Genetic Algorithm Design of Neural Network and Fuzzy Logic Controllers

KUAN-SHIU CHIU

Dept. of Information Management, Aletheia University, Taiwan.

ANDREW HUNTER

School of Computing and Engineering Technology

University of Sunderland, Sunderland SR6 0DD, England, U.K.(✉)

E-mail: Andrew.Hunter@sunderland.ac.uk, cs0ksc@isis.sunderland.ac.uk
Phone: (0191) 515 2778
Fax: (0191) 515 2781

Abstract

This paper discusses the design of neural network and fuzzy logic controllers using genetic algorithms, for real-time control of flows in sewerage networks. The soft controllers operate in a critical control range, with a simple set-point strategy governing "easy" cases. The genetic algorithm designs controllers and set-points by repeated application of a simulator. A comparison between neural network, fuzzy logic and benchmark controller performance is presented. Global and local control strategies are compared. Methods to reduce execution time of the genetic algorithm, including the use of a Tabu algorithm for training data selection, are also discussed. The results indicate that local control is superior to global control, and that the genetic algorithm design of soft controllers is feasible even for complex flow systems of a realistic scale. Neural network and fuzzy logic controllers have comparable performance, although neural networks can be successfully optimised more consistently.

Genetic Algorithms Real-Time Control Fuzzy Logic Control Neural Network Control

1. Introduction

Combined sewerage systems are used in many cities and countries. The same pipes carry foul and storm flows in the system. Most of the time they function normally. During heavy rainfall inflow may exceed capacity, leading to overflows which are environmentally damaging, and expensive to the company. Overflows can be largely or entirely eliminated by construction of new storage tanks and sewers; however, such schemes are expensive and disruptive, as the systems are

distributed across wide geographical areas, particularly in major conurbations. Since inflows are seldom constant across the entire system, the incidence of overflow can in principle be reduced by real-time control (RTC) strategies, which adjust the flow among parts of the system under different loading. In practice, this may be achieved by the installation of actuated penstocks that can be opened or closed to control the flow past a certain point, together with sensor systems and controllers that operate the penstocks based upon measured depths at related points [11].

Standard optimization techniques, such as linear and dynamic programming, have been applied to this type of problem, but without great success other than for very simple networks [15]. Linear programming can be 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.

We have previously shown that genetic design of neural networks [11] and fuzzy logic controllers [2] is a feasible approach. However, there are key issues that these earlier papers did not address: the extremely heavy computational requirements, and the complexity issues raised by realistically scaled problems (the "curse of dimensionality" and "combinatorial explosion"). This paper introduces techniques to improve controller performance, to limit controller complexity and to reduce the training time.

In section 2 standard approaches to control of flow systems are briefly introduced. Section 3 describes how the soft controllers are implemented, and how they are optimised using a genetic algorithm. Section 4 discusses a novel Tabu-based algorithm that selects a subset of the training data for the genetic algorithm, maintaining selective pressure with greatly reduced execution time. Section 5 presents experimental results, and section 6 concludes the paper.

2. Standard Approaches to Flow Control

The most common approaches to flow control are based upon the On-Off control strategy [13]. The output of the controller is a binary signal, interpreted as either

On or Off (i.e. the penstock is either fully open or fully closed). The controller output signal is changed if the detected fluid level crosses a specified depth (the set point). The set-points are usually chosen heuristically by an expert, although it is also possible to deploy optimisation algorithms to select them.

On-Off control performs very adequately if a good set point is selected, but not optimally. One reason for the restricted performance is that the penstock may be opened and closed repeatedly during heavy rainfall. While closed, no flow at all is permitted, whereas in fact there is almost certainly some (reduced) capacity in the lower tanks.

In practice, Two-point or Three-point control is usually applied instead of On-Off control. Two-point control is the simplest and most frequently applied method of discrete control. An example for Two-point control is a pump switch to fill a reservoir. The pump switches On at a low level and Off at a high level. Three-point controllers are typically used for such regulators as sluice gates and weirs. The output signal changes according to the three input points: minimum, middle, and maximum [17]. Since these approaches are similar, we use On-Off controllers for benchmarking.

3. Soft Controllers

A wide range of soft computing techniques may be used to implement controllers, including neural networks [11], fuzzy logic [2], genetic programming [14], and classifier systems [10]. In all cases, the controller's output signal gives the penstock setting, and the input signals are measurements of fluid depth in appropriate tanks. In global control, a single controller takes inputs from throughout the system, and has outputs to all penstocks. In a local control system, inputs are from nearby tanks (typically those above and below the controller), and there is a single output. In either case, processing of continuous variables is required to represent the inputs and outputs, making neural networks and fuzzy logic systems a sensible choice. Both of them are well suited to handle continuous variables.

