5 Data-Fitting, Evolutionary, and Qualitative Modeling

Clearly, all model outputs depend on model inputs. The optimization and simulation models discussed in the previous chapters are no exception. This chapter introduces some alternative modeling approaches that depend on observed data. These approaches include artificial neural networks and various evolutionary models. The chapter ends with some qualitative modeling. These data-driven models can serve as substitutes for more process-based models in applications where computational speed is critical or where the underlying relationships are poorly understood or too complex to be easily incorporated into calculus-based, linear, nonlinear, or dynamic programming models. Evolutionary algorithms involve random searches based on evolutionary or biological processes for finding the values of parameters and decision variables that best satisfy system performance criteria. Evolutionary algorithms are popular methods for analyzing systems that require complex simulation models to determine values of performance measures. Qualitative modeling approaches are useful when performance measures are expressed qualitatively, such as “I want a reliable supply of clean water at a reasonable cost,” where there can be disagreements among different stakeholders and decision makers with respect to specifying just how reliable, how clean, and how affordable.

5.1 Introduction

Most models used for water resources planning and management describe, in mathematical terms, the interactions and processes that take place among the various components of the system. These mechanistically or process-based models usually contain parameters whose values are determined from observed data during model calibration. These types of models are contrasted to what are typically called “black-box” models, or statistical models. Such models do not describe physical processes. They attempt to convert observed inputs (e.g., rainfall and runoff, inflows to a reservoir, pollutants entering a wastewater treatment plant or effluent concentrations discharged to a river) to observed outputs (e.g., runoff, reservoir releases, pollutant concentrations) using any set of mathematical equations or expressions that does the job. One type of such models is regression.

Regression equations, such as of the forms

$$\text{Output variable value} = a + b(\text{input variable value}) \tag{5.1}$$

$$\text{Output variable value} = a + b(\text{input variable value})^{C} \tag{5.2}$$

$$\begin{aligned}\text{Output variable value}\\ = a + b_1(\text{input variable}_1\text{value})^{C1}\\ + b_2(\text{input variable}_2\text{value})^{C2}\end{aligned} \tag{5.3}$$

are examples of such data-fitting or statistical models.

They depend on observed inputs and observed outputs for the estimation of the values of their parameters (*a*, *b*, *c*, etc.) and for further refinement of their structure. They lack an explicit, well-defined representation of the processes involved in the transformation of inputs to outputs. While these statistical models are better at interpolating within the range of data used to calibrate them, rather than extrapolating outside that range (as illustrated in Fig. 5.1), many have proven quite successful in representing complex physical systems.

Other examples of data-driven models are based on biological principles and concepts. These are a class of probabilistic search procedures known as evolutionary algorithms (EAs). Such algorithms include genetic algorithms (GAs), genetic or evolutionary programming (GP or EP), and evolutionary strategy (ES). Each of these methods has many varieties but all use computational methods based on natural evolutionary processes and learning. Perhaps the most robust and hence the most common of these methods are genetic algorithms and their varieties used to find the values of parameters and variables that best satisfy some objective. Alternatively, an extension of regression is artificial neural networks (ANN). The development and application of black-box models like GA, GP,

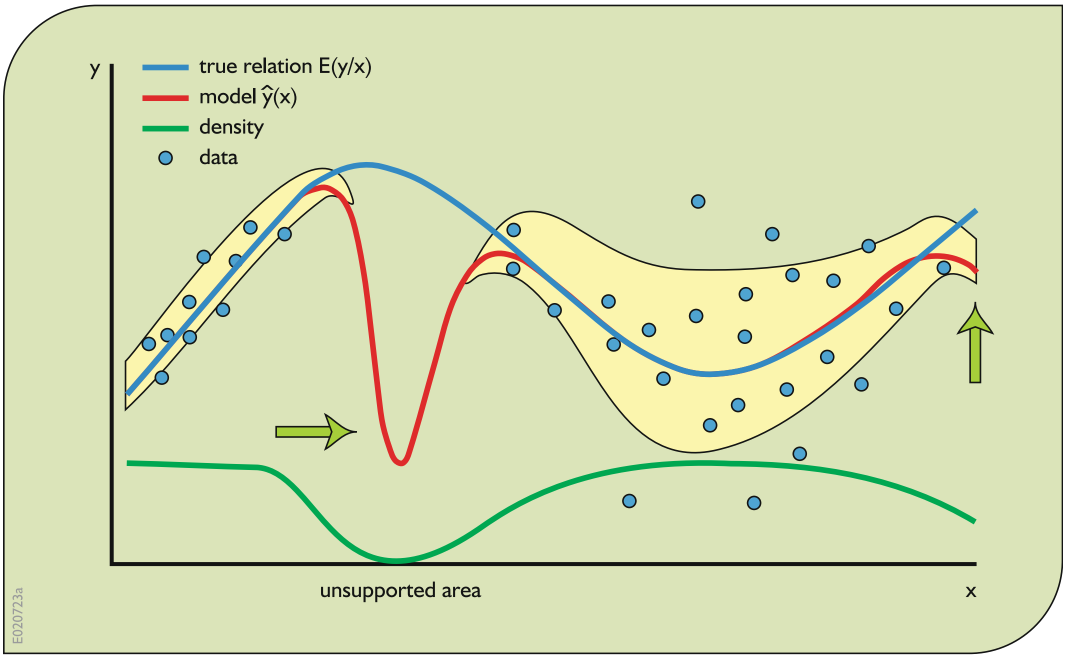


Fig. 5.1 Data-fitting models are able to estimate relatively accurately within their calibrated ranges, but not outside those ranges. The *bottom curve* represents the relative density of data used in model calibration. The *arrows point* to where the model does not predict well

and ANNs emulate larger, deterministic, process-oriented models. Once calibrated, their use may be advantageous if and when it is quicker to use them to obtain the information needed rather than using process-oriented models that typically take longer to solve. Process-oriented models are sometimes used to calibrate artificial neural networks, which are then used to more quickly explore and evaluate the range of solution outputs associated with varying inputs.

