Optimization of a Model to Predict Salinity Intrusion in San Francisco Bay Estuary Using a Genetic Algorithm

T.Rajkumar[a] **and David E. Thompson**[b]

[a] *SAIC, NASA Ames Research Center, M.S. 269-2 Moffett Field, California 94035*
(rajkumar@mail.arc.nasa.gov)

[b] *NASA Ames Research Center, M.S. 269-2 Moffett Field, California 94035*
(dethompson@mail.arc.nasa.gov)

Abstract: Salinity intrusion is a significant issue in the San Francisco Bay Delta because this estuary contributes substantially to fresh water supply for much of northern California. Traditionally, salinity along the estuary has been measured using monitoring instruments installed at various locations within the Sacramento-San Joaquin River Delta system, and these measurements are used to predict future conditions and evaluate the water management alternatives. The California Department of Water Resources Delta Simulation Model (DWRDSM) is used for this type of analysis. DWRDSM is an unsteady one-dimensional hydrodynamic and salt transport model. It can provide estimates of salinity at almost any location within the Bay-Delta system. However, like many numerical models, when simulating the dynamics of a large complex system, the processing time can be extremely long, which makes it difficult to evaluate a large number of management scenarios in a short time. Statistically-based models of the flow and salinity have been tried and found lacking; classical time series analysis is often linear or requires transformation to a stationary time series that renders the resulting model unsuitable, since the salinity intrusion problem is a complex non-linear problem. Consequently, there remains a need for a reliable and fast method to predict salinity intrusion in the Sacramento-San Joaquin River Delta system. In this paper, such a method is advocated: Genetic Algorithms (GA) are used to optimize a previously built Artificial Neural Network (ANN) that reliably predicts periodic salinity intrusion in the San Francisco Bay Estuary. Results indicate that the GA provided an efficient method of optimizing the ANN model to predict salinity. The reliability of the ANN using the GA includes prediction of salinity to an accuracy of ±10% at hourly intervals for any time period in the year.

Keywords: Salinity intrusion; San Francisco bay delta modeling; Neural network; Genetic algorithm optimization

1. INTRODUCTION

Sources of freshwater should be managed in a sustainable manner that will improve socio-economic development. The Sacramento-San Joaquin delta is a unique and valuable resource and California's water supply network. Water projects divert water from the delta to meet the needs of approximately two-thirds of the state's population. In addition to these exports, there are about 2000 local irrigation diversions located within the delta. Most of the remaining water constitutes the final delta outflow into San Francisco Bay. Saline water from the bay is normally kept from intruding into the delta by these freshwater outflows. Traditionally salinity is calculated using monitoring instruments installed at the boundary of the delta. By using this data, future conditions can be predicted to evaluate the remediation alternatives. The California Department of Water Resources Delta Simulation Model (DWRDSM) [Chung 1999] is used for this type of analysis. DWRDSM is an unsteady one-dimensional hydrodynamic salt transport model. However, like many numerical models, when simulating the dynamics of a large complex system, the processing time can be extremely long, which makes it difficult to evaluate a large number of management scenarios in a short time. Statistically-based models of the flow and salinity have been tried and found lacking due to inbuilt linearity [Winkler 1985]; classical time series analysis is often linear or requires transformation to a stationary time series which renders the resulting model unsuitable, since the salinity

intrusion problem is a complex non-linear problem. Consequently, there remains a need for a reliable and fast method to predict salinity intrusion in the Sacramento-San Joaquin River Delta system. In this paper, such a method is advocated: Genetic Algorithms (GA) [Holland 1975] are used to optimize a previously built Artificial Neural Network (ANN) that reliably predicts periodic salinity intrusion in the San Francisco Bay Estuary (Figure 1) [Chung 1999; Rajkumar and Johnson 2001].