To optimise the controllers, a learning algorithm is required. Reinforcement learning methods, such as genetic algorithms, are feasible solutions (by reinforcement learning we mean search algorithms that work by generating proposed solutions and testing these to obtain a "fitness" or "error" value, that constitutes the sole feedback on performance to the optimisation algorithm). The genetic algorithm generates candidate controllers. To evaluate the fitness, each controller is embedded into a flow system simulator, which is executed one hundred times under different simulated weather conditions. The inflow sequences were generated using the Hydroworks simulation package [18], and are consistent with real weather patterns. Each simulation has about 500 time-steps. A count of the number of simulations that overflow provides a fitness value which is passed back to the genetic algorithm. Possible alternative optimisation approaches include Simplex [16] and Simulated Annealing [4], but these are not considered further here.

In recent papers the authors have shown how genetic algorithms can be used to optimise the weights and biases of neural network controllers in very simple systems [11], and the membership functions and rule base of fuzzy logic controllers [2]. The experimental system architecture is shown in figure 1.

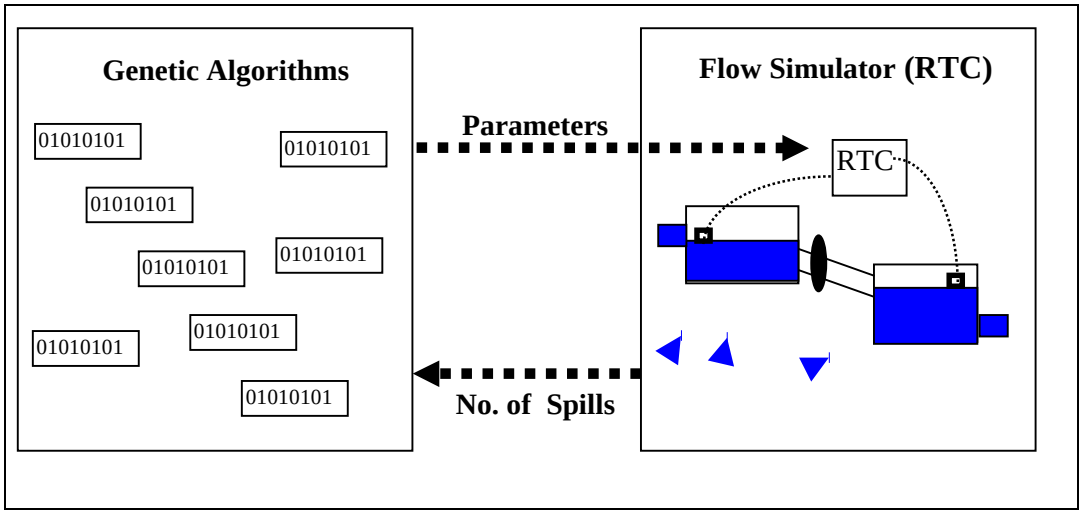


Figure 1 The system configuration

3.1 Hybrid Crisp/Soft Controllers

On-Off control achieves reasonably good results, as the strategy of allowing flows through the system as rapidly as possible when it is not heavily loaded is basically sound. Recognising this, we can improve the performance of the soft controllers by deploying a simple set point rule (a crisp controller) under light or medium loading, and only switching on the soft controller during periods of heavy loading. We therefore include a set point in our systems; when the fluid level is below the set point, the penstock is fully opened. When the fluid level is above the set point, the soft controller is used to determine how far open the penstock should be held. The soft controller is able to base this decision on the exact level of loading above and below the penstock, and (depending on the type of controller) on more global information.

3.2 Computational Complexity

There are two major issues in genetic design of soft controllers: the extremely high computational cost, and the computational complexity (the so-called "combinatorial explosion" problem).

In our experiments, 100 different weather simulations are used, each containing about 500 time-steps, so that the system is optimised to respond to a representative range of possible conditions. During evaluation of a controller, the simulator is executed (solving flow equations and running the soft controllers to operate the simulated penstocks) until an overflow occurs, or the simulation completes without overflow. The evaluation of a single chromosome therefore requires up to 50,000 executions of each controller. 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, implying 500,000,000 controller executions in total.

With these computational requirements, it takes several hours to evolve the controllers on a Sun SPARCstation 5 workstation even for a simple two tank system; a thirty one tank system takes several days. Fortunately, this issue only applies during the initial evolution of the controllers. Once evolved, they can be

embedded in the system and executed in real time. However, it is clearly still desirable to reduce the time spent in evolving the solution.

