1998 International Computer Symposium
Workshop on Artificial Intelligence
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

# GA-TABU DESIGN NEURAL NETWORK CONTROLLER

*Kuan-Shiu Chiu*, and Andrew Hunter*

School of Computing and Information Systems,
University of Sunderland, Sunderland, England, U.K.
Email:{cs0ksc,cs0ahu}@isis.sunderland.ac.uk
* also Computer Center, Tamsui Oxford University College, Taiwan

## ABSTRACT

This paper discusses the use of GAs (Genetic Algorithms) and TS (Tabu Search) to design NNCs (Neural Network Controllers) for Real-Time control of flows in sewerage networks. Genetic Algorithms evolve the weights for Neural Networks Controllers. We apply a modified Tabu Search algorithm in a novel fashion, to select the most relevant training data, in order to reduce the training time.

The comparison between this approach and various fixed penstock control settings, and genetically-designed Neural Networks, is discussed. This paper reports experiments demonstrating that GAs are both effective and robust to design Neural Networks controllers in sewerage network control problems. To confirm whether the GA-Tabu training algorithm has statistically significant better performance than other data selecting algorithms, a t-test with a 5% significance level is examined. Use of the Tabu algorithm reduces the training time without affecting the results.

## 1. INTRODUCTION

Combined sewerage systems are used in many cities and countries. The same pipes carry foul and storm flows in the systems. Most of the time they function normally. During heavy rainfall the system faces the problem of overflows which occur when the system cannot carry all the flows entering the sewers. Construction of new storage and sewers could be the solution to completely eliminate overflow problems. However, such schemes are expensive and can also be extremely disruptive because the sewerage networks may extend across wide geographical areas. Since inflows are seldom constant across an entire system, we can reduce the overflow spills by controlling the flow between parts of the system which are under different loading. This may be achieved by the installation of automatically actuated penstocks which can be opened or closed to control the flow past a certain point [1].

Standard optimization techniques, such as linear and dynamic programming, have been applied to these type of problems, but without great success other than for very simple networks [2]. Linear programming is applied in simplified problems, but alternative solutions are difficult to evaluate. Dynamic programming could be successful if all possible configurations are tested, which requires sufficient computing resources. For a complex system, it is unrealistic.

Genetic designed Neural Networks Controllers and Genetic designed Fuzzy Logic Controllers have showed to be feasible approaches [1][3]. However, Genetic designed controllers require heavy training time. If 100 different weather simulations are used, it may take several hours to train the controllers on a Sun SPARCstation 5 workstation even for a simple 2 tank system.

To reduce the training time, we apply Tabu Search algorithm to choose the proper training data for the Genetic designed Neural Network controllers. The Algorithm saves a significant amount of training time without degrading results.

In this paper, we describe the used methods and the previous work in this area. The differences between the conventional Tabu Search Algorithms and our TS algorithm will be discussed. Then GA-Tabu training scheme is implemented. Finally we compare the simulations results for both two and three tank systems.

## 2. OVERVIEW OF TECHNOLOGY USED

To implement the controllers, there are a wide range of choices, for example, Neural Networks, Fuzzy Logic Systems, Genetic Programming, and Classifier Systems [1]. The Choice of paradigm must satisfy the two factors:

**1. Control penstock setting.**

**2. Minimise the amount of spillage across the systems.**

The controllers must be able to adjust the penstock setting according to the inflow, current tank level, and the next adjacent tank level. Real-valued processing is required to represent the three inputs and one output, making neural networks and fuzzy logic systems a sensible choice. Both of them are well-suited to handle real values.

To optimise the controllers, a learning algorithm is required. Reinforcement learning methods, such as Genetic Algorithms, are feasible solutions. Genetic design of neural network controllers and Genetic design of fuzzy logic controllers are the two methods described in this paper.

### 2.1 Genetic Algorithms

Genetic Algorithms (GAs) were first developed by John Holland, his colleagues, and his students at the University of Michigan in 1970s. GAs are search algorithms that mimic biological evolution. They use a constant-size population of individuals, each one representing a possible

```
begin GA
    g:=0 { generation counter }
    Initialize population P(g)
    Evaluate P(g) { compute individuals' fitness values }
    while not satisfy the stop conditions do
        g:=g+1
        Select P(g) from P(g-1)
        Crossover P(g)
        Mutate P(g)
        Evaluate P(g)
    end while
end GA
```

**Figure 1: Pseudo-code of the simple Genetic Algorithms**

solution in a given problem space. An individual is usually coded as a binary string. The problem space is referred to as the search space, comprising all possible solutions to the problem. The search space is usually too large to use an exhaustive search.