Our specific objective is to predict the salinity and stage at the Carquinez Strait (RSAC054) for hourly intervals. We selected the Carquinez Strait because it lies at the western edge of Suisun bay and is generally near the location of biologically productive salt-fresh water mixing zone. Also, monitoring data are available for Carquinez Strait, which allows model development and testing. Over 20 different input parameters were evaluated to characterize the dynamics of flow and salinity in the system using the ANN [Masters 1993]. Different neural network architectures and learning algorithms were tried to predict the best salinity intrusion model at any given point. Out of all training algorithms tested, the back propagation method using the Levenberg-Marquardt algorithm was the best predictor of salinity intrusion [Rajkumar and Johnson 2001]. However, it took many hours to train the neural network. To increase the efficiency of the training, GA's were used to autonomously select both the parameters for training and the input data necessary to capture the dynamics of the system [as in Miller et. al 1989]. GA's offer a faster and more intelligent methodology for searching for optimal combinations of parameters for ANN's when compared to exhaustive searches of all combinations of parameters [Goldberg 1988]. Our GA searched through combinations of ANN input parameters looking for the set of inputs that optimized the training criteria (minimum error of tolerance in neural network). The GA evaluated the usefulness of each input parameter in conjunction with all other input parameters, and it did not become trapped in a "local minimum" as could occur with other optimization techniques. The following sections discuss the need for genetic optimization and our experiments.

2. NEED FOR OPTIMIZATION OF NEURAL NETWORK

The problem of neural network design comes down to searching for an architecture that performs best on some specified task according to explicit performance criteria. This process in turn can be viewed as searching the surface defined by levels of trained network performance above the space of possible neural network architectures. Since the number of possible hidden neurons and connections is unbounded, the surface is infinitely large. Since changes in the number of hidden neurons or connections must be discrete, and can have a discontinuous effect on the network's performance, the surface is undifferentiable. The mapping from network design to network performance after learning is indirect, strongly epistatic, and dependent on initial conditions (e.g. random weights), so the surface is complex and noisy [Miller et. al 1989]. Structurally similar networks can show very different information processing capabilities, so the surface is deceptive; conversely, structurally dissimilar networks can show very similar capabilities, so the surface is multimodal. Hence we seek an automated method for searching the vast, undifferentiable, epistatic, complex, noisy, deceptive, multimodal surface.

If a neural network has too many hidden neurons, it will almost exactly learn, or memorize, the training examples, but it will not perform as well recognizing new data after the training process is complete. If a neural network has too few hidden neurons, it will have insufficient memory capacity to learn all the different types of training examples. A genetic algorithm is used to optimize the minimum number of training data sets required to train the neural network and the minimum number of hidden neurons in a three layer neural network architecture. The objective of the genetic algorithm to eliminate training cases that make it difficult for a neural network to differentiate all output classifications and to avoid discarding data [Jurgen 1995; Hancock 1992; Hochman et. al 1996; Kroning 1994]. The fitness function used for the genetic algorithm is chosen such that it will satisfy the conflicting requirements of training-data size reduction. The fitness function for our genetic algorithm performs the following calculations for each chromosome in the population:

- Count the number of inputs ignored.
- Train the neural network for 500 learning cycles. Sum the training error for the last 40 cycles. Beyond this training, the convergence of the neural network is not very significant.
- Calculate the fitness value for a chromosome based on cumulative learning error, the number of input neurons that are ignored, and the number of hidden layer neurons.

The fitness function should minimize the training error, the number of hidden neurons and the

number of inputs that are ignored (i.e., avoids discarding training cases except when absolutely necessary).

In order to optimise the structure of neural network using genetic algorithm, chromosome is encoded using information from input as well hidden neurons. We adopted at least 15 hidden neurons, and this value can be encoded in four bits. At least one bit in the chromosome represents information from the input neuron. When a fit chromosome is found, that chromosome is used to specify the number of hidden layer neurons.