Examples of such situations where multiple solutions of a model must be obtained include sensitivity or uncertainty analysis, scenario evaluations, risk assessment, optimization, inverse modeling to obtain parameter values given the values of the decision variables, and/or when model runs must be extremely fast, as for rapid assessment and decision support systems, real-time predictions/management/control, and so on. Examples of the use of data-fitting models for model emulation are given in the next several sections.

Genetic algorithms and *genetic programming* are automated, domain-independent methods for evolving solutions to existing models or for producing new models that emulate actual systems, such as rainfall–runoff relationships in a watershed, wastewater removal processes in a treatment plant, or discharges of water from a system of natural lakes, each subject to random inputs. Search methods such as genetic algorithms and genetic programming are inspired by our understanding of biology and natural evolution. They start initially with a number of sets of randomly created values of the unknown variables or a number of black-box models, respectively. The variable values or structure of each of these models are progressively improved over a series of generations. The evolutionary search uses the Darwinian principal of “survival of the fittest” and is patterned after biological operations including crossover (sexual recombination), mutation, gene duplication, and gene deletion.

Artificial neural networks are distributed, adaptive, generally nonlinear networks built from many different processing elements (PEs) (Principe et al. 2000). Each processing element receives inputs from other processing elements and/or from itself. The inputs are scaled by adjustable parameters called weights. The processing elements sum all of these weighted inputs to produce an output that is a nonlinear (static) function of the sum. Learning (calibration) is accomplished by adjusting the weights. The weights are adjusted directly from the training data (data used for calibration) without any assumptions about the data’s statistical distribution or other characteristics (Hagan et al. 1996; Hertz et al. 1991).

The following sections are intended to provide some background helpful to those who may be selecting one among all the available computer codes for implementing a genetic algorithm, genetic program, or artificial neural network.

5.2 Artificial Neural Networks

5.2.1 The Approach

Before the development of digital computers, any information processing necessary for thinking and reasoning was carried out in our brains. Much of it still is. Brain-based information processing continues today (e.g., see Fig. 2.1) and will continue in the future even given our continually improving electronic digital processing capabilities. While recent developments in information technology (IT) have mastered and outperformed much of the information processing one can do just using brain power, IT has not mastered the reasoning power of our brains. Perhaps because of this, some computer scientists have been working on creating information processing devices that mimic the human brain. This has been termed *neurocomputing*. It uses ANNs representing simplified models of the brain. In reality, it is just a more complex type of regression or statistical (black-box) model.

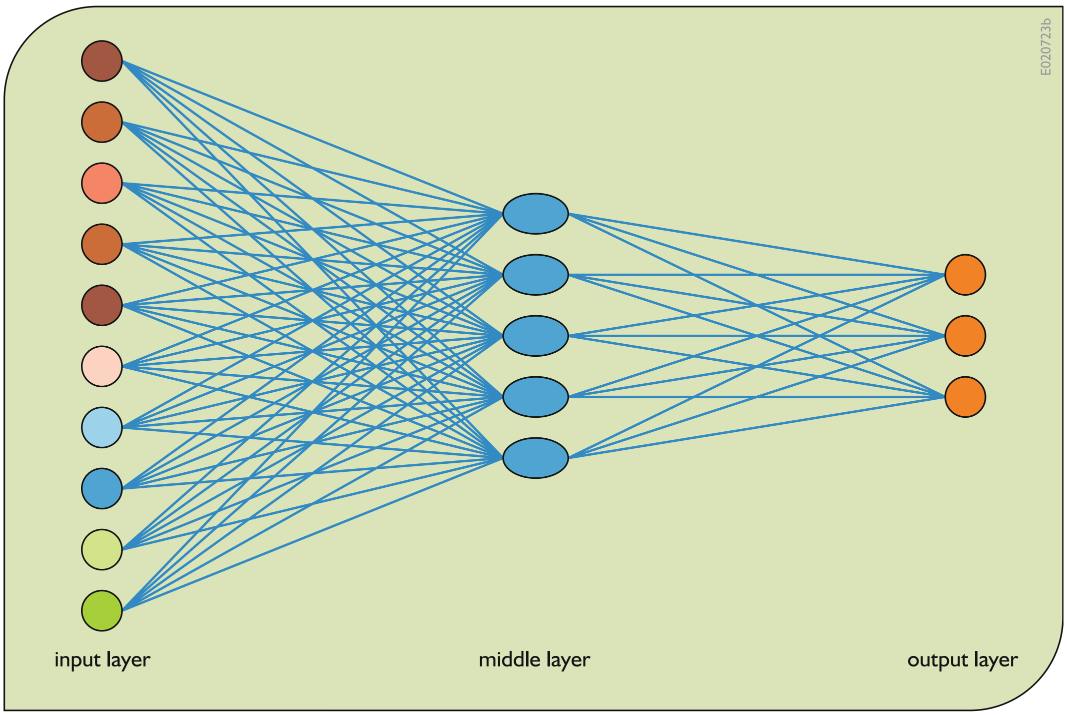


Fig. 5.2 A typical multilayer artificial neural network showing the input layer for ten different inputs, the hidden layer (s), and the output layer having three outputs