GAs generate a initial population randomly. Each individual in the population is decoded and evaluated (given a 'Fitness') by some predefined quality criteria. Three operators are applied to the population: reproduction, crossover, and mutation. Reproduction is a process in which individuals are selected according to their fitness, and copied. Many selection methods are currently in-use. Holland's fitness-proportionate selection, where individuals are selected with a probability proportional to their relative fitness, is one of the simplest. It ensures that the expected number of times an individual is chosen is approximately proportional to its relative performance in the population. Thus, high-fitness individuals stand a better chance to be reproduced, while low-fitness ones are more likely to disappear. The crossover operator is performed according to a crossover probability (crossover rate), between two selected individuals, called parents, by exchanging parts of their code to form two new individuals, called offspring. The mutation operator is used with a small probability (mutation rate), flipping bits or re-generating bits at random. In this manner, a new population is generated. This is called one generation. The process after initialization of the population repeats until a stop condition is satisfied. One of the most commonly used stop conditions is the end of generation [4][5]. The algorithm is shown in Figure 1.

## 2.2 Neural Networks

Artificial Neural Networks (also known an ANNs, or simply Neural Networks) are computer models inspired by biological nerves systems. They consist of large numbers of simple processing units called neurons, connected together by links of various strengths called weights. Neural Networks can be built in special hardware or simulated on normal computers. A simple model of an Artificial Neuron

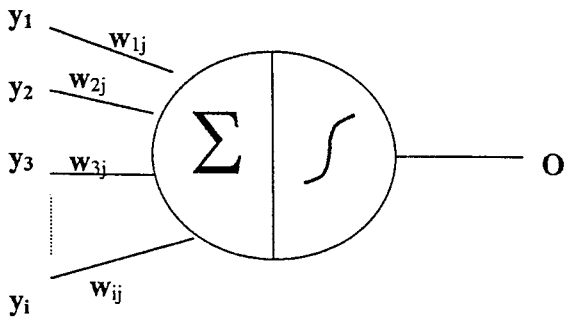


**Figure 2: A simple model of a neuron**

is shown in Figure 2.

In Figure 2, the input signals (y) to a neuron are each multiplied by the synaptic strengths (w), then are added together. The result is called the activation level of the neuron. The activation is passed as input to a non-linear sigmoid function to produce the neuron's output [6].

In order to mimic brain mechanisms to simulate intelligent behaviour, learning methods are used. Learning is the process that adjusts synaptic weights so that the network learns to perform a good mapping between inputs and outputs. This is also called training the network. Three types of learning algorithms are widely used: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. In Supervised learning algorithms, the network trainer states what outputs are expected from the training pattern inputs. In Unsupervised learning algorithms, the network learns to classify inputs based solely on their similarity with each other. In Reinforcement learning algorithms, there is no training pattern outputs, but a reinforcement signal is available which indicates how successful a network is. It is a trial-and-error learning scheme, based on feedback of the network performance.

There have been many artificial neural networks models proposed in the past 10 years. The obvious difference lies in the network architectures and learning algorithms. One of the most popular models is the multi-layered feedforward network. The multi-layered feedforward network has a number of sequential layers; every neuron in each layer is connected to every neuron in the next layer. The first layer is called the input layer, and receives signals from the outside world. The last layer is called the output layer, and propagates signals to the outside world. The other layers are called hidden layers because they are not directly accessible from the outside world - they are purely involved in decision-making [7].

## 2.3 PVM GA-NNC Training Algorithm

A previous paper [1] shows how GAs can be used to optimise the weights of Neural Network controllers. In another paper [3], the authors applied GAs to design membership function and rule bases for Fuzzy Logic Controllers. The training time for a simple 2 tank system requires several

