Coevolution of Simulator Proxies and Sampling Strategies for Petroleum Reservoir Modeling

Tina Yu *Member, IEEE* and Dave Wilkinson

Abstract— **Reservoir modeling is an on-going activity during the production life of a reservoir. One challenge to constructing accurate reservoir models is the time required to carry out a large number of computer simulations. This research investigates a competitive co-evolutionary algorithm to select a small number of informative reservoir samples to carry out the computer simulation. The simulation results are also used to co-evolve the computer simulator proxies. We have developed a co-evolutionary system incorporating various techniques to conduct a case study. Although the system was able to select a very small number of reservoir samples to run the computer simulations and use the simulation data to construct simulator proxies with high accuracy, these proxy models do not generalize very well on a larger set of simulation data generated from our previous study. Nevertheless, we have identified that including a test-bank in the system helped mitigating the situation. We will conduct more systematic analysis of the competitive co-evolutionary dynamics to improve the system performance.**

I. INTRODUCTION

RESERVOIR modeling is an on-going activity during the production life of a reservoir. When an oil field is first discovered, the reservoir model is constructed using geological data, such as porosity and permeability of the reservoir rocks. Once the field enters into the production stage, many changes take place in the reservoir. For example, the extraction of oil/gas/water from the field can cause the fluid pressures of the field to change. In order to obtain the most current state of a reservoir, these changes need to be reflected in the model. History matching is the process of updating reservoir model descriptor parameters to reflect such changes, based on the production data collected from the field. Using the updated models, petroleum engineers can make more accurate production forecasts. The results of history matching and subsequent production forecasting strongly impact reservoir management decisions.

Reservoir history matching is usually carried out through computer simulation. Normally, multiple simulation runs are conducted to identify reservoir models that generate fluid flows matching the historical production data. Since computer simulation is very time consuming, only a small number of simulations are conducted and the history matching results are associated with uncertainty.

Previously, we have introduced a framework applying genetic programming (GP) [11] to construct a reservoir simulator proxy to enhance the history matching process [18]. Unlike the full reservoir simulator, which gives the flow of fluids of a reservoir, this proxy only labels a reservoir model as "good" or "bad", based on whether or not its flow outputs match well with the production data. In other words, this proxy acts as a classifier to separate "good" models from "bad" models in the reservoir descriptor parameter space. Using this "cheap" proxy as a surrogate of the full-simulator, we can examine a large number of reservoir models in the parameter space in a short period of time. Collectively, the identified good-matching reservoir models provide us with comprehensive information about the reservoir with a high degree of certainty.

In order to construct a simulator proxy, training data need to be collected. One common data sampling approach is design of experiment (DOE) [3]. In that work, we adopted one such design called uniform design[4] to select 1,000 reservoir model samples for computer simulation runs. The simulation data were then used by a GP system to train a symbolic regression as the simulator proxy to evaluate more reservoir models. Although the project was a success in that better history matching results were produced due to the information revealed from the large number of "good" reservoir models identified by the simulator proxy, the number of simulation data required to train a quality proxy is still too large for a typical size reservoir.

In this work, we investigate a more intelligent sampling strategy that can select a smaller number of more informative samples to train better quality proxies. The technique we adopted is the estimation-exploration algorithm developed by Bongard and Lipson [1]. In this algorithm, two co-evolving populations are maintained: one evolves candidate models (simulator proxy) and the other evolves test samples (reservoir samples). The fitness of a candidate model is its ability to explain behavior of the target system (reservoir simulator) observed in response to all tests carried out (simulation data) so far; the fitness of a candidate test is its ability to make the models disagree in their predictions. In other words, the fitness of individuals in one population is designed to against the fitness of individuals in the other population. This competitive coevolution can lead to an "arms race" between the two populations and improve the performance of both populations: more informative tests and better quality models are evolved. However, in order to generate disagreements among the models' predictions, these models need to have high diversity . We have integrated an additional technique *competitive fitness sharing* [15] [12] [16] to promote the model population diversity. We also

Tina Yu is with the Department of Computer Science, Memorial University of Newfoundland, Canada (phone: 709-737-6943; fax: 709-737-2009; email: tinayu@cs.mun.ca). Dave Wilkinson is with Chevron Energy Technology Company, U.S.A. (phone: 925-842-6200; fax: 925-842-2111; email: DavidAWilkinson@chevron.com).

devised different crossover and mutation operations to better trace the samples that can create high-disagreement among models. Another integrated technique is "test bank" [2] which was used to prevent population disengagement. With all the integrated techniques, the coevolutionary system was able to select more informative samples which in turn allow better simulator proxies to be trained.

The paper is organized as follows. Section II provides background and related work that have influenced the development of this research. In Section III, the co-evolutionary system tailored for this application is presented. Section IV reports a preliminary case study of this system on an oil field situated in offshore Africa. The results are analyzed in Section V. Finally, Section VI concludes the paper with outlines of our future work.