Problem complexity is a key issue in scaling up to real-world size problems. Consider first the case of fuzzy logic controllers. Fuzzy logic controllers are usually applied to problems with small number of inputs. An increase in the number of input variables causes an exponential growth in the number of rules generated. For example, in a thirty one tank system using a single global network controller, there are thirty input variables, and the maximum number of generated rules is $2e^{14}$, assuming that 3 fuzzy sets are used for each input variable. We investigated the use of the Combs method to address this issue, but this proved largely unsuccessful; a more detailed discussion may be found in [3]. The "curse of dimensionality" issue makes fuzzy logic global control impossible for large sewerage networks. Global neural network controllers also have problems, if not on quite the same scale as fuzzy logic controllers.

Local control avoids the complexity issue, at the price at sacrificing the possibility of optimal performance. Each controller takes only local inputs. The input variables are selected so that the same controller can be deployed anywhere in the system (e.g. depths are expressed as a proportion of the tank depth, rather than explicitly). The generic local controller is then embedded throughout the system, and "globally" optimised, in the sense that the genetic algorithm selects for controllers which when deployed *en masse* have desirable global dynamics.

The controllers, whether neural networks or fuzzy logic, are combined with simple On-Off control. Each penstock is partially controlled using a set point on the tank depth immediately below the penstock. The set point is also selected by the genetic algorithm. If the fluid level does not exceed the set point, the penstock is kept open. Only when the fluid level is above the set point is the soft controller deployed, thus allowing it to concentrate on modelling the critical part of the response curve.

The soft controller takes two input variables - the fluid depths in the tanks above and below. The level below is measured from the set point, rather than from the bottom of the tank.

3.3 Controller Representation

Floating-point chromosomes are used to encode both neural networks and fuzzy logic controllers. There has been some dispute in the literature as to whether such a high-cardinality representation is appropriate; Goldberg argues [8] that a binary encoding maximises the number of schemata per bit, whereas Antonisse [1] interprets the schema definition differently, and concludes that the higher cardinality representation has more schemata. In a later paper, Goldberg [9] also recognises that high cardinality representations are sometimes superior.

For neural networks (standard feedforward Multilayer Perceptron networks are used) the number of hidden units was fixed in preliminary experiments. Two hidden units are used for local controllers; the number varies in global controllers according to system complexity. The chromosome contains all the weights and biases of the network (plus an extra value for the set point). For the fuzzy logic controllers, the Mamdani approach with Min-Max-COG is used. Each input/output variable has three triangular fuzzy sets, requiring two floating-point numbers for each variable. The two floating-point numbers are used for the base lengths of the fuzzy sets, one for the first/third set and the other for the second. The centres of the fuzzy sets are fixed. This keeps the chromosome representation as simple as possible. A position dependent encoding is implemented to construct the fuzzy rule base, where every position corresponds to a rule. The value of an allele represents the output fuzzy set of the rule, and its locus the combination of input fuzzy sets. Further details on the encoding of the fuzzy logic controllers are given in [3].

4. Selection of Data using Tabu Search

The Tabu Search, a modified hill-climbing algorithm, was introduced by Glover [6] [7]. The algorithm uses a flexible memory structure, the Tabu list, to prevent the search from becoming trapped at locally optimal solutions. Tabu restrictions and aspiration criteria are used to drive the search into new regions. Tabu

restrictions discourage the reversal (or sometimes repetition) of past moves, whereas aspiration criteria allow moves that would normally be disallowed by Tabu status, if there are special circumstances that imply the status should be overridden.

We use a Tabu-like algorithm to select a subset of the training data (weather simulations) for use on each generation. Only a subset of the data has discriminatory value at each generation (specifically, only a simulation that causes some members of the population of controllers to overflow, and some not to, has discriminatory value). The set of discriminatory data changes over time - simulations that are easy to control without flooding can be discarded as the genetic algorithm converges, and simulations that initially caused flooding in the entire population can be usefully introduced later. Using such a subset reduces the execution time on each generation, while maintaining selective pressure.

In the first generation, the entire set of 100 inflow sequences is used as the training data. Each controller is tested using the 100 sequences. The number of times that each causes an overflow is counted. After training, each sequence has an overflow count between 0 and 100. An overflow count of 0 means that the sequence has no discriminatory value, since all controllers passed the test; these sequences are discarded. On the other hand, an overflow count of 100 implies that it is extremely difficult to handle the sequence. No controllers passed the test at the first generation, and possibly no controllers can pass throughout the entire optimisation procedure. Such sequences are put in the Tabu list for 1-15 tenures (the number chosen randomly).

In subsequent generations, 10 sequences are chosen. Sequences with a spill count below 20 are considered as unchallenging, and are placed in the *discard list*. The rest of the sequences are treated as candidates for selection in the next generation.

