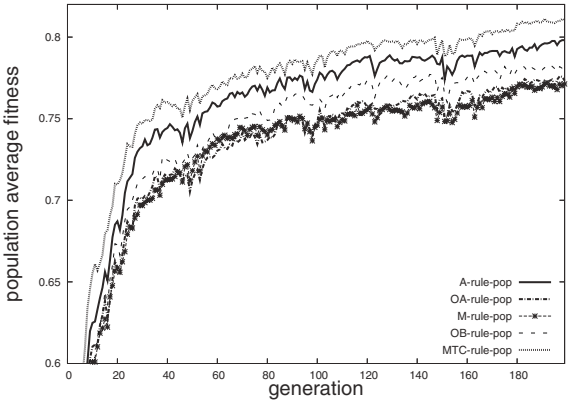

Fig. 4. Average fitness of the 5 co-evolved populations.

These results suggest that MTC-rule has the least impact on the overall classification results. Once $best$ rules from

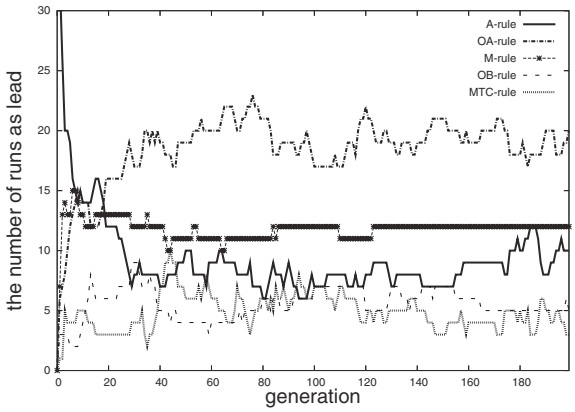

Fig. 5. The number of runs that each rule is selected as the lead.

other populations are integrated, any MTC-rule in the population can produce good classification results. We examined the number of runs that each rule is selected as the lead classifier (see Figure 5) and found that MTC-rule is the least selected one, indicating that it is of the least impact. This result is consistent with our hypothesis.

The two most selected rules as lead classifiers in the team are OA-rule and M-rule. Figure 4 also shows that these two populations have the lowest average fitness; the $best$ rules from other populations are not sufficient for the team to produce good classification accuracy. Good OA and M rules are critical for a team to produce good classification accuracy. These evolutionary dynamics suggests that OA-rule and M-rule have more impact on the overall classification results than other rules.

One factor that might have contributed to this evolutionary dynamics is the number of training data for each class. In general, training process biases toward the class that has a larger number of training data. As a result, that classification rule has a higher impact on the final classification results. In this data set, MTC class has the smallest number (4) of data points while OA and M classes have the largest number (14 and 19). It is understandable that the training process ignored MTC-rule but focusing on evolving good OA and M rules to produce good overall classification results.

The best team produced from the 50 runs gives classification accuracy of 90%: 5 of the 50 depositional units were mis-classified. Table III gives the details.

TABLE III

CLASSIFICATION ACCURACY OF THE BEST TEAM.

	A	OA	M	OB	MTC	A	OA	M	OB	MTC
A	4	0	0	0	0	100%	0%	0%	0%	0%
OA	0	9	0	0	0	0%	100%	0%	0%	0%
M	0	0	13	1	0	0%	0%	93%	7%	0%
OB	0	0	1	18	0	0%	0%	5%	95%	0%
MTC	0	0	1	2	1	0%	0%	25%	50%	25%

Among the 5 mis-classified labels, two MTC were mis-labeled as OB, one MTC was mis-labeled as M, one OB was mis-labeled as M and one M was mis-labeled as OB. As mentioned previously, M, OB and MTC have similar depositional ingredients. These mis-classifications, hence, are not considered to be serious.

Figure 6 gives the A-rule tree in the best team. Its interpretation is straight-forward: symbol a (which represents the smallest Vsh value) has to dominate the segments in the depositional unit. This model accurately describes the composition of channel-axis deposition: massive sandstone without shale (see Section II). Due to space restriction, the other 4 rules are not included in the paper but can be found in http://www.cs.mun.ca/∼tinayu/cec2008.htm. All of them are geologically sensible models.

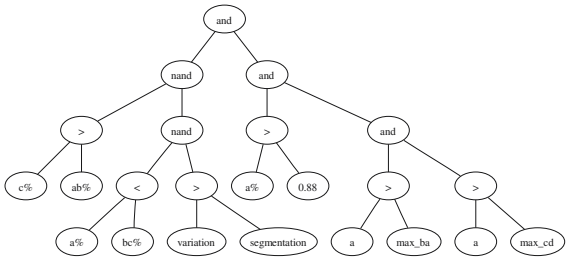

Fig. 6. The rule that classifies depositional type "A".