An example of the basic structure of an ANN is shown in Fig. 5.2. There are a number of input layer nodes on the left side of the figure and a number of output layer nodes on the right. The middle column(s) of nodes between these input and output nodes are called *hidden layers*. The number of hidden layers and the number of nodes in each layer are two of the design parameters of any ANN. Most applications require networks that contain at least these three types of layers:

- *The input layer* consists of nodes that receive an input from the external environment. These nodes do not perform any transformations upon the inputs but just send their weighted values to the nodes in the immediately adjacent, usually "hidden," layer.
- *The hidden layer(s)* consist(s) of nodes that typically receive the transferred weighted inputs from the input layer or previous hidden layer, perform their transformations on it, and pass the output to the next adjacent layer, which can be another hidden layer or the output layer.
- *The output layer* consists of nodes that receive the hidden layer output and send it to the user.

The ANN shown in Fig. 5.2 has links only between nodes in immediately adjacent layers or columns and is often referred to as a multilayer perceptron (MLP) network, or a feedforward (FF) network. Other architectures of ANNs, which include recurrent neural networks (RNN), self-organizing feature maps (SOFMs), Hopfield networks, radial basis function (RBF) networks, support vector machines (SVMs), and the like, are described in more detail in other publications (for example, Haykin 1999; Hertz et al. 1991).

Essentially, the strength (or weight) of the connection between adjacent nodes is a design parameter of the ANN. The output values O_j that leave a node j on each of its outgoing links are multiplied by a weight, w_j. The input I_k to each node k in each middle and output layer is the sum of each of its weighted inputs, w_jO_j, from all nodes j providing inputs (linked) to node k.

Input value to node k:

$$I_k = \sum w_j O_j \qquad (5.4)$$

Again, the sum in Eq. 5.4 is over all nodes j providing inputs to node k.

At each node k of hidden and output layers, the input I_k is an argument to a linear or nonlinear function $f_k(I_k + \theta_k)$, which converts the input I_k to output O_k. The variable θ_k represents a bias or threshold term that influences the horizontal offset of the function. This transformation can take on a variety of forms. A commonly used transformation is a sigmoid or logistic function as defined in Eq. 5.5 and graphed in Fig. 5.3.

$$O_k = 1/[1 + \exp\{-(I_k + \theta_k)\}] \qquad (5.5)$$

The process of converting inputs to outputs at each hidden layer node is illustrated in Fig. 5.4. The same process also happens at each output layer node.

The design issues in artificial neural networks are complex and are major concerns of ANN developers. The number of nodes in the input as well as in the output layer is usually predetermined from the problem to be solved. The number of nodes in each hidden layer and the number of hidden layers are calibration parameters that can be varied in experiments focused on

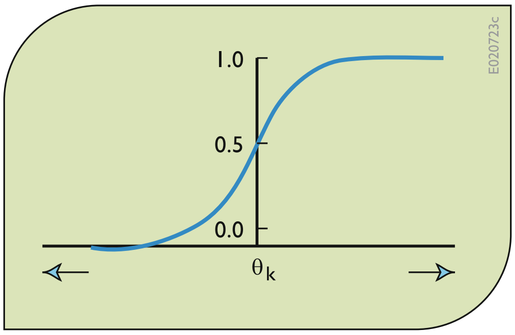


Fig. 5.3 The sigmoid or logistic threshold function with threshold $\boldsymbol{\theta_k}$

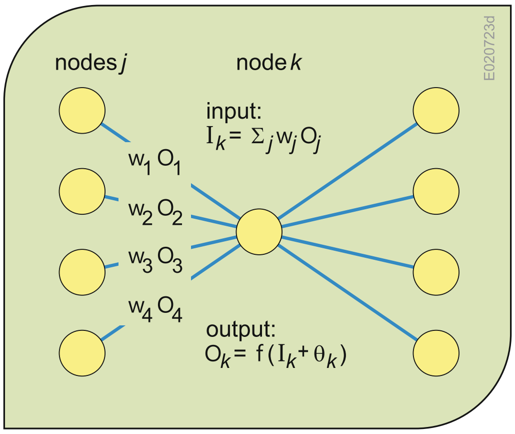


Fig. 5.4 A middle-layer node k converting input values to an output value using a nonlinear function f (such as defined by Eq. 5.5) in a multilayer ANN

getting the best fit of observed and predicted output data based on the same input data. These design decisions, and most importantly the determination of the values of the weights and thresholds of each connection, are "learned" during the "training" of the ANN using predefined (or measured) sets of input and output data.

Some of the present-day ANN packages provide options for building networks. Most provide fixed network layers and nodes. The design of an ANN can have a significant impact on its data-processing capability.