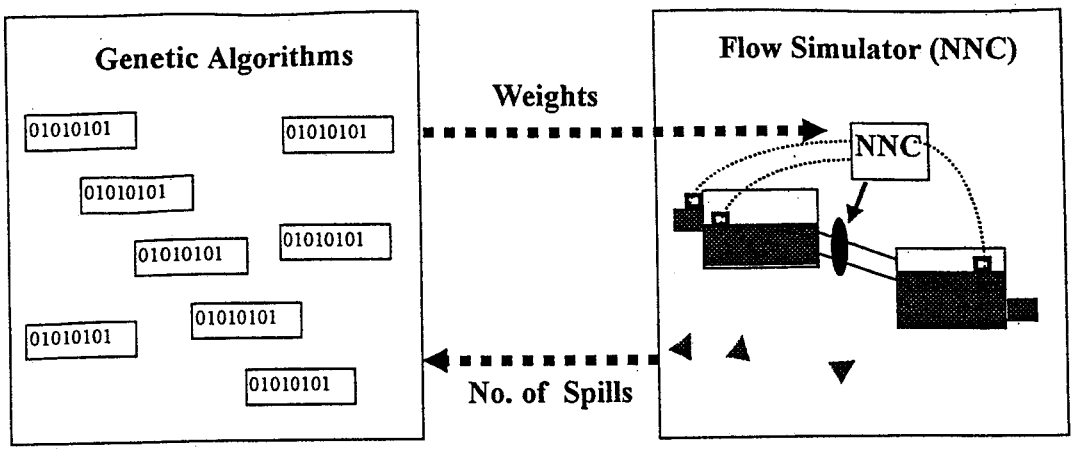


**Figure 3: The GA-NNC system architechture**

hours on a Sun SPARCstation 5 machine. For 3 tank system, it needs more than 10 hours to complete the training for both controllers. In this paper, we focus on the feasible methods to overcome the extreme computational cost. The methods in this section and next section are implemented in Genetic designed Neural Networks Controllers. The GA-NNC system architecture is shown in Figure 3. This section discusses the use of Parallel Virtual Machine (PVM) software to reduce the training time.

Genetic Algorithms have been called "embarrassingly parallelizable." The population-based approach makes it possible to evaluate a large number of entities simultaneously simple form of concurrency. The experiments have been conducted by using PVM software for the concurrent processing strategies.

PVM is a software system that allows a heterogeneous collection of Unix computers networked together to be viewed by a user's program as a single parallel computer. PVM is the mainstay of the Heterogeneous Network Computing research project, a collaborative venture between Oak Ridge National Laboratory, the University of Tennessee, Emory University, and Carnegie Mellon University (see Preface in [8]).

In our experiments, 25 "Sun SPARCstation 5" machines are used to run the simulations. Two programs are implemented: a main program and a simulator program. Genetic Algorithms in the main program generate weights and pass them to the Neural Network Control simulators. After 100 different simulations, each simulator returns the number of spillages as fitness to the main program. For population size of 100, there is one process for the main program and there are 100 processes for simulator programs run on 25 "Sun SPARCstation 5" machines.

On each generation, the main program passes weights to the simulators. It waits for the number of overflows to be returned from the simulators. The Genetic Algorithm uses the returned values as a fitness. Then the 3 genetic operators: reproduction, crossover, and mutation evolve optimum weights for the Neural Network Controllers.

For the 2 tanks system without PVM, the training time was about 8 hours. With the parallel processing of PVM, it takes about one hour to finish the training on 25 Sun-SPARCstation 5 machines.

## 3. GA-TABU DESIGNED NN CONTROLLERS

We first discuss the difference between our Tabu Search algorithms and Glover's. Then GA-Tabu training scheme is described.

### 3.1 Tabu Search algorithm

Tabu Search was introduced by Glover. It is a modified version of hill-climb search algorithms. TS introduces a flexible memory structure, Tabu list, to prevent the search from becoming trapped at locally optimal solutions. The method utilizes Tabu restrictions and aspiration criteria to drive the search into new regions. Tabu restriction discourages the reversal (or sometimes repetition) of certain moves, whereas aspiration criteria allow a move to be selected regardless of its Tabu status [9][10][11].

Tabu Search has been used to solve many combination problems. The application areas include scheduling, transportation, layout and circuit design, telecommunications, graphs, probabilistic logic expert systems, and neural networks [11]. In this paper we apply TS in a new application area, selecting proper training data efficiently for Genetic designed Neural Network controllers.

There are several differences between our Tabu algorithm and Glover's Tabu Search.

1. The algorithm works in dynamic environment. GAs evolve the weights for Neural Networks controllers. The weights change from generation to generation. Therefore, the Tabu algorithm selects the proper training data to fit in different generations.