3. GENETIC ALGORITHM

The basic genetic algorithm comprises four important steps [see Goldberg 1988]:

- The first step is the creation of the initial population of chromosomes either randomly or by perturbing an input chromosome. How the initialization is done is not critical as long as the initial population spans a wide range of variable settings (i.e., has a diverse population). Thus, if explicit knowledge about the system being optimized is available that information can be included in the initial population.
- In the second step, evaluation and the fitness function is computed. The goal of the fitness function is to numerically encode the performance of the chromosome. For this problem of optimization, the choice of fitness function is the most critical step.
- The third step is the exploitation or natural selection step. In this step, the chromosomes with the largest fitness scores are placed one or more times into a mating subset in a semi-random fashion. Chromosomes with low fitness scores are removed from the population. There are several methods for performing exploitation. In the binary tournament mating selection method, each chromosome in the population competes for a position in the mating subset. Two chromosomes are drawn at random from the population, the chromosome with the highest fitness score is placed in the mating subset. Both chromosomes are returned to the population and another tournament begins. This procedure continues until the mating subset is full. A characteristic of this scheme is that the worst chromosome in the population will never be selected for inclusion in the mating subset.
- The fourth step, exploration, consists of recombination and mutation operators. Two chromosomes (parents) from the mating subset are randomly selected to be mated. The probability that these chromosomes are recombined (mated) is a user-controlled option and is usually set to a high value (e.g., 0.95). If the parents are allowed to mate, a recombination operator is employed to exchange genes between the two parents to produce two children. If they are not allowed to mate, the parents are placed into the next generation unchanged. The two most common recombination operators are the one-point and two-point crossover methods. In the one-point method, a crossover point is selected along the chromosome and the genes up to that point are swapped between the two parents. In the two-point method, two crossover points are selected and the genes between the two points are swapped. The children then replace the parents in the next generation. A third recombination operator, which has recently become quite popular, is the uniform crossover method. In this method, recombination is applied to the individual genes in the chromosome. If crossover is performed, the genes between the parents are swapped and if no crossover is performed the genes are left intact. This crossover method has a higher probability of producing children that are very different than their parents, so the probability of recombination is usually set to a low value (i.e. 0.1). The probability that a mutation will occur is another user-controlled option and is usually set to a low value (e.g., 0.01) so that good chromosomes are not destroyed. A mutation simply changes the value for a particular gene.

After the exploration step, the population is full of newly created chromosomes (children) and steps two through four are repeated. This process continues for a fixed number of generations. For this application, the most widely used binary coded GA is used for encoding genes. In binary coding each chromosome is comprised of zeroes and ones where each bit represents a gene. To formulate the chromosome for optimization, the bit string is concatenated with the bit strings from the other variables to form one long binary string. We adopted a binary coding mechanism for creating the chromosomes.

4. EXPERIMENTS

The record of salinity fluctuations at gauging stations is a record of cyclic salinity intrusion within an estuary, and as such, the dominant signal is from tidal forcing (see Figure 2.). The observed data and hence the selected training data set is a time series. The time series is a well-defined pattern for this problem; this means that a few data points that are not well fit to the time series should not be used to create the neural network training set for learning because these data will make the learning much harder. The data available for years prior to 1996 were collected by multiple agencies, each using their own time intervals, and hence are of poor quality for testing long-term trends in salinity in the Bay Delta region. Since 1997, data has been collected by the State using common time intervals at all sites. In the previous neural network paper [Rajkumar and Johnson 2001], we considered two data sets (i.e., April 97 and August 98 for prediction) (http://www.iep.ca.gov). The training data pairs (15 inputs and 1 output) for this neural network are from 1 January 1996 to 30 March 1997 (7669 pairs). The April 1997 prediction data consists of 15 input variables and 1 output variable with 260 training pairs (from 2 - 14 April 1997). The input variables have been selected throughout the Bay Delta area to characterise the entire fluctuation of salinity intrusion [Rajkumar and Johnson 2001]. August 1998 had similar types of inputs and outputs, but the training pairs were increased to 17,154 (1 Jan 1996 to 31 Jul 1998).