There are two major connection topologies that define how data flows between the input, hidden, and output nodes. These main categories are:

- *Feedforward networks* in which the data flow through the network in one direction from the input layer to the output layer through the hidden layer(s). Each output value is based solely on the current set of inputs. In most networks, the nodes of one layer are fully connected to the nodes in the next layer (as shown in Fig. 5.2); however, this is not a requirement of feedforward networks.
- *Recurrent or feedback networks* in which, as their name suggests, the data flow not only in one direction but in the opposite direction as

well for either a limited or a complete part of the network. In recurrent networks, information about past inputs is fed back into and mixed with inputs through recurrent (feedback) connections. The recurrent types of artificial neural networks are used when the answer is based on current data as well as on prior inputs.

Determining the best values of all the weights is called training the ANN. In a so-called supervised learning mode, the actual output of a neural network is compared to the desired output. Weights, which are usually randomly set to begin with, are then adjusted so that the next iteration will produce a closer match between the desired and the actual output. Various learning methods for weight adjustments try to minimize the differences or errors between observed and computed output data. Training consists of presenting input and output data to the network. These data are often referred to as training data. For each input provided to the network, the corresponding desired output set is provided as well.

The training phase can consume a lot of time. It is considered complete when the artificial neural network reaches a user-defined performance level. At this level the network has achieved the desired statistical accuracy as it produces the required outputs for a given sequence of inputs. When no further learning is judged necessary, the resulting weights are typically fixed for the application.

Once a supervised network performs well on the training data, it is important to see what it can do with data it has not seen before. If a system does not give a reasonable output for this test set, this means that the training period should continue. Indeed, this testing is critical to ensure that the network has learned the general patterns involved within an application and has not simply memorized a given set of data.

Smith (1993) suggests the following procedure for preparing and training an ANN:

1. Design a network.
2. Divide the data set into training, validation, and testing subsets.
3. Train the network on the training data set.
4. Periodically stop the training and measure the error on the validation data set.
5. Save the weights of the network.
6. Repeat Steps 2, 3, and 4 until the error on the validation data set starts increasing. This is the moment where the overfitting has started.
7. Go back to the weights that produced the lowest error on the validation data set, and use these weights for the trained ANN.
8. Test the trained ANN using the testing data set. If it shows good performance, use it. If not, redesign the network and repeat entire procedure from Step 3.

There is a wide selection of available neural network models. The most popular is probably the multilayer feedforward network, which is typically trained with static back propagation. They are easy to use, but they train slowly, and require considerable training data. In fact, the best generalization performance is produced if there are at least 30 times more training samples than network weights (Haykin 1999). Adding local recurrent connections can reduce the required network size, making it less sensitive to noise, but it may get stuck on a solution that is inferior to what can be achieved.

5.2.2 An Example

To illustrate how an ANN might be developed, consider the simple problem of predicting a downstream pollutant concentration based on an upstream concentration and the streamflow. Twelve measurements of the streamflow quantity, velocity, and pollutant concentrations at two sites (an upstream and a downstream site) are available. The travel times between the two measurement sites have been computed and these, plus the pollutant concentrations, are shown in Table 5.1.

Table 5.1 Streamflow travel times and pollutant concentrations

travel time (days)	concentration upstream	concentration downstream
2.0	20.0	6.0
2.0	15.0	4.5
1.5	30.0	12.2
1.0	20.0	11.0
0.5	20.0	14.8
1.0	15.0	8.2
0.5	30.0	22.2
1.5	25.0	10.2
1.5	15.0	6.1
2.0	30.0	9.0
1.0	30.0	16.5
0.5	25.0	18.5

Assume at first that the ANN structure consists of two input nodes, a hidden node, and a single output node. One of the input nodes is for the upstream concentration and the other input node is for the travel time. The single output node represents the downstream concentration expressed as a fraction of the upstream concentration. This is shown in Fig. 5.5.

The model output is the fraction of the upstream concentration that reaches the downstream site. That fraction can be any value from 0 to 1. Hence the sigmoid function (Eq. 5.5) is applied at the middle node and at the output node. Using two or more data sets to train or calibrate this ANN (Fig. 5.5) results in a poor fit as measured by the minimum sum of absolute deviations between calculated and measured concentration data. The more data samples used, the worse the fit. This structure is simply too simple. Hence, another node was added to the middle layer. This ANN is shown in Fig. 5.6.

Using only half the data (six data sets) for training or calibration, the weights obtained provided a near perfect fit. The weights obtained are shown in Table 5.2.

Next the remaining six data sets were applied to the network with weights set to those values shown in Table 5.2. Again the sum of absolute deviations was essentially 0. Similar results were obtained with increasing numbers of data sets.