II. Background and Related Work

Proxies, or fitness approximation models, is an active research area in the evolutionary computing (EC) community [8]. Although surrogate is a more commonly used term in the EC community, we decided to use proxy in this article as it is the term used in the petroleum engineering community and in our previous work. A proxy model can be integrated in an EC system for various purposes. The focus of this article is on on-line sampling strategy, i.e. dynamic sample selection to train quality proxy models.

In [14], the author applied Kriging to build a proxy model for an objective function. While applying an EC system to optimize parameter values for the objective function, the fitness evaluation is carried out by the proxy for a number of generations. After that, some of the individuals in the populations are selected and evaluated by the objective function. The evaluation outputs are then used to update the Kriging proxy model. The evolutionary run then continued using the updated proxy model for fitness evaluation. This process is repeated every n generations until the evolutionary run is completed. The author reported that selecting the individual that has the best Kriging proxy fitness to carry out the objective function evaluation is the best strategy. However, when applied to a more complex optimization task (structure design), this sampling strategy could not produce quality proxy, hence did not perform well on the optimization task.

Various researchers have explored ways to co-evolve models and test samples[12][15], starting with the seminal work of Hillis [6] on sorting networks. All these works contained a similar co-evolution framework: there are two populations, one evolve models and the other evolves test samples. The fitness of an individual in one population is based on direct competition with some individual(s) from the other population. The estimation-exploration algorithm [1] is one of the algorithms that is based on this competitive co-evolution framework.

This algorithm has a two-phase cycle. In the estimation phase, model population evolves models to describe the target system based on a set of input-output data obtained from the system. In the exploration phase, the test population explores the target system parameter space to discover test samples that provide new information of the target system. Once such a test is evolved, it is applied to the target system and the outputs, paired with the test inputs, are added to the training set for the model population to evolve updated and improved models. These two phases alternate until the termination criterion is met. This algorithm flow is given in Figure 1.

```
1. Initialization
     a. Select the initial sample set, evaluate it on the target system
        and add them to the training set. Go to 2.
2. Estimation Phase
     a. Evolve candidate models using the current training set.
     b. Fitness of a model is its performance on the current training
        set.
     c. Send the best models to 3.
3. Exploration Phase
     a. Evolve candidate sample tests.
     b. Fitness of a sample test is the disagreement it caused among
        the models sent from 2.
     c. Select the sample test that causes the most disagreement and
        evaluate it on the target system. The data is added into the
        training set.
     d. Go to 2.
4. Termination
     a. Repeat step 2 and 3 until the population of models achieve
        satisfying performance.
```

Fig. 1. Estimation-Exploration algorithm flow [1].

The central concept of the estimation-exploration algorithm is to select test samples that provide the most new information about the target system and then use these samples to improve the generalization ability of the estimation models. But what kind of test samples can meet this objective? This algorithm adopted a well-known active learning technique in the machine learning community: select the sample that causes the maximal disagreement among a team of good quality models evolved so far.

For example, in query by committee [17] [5], the data points that were selected for application are those that cause the maximal disagreement among a set of candidate models. Similarly, the work in [13] first trained different classifiers based on different views of the data set (disjoint subsets of features that can be used for learning) and then used the disagreement among these classifiers to select data. Another work that also selected samples based on the disagreement of trained classifiers is reported in [10]. In that work, there were two classifiers: one was trained on labeled data and the other was trained on one data point with unknown label, in addition to the labeled data. The unlabeled data point is assigned with both positive and negative labels before being added into the training set. After the two classifiers were trained, their classification disagreements on the data set decided which data points were selected and which ones were not. This same criterion is used in the exploration phase of the algorithm for samples selection.

In order to generate disagreement among model predictions, there must be sufficient diversity within these models. One technique to promote population diversity in competitive coevolution is *competitive fitness sharing* [15][12][16], in which models receive higher fitness if they are able to give correct prediction on test samples that only a small number of models can predict correctly. In [15], the competitive sharing fitness of a model is defined as $\sum_{i=1}^{n} \frac{1}{N_i}$, where N_i is the total number of models in the population can give correct prediction on test sample i and n is the total number of test samples. Table I gives examples showing how this fitness is calculated.

For example , model M_1 receives fitness $\frac{1}{3}$ for predicting test case t_1 correctly since there are 3 models in the population which can give correct prediction on this test sample. Similarly, it receives fitness $\frac{1}{2}$ for predicting test case t_2 correctly and $\frac{1}{2}$ for predicting test case t_3 correctly. The total fitness for M_1 is therefore $\frac{4}{3}$. With this fitness mechanism, the population would more likely to evolve models that solve test cases that are not solved by other models, hence increase population diversity. We have incorporated this technique in our competitive co-evolutionary system to promote more diversified models being evolved in the model population.

TABLE I

EXAMPLES OF COMPETITIVE FITNESS SHARING [15]

	t_1	t_2	t_3	t_4	fitness
M_1	1	1	1	0	$\frac{1}{3}+\frac{1}{2}+\frac{1}{2}=\frac{4}{3}$
M_2	1	1	0	0	$\frac{1}{3}+\frac{1}{2}=\frac{5}{6}$
M_3	1	0	0	0	$\frac{1}{3}$
M_4	0	0	1	1	$\frac{1}{2}+\frac{1}{1}=\frac{3}{2}$
N_i	3	2	2	1	

In competitive co-evolutionary systems, disengagement may occur when one population is entirely superior to another population. Consequently, the individual fitness in the dominated population become identical and selection pressure is lost. In the case of co-evolving models and test samples, this may occur when a test sample is too difficult for the model population to make correct prediction, and all models receive the same low fitness. One remedy to this problem is "test bank" [2], in which difficult test cases are temporarily withdrawn from the training set and re-introduced in the following or later algorithm cycles. We have integrated this technique in our competitive co-evolutionary system to manage populations disengagement.

III. System Design and Implementations

For our application, the two populations are simulator proxies and sample points (reservoir model descriptive parameter values) that the reservoir simulator will be run on. The two populations are asymmetric in that the proxies are represented as S-expressions and evolved using a GP while the test samples are represented as linear vectors of real numbers and evolved using a genetic algorithm (GA) [7].

Initially, we ran reservoir simulator on a small number of samples selected by uniform design [4] as we have observed that uniform-design samples are suitable for evolutionary systems to construct proxies [18]. These simulation data becomes the initial training data for the proxy population to train proxy models at the initial estimation phase.

During each estimation phase, the proxy population is seeded with randomly generated s-expressions so that GP is free to explore more diversified proxy models. The fitness function for a proxy model m is the product of its classification accuracy and the competitive fitness sharing value (explained in the previous section) on the current training data set:

$$f_m = \frac{\sum_{i=1}^{n} c_i}{n} \times \sum_{i=1}^{n} \frac{1}{N_i} \qquad (1)$$

where c_i is 1 if model m gives correct prediction on training data i and 0 otherwise. N_i is the number of models in the population that gives correct prediction on training data i and n is the total number of training data. The higher the value f_m is, the better the proxy model is.

The GP carries out selection, crossover and mutation in a normal fashion. After a pre-defined number of generations is reached, the best k number of (hopefully diversified) proxies is sent to the sample population to start the exploration phase.

At each exploration phase, the sample population is seeded with linear vectors of reservoir descriptive parameter values which are chosen randomly within their predefined ranges. The fitness of a sample f_s is the amount of disagreement it induces from the predictions of the k models.

$$f_s = \frac{\frac{2}{k} - |\frac{2}{k} - \sum_{j=1}^{k} a_j|}{k} \times 2 \qquad (2)$$

where a_j is the prediction that the sample received from model j. Since the model is a binary classifier, there are only two possible values: good-match (denoted by 1) and bad-match (denoted by 0) to the reservoir production data. The disagreement is normalized to be between 0 and 1; the higher the value f_s is, the larger the disagreement and the fitter the sample s is. A sample that receives one prediction (good-match) from 50% of the k models and another prediction (bad-match) from the other 50% of the k models receives the highest fitness of 1.0.

The GA explores reservoir model parameters space using three genetic operators: *average crossover*, *attractor* and *repeller mutations*. Average crossover takes two samples (each of which is a linear vector of 10 values) and produce one offspring whose vector values are averages of the two parent samples. This operation is performed on selected samples (we used tournament selection of size 2) which have high disagreement ($f_s >= t$, where t is a users defined threshold). Attractor and repeller mutations work by adding/subtracting a δ to each reservoir parameter value of a sample vector. They are applied on selected samples which have low disagreement ($f_s < t$). The value of δ_i for reservoir parameter x_i is defined as follow:

$$\delta_i = min(x_u - x_i, x_i - x_l) \times \theta \qquad (3)$$

where x_u is the upper bound of x_i and x_l is the lower bound of x_i. When the highly agreed prediction on the sample is "good-match", the sample is considered as an attractor as there are a much fewer number of good-match samples than bad-match ones in the reservoir parameter search space. We therefore want to sample more of this kind, even though we don't know whether this is really a good match to the reservoir model or not until we run it on the computer simulator. In this case, a small θ (0.1) is used in Equation 3. If the highly agreed prediction on the sample is "bad-match", the sample is considered as a repeller and a larger θ (0.9) is used in the equation. The mutated value $x_i' = x_i \pm \delta_i$.

When the GA has completed evolving a pre-defined number of generations, the best individual in the sample population is selected to run reservoir simulator. The simulation data is a new data record that serves two purposes. First, it is used to blind test if the quality of the proxy model is satisfactory. If the best proxy model is able to predict the new sample correctly, the evolutionary run terminates. If this is not the case, the data will serve the second purpose as a new training data to improve proxy quality. After the new data is added to the training set, the system enters a new cycle of estimation phase.

The proxy model training in the estimation phase is incremental: new data is added one at a time to gradually improve the previously best trained model. This is a safe approach as the training difficulty can be monitored. If too many data points are added to the training set at the same time, it would be harder to identify which data is successfully learned and which one is not. Another shortcoming is that the evolution might be directed into more than one direction if the multiple new data are very different in their mapping between the inputs (reservoir parameter values) and outputs (predictions).

To start a new estimation phase, the best proxy model evolved from the previous cycle is copied over to the current proxy population. Meanwhile, the rest of the proxy population is seeded with randomly generated s-expressions. As GP evolution progresses, the population improvement can be monitored. If the training of the new single sample is successful, the best proxy at the end of the cycle would achieve a certain fitness value (d). If this is not the case, it is an indication that the new sample is too hard for the current proxy population to digest. This new data is withdrawn from the training set and deposited in a test bank for later recycle. At the same time, old test samples stored in the test bank are evaluated by the current best proxy model one at a time. If a test sample can be predicted correctly by the model, it is re-introduced to the training set. Otherwise, the data remains in the test bank.

When there is an update on the training set (due to the removal of difficult data or reintroduction of test-bank data), the estimation phase will be initiated. Since this cycle is triggered by an unsuccessful run, previous training results won't serve as a good base for incremental learning. The model population is therefore seeded entirely with randomly generated s-expressions. GP starts with a pool of new materials to evolve proxy models using the updated training set.

This alternation of estimation and exploration phases continues until either the best proxy has meet the specified accuracy and passed the blind test (defined in Table III) or the pre-determined number of cycles is reached. Fig. 2 gives the proxy-sample competitive co-evolutionary system workflow.

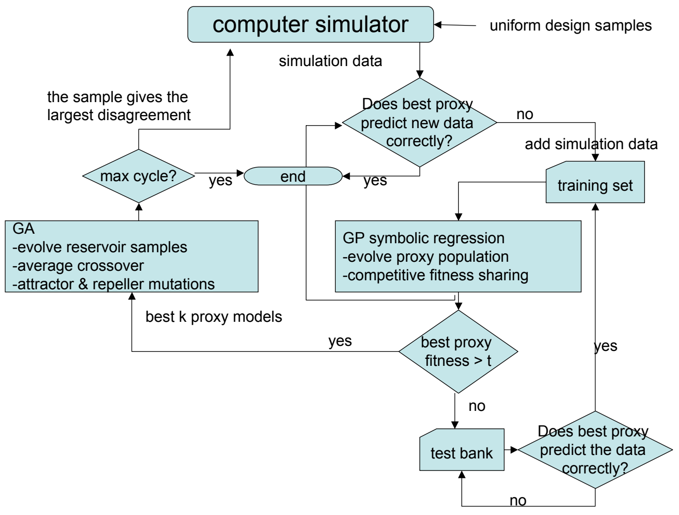

Fig. 2. The proxy-sample competitive co-evolutionary system workflow.

IV. A Case Study

We conducted a case study using the developed system on a complex clastic channel reservoir situated offshore Africa. The primary reservoirs are sandstones deposited in the channelized system and have been in production since 1998.

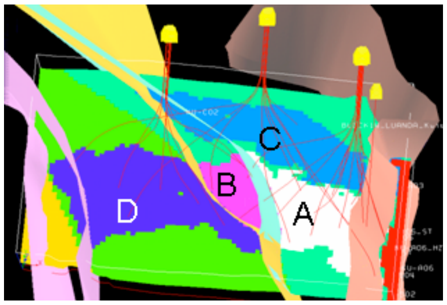

Fig. 3. The studied oil filed in offshore Africa.

In our previous study, we have conducted computer simulation on 4 blocks (A, B, C, D) of the field (see Fig. 3). The parameters and their value ranges used to conduct the simulation are listed in Table II. Note that the 6 multiplier parameters are in log-10 scale. They are applied to the base values in each of the reservoir model grid during computer simulation.

TABLE II

RESERVOIR PARAMETERS AND THEIR VALUE RANGES.

Parameters	Min	Max	Parameters	Min	Max
KRW_A	0.3	0.7	ZPERM_A Multiplier	-2	0
KRW_B	0.1	0.5	ZPERM_B Multiplier	-2	0
KRW_C	0.1	0.5	ZPERM_C Multiplier	-2	0
KRW_D	0.1	0.5	ZPERM_D Multiplier	-2	0
XPERM Multiplier	1	2	FAULT_A_B Multiplier	-4	0

The simulation was conduced on 1,000 samples selected by uniform design[1]. Among them, 894 runs were successful while the other 106 did not make to the end of the simulation run due to system failure. The 894 simulation data will be used in this case study to measure the generalization capability of the proxy models evolved by the developed competitive co-evolutionary system.

For this preliminary study, the co-evolutionary system is not connected to the computer simulator. Instead, we used the simulator proxy model generated from our previous study [18] as a surrogate. In this way, this proof-of-concept study can be completed faster, avoiding the time-consuming computer simulation.

Each sample in the sample population is a linear vector containing 10 reservoir parameter values. Initially, 20 samples were selected based on uniform design[2]. These 20 samples were evaluated by the surrogate model and the predictions, along with the sampled parameter values, become the initial training set for the proxy population. Each model in the proxy population is a classifier that takes inputs of the 10 reservoir parameter values and produces a prediction of "good-match" or "bad-match" to the historical production data. A "good-match" indicates the parameter values are close to reality while "bad-match" indicates the opposite.

TABLE III

THE GP SYSTEM PARAMETERS SETUP.

Parameters	Values
Functions	+, -, $\times$, /, abs
Terminals	10 reservoir parameters, constants
Initial tree depth	6
Population size	200
Genetic operator	point mutation, homologous crossover
Crossover rate P_c	50%
Mutation rate P_m	50%
Selection method	tournament of size 2
Population model	steady-state with 75% replacement
No. of generations per cycle	60
No. of cycles	20

Table III gives the GP parameters setup to evolve the proxy models. Each experimental run terminates when a proxy with 100% accuracy on both the training and the blind test data is found or the maximum number of cycles (20) is completed.

TABLE IV

THE GA SYSTEM PARAMETERS SETUP.

Parameters	Values
Population size	100
k (number of judging models)	20
Genetic operator	attractor & repeller mutations average crossover
Crossover rate P_c	50%
Mutation rate P_m	50%
Disagreement threshold t	60%
Test difficulty threshold d	80%
Selection method	tournament of size 2
Population model	steady-state with 90% replacement
No. of generations per cycle	30
No. of cycles	20

Table IV gives the GA parameters setup to evolve test samples. A new data is considered "difficult" if the best proxy trained using the data set including this new data has prediction accuracy lower than 80%. When this happens, this data is extracted from the training set and deposited in the test bank for later recycle.

The disagreement threshold t for a sample to carry out average crossover is 60%. If a selected sample has disagreement value lower than that, attractor/repeller mutation is performed. Note that the crossover and mutation rates are applied differently from that in the standard GA. Here, only one of the 3 genetic operators is applied on the selected sample depending on its disagreement value. If the operator is mutation, each parameter value has P_m probability to be mutated. If the operator is average crossover, each parameter value has P_c probability to be averaged with the value in the other parent to generate offspring. Since each sample can only be applied by one of the operators (unlike the standard GA where an individual is applied by both crossover and mutation operators), 50% seems to be a reasonable rate for P_c and P_m. We will conduct experiments using different P_c and P_m values in our future work.

To evaluate the effectiveness of the system with each of the add-on techniques, we conducted 6 sets of experiments, each consists of 50 independent runs. The setups are given in Table V and the results are analyzed in the following section.

TABLE V

SIX EXPERIMENTAL SETUPS.

Setup	Proxies Population	Samples Population
(*a*)	GP	Random Sampling
(*b*)	GP with test bank	Random Sampling
(*c*)	GP	GA with point crossover & bit mutation
(*d*)	GP with test bank	GA with point crossover & bit mutation
(*e*)	GP	GA with 3 developed genetic operators
(f)	GP with test bank	GA with 3 developed genetic operators

V. RESULTS AND ANALYSIS

Fig. 4 shows the final best proxy models prediction accuracies on training data, averaged over 50 runs, under 6

[1]The number of factors for each of the 10 parameters to conduct the design is: 8, 8, 8, 8, 16, 32, 32, 32, 32, 64.

[2]The number of factors for each of the 10 parameters to conduct the design is: 5, 5, 5, 5, 5, 5, 5, 5, 5, 5. See the design at http://www.math.hkbu.edu.hk/UniformDesign/

different experimental setups. Their prediction accuracies on 894 simulation data, averaged over 50 runs, are given in Fig. 5. As shown, the training data accuracies are close to each other between 92% to 96% while the simulation data accuracies are a little bit farther apart between 72% and 80%. The prediction accuracies gap between the two sets of data is more than 10%, indicating that the evolved proxy models do not generalized very well. This result, although disappointing, should not be interpreted as a failure of the method. On the contrary, if we consider the number of samples that each proxy was trained on (maximum of 40, where 20 was from the uniform design and the average numbers of samples selected by the GA to train the proxies were: 12.92, 16.94, 14.72, 17.6, 16.78 and 16.9 for the 6 different setups), this result is actually quite impressive.

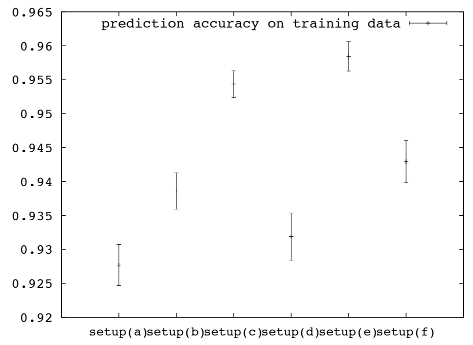

Fig. 4. The final proxy models prediction accuracies on training data.

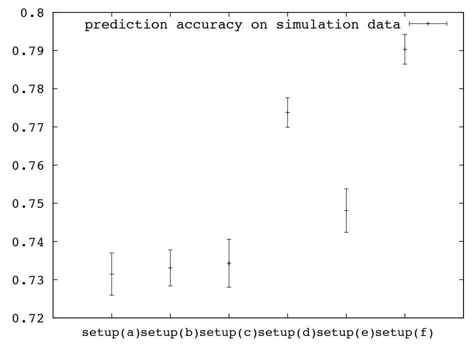

Fig. 5. The final proxy models prediction accuracies on simulation data.

For the parameter values of this reservoir, random sampling (setup a) did an average job (93% accuracy on training data), suggesting that the fitness landscape has many local optima. Any subset of the samples could provide fitness gradients for GP to evolve a sub-optimal proxy model. These sub-optimal models, however, do not generalize well: the prediction accuracy is 20% lower on the simulation data.

Adding a test bank to the GP system to remove and reintroduce randomly generated samples (setup b), the evolved proxies give better accuracy on training data (94%). However, their prediction accuracies on simulation data do not improve, i.e. their generalization ability is still the same.

Using the standard GA one-point crossover and point-mutation to evolve samples for GP to co-evolve proxy models (setup c), the proxies prediction accuracy on training data is increased to 95%. This indicates that the co-evolutionary sampling method is more intelligent than random sampling in that the selected samples provide more information about the reservoir model parameter space. However, these selected samples seem to bias toward a certain area of the parameter space hence the trained proxies do not generalize better than the proxy trained using random samples. This situation is changed when the GP system has a test-bank to remove and reintroduce the selected samples (setup d). Although the evolved proxies have a lower accuracy on training data (93%), their prediction accuracy on simulation data is improved to 76%. In other words, the test-bank has helped GP to evolve more robust proxy models.

When the GA standard operators were replaced by the three designed genetic operators (setup e), the evolved proxies give higher prediction accuracy on both training (96%) and simulation data (75%). This indicates that the three designed GA operators are more effective than the standard operators in selecting samples that maximize the disagreement of model predictions to help train better proxies.

Similar to the results produced by the GA standard operators, when the GP system is combined with a test-bank (setup f), the three designed GA operators selected samples that trained proxies models with lower accuracy on training data (94%) but with higher accuracy on simulation data (79%). Meanwhile, with test-bank, the competitive co-evolution framework (setup d and f) produced proxies that have the smallest prediction accuracy gap (15%) between the two data sets. This indicates that withdrawing challenging samples evolved by GA temporary has changed the dynamics of the GP learning.

The experimental results show that the three designed GA operators did excellent job in selecting difficult samples which need to be included in the training set. However, there is a gap between the difficulty level of the new sample and other samples in the current training set. If the difficult sample is kept in the training set, GP might eventually learn the sample but the trained models also bias the selection of later samples: the trained proxies will cause GA to evolve even more difficult samples to create prediction disagreement. However, if the difficult sample is temporarily removed from the training set and replaced with an intermediate level difficult sample, GP is more likely to learn gradually and to produce proxies with better generalization capability. This is demonstrated in the experimental results of setup d and f.

For all test samples selected by the 6 different experimental setups, we calculated the sample distributions of the 10 parameters. Each parameter value is partitioned into 5 ranges and the number of samples that selects value at each range is calculated. We do not include the 20 seeding samples from uniform design in this calculation. In fact, including these samples will not change the distribution density since the design gives balanced distribution over the 5 parameter value ranges: 4 data points for each range. Fig. 6, 7, 8, 9, 10 and 11 give the calculated distributions. A similar calculation was done on the 894 simulation data and the results are given in Fig. 12.

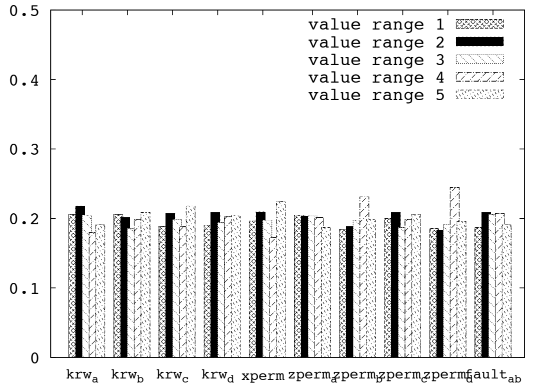

Fig. 6. Parameter values distribution of samples selected by setup (a)

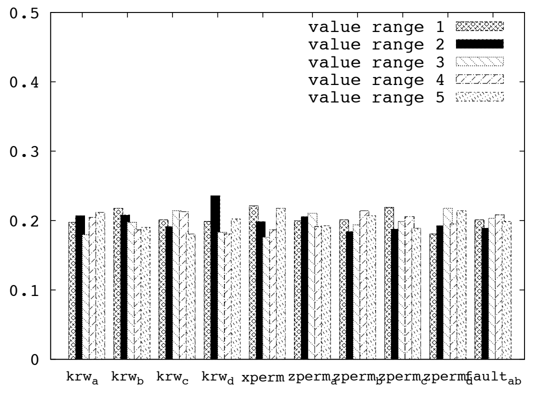

Fig. 7. Parameter values distribution of samples selected by setup (b).

Random sampling, as shown in Fig. 6 and 7, sampled parameter values over all ranges. By contrast, the GA standard operators sampled more data values on the boundary (high and low), suggesting that they are difficult points and caused high-disagreement among proxy models (see Fig. 8). This tendency is more evident when the three designed GA operators were used to select samples (see Fig. 10). These three greedy operators aggressively sought high-disagreement samples. As a result, more of these difficult samples (parameter values are around the boundaries) were selected. This is another indication that the three designed operators are more effective than the standard one-point crossover and point mutation in selecting difficult samples.

This tendency of over-selecting samples with boundary-values can be balanced by the ”test-bank” technique. As shown in Fig. 9 and 11, there is slight tendency of sampling boundary-value cases. Yet, other values ranges were also well sampled. The sample distribution is similar to the

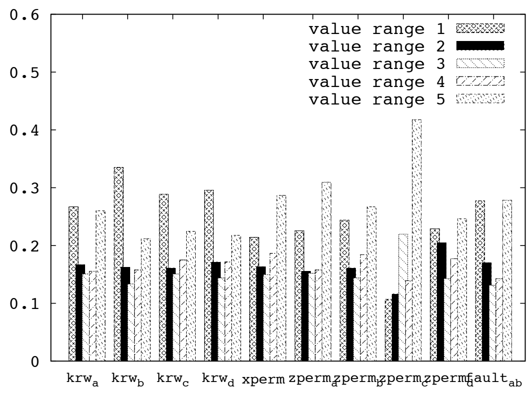

Fig. 8. Parameter values distribution of samples selected by setup (c).

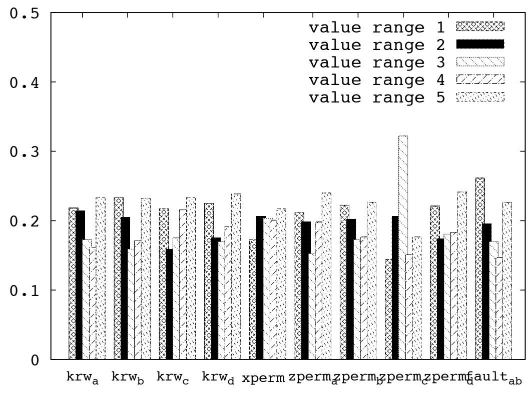

Fig. 9. Parameter values distribution of samples selected by setup (d).

distribution of the 894 simulation data (see Fig. 12[3]). Using these selected samples, GP was able to evolve proxy models which generalized to the simulation data better than those based on samples selected by the same algorithm without the test bank mechanism.

Why did the test-bank cause the GA operators (both standard and designed) changing their sampling distribution? Every training data point directs GP evolutionary learning toward models which have the ability to handle that particular data. When a data point is difficult and the current GP population could not digest it (all models do equally bad), the evolved proxies will guide GA to select more difficult samples. The over-sampling of boundary values might be misled by the sub-optimal proxy models fine tuned by these challenging samples. In other words, the two populations (proxy and sample) “conspired“ with each other to evolve proxy models that only work well on the difficult samples (those with boundary values) but not other values. When these challenging samples are withdrawn from the training set and re-introduced later, i.e. changing the order of GP learning, the over-sampling phenomenon no longer existed.

[3]Uniform design should sample all values ranges uniformly. However, due to 106 simulation runs did not complete successfully, the rest 894 data only sample part of what was sampled by uniform design.

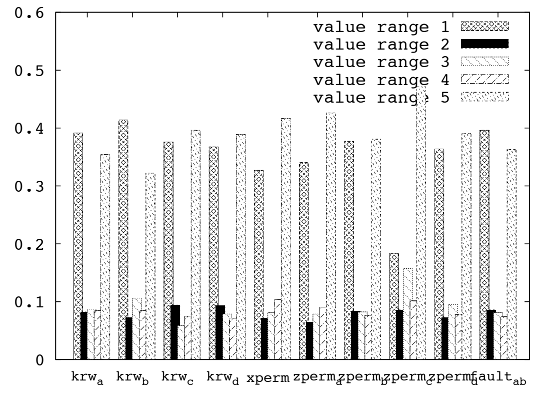

Fig. 10. Parameter values distribution of samples selected by setup (e).

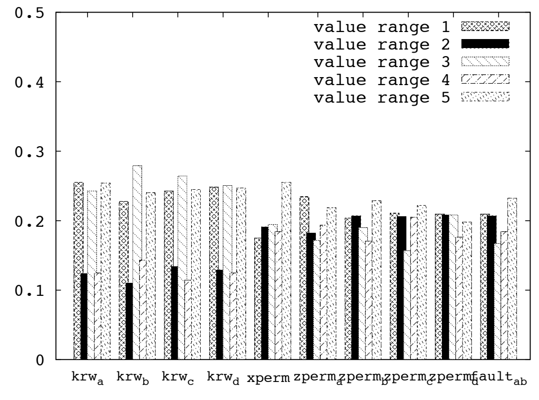

Fig. 12. Parameter values distribution of the 894 simulation data.