2. There is no candidate list; the data are chosen randomly from the data not in the Tabu list.

3. The Tabu structure is a vector, not a matrix. We only use a one dimensional array to record the restriction tenure for each training data.

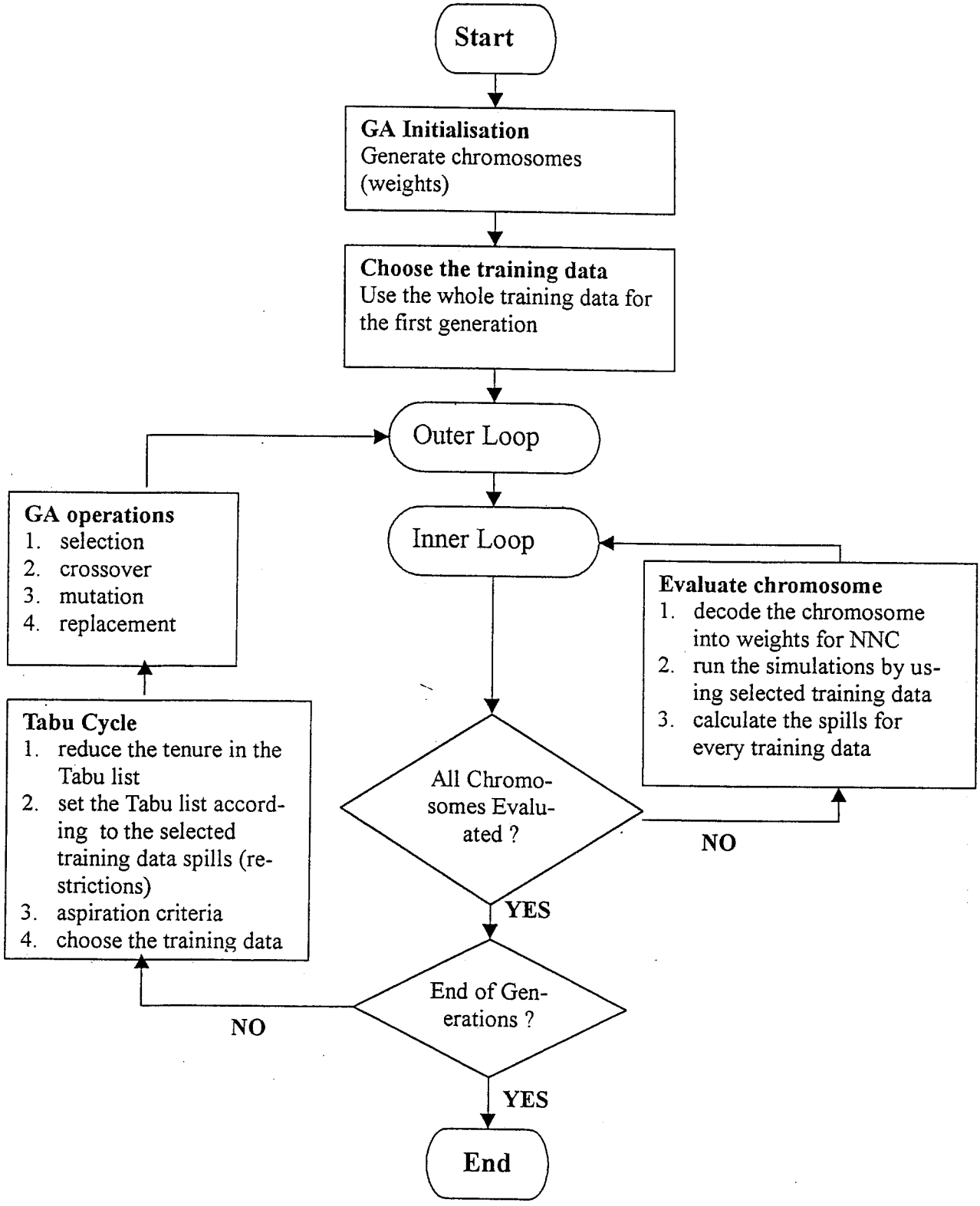


**Figure 4: GA-Tabu training scheme**

4. Tabu restrictions are applied to prevent the use of over- or under-challenging simulations on each generation.

5. Aspiration criteria are used when not enough data is selected, not because we have the best candidate so far.

### 3.2 GA-Tabu Training Algorithms

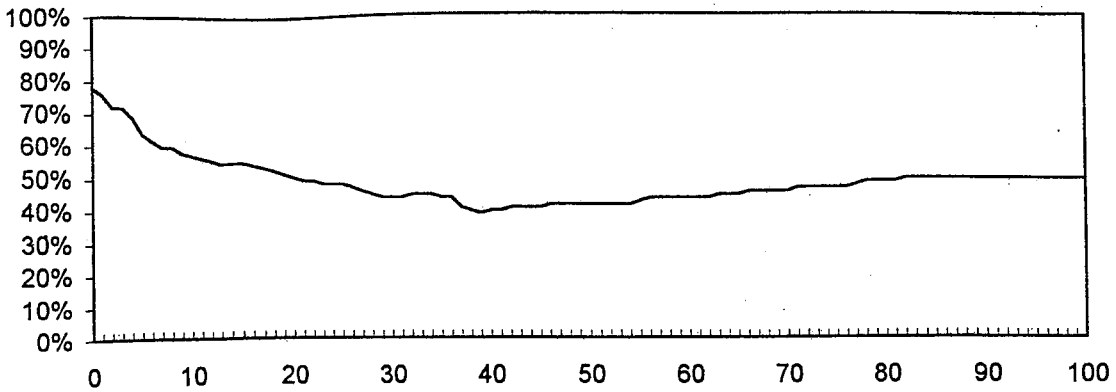


**Figure 5: Two tanks fixed aperture flood profile**

The Genetic Algorithm selects improved structures by evaluating the fitness of each structure. In this application, fitness is measured by running the system through a number of weather simulations (100), and counting how many times the system floods. In the basic approach, the same simulation runs are used on each generation. However, the GA only gets useful discriminatory feedback from simulations which cause some of the current generation to fail. Simulations which are too easy or too challenging (i.e. all chromosomes flood, or all are OK) absorb computer time with no benefit. However, as training proceeds the set of useful simulations changes (in particular, the more challenging ones become more important), and so selecting training data maintains constant selective pressure.

Tabu search uses the feedback from the Neural Network simulations to choose proper training data efficiently for training on the next generation. The details of the GA-Tabu training algorithm are as follows:

The Genetic Algorithm randomly generates the weights for the Neural Network controllers. In the first generation, we use the whole 100 different weather simulations as the training data. Each controller is tested using 100 simulations. We record and count the simulations which cause spills. After training, each simulation has a spill count between 0 and 100. A spill count 0 means the training data presents no challenge since all controllers passed the test; these simulations are discarded. On the other hand, a spill count of 100 shows the weather is terrible. No controllers passed the test at the first generation and possibly no controllers can pass through the entire 100 generations. We put such simulations in the Tabu list for 1-15 tenures randomly. Due to the heavy computation time, we randomly choose 10 training data after the first generation. The simulations with spill count below 20 would be considered as unchallenging, so we put them in the discard list. The rest of the training data are candidates to be selected for the next generation.

A difficulty could arise if most of the data are put into the Tabu list, so that not enough training data is available. In this case, another Tabu condition, aspiration criteria, is put into action. This reuses the data in the discard list until enough data is selected. The whole training scheme is shown in Figure 4.

## 4. EMPIRICAL TESTING

Our experiments are tested on several Sun SPARCstation 5 machines. We use the C language to write the simulator and Neural Networks controllers integrated with SUGAL, The SUnderland Genetic Algorithms Library [12]. Before the experiments, we briefly discuss the simulations.

### 4.1 Simulation design

A typical Genetic Algorithm for a reasonably simple problem might involve a population of one hundred chromosomes run for one hundred generations, requiring a total of 10,000 chromosome evaluations. One hundred inflow sequences were designed to simulate the different weather conditions. These were initially generated using the Hydroworks simulation package to present varying degrees of challenge to the system. Each simulation has about 500 time-steps. On each time-step a neural network controller in the system is executed and a set of flow-equations is solved. So we have about 50,000 time-steps for each of 10,000 chromosome evaluations, resulting in approximately 500,000,000 simulation time-steps in total.

With the implementation of Tabu algorithm, 100 different weather conditions are used in the first generation. Then the system choose 10 different inflow sequences for the rest of each generation. There are 54,500,000 simulation time-steps in total. The training time is about 9 times faster than the previous method.

### 4.2 Two tank experiments

The first group of experiments is done by using the two tank system (see Figure 3). This system has two interconnected tanks, with an inflow to the upper tank and an outflow from the lower. A penstock is located on the jointing pipe. A spill is deemed to occur if the water level in either tank passes the top. 100 simulations are tested using the following benchmarks.

1. The optimal spillage rate.
2. Various fixed penstock setting.
3. GA-Tabu design of Neural Network Controllers.

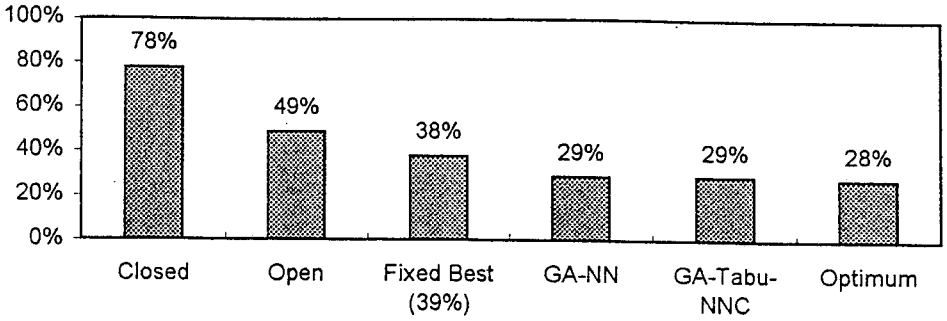


**Figure 6: Number of spills comparisons in two tank systems**

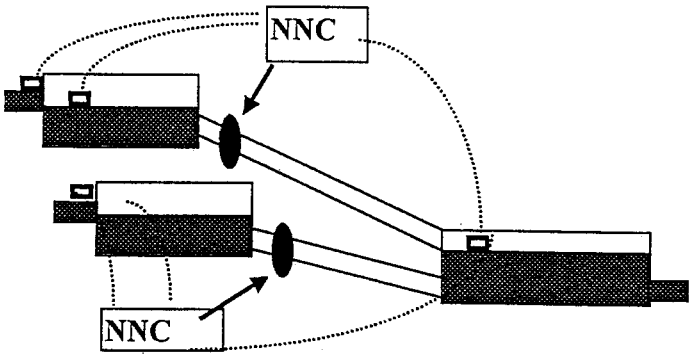


**Figure 7: Three tanks system**

4. Genetic design of Neural Network controllers (discussed in reference [1]).

Since water can only exit the system through the outflow in the lower tank and the speed of outflow is related to the depth in the lower tank, to get the optimal spillage rate, we maximise the rate of outflow by keeping the lower tank as full as possible. This is done by initially opening the penstock fully, then as the level of the bottom tank nears the top, opening the penstock just sufficiently to keep it fully topped up.

Our second experiment for the two tanks system fixes the penstock to a particular setting throughout the 100 simulations. The fixed penstock settings are tested from 0 to 100% open in steps on 1%. Not surprisingly, the largest number of spills (78%) occurs when the penstock is full closed throughout the simulations. However, the fully open penstock produces 49% spillage, which is not the lowest number. The lowest number of spills (38%) results from a fixed aperture of 39%. The results are shown in Figure 5.

For the GA-Tabu-NNC and GA-NNC, we use 3 sensor inputs; the inflow to the upper tank and the levels in the two tanks. The only output is the aperture setting (see Figure 3). The experiment is repeated five times. Both produce the best results ranging from 29% to 30% spillage. These are almost optimal values. The performance of GA-Tabu-NNC is similar to GA-NNC. Figure 6 displays the comparisons of our benchmarks.

## 4.3 Three tank experiments

The three tanks system is used for second group of experiments (see Figure 7). The system contains three interconnected tanks, two up-stream tanks feeding into a single downstream tank. The optimal strategy in the two tanks system can no longer be used, since one up-stream tank could have a heavy inflow and the other a light inflow. In this case the second tank should close its penstock to allow the first tank to empty more rapidly through the shared lower tank. However, if both upper tanks are under lower pressure, they should be closed to clear the lower tank.

We use the same inflow simulations for the two up-stream tanks. In order to simulate a weather pattern moving across a geographical area and increasing in intensity, a delay and scaling factor is used for the inflow into the second up-stream tank.

The results are shown in Figure 8. We note that the GA-Tabu-NNC outperforms the best fixed aperture setting. Both simulation results of GA-Tabu-NNC and GA-NNC are similar.

## 4.4 Comparison with other selecting training data methods

Two important issues in GAs are premature convergence and slow finishing. Premature convergence occurs when a few outstanding chromosomes exist. GAs will focus on exploiting these super-fit individuals. The results are fast convergence without finding other possible good solutions. On the other hand, slow finishing problems happen when most of the chromosomes' fitness is similar. GAs will have difficulty in convergence.

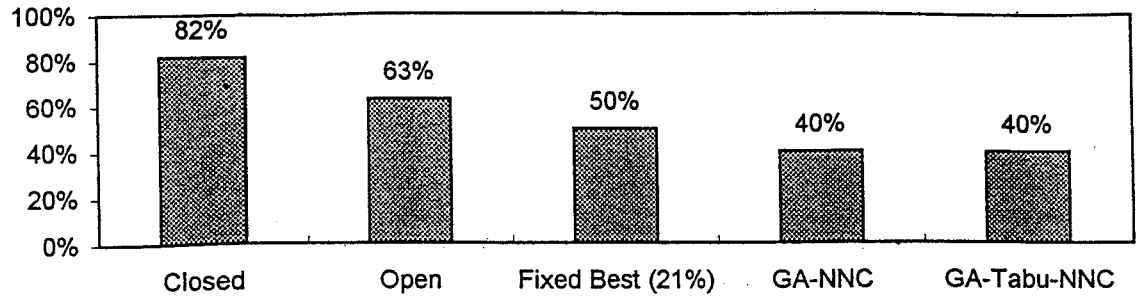


**Figure 8: Number of spills comparisons in three tanks system**

| | 2 Tank System | | | | 3 Tank System | | | |
|---|---|---|---|---|---|---|---|---|
| | Min | Max | Mean | t-test | Min | Max | Mean | t-test |
| Benchmark 1 | 29 | 37 | 30.38 | -4.291 | 40 | 50 | 43.42 | -6.042 |
| Benchmark 2 | 29 | 34 | 30.523 | -5.627 | 40 | 50 | 44.149 | -7.411 |
| Benchmark 3 | 28 | 30 | 29.3 | 1.264 | 40 | 43 | 40.86 | 1.486 |
| GA-Tabu | 28 | 30 | 29.468 | | 40 | 43 | 41.04 | |

**Table 1: Statistically figures for the 4 benchmarks (50 runs)**

Generally speaking, using a small part of the training data has higher selective pressure than using the whole training data. Using the whole training data is more likely to cause slow finishing problems.

The issues discussed are for the use of a fixed training data. For a random selection method, successful training also depends on the proper selection of data. GA-Tabu algorithm uses the Tabu list to help the proper data selection in each generation. To see if the algorithm we proposed would be better than other selection methods, we compare with the following 3 benchmarks:

1. Selecting 10 training data randomly without regarding to their degree of challenge.

2. Selecting 10 training data randomly from the moderately challenging data (initial number of spills greater than 20 and less than 100).

3. Use the whole 100 training data.

The first 2 benchmarks use 100 training data for the first generation and 10 for the rest of generations. Each experiment involved 50 independent runs of the algorithm, with different random seeds. On a single run, the performance at each generation is measured using the population minimum fitness (i.e. best fitness across the population; these are function minimisation problems). The performance of the algorithm is then assessed by taking the aggregate experimental mean of population minimums.

To confirm whether the GA-Tabu training algorithm has statistically significant better performance than other selection algorithms, we gathered mean and standard deviation values of the 50 population minimums for each benchmark, and use a t-test with a 5% significance level. With these choices, a t-value less than -1.96 indicates significantly better performance [13]. The simulation results are shown in Table 1.

In the first benchmark, the 100 training data have an equal opportunity to be chosen no matter in which generation. The training data will be meaningless if we select the over- or under-challenging data. The fitness of the better chromosomes will not be so obvious, reducing "selective pressure". It also encourages the crossover of better and worse chromosomes. The situation will be worse if we have too much of this kind of extreme training data. The t-test values are -4.291 (2 tanks system) and -6.042 (3 tanks system). They are less than -1.96, which means GA-Tabu is much better than the algorithm.

In the second benchmark, the selected training data is biased. The training prevents the use of the extreme data and focuses on the moderately challenging data after the first generation. However, it neglects the nature of GAs: the chromosomes in each generation will perform better than the previous generation. The storm weathers cause spills in the current generation. They could be prevented from overflows in the next generation. In this selection approach the controllers will not learn to adapt to tough weathers. So the optimum control is not easy. In our experiments, the algorithm is even worse than benchmark 1.

The last benchmark uses the whole 100 training data. From Table 1, it is the best training strategy. But there is no significant difference from the GA-Tabu algorithm. On the other hand, the training time is about 9 times longer than the selection methods.

In GA-Tabu design Neural Network controllers, we combined the advantages of the first two methods and prevent the disadvantages. Selection of extreme weather conditions is reduced, and the under-challenging data is discarded. As the population increases its fitness, we increase the selection of more challenging data. The goal fitness level is measured by the difference between the best chromosome's fitness and the mean fitness of the chromosomes' pool. This maintains a moderate challenge for each generation.

The algorithm has a significantly better performance than other selection approaches, and is similar to the use of the whole 100 training data.

## 5. FUTURE WORK

The application of adaptive computational techniques in flow control systems is poorly studied, and particularly in Real-Time Control of sewerage flow systems. Two previous papers [1][3] have showed Genetic designed Neural Networks Controllers and Genetic designed Fuzzy Logic Controllers to be feasible approaches. The major disadvantage of the techniques is its extreme computational cost. PVM software can be implemented to speed up the training time, but it requires more workstations. In this paper, we proposed a GA-Tabu training algorithm without increasing any machines. The experiments reported the training time is similar to the use of 25 Sun SPARCstation 5 workstations by PVM software.

For the future work, there is an issue of Local versus Global control. In one respect, the simplest system is a single Global network controller, which draws inputs from sensors across the entire flow system, and has one output to each penstock. Such a network scales rapidly in complexity as the flow system grows. Not only does the execution time grow, but the complexity of training grows. This approach is probably unrealistic for very large flow systems. In another respect, the hybrid of conventional control methods and adaptive approaches is under development. The conventional control methods are used as Local control. The adaptive approaches act as a Global overseer to guide the Local control.

## 6. CONCLUSION

The successful training of Neural Networks Controller depends on proper training data. Normally, all of the available training data are used through the whole training process. This is not efficient in reinforcement learning algorithms such as GAs, where the discriminatory power of each simulation may vary from generation to generation.

We propose a Tabu search algorithm to select the proper training data to fit in different generations. The results demonstrate a significant saving in training time without degrading performance.

## 7. REFERENCES

[1] Andrew Hunter (1997) "Genetic Design of Real-Time Neural Network Controllers." Neural Computing & Applications. V4. N3. 1997. Springer-Verlag.

[2] SO. Petersen (1987) Real Time Control of Urban Drainage Systems. M.sc. Thesis. Dept. of Environmental Engineering, Technical University of Denmark, August 1987.

[3] Kuan-Shiu Chiu and Andrew Hunter (Oct. 1997) "Genetic Design of Real-Time Fuzzy Logic Controllers." EXPERSYS-97, Sunderland, UK, Oct. 1997.

[4] Moshe Sipper (1996) "A Brief Introduction to Genetic Algorithms." http://lslwww.epfl.ch/%7Emoshes/ga_main.html accessed March 23, 1998.

[5] David E. Goldberg (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.

[6] Andrew Hunter and John MacIntyre (1996) Neural Networks for Industry Course, course notes, University of Sunderland, Oct. 1996.

[7] Andrew Hunter (1995) "Introduction to Neural Networks" http://osiris.sunderland.ac.uk/~cs0ahu/ac/nnintro.html accessed March 23, 1998.

[8] Al Geist etc. (1994) PVM: Parallel Virtual Machine, The MIT Press, Cambridge, MA, 1994.

[9] F. Glover (1989) "Tabu Search - Part I," ORSA Journal on Computing, Vol.1, No.3, pp. 190-206, 1989.

[10] F. Glover (1990) "Tabu Search - Part II," ORSA Journal on Computing, Vol.2, No.1, pp. 4-32, 1990.

[11] F. Glover and M. Laguna (1997) Tabu Search, Kluwer Academic Publishers, July 1997.

[12] Andrew Hunter The SUGAL Genetic Algorithm Simulator. http://www.trajan-software.demon.co.uk/sugal.htm accessed March 23, 1998.

[13] Joseph Newmark (1988) Statistics and Probability in Modern Life, 4th Edition, Saunders College Publishing, New York, 1988.