The values of the weights in Table 5.2 indicate something water quality modelers typically assume, and that is that the fraction of the upstream pollutant concentration that reaches a downstream site is independent of the actual upstream concentration (see Chap. 4). This ANN

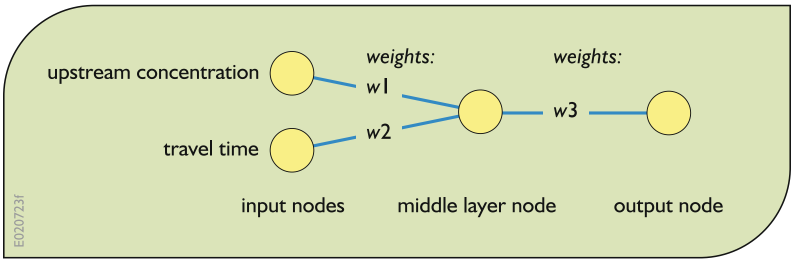


Fig. 5.5 Initial ANN for example problem

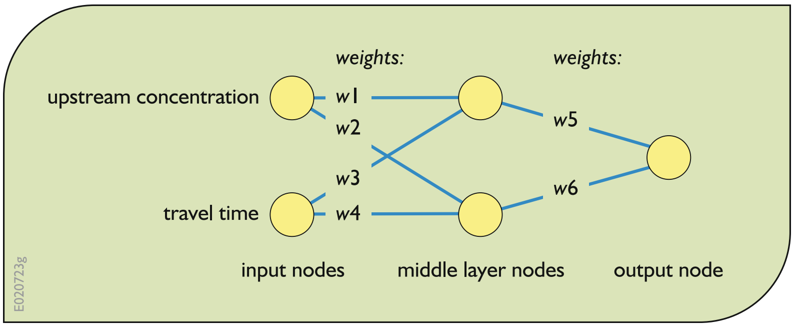
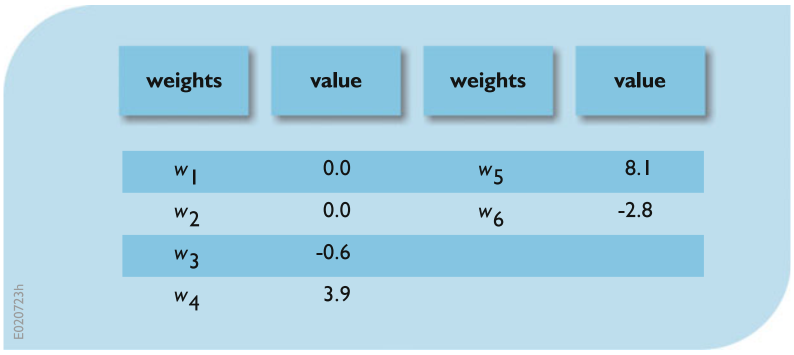


Fig. 5.6 Modified ANN for example problem

Table 5.2 Weights for each link of the ANN shown in Fig. 5.6 based on six data sets from Table 5.1. All bias variables ($\boldsymbol{\theta}_{\mathbf{k}}$ in Eq. 5.5) were 0



weights	value	weights	value
w_1	0.0	w_5	8.1
w_2	0.0	w_6	-2.8
w_3	-0.6		
w_4	3.9		

could have had only one input node, namely that for travel time. This conforms to the typical first-order decay function:

$$\text{Fraction of pollutant concentration downstream per unitconcentration upstream} = \exp\{-k(\text{travel time})\}, \tag{5.6}$$

where the parameter k is the decay rate constant having units of 1/travel time (travel time units^{-1}).

5.3 Evolutionary Algorithms

Evolutionary algorithms (EA) represent a broad spectrum of heuristic approaches for simulating biological evolution in the search for improved "fitness," i.e., the best values of decision variables and parameters based on an objective or fitness function. Evolutionary algorithms are broadly based on the repeated mutation and recombination and selection: in each generation (iteration) to define new individuals (candidate solutions). These are generated by variation, usually in a stochastic way, and then some individuals are selected for the next generation based on their relative fitness or objection function value. Over the generation sequence, individuals with increasingly better fitness values are generated (Simon 2013).

Primary examples include genetic algorithms (Holland 1975), evolutionary strategies (Rechenberg 1973; Schwefel 1981), evolutionary programming (Fogel et al. 1966), and genetic programming (Koza 1992). These methods are comprised of algorithms that operate using a population of alternative solutions or designs, each represented by a potential decision vector. They rely on randomized operators that simulate mutation and recombination to create new individuals, i.e., solutions, who then compete to survive via the selection process, which operates according to a problem-specific fitness or objective function. In some cases this function can be a complex simulation model dependent on the values of its parameters and decision variables derived from the EA. EA popularity is, at least in part, due to their potential to solve nonlinear, nonconvex, multimodal, and discrete problems for which deterministic gradient-based search techniques incur difficulty or fail completely. The growing complexity and scope of environmental and water resources applications has served to expand EAs' capabilities.

Currently, the field of biologically inspired search algorithms mostly include variations of evolutionary algorithms and swarm intelligence algorithms, e.g., ant colony optimization (ACO), particle swarm optimization (PSO), bees algorithm, bacterial foraging optimization (BFO), and so on, many of which have been used to analyze water resources planning and management problems. This is especially true for application of genetic algorithms, arguably among the most popular of the several types of EAs. EAs are flexible tools that can be applied to the solution of a wide variety of complex water resources problems. Nicklow et al. (2010) provides a comprehensive review of state-of-the-art methods and their applications in the field of water resources planning and management. EAs have been successfully applied to the study of water distribution systems, urban drainage and sewer systems, water supply and wastewater treatment, hydrologic and fluvial modeling, groundwater systems, and parameter identification, to name a few. Nicklow et al. also identify major challenges and opportunities for the future, including a call to address larger scale problems that involve uncertainty and an expanded need for collaboration among multiple stakeholders and disciplines. Evolutionary computation methods will surely continue to evolve in the future as analysts encounter increased problem complexities and uncertainty and as the societal pressure for more innovative and efficient solutions rises.