A difficulty can arise if more than ninety sequences are put into the Tabu list, so that not enough are available for controller evaluation. In this case, another Tabu concept, the aspiration criteria, is brought into action. Sequences are randomly reselected from the discard list to form a full set of training data.

Extensive experimental results are reported in [3], but have been omitted from this paper for brevity. In summary, they show that the algorithm is just as effective as using the full set of data for training on each generation, but with a speed-up factor of nine.

5. Experimental Results

Experiments were conducted on a network of Sun SPARCstation 5 machines. The simulator is custom-written in the C programming language, as are the neural network and fuzzy logic controllers, whereas the inflow sequences were generated with a commercial simulation package, HydroWorks. The genetic algorithm is provided by SUGAL, The SUnderland Genetic Algorithms Library [12]. The simulator is capable of parallel execution using a Parallel Virtual Machine (PVM) environment [5]; however, as for the purposes of this paper a large number of simulations needed to be run, concurrent execution was not used in this case.

Experiments were conducted using four systems of varying complexity, with two, three, ten and thirty one interconnected tanks respectively. The two tank system is the simplest possible system, yet real-time control is still demonstrably superior to simpler techniques. The three tank system is the simplest system that requires balancing of conflicting requirements. The ten tank system is sufficiently large to illustrate the contrasting performance of local and global control strategies, and the thirty one tank system is of comparable complexity with some real world flow systems. We use a single source file for each weather simulation. 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 up-stream tanks; see [3] for more details.

The experiments with soft controllers were conducted five times, and the bar charts show the maximum and minimum performance across the five experiments.

5.1 Two Tank Experiments

The first group of experiments uses a minimal two tank system (see figure 2). 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 fluid level in either tank passes the top. The soft controllers are compared with several benchmarks, discussed further below. We also report results for soft controllers that act throughout the entire control range, and hybrid crisp/soft controllers that combine a set-point with a soft controller for the critical range.

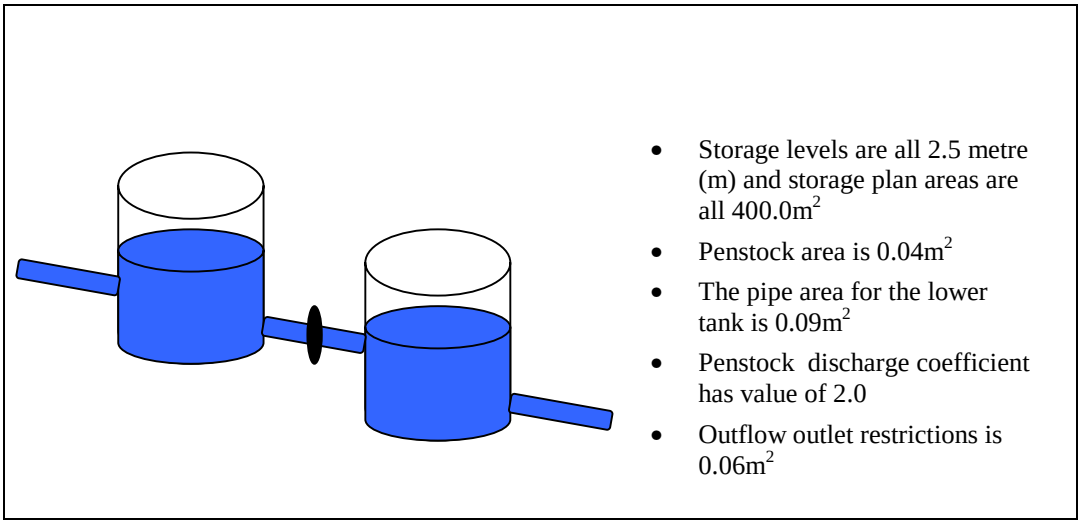


Figure 2 A two tank system

The Closed and Open benchmarks represent the situations where the penstock is either kept fully open or fully closed throughout the simulation. In the Fixed Best benchmark, the penstock is kept at 42% throughout - this particular aperture was experimentally determined to be the best fixed aperture.

In the On-Off control benchmark, a simple On-Off controller is applied, with the set-point at 96% of the tank depth; again, this was experimentally determined to be the best set-point.

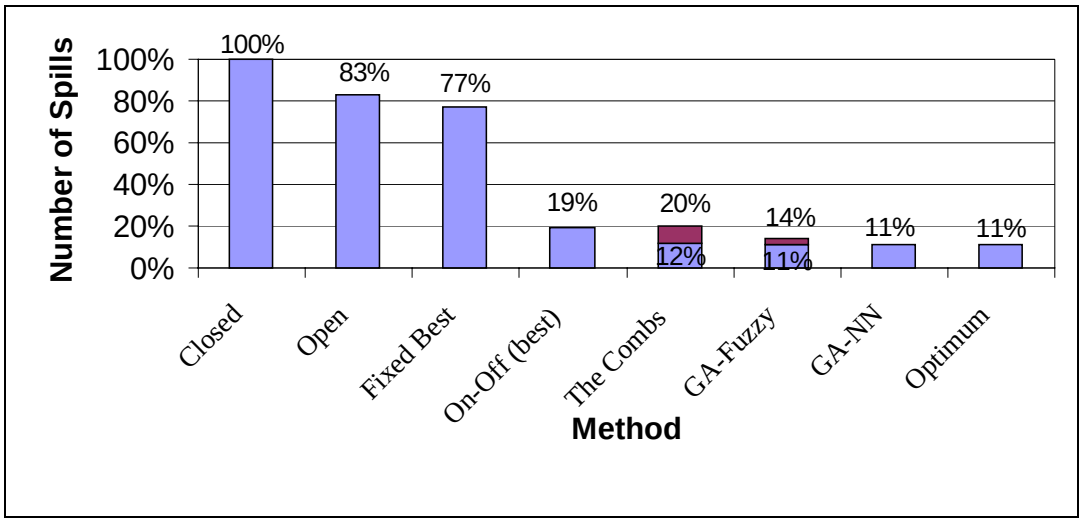
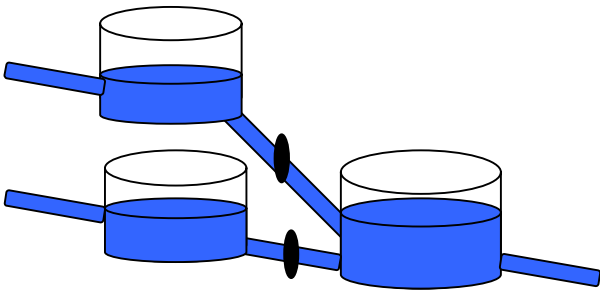


Figure 3 A comparison of the number of spills in a two-tank system

The optimum spillage rate is calculable for this simple flow system, although not for any more complex system. Since water can only exit the system through the outflow in the lower tank, and the speed of the outflow is related to the depth in the lower tank, to get the optimal performance 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. We calculate the optimal aperture on each iteration by experimentally determining the largest aperture that does not cause overflow.

It is noticeable that, even on this very simple system, On-Off control does not achieve optimal performance, whereas the hybrid soft controllers do. The exception is the Combs method fuzzy controller, which performed relatively poorly.

5.2. Three Tank Experiments



- Storage levels are all 2.5 metre (m)
- Storage plan areas are 450.0m^2, 400.0m^2, and 600.0m^2 for the two upper tanks and the lower tank, respectively
- Penstock areas are 0.04m^2 and 0.05m^2 for the upper penstock and the lower penstock, respectively
- The pipe area for the lower tank is 0.09m^2
- Penstock discharge coefficient has value of 2.0
- Outflow outlet restrictions is 0.06m^2

Figure 4 A three tank system

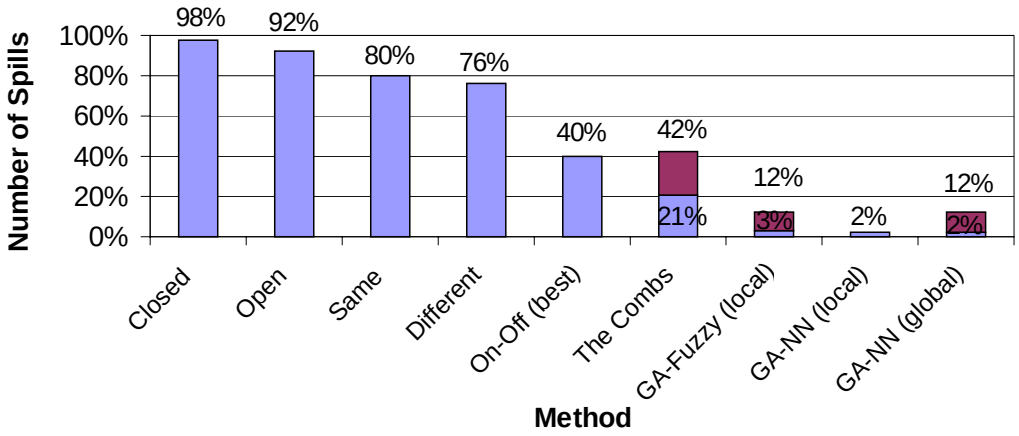


Figure 5 A comparison of the number of spills in a three-tank system

The three tank system is shown in figure 4. It is no longer possible to detemine the theoretical optimum performance, as the simple strategy used in the two tank case no longer applies. The fixed penstock benchmark is complicated by the fact that the two penstocks could have different fixed settings, and therefore two versions of the benchmark are illustrated (one where the two have the same aperture, and one where they have different apertures). Figure 5 shows the results with the three tank system.

Even the very modest increase in complexity from the two tank system has noticeable effects. We again observe that the Combs method is inferior, and this was abandoned for subsequent experiments. On-Off control proves noticeably inferior to the hybrid soft controllers. With this simple system, there is no distinguishable difference in performance between the local and global controllers.

Neural networks show a modest superiority over fuzzy logic in this instance, although it would be dangerous to draw general conclusions from this.

5.2 Ten Tank Experiments

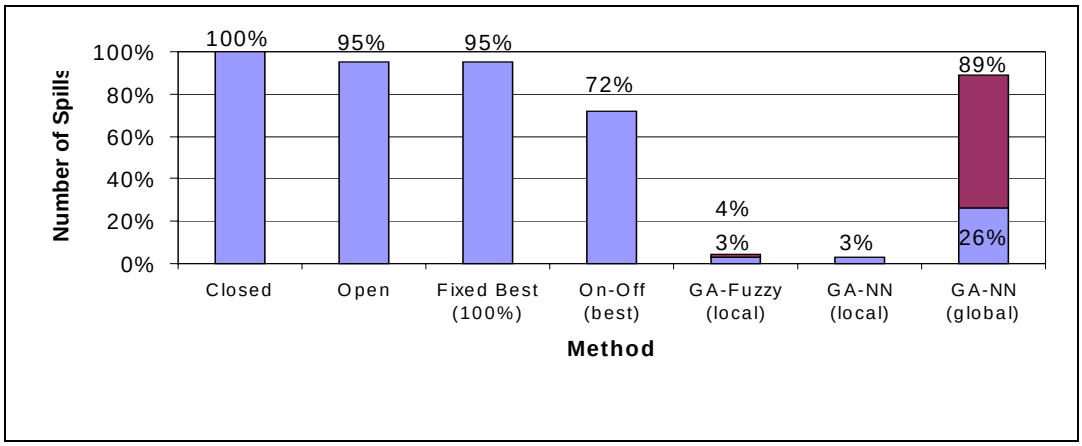


Figure 6 Local versus global control in a ten-tank system

To illustrate the difference in performance of local and global control strategies, a system containing ten tanks is presented. Global fuzzy control was not attempted, as the combinatorial explosion makes the Mamdani approach infeasible, and the Combs method proved markedly inferior even on the simple two tank and three tank systems. The results are shown in figure 6. They demonstrate quite clearly the superiority of the simpler local control strategies.

As the global controller is required to model quite a complex transformation function, we experimented with a range of network complexities, containing from two to six hidden units, and both one and two hidden layers. The result reported is for the best of these networks; see [3] for further details.

5.4 Thirty-one Tank Experiments

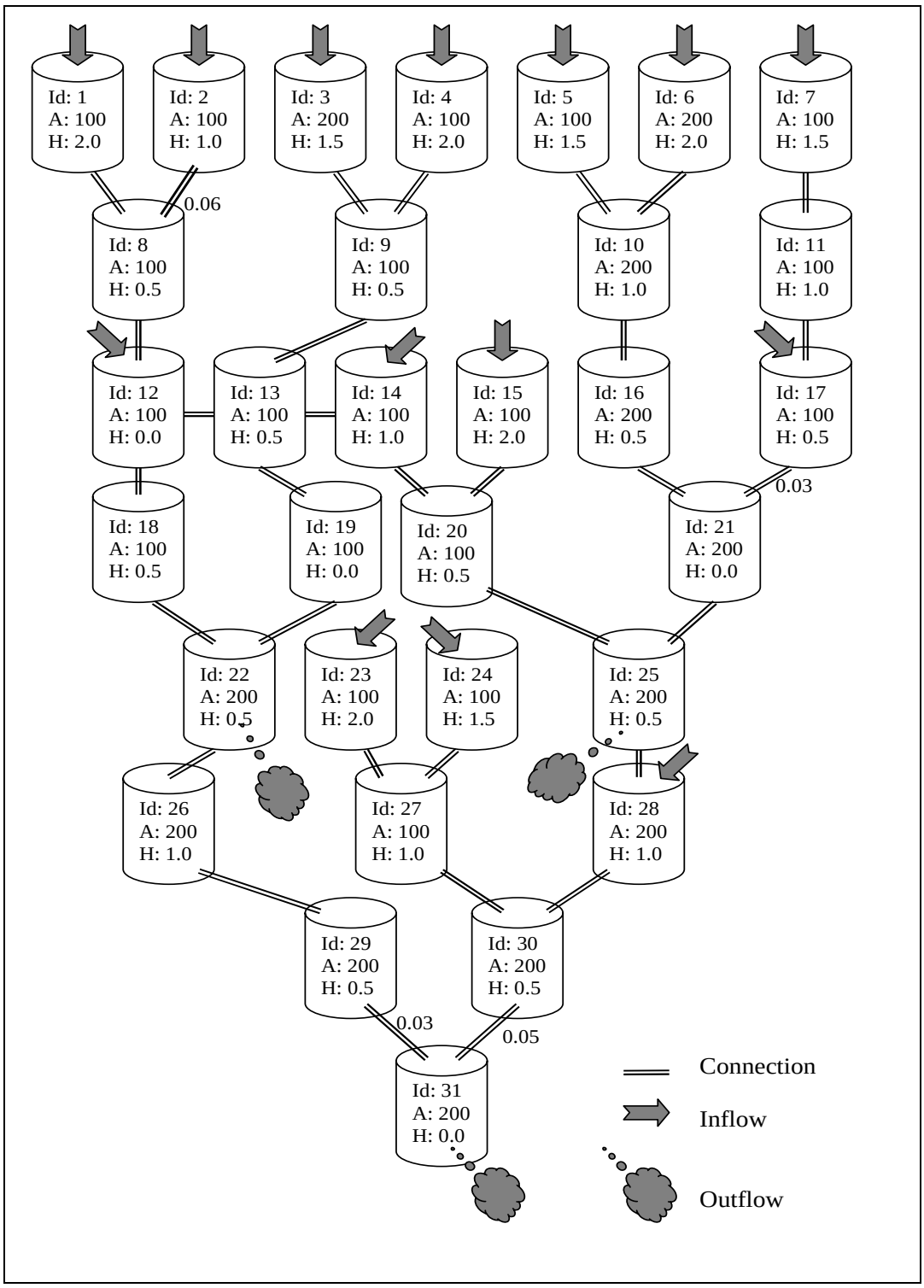


Figure 7 A 31 tank system

- Penstocks are installed in each connection
- Storage levels are all 2.5 metre (m)
- Penstock areas are all 0.04 m^2 unless stated by the connection symbol

- Penstock discharge coefficient all have value of 2.0
- Outflow outlet restrictions are all 0.6 m^2
- A indicates the storage plan area (m^2)
- H indicates the height of the tank base above a baseline

The thirty one tank system contains 14 up-stream tanks feeding into 17 downstream tanks; see figure 7. The system includes reverse-flow dynamics (where flow can reverse along a near-horizontal pipe), and is of comparable scale to some real world problems [3]. Since this system includes a large number of tanks and penstocks, it seems reasonable that in On-Off control different set-points might be used for each controller. Two On-Off benchmarks are therefore illustrated. In the first, a single common set-point is used (experimentally optimised). In the second, a genetic algorithm was used to select different set-points for each controller. The results are shown in figure 8. Once again, it is clear that the soft controllers have superior performance.

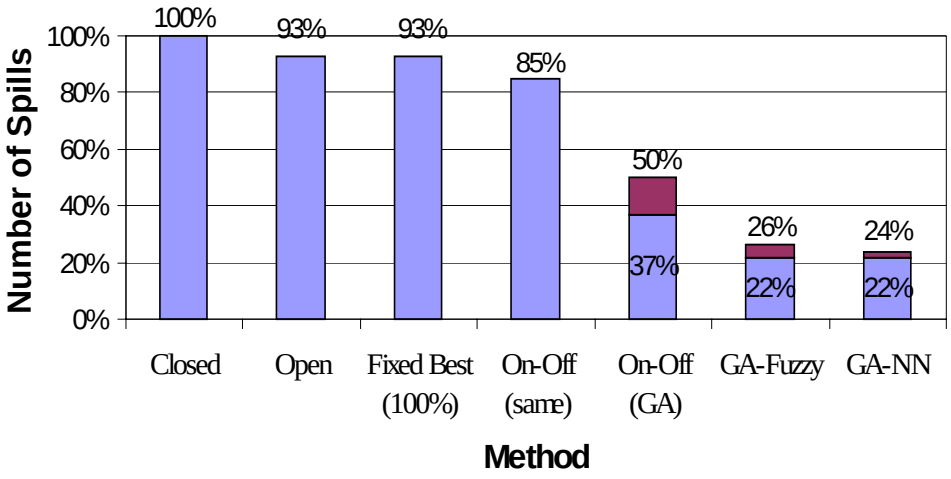


Figure 8 A comparison of the number of spills in a 31-tank system

6. Conclusion

Real-time control of the flow in complex network systems such as sewerage systems is an extremely difficult problem. The interaction between control actions in different parts of the system, combined with exogenous effects from rainfall, make it almost impossible to design good control strategies "by hand."

Soft computing techniques can be used to implement controllers, by embedding the controllers into a simulator, evaluating performance under a range of conditions, and feeding back an assessment of performance to guide a genetic algorithm for optimisation.

Our experiments have indicated that this approach is feasible for networks of realistic scale, although to date we have used simplified flow dynamics in the simulator. However, it is not anticipated that the inaccuracies in the simulator invalidate the conclusion that automatic design is possible.

Two forms of soft controller have been investigated: neural networks and genetic algorithms. Our experiments show that the performance of the two methods is comparable; neural networks optimise well more consistently, but with sufficient runs fuzzy logic controllers usually match their performance. Fuzzy logic controllers also have the benefit of interpretability, and may be preferable in some cases for this reason.

Global control has been shown to be infeasible, at least with the techniques we have so far deployed. Possible alternatives that might address the problem include sparsely connected networks, and/or weight decay/elimination techniques, both the subject of future work. Local control achieves significantly better control than the conventional benchmark methods that were examined.

A key issue is the extremely high computational cost involved in optimising the controllers. We have addressed this issue using several techniques: a hybrid scheme including a set point that simplifies the task required of the soft controllers; a PVM based concurrent genetic algorithm simulator; and a novel Tabu-based algorithm that selects a training data subset on each generation of the genetic algorithm without reducing selective pressure.

It is interesting to consider whether there are general classes of problem to which this approach can be applied. We identify the following critical features in our system:

- It consists of a directed network through which some "commodity" travels;
- The network has a definite and limited "loading capacity" within each locale, and overloading is problematic;
- The system may be under uneven load;

The generic approach is to adaptively restrict flow so as to avoid local overloading. Other problem-domains where this approach might apply, besides other fluid flow problems, include goods supply networks, manufacturing system and discrete event simulations.

7. References

1. Antonisse H. (1989) "A New Interpretation of Schema Notation that Overturns the Binary Encoding Constraint," *Proceedings of the Third International Conference on Genetic Algorithms*, 86-91, San Mateo, CA: Morgan Kaufmann

2. Chiu K. and Hunter A. (1997) "Genetic Design of Real-Time Fuzzy Logic Controllers," *International conference EXPERSYS-97*, Sunderland, U.K, 245-250

3. Chiu K. (1999) *Adaptive Optimization of Intelligent Flow Control*, Ph.D. Thesis, School of Computing, Engineering and Technology, University of Sunderland

4. Davis L. ed. (1987) *Genetic Algorithms and Simulated Annealing*, San Mateo, CA: Morgan Kaufmann

5. Geist A. etc. (1994) *PVM: Parallel Virtual Machine*, Cambridge, MA: The MIT Press

6. Glover F. (1989) "Tabu Search - Part I," *ORSA Journal on Computing*, 1,3:190-206

7. Glover F. and Laguna M. (1997) *Tabu Search*, Kluwer Academic Press

8. Goldberg D. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.

9. Goldberg D. (1990) "Real-Coded Genetic Algorithms, Virtual Alphabets, and Blocking," University of Illinois at Urbana-Champaign, *Technical Report No. 90001*

10. Holland J. (1992) *Adaptation in Natural and Artificial Systems* (2nd ed.), Cambridge, MA: MIT Press

11. Hunter A., Hare G., and Brown K. (1997) "Genetic Design of Real-Time Neural Network Controllers," *Neural Computing & Applications*, 6:12-18

12. Hunter A. (1998) "Crossing Over Genetic Algorithms: The Sugal Generalised GA," *Journal of Heuristics*, 4,2:179-192

13. IAWPRC Task Group on RTC of UDS (1989) *Real Time Control of Urban Drainage Systems - The State of the art*, (ed. W. Schilling), Pergamon Press

14. Koza J. (1992) *Genetic Programming: On the Programming of Computers by means of Natural Selection*, Cambridge, MA: MIT Press

15. Petersen S. (1987) *Real Time Control of Urban Drainage Systems*, M.sc. Thesis, Dept. of Environmental Engineering, Technical University of Denmark, August 1987, ISBN 87-89220-04-8

16. Press W., Teukolsky, S., Vetterling, W. and Flannery, B. (1992) *Numerical recipes in C: the art of scientific computing, 2nd ed*, Cambridge University Press

17. Schilling W. (1992) "The Feasibility of Real Time Control of Combined Sewer Overflows," *Civil Engineering Practice*, 7,2:17-26

18. Wallingford (1997) HydroWorks V3.3 Software http://www.wallingford-software.co.uk/ accessed June 9, 1999