VII. EVOLVING FINITE STATE TRANSDUCERS

With the establishment of the 5 classifiers, we can proceed to the last step of FSTs training. As mentioned previously, a FST is a model that translates strings from an input language to strings from an output language. In our case, the input string is a sequence of symbols representing Vsh values and the output string is a sequence of depositional labels.

There are various ways to represent a FST, such as a directed graph. In this research, we followed the work of [4] and used two tables to represent a FST. The first table is transition table (see Figure V), which gives the next state of a FST based on the input symbol and the current state. The second table is output table (see Figure VI) which gives the output symbol also based on the input symbol and the current state. In our case, the entries in the output table are not output symbols themselves. Instead, they are names of the 5 classification rules. The following sub-section describes the FST operations.

A. FST Operating Procedures

Figure 7 gives the workflow of the FST operation. At each step, one symbol and the thickness of the segment that the symbol represents are processed. The information is used to update the attribute values in the database listed in Table I. Meanwhile, the input symbol and the current state are used to look up the classification rule in the output table. The name of the rule is the proposed depositional type for the segments processed so far yet been labeled.

After the rule is applied on the updated attribute values, it returns a value of `true` or `false`. If the value is `true`, it means that the segments satisfy the criteria of that depositional type. The associated deposition label becomes

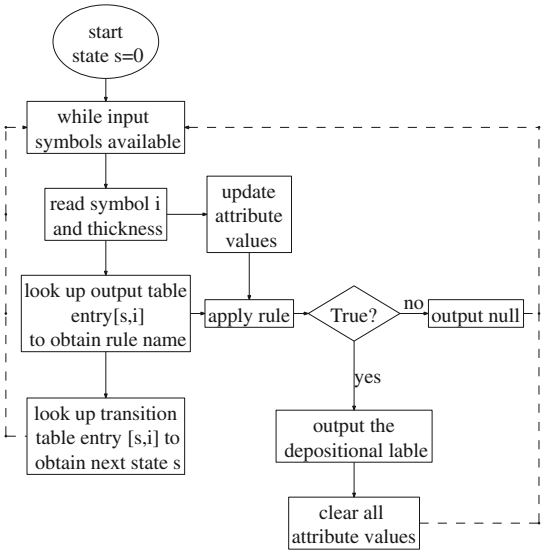

Fig. 7. The operating procedure of a FST.

the output symbol. Once a label is assigned to the current segments, all attributes values in the database are set to 0. The system is ready to process the next symbol and decides where to assign the next deposition label on the symbol sequence.

If the classification rule returns `false`, it means that the current segments do not satisfy the criteria of the proposed depositional type. An output symbol null is given and the system continues to process the next symbol and update the attribute values in the database. These steps repeat many times until all input symbols are processed.

Using the FST in Table V and VI, we show its operation on input symbols $(d, 96, d, 3, cb, 12.5)$. With the initial state 0 and input symbol d, the proposed classification rule is OB, according to the output table. After updating the attribute information, OB rule is fired. Assume the rule returns `true` on the attribute value, OB is the output symbol. All attribute information in the database are cleared and the system moves to state 18 according to the transition table. Next, symbol d is processed and the attribute values are updated. The proposed rule according to output table is OA. Assume the rule returns `false` on the attribute values, null is the output symbol. The system moves to state 10 without erasing the attribute values.

The next symbol to process is cb. The attribute values will now reflect the information of two segments (d and cb). The proposed classification rule is A. Assume the rule returns `true` on the updated attribute values, A is the output symbol. Since there is no more input symbol, the interpretation process terminates. The system clears the attributes memory and moves to state 7. The output sequence produced by the FST on the given input symbols is $(OB, null, A.)$

B. The Evolutionary System

The two tables of the FST have the same size of $N_Q \times N_I$, where N_Q is the number of states and N_I is the number of input symbols. In this study, N_I is 10: $a, ab, \ldots d$ and N_Q is 20, which was decided based on the complexity of the problem. With 5 possible output symbols (N_O), the number of possible FSTs is:

$$S = N_Q^{N_Q \times N_I} \times N_O^{N_Q \times N_I} \approx 10^{400} \quad (5)$$

This is a huge search space and stochastic search methods such as evolutionary algorithms are good candidates to locate a decent solution within a reasonable time frame.