5.3.1 Genetic Algorithms

Genetic algorithms are randomized general-purpose search techniques used for finding the best values of the parameters or decision variables of existing models. It is not a model-building tool like genetic programming. Genetic algorithms and their variations are based on the mechanisms of natural selection (Goldberg 1989). Unlike conventional optimization search approaches based on gradients, genetic algorithms work on populations of possible solutions, attempting to find a solution set that either maximizes or minimizes the value of a function of those parameters and decision variables. This function is called an objective function. Some populations of solutions may improve the value of the objective function, others may not. The ones that improve its value play a greater role in the generation of new populations of solutions than those that do not. This process continues until no significant improvement in model output is apparent. Just how good or "fit" a particular population of parameter and decision variable values is must be evaluated using a model of the

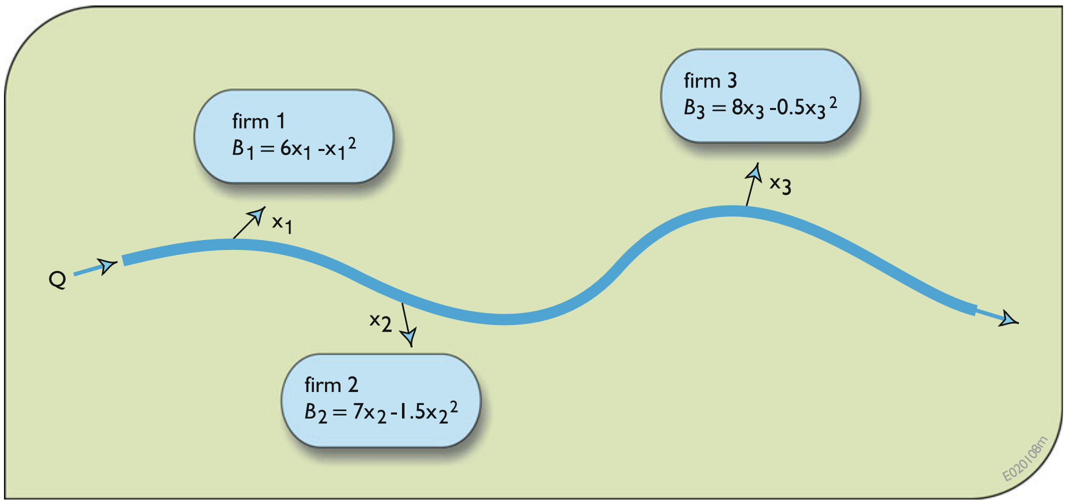


Fig. 5.7 Water allocation to three users from a stream having a flow of ***Q***

system that contains these parameters and decision variables. This system model is separated from the GA model. This model separation makes GA applicable for estimating the best parameter and decision variable values of a wide variety of simulation models used for planning, design, operation, and management.

Each individual solution set of a GA model contains the values of all the parameters or variables whose best values are being sought. These solutions are expressed as strings of values. For example, if the values of three variables *x, y,* and *z* are to be obtained, these variables are arranged into a string, *xyz*. Assuming each variable is expressed using three digits, then the string 056004876 would represent $x = 56$, $y = 4$, and $z = 876$. These strings are called *chromosomes*. A chromosome is an array of numbers. The numbers of the chromosome are called *genes*. Pairs of chromosomes from two parents join together and produce offspring, who in turn inherit some of the genes of the parents. Altered genes may result in improved values of the objective function. These genes will tend to survive from generation to generation, while those that are inferior will tend to die and not reappear in future population sets.

Chromosomes are usually represented by strings of binary numbers. While much of the literature on genetic algorithms focuses on the use of binary numbers, numbers of any base may be used.

To illustrate the main features of genetic algorithms, consider the problem of finding the best allocations of water to the three water-consuming firms shown in Fig. 5.7. Assume only integer solutions are to be considered. The maximum allocation, x_i, to any single user *i* cannot exceed 5, and the sum of all allocations cannot exceed the value of Q, say 6.

$$0 \leq x_i \leq 5 \quad \text{for} \quad i = 1, 2, \text{ and } 3. \tag{5.7}$$

$$x_1 + x_2 + x_3 \leq 6 \tag{5.8}$$

The objective is to find the values of each allocation that maximizes the total benefits, $B(X)$, while satisfying (5.7) and (5.8).

$$\begin{aligned}\text{Maximize } B(X) = {} & (6x_1 - x_1^2) + (7x_2 - 1.5x_2^2) \\ & + (8x_3 - 0.5x_3^2)\end{aligned} \tag{5.9}$$

A population of possible feasible solutions is generated randomly. The best size of the sample solution population—the number of solutions being considered—is usually determined by trial and error.

Using numbers to the base 10, a sample individual solution (chromosome) could be 312, representing the allocations $x_1 = 3$, $x_2 = 1$, and $x_3 = 2$. Another individual solution, picked at random, might be 101. These two individuals or chromosomes, each containing three genes, can pair up and have two children.