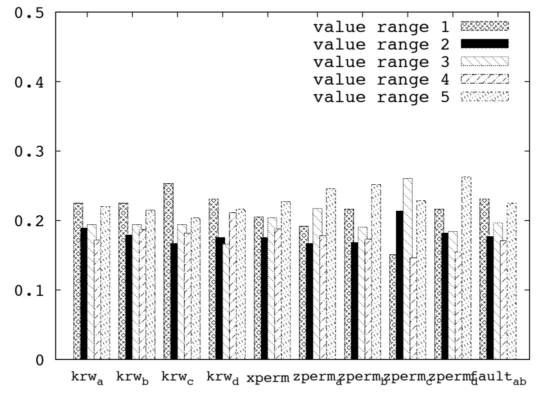

Fig. 11. Parameter values distribution of samples selected by setup (f).

VI. Conclusion

We have reported a preliminary study on co-evolution of simulator proxies and sampling strategies for petroleum reservoir modeling. The objective is to select a small number of informative samples to train a quality simulator proxy to improve reservoir history matching process and reduce uncertainty. This is a challenging task as there is a cycling effect of the two desired outputs: the selected samples impact the training of the proxy models which in turn effect the selection of new samples. As shown in the study, although the competitive co-evolutionary system produced high prediction-accuracy proxies on training data, they don't generalize very well on simulation data. Nevertheless, this result is still very impressive given the very small number of samples selected by the co-evolutionary system to construct the simulator proxy models.

The test-bank technique has shown potential in mitigating this situation. By re-arranging the learning order of the selected samples, the trained proxy models generalize better on simulation data. There are also different proxies quality measures [9] that might be more suitable in this type of reservoir parameter space. We will continue investigating the coevolutionary dynamics of the proxy and sample populations and other quality measurements to improve the system performance.

References

[1] J. C. Bongard and H. Lipson, "Nonlinear system identification using coevolution of models and tests" *IEEE Transactions on Evolutionary Computation*, vol. 9(4), August pp. 361–384, IEEE, 2005.

[2] J. C. Bongard and H. Lipson, "Managed challenge, alleviates disengagement in coevolutionary system identification" *Proceeding of Genetic and Evolutionary Computation Conference*, pp. 531-538, ACM 2005.

[3] G. E. P. Box, J. S. Hunter and W. G. Hunter, *Statistics for Experimenters: Design, Innovation, and Discovery, second edition*, Wiley, 2005,

[4] K-T Fang, *Uniform Design: Application of Number Theory in Test Design*, ACTC Mathematicas Applicatae Sinica. 1980.

[5] Y. Freund, H. S. Seung, E. Shamir and N. Tishby, "Selective sampling using the query by committee algorithm" *Machine Learning Journal*, vol. 28, pp. 133–168, Kluwer, 1997.

[6] W. D. Hills, "Coevolving parasites improve simulated evolution as an optimization procedure" *Physica D*, vol. 42, pp. 228–234, 1990.

[7] J. Holland, *Adaptation in Natural and Artificial Systems* MIT Press, 1992,

[8] Y. Jin, "A Comprehensive survey of fitness approximation in evolutionary computation" *Soft Computing Journal*, vol. 9(1), pp. 3–12, 2005.

[9] Y. Jin, M. H*ü*sken and B. Sendhoff "Quality measures for approximate models in evolutionary computation" *Proceedings of the GECCO Workshop on "Learning, Adaptation and Approximation in Evolutionary Computation"*, pp.170-174, Chicago, 2003.

[10] P. Juszczak and R. W. Duin "Selective sampling based on the variation in label assignments." *Proceedings of the 17th International Conference on Pattern Recognition*, pp. 375–378, 2004.

[11] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* MIT Press, 1992.

[12] H. Juill*é* and J. B. Pollack "Dynamics of co-evolutionary learning," *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pp. 526-534, MIT Press, 1996.

[13] I. Muslea, S. Minton and C. A. Knoblock "Selective sampling with redundant views." *Proceedings of the 7th National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pp. 621–626, 2000.

[14] A. Ratle, "Optimal sampling strategies for learning a fitness model," *Proceedings of 1999 Congress on Evolutionary Computation*, vol. 3, pp. 2078–2085, IEEE, 1999.

[15] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution" *Evolutionary Computation Journal*, vol. 5(1), pp. 1–29, 1997.

[16] J. Werfel, M. Mitchell and J. P. Crutchfield, "Resource sharing and coevolution in evolving cellular automata" *IEEE Transactions on Evolutionary Computation*, vol. 4(4), November, pp. 388 – 393, 2000.

[17] H. S. Seung, M. Opper and H. Sompolinsky, "Query by committee," *Proceedings of the fifth annual ACM Workshop on Computational Learning Theory*, pp. 287–294, ACM, 1992.

[18] T. Yu, D. Wilkinson and A. Castellini, "Applying genetic programming to reservoir history matching problem," *Genetic Programming, Theory and Practice IV* (Chapter 13), Edited by R. Riolo, T. Soule and B. Worzel, pp. 187-201, Springer, 2006.