The main objectives of optimizing the neural network [Bigus and Bigus 2001; Watson 1997] are to (i) minimize the number of training data sets, and (ii) minimize the number of hidden neurons. In both the previous paper and this one, the April 1997 data set is used for reduction in training data size and reduction in hidden number of neurons. Both 1997 and 1998 were classified as (abnormally) wet years by the California Department of Water Resources; the 1997 wet year was worse than 1998 wet year. We used the April1997 wet year data pairs for training data size reduction because it shows more variability and distinct features that enhance the neural network training. (i) To accomplish minimizing the number of training data sets, we presented the January 96 – March 97 training pairs (7669) for data size reduction. During the first pass of training, 5 data sets were eliminated. In the next round of training, 3 data pairs were eliminated. Twenty training cycles were performed on the same data, and 72 data pairs were eliminated from the original data, reducing it to 7597 pairs. That is, approximately 1% of total data were eliminated. These data were eliminated because they were considered outliers to the time series record, and including them merely delayed convergence for neural network training. Beyond 20 training cycles, very few data pairs were eliminated, and elimination required more time, so 20 cycles were considered sufficient for appropriate data size reduction. (ii) To accomplish optimizing the number of hidden neurons in the neural network, we used this same 7597 data pairs that had been used for training. Based on these data pairs, the number of hidden neurons were reduced from 30 (chromosome 5 bit encoding) to 8 hidden neurons. This resulted in a fast prediction of salinity at Carquinez Strait. The final architecture of the neural network is thus 15-8-1 [input-hidden-output].

Figure 2 presents a comparison between real-time salinity data monitored at Carquinez Strait, plotted against the neural network prediction and the GA-optimised neural network prediction for this data. The figure shows substantial improvement of fit when the network is optimized by the genetic algorithm. Note that in the first week of April, the GA optimised neural network has performed very well compared to the second week of April. In the first week (early spring) of April, there are not many spikes in the pattern, whereas the second week, the pattern has alternate spikes, which is varying between 5000 to 25000 Micromhos/cm. These spikes can be expained as follows. In the Bay, salinity intrusion will reach its greatest starting from mid-spring to mid-summer. By mid-spring, effects other than simple tidal-forcing are coming into play. First, agriculture districts are drawing out substantially more fresh water allowing for greater salinity intrusion. Second, dam controls are regulating fresh water flow into the delta from spring runoff. Hence the time series is more complex than the originally trained network would anticipate, and the GA-optimized neural net reflects this variability through recognizing the new spikes in the pattern. This is why we also selected August data as a calibration data set for our system.

As a performance index of the prediction of the neural network, the correlation coefficient of the measured time series data with respect to the neural network predicted time series is 0.747. The GA optimised neural network predicted a much closer time series for the Carquinez Strait (correlation coefficient 0.904); it predicted 82.63 % over fitted and 17.37 % under fitted. The Neural network without optimization predicted 59% over fitted and 41 % under fitted. Over fitting and under fitting are computed with respect to the measured time series. The under fitting or over fitting was in ±10% data range. The results of the GA optimised neural network are much

better than the neural network prediction above. The software was developed in Java [Bigus and Bigus 2001; Watson 1997] and the results were obtained using a PC of 1 Ghz Pentium IV. It took 8 hrs to optimise the training data and neural network architecture for genetic algorithm.

5. CONCLUSIONS

When not much is known about the response surface and computing the gradient is either computationally intensive or numerically unstable, a genetic algorithm is efficient. In our problem, we would like to get an optimized neural network architecture and minimum data set. This has been accomplished within 500 training cycles of a neural network. Repeated execution of a neural network takes 8 hours to create the optimized data set and architecture. We tried to use the original data set for the GA optimised neural network to predict the salinity at Carquinez Strait, but it has not performed efficiently. After removing 72 training pairs (outliers), GA has produced much better results. The neural network constructed is a feed forward neural network with a back propagation learning mechanism. The main goal has been to free the network design process from constraints of human biases, and to discover better forms of neural network architectures. The automation of the network architecture search by genetic algorithms seems to have been the best way to achieve this timely goal. We further more believe that this method will generalize to other similar environmental problems.

6. ACKNOWLEDGEMENTS

The authors wish to thank Jorge Bardina, Joan Walton and Tarang Patel for their valuable suggestions for improvement.

7. REFERENCES

Bigus P.J, Bigus J, Constructing intelligent agents using java, Wiley computer publishing, 2001.

Chung F, Modeling flow salinity relationships in the Sacramento-San Joaquin delta using artificial neural networks, Technical report OSP-99-1, Department of Water resources office of SWP planning, California, 1999.

Goldberg D.E, Genetic algorithms in search, optimization and machine learning, Addison-Wesley 1988.

Hancock, P. J. B., Pruning Neural Nets by Genetic Algorithm, Proceedings of the International Conference on Artificial Neural Networks, Brighton, 1992.

Hochman, R.; Khoshgoftaar, T.M.; Allen, E.B.; Hudepohl, J.P, Using the genetic algorithm to build optimal neural networks for fault-prone module detection, Proceedings of the The Seventh International Symposium on Software Reliability Engineering (ISSRE '96), IEEE Press 1996.

Holland J.H, Adaptation in natural and artificial systems, Ann Abor, MI : University of Michigan press, 1975.

Jurgen B., Evolutionary algorithms for neural network design and training, Proceedings of the first Nordic workshop on genetic algorithms and its applications, Vasa Finland, 1995.

Korning P.G, Training of neural networks by means of genetic algorithm working on very long chromosomes, Technical Report, Computer Science Department, Aarhus, Denmark, 1994.

Masters T, Practical neural network recipes in C++, Academic press Inc. 1993.

Miller G, Todd P., and Hegde.S, Designing neural networks using genetic algorithms, Proceedings of third international conference on genetic algorithms, George Mason University, 1989.

Rajkumar T., Johnson M.,. Prediction of salinity in San Francisco bay delta using neural network, Proceedings of IEEE SMC., Arizona, 2001.

Watson M, Intelligent Java applications, Morgan Kaufmann publishers, 1997.

Winkler E.D., Preliminary development of a statistical approach to salinity modelling in the Western Sacramento-San Joaquin Delta and Suisun Bay, California department of water resources, Technical information record, Study code no. 1463-CD-04, 1985.

Appendix I

Locations in Bay-Delta Region

RSAC045	Selby (Wickland Oil Pier)
SHWSF001	Central Bay at Presidio Fort Point
RSAC054	Sacramento River at Martinez (Carquinez Strait)
RSAC064	Sacramento River at Port Chicago
SLMZU011	Montezuma Slough at Beldons
RCSM075	Consumnes River at Michigan Bar
RSAC101	Sacramento River at Rio Vista Bridge
RSAC155	Sacramento River at Freeport
CHGRL009	Grantline Canal at Tracy Blvd Bridge
ROLD040	Old River at Clifton Court Ferry
RSAN007	San Joaquin River at Antioch

	betweens lights 7 & 8
RSAN112	San Joaquin River at Vernallis
CHDMC006	Delta Mendota Canal at the head (beginning) of the concrete liner
CHSWP003	State water project California at Harvey O.Banks Delta Pumping Plant
CHDMC004	CDEC TRD Delta Mendota Canal at Tracy Pumping plant
CHCCC006	CCWD Pumping station, Contra Costa Canal at Pumping Plant 1