The genes of the children are determined by crossover and mutation operations. These pairing, crossover and mutation operations are random. Suppose a crossover is to be performed on the pair of strings, 312 and 101. Crossover involves splitting the two solution strings into two parts, each string at the same place. Assume the location of the split was randomly determined to be after the first digit,

$$\begin{array}{c} 3\,|\,1\,2 \\ 1\,|\,0\,1 \end{array}$$

Crossover usually involves switching one part of one string with the corresponding part of the other string. After a crossover, the two new individuals are 301 and 112.

Another crossover approach is to determine for each corresponding pair of genes whether or not they will be exchanged. This would be based on some preset probability. For example, suppose the probability of a crossover was set at 0.30. Thus, an exchange of each corresponding pair of genes in a string or chromosome has a 30% chance of being exchanged. Assume as the result of this "uniform" crossover procedure, only the middle gene in the pair of strings 312 and 101 is exchanged. This would result in 302 and 111. The literature on genetic algorithms describes many crossover methods for both binary as well as base 10 numbers. The interesting aspect of GA approaches is that they can be, and are, modified in many ways to suit the analyst in the search for the best solution set.

Next consider mutation. Random mutation operations can apply to each gene in each string. Mutation involves changing the value of the gene being mutated. If these strings contain binary numbers, a 1 would be changed to 0, and a 0 would be changed to 1. If numbers to the base 10 are used as they are here, mutation processes have to be defined. Any reasonable mutation scheme can be defined. For example, suppose the mutation of a base 10 number reduces it by 1, unless the resulting number is infeasible. Hence in this example, a mutation could be defined such that if the current value of the gene being mutated (reduced) is 0, then the new number is 5. Suppose the middle digit 1 of the second new individual, 112, is randomly selected for mutation. Thus, its value changes from 1 to 0. The new string is 102. Mutation could just as well increase any number by 1 or by any other integer value. The probability of a mutation is usually much smaller than that of a crossover.

Suppose these paring, crossover, and mutation operations have been carried out on numerous parent strings representing possible feasible solutions. The result is a new population of individuals (children). Each child's fitness, or objective value, can be determined. Assuming the objective function (or fitness function) is to be maximized, the higher the value the better. Adding up all the objective values associated with each child in the population, and then dividing each child's objective value by this total sum yields a fraction for each child. That fraction is the probability of that child being selected for the new population of possible solutions. The higher the objective value of a child, the higher the probability of its being selected to be a parent in a new population.

In this example, the objective is to maximize the total benefit derived from the allocation of water, Eq. 5.9. Referring to Eq. 5.9, the string 301 has a total benefit of 16.5. The string 102 has a total benefit of 19.0. Considering just these two children, the sum of these two individual benefits is 35.5. Thus the child (string) 301 has a probability of 16.5/35.5 = 0.47 of being selected for the new population, and the other child (string 102) has a probability of 19/35.5 = 0.53 of being selected. Drawing from a uniform distribution of numbers ranging from 0 to 1, if a random number is in the range 0–0.47, then the string 301 would be selected. If the random number exceeds 0.47, then the string 102 would be selected. Clearly in a more realistic example the new population size should be much greater than two, and indeed it typically involves hundreds of strings.

This selection or reproduction mechanism tends to transfer to the next generation the better (more fit) individuals of the current generation. The higher the “fitness” (i.e., the objective value) of an individual—in other words, the larger the relative contribution to the sum of objective function values of the entire population of individual solutions—the greater will be the chances of that individual string of solution values being selected for the next generation.

Genetic algorithms involve numerous iterations of the operations just described. Each iteration (or generation) produces populations that tend to contain better solutions. The best solution of all populations of solutions should be saved. The genetic algorithm process can end when there is no significant change in the values of the best solution that has been found. In this search process, there is no guarantee this best solution will be the best that could be found, that is, a global optimum.

This general genetic algorithm process just described is illustrated in the flow chart in Fig. 5.8.

5.3.2 Example Iterations

A few iterations with a small population of ten individual solutions for this example water allocation problem can illustrate the basic processes of genetic algorithms. In practice, the population typically includes hundreds of individuals and the process involves hundreds of iterations. It would also likely include some procedures the modeler/programmer may think would help identify the best solution. Here we will keep the process relatively simple.

The genetic algorithm process begins with the random generation of an initial population of feasible solutions, proceeds with the paring of these solution strings, performs random crossover and mutation operations, computes the probability that each resulting child will be selected for the next population, and then randomly generates the new population. This process repeats itself with the new population and continues until there is no significant improvement in the best solution found from all past iterations.

For this example, we will

1. Randomly generate an initial population of strings of allocation variable values, ensuring that each allocation value (gene) is no less than 0 and no greater than 5. In addition, any set of allocations A_1, A_2, and A_3 that sum to more than 6 will be considered infeasible and discarded.
2. Pair individuals and determine if a crossover is to be performed on each pair, assuming the probability of a crossover is 50%. If a crossover is to occur, we will determine where in the string of numbers it will take place, assuming an equal probability of a crossover between any two numbers.
3. Determine if any number in the resulting individual strings is to be mutated, assuming the probability of mutation of any particular number (gene) in any string (chromosome) of numbers as 0.10. For this example, a mutation reduces the value of the number by 1, or if the original number is 0, mutation changes it to 5. After mutation, all strings of allocation values (the genes in the chromosome) that sum to more than 6 are discarded.
4. Using Eq. 5.9, evaluate the “fitness” (total benefits) associated with the allocations represented by each individual string in the population. Record the best individual string of allocation values from this and previous populations.
5. Return to Step 1 above if the change in the best solution and its objective function value is significant; Otherwise terminate the process.