To work with the two-table FST representation, genetic operators need to be modified. This system only uses mutation operator, which works the same way as that in [4]. First, a decision is made with equal probability to either mutate the transition table or the output table. A random location is then selected in the chosen table, and the entry there is modified. This ensures that mutation causes at least one change. After that, an iteration is performed over all the table entries apart from the entry just modified, changing each entry with a probability of $\frac{1}{N_Q \times N_I}$. When an entry is modified, a symbol is chosen from a uniform distribution of all possible symbols except the current symbol. According to [4], a single call to the mutation operator is most likely to produce one or two changes to the FST tables, but can also produce more.

The fitness of a FST is based on the output symbols it produces. After processing an input symbol, a FST always produces an output symbol, either it is a depositional label or null. The length of the produced output symbols, hence, is always the same as the length of the input symbols.

The produced output symbols are aligned with the depositional labels produced by the stratigrapher and the number of mis-match between the two is the FST's fitness. A FST that produce all depositional labels correctly at the correct segment position has fitness value 0.

To encourage FSTs to produce less serious mis-match, we used the same distance measure used to co-evolve classification team as the secondary criteria for selection (see Section VI-A). If two FSTs have the same number of mis-match, the one with a smaller distance error is the winner during selection. Table IV lists the parameter values used to conduct 100 runs.

TABLE IV
GA PARAMETERS FOR EXPERIMENTAL RUNS.

parameter	value	parameter	value
tournament selection	size 2	elitism	size 1
population size	50	N_Q	20
max generation	1000	mutation rate	2%

C. Results

Figure 8 gives the average and the best fitness of the population averaged over 100 runs. It shows that they improved consistently throughout the run of 1,000 generations.

Among the FSTs generated from the 100 runs, the best one produced 7 mis-matches for the input of 82 symbols:

- two M were mis-labeled as OA;
- one A was mis-labeled as OA;

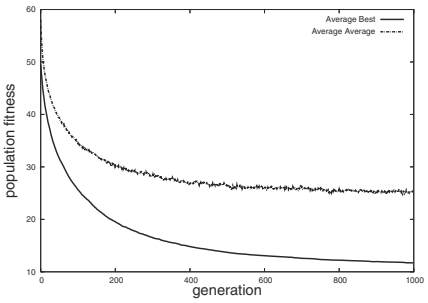

Fig. 8. Average and best population fitness of evolved FST.

- two OB were not labeled (null output). The segments were then combined with the following segments and labeled as M, which was the correct label;
- two OB were mis-labeled as M.

These results are comparable to that produced by the stratigrapher on this data set. We list the transition table and the output table in Table V and Table VI.

TABLE V

THE BEST FINITE STATE TRANSDUCER TRANSITION TABLE.

input	a	ab	ba	b	bc	cb	c	cd	dc	d
S0	S8	S2	S19	S9	S1	S14	S11	S7	S18	S18
S1	S9	S17	S4	S5	S3	S2	S14	S12	S2	S10
S2	S9	S18	S1	S10	S3	S9	S16	S4	S1	S3
S3	S15	S9	S15	S0	S16	S13	S14	S17	S16	S2
S4	S0	S0	S17	S8	S7	S9	S3	S6	S6	S13
S5	S14	S12	S9	S0	S14	S16	S6	S3	S3	S8
S6	S1	S14	S12	S19	S3	S1	S16	S1	S3	S13
S7	S17	S19	S4	S19	S3	S10	S6	S5	S15	S15
S8	S12	S6	S5	S13	S16	S1	S4	S14	S16	S3
S9	S3	S19	S4	S19	S11	S1	S2	S15	S16	S8
S10	S7	S9	S19	S6	S16	S7	S11	S15	S7	S6
S11	S4	S13	S19	S18	S10	S8	S19	S15	S2	S12
S12	S19	S1	S6	S14	S11	S9	S3	S18	S3	S10
S13	S10	S11	S10	S11	S7	S8	S3	S15	S17	S6
S14	S9	S16	S0	S3	S4	S3	S8	S15	S5	S3
S15	S13	S13	S3	S6	S9	S8	S3	S7	S18	S6
S16	S18	S6	S2	S5	S0	S14	S10	S14	S11	S4
S17	S9	S16	S4	S6	S7	S6	S13	S7	S9	S4
S18	S1	S12	S19	S6	S2	S9	S0	S0	S5	S10
S19	S13	S2	S15	S18	S14	S0	S18	S2	S12	S0

As the project is at the concept-proofing stage, we don't have a second data set to test the generated FST. Nevertheless, we have demonstrated that the developed methodology and the designed workflow can produce FST models that interpret deepwater reservoir depositional environments adequately.

VIII. CONCLUDING REMARKS

Reservoir characterization continues to present new challenges as exploration moves to new territories, such as deepwater. We have recently developed a stratigraphic interpretation framework that analyzes depositional information to improve reservoir characterization and the prediction of productivity. This approach requires the identification and labeling of stratigraphic components in order to conduct comparative study. The quality of the component interpretation, hence, is very importance to the characterization of a reservoir and can impact reservoir management decisions.

This research developed methods and designed workflow to automat this interpretation process. In particular, we treat the problem as a language translation problem and used evolutionary systems to train FST models to carry out the translation task. Step by step, we illustrated the generation of FST models using a well log data set. The final FST model produces interpretation results that are comparable to the interpretation of the stratigrapher on this data set.

TABLE VI

THE BEST FINITE STATE TRANSDUCER OUTPUT TABLE.

input	a	ab	ba	b	bc	cb	c	cd	dc	d
S0	OA	OB	OB	OB	OA	A	MTC	M	OB	OB
S1	OA	MTC	OB	MTC	MTC	M	M	M	OB	M
S2	OB	OA	MTC	OA	MTC	M	OB	M	OA	A
S3	M	OA	OA	A	M	MTC	OB	M	OB	OB
S4	M	MTC	A	OB	M	OB	OA	OB	MTC	OB
S5	M	A	M	OA	M	OA	A	A	MTC	A
S6	A	M	OA	MTC	MTC	OA	OB	OA	M	A
S7	OA	M	M	OB	M	A	M	OA	MTC	M
S8	OA	OB	MTC	MTC	OA	OB	MTC	MTC	M	M
S9	OA	OA	A	A	MTC	MTC	MTC	M	OA	MTC
S10	OA	A	OA	OA	OB	A	MTC	OA	A	OA
S11	MTC	OB	OB	OB	A	MTC	M	A	MTC	OA
S12	M	M	M	MTC	OA	A	OB	MTC	MTC	OB
S13	OA	MTC	M	M	OA	MTC	OB	OB	OA	OA
S14	A	MTC	MTC	MTC	MTC	OA	OB	M	MTC	M
S15	M	OB	OB	A	MTC	A	OB	MTC	A	OB
S16	OA	MTC	A	MTC	MTC	OB	M	MTC	OA	OA
S17	A	OA	A	OA	M	OA	OB	OB	M	M
S18	A	MTC	OB	OA	MTC	A	OB	MTC	OB	OA
S19	A	MTC	A	A	OA	OB	MTC	A	OA	MTC

With this encouraging initial result, we will continue this research in the following areas:

- acquire new data sets to test the generality and scalability of the developed methods.
- help stratigraphers to understand the FST model training process.
- apply the same method to interpret the higher-level components in the hierarchical framework.

REFERENCES

[1] Daniel Gildea and Daniel Jurafskyd. Automatic Induction of Finite State Transducers for Simple Phonological Rules. In *Proceedings of the 33rd Conference on Association for Computational Linguistics*, pp 9–15, 1995.

[2] C.-N. Hsu and M.-T. Dung Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web. *Information Systems*, vol. 23, no. 8, pp. 521-538, 1998.

[3] J. Lin and E. Keogh and S. Lonardi and B. Chiu 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*, 2003.

[4] S. M. Lucas and T. J. Reynolds Learning Finite State Transducers: Evolution Versus Heuristic State Merging. *IEEE Transactions on Evolutionary Computation*, vol. 11 no. 3, pp. 308-325, June 2007.

[5] M. A. Potter and K. A. De Jong. A Cooperative Coevolutionary Approach to Function Optimization. In *Parallel Problem Solving from Nature – PPSN III*, pages 249–257, Berlin, 1994. Springer.

[6] G. Shanmugam. Deep-water Processes and Facies Models : Implications for Sandstone Petroleum Reservoirs, Elsevier:Oxford, 2006

[7] T. Yu and C. Clack. PolyGP: A Polymorphic Genetic Programming System in Haskell. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 416–421. Morgan Kaufmann, 1998.

[8] T. Yu and D. Wilkinson A Fuzzy Symbolic Representation for Intelligent Reservoir Well Logs Interpretation. In O. Castillo, P.Melin, W. Pedrycz and J. Kacprzyk, editors, *Hybrid Intelligent Systems using Soft Computing*, Springer Verlag, 2007.

[9] T. Yu and D. Wilkinson, A Co-evolutionary Fuzzy System for Reservoir Well Logs Interpretation. In T. Yu, L. Davis, C. Baydar, and R. Roy, editors, *Evolutionary Computation in Practice*, Chapter 9, pages 199–218, Springer, 2008.