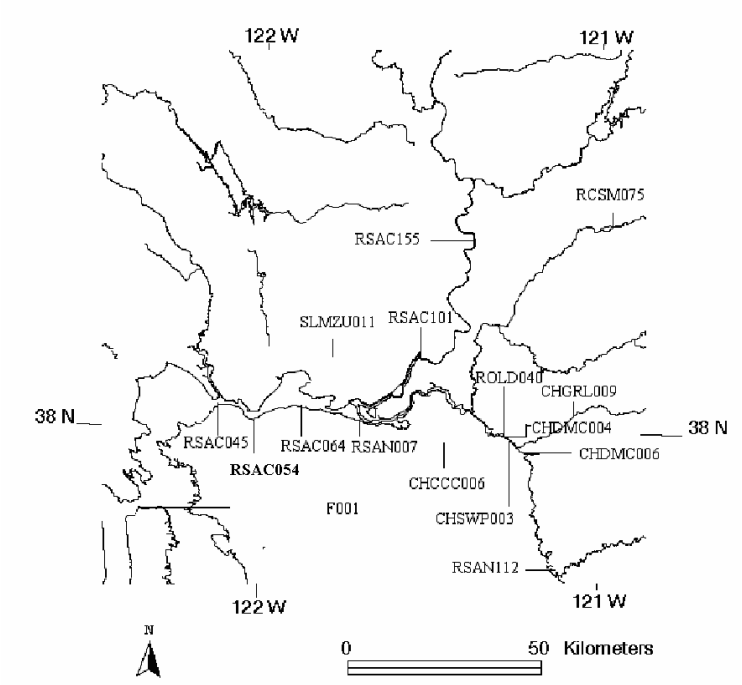

Figure 1. Location of USGS, NOAA and CADWR monitoring stations in the San Francisco Bay Delta

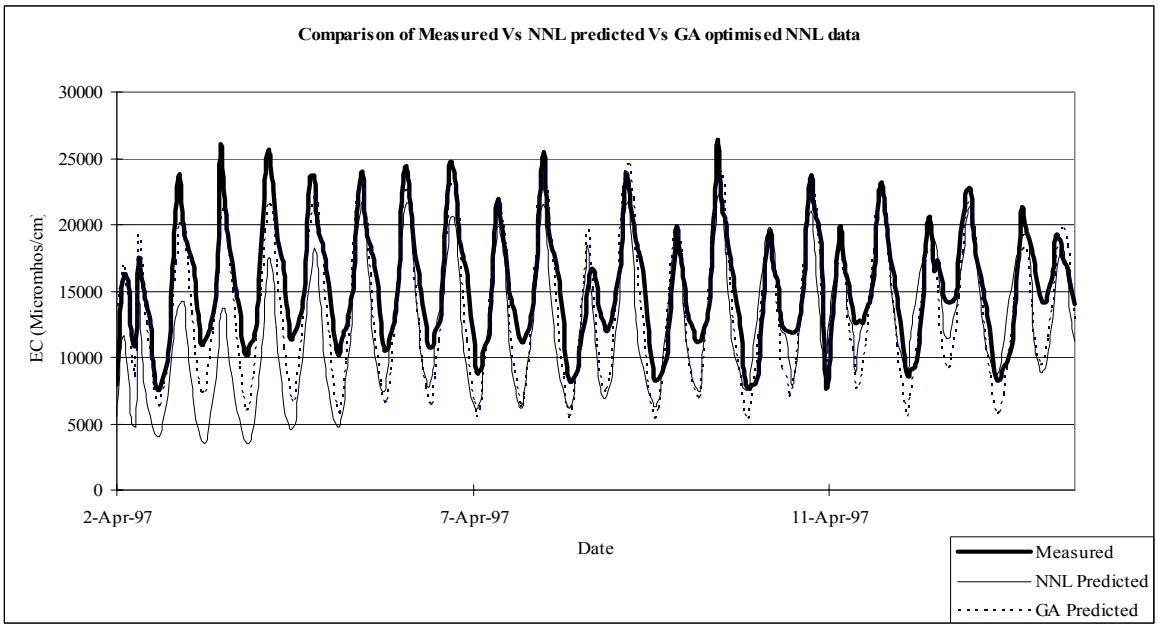

Figure 2. Comparison of Salinity prediction at Carquinez Strait with neural network and GA optimised neural network