These steps are performed in Table 5.3 for three iterations using a population of 10.

The best solution found so far is 222: that is, $x_1 = 2$, $x_2 = 2$, $x_3 = 2$. This process can and should continue. Once the process has converged on the best solution it can find, it may be prudent

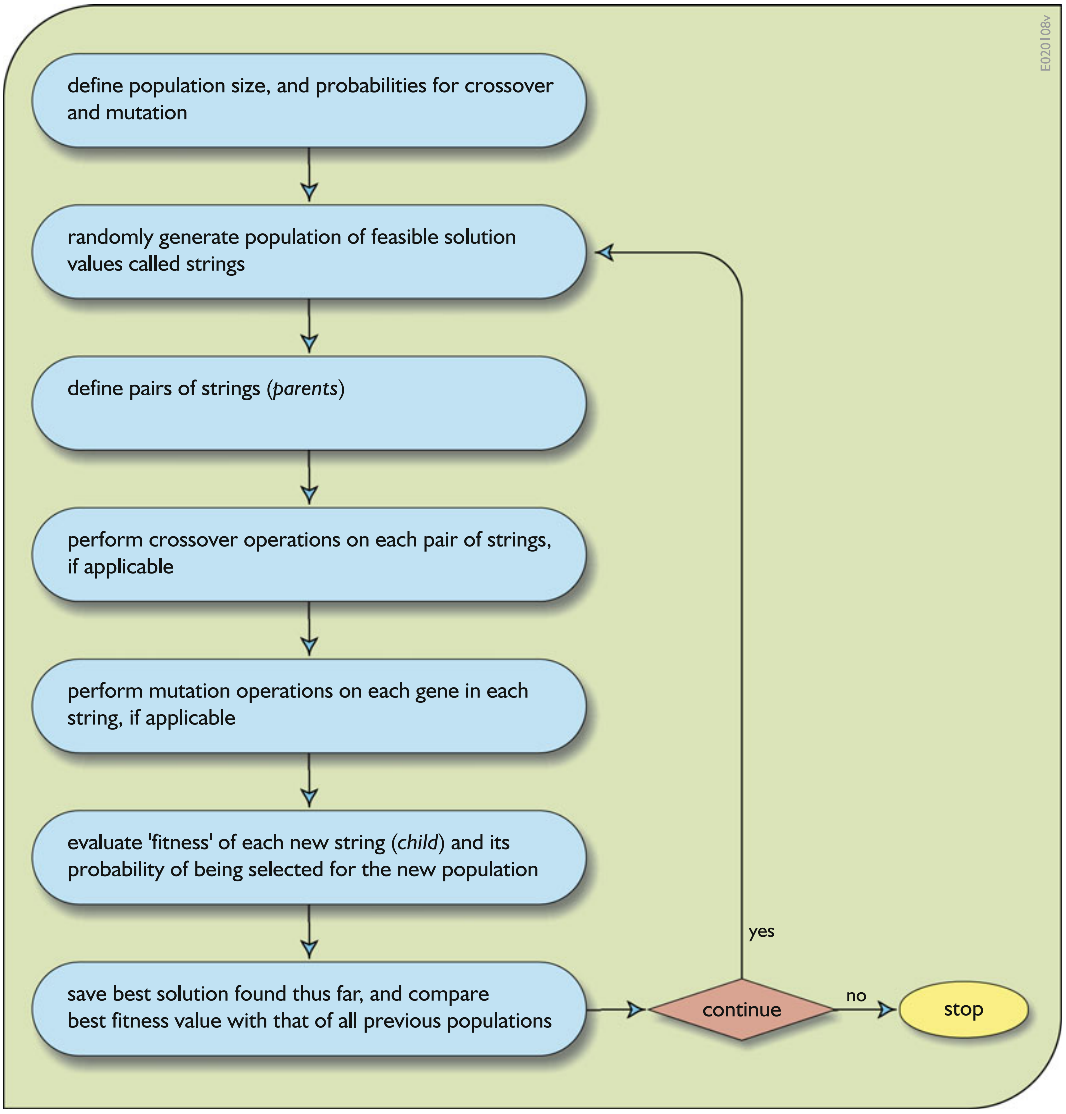

E020108v
define population size, and probabilities for crossover and mutation
randomly generate population of feasible solution values called strings
define pairs of strings (*parents*)
perform crossover operations on each pair of strings, if applicable
perform mutation operations on each gene in each string, if applicable
evaluate 'fitness' of each new string (*child*) and its probability of being selected for the new population
save best solution found thus far, and compare best fitness value with that of all previous populations
continue
yes
no
stop

Fig. 5.8 Flow chart of genetic algorithm procedure

to repeat the process, but this time, change the probabilities of crossover or mutation or let mutation be an increase in the value of a number rather than a decrease. It is easy to modify the procedures used by genetic algorithms in an attempt to derive the best solution in an efficient manner.

Note that the above description of how genetic algorithms work permits the use of any "fitness function" for comparing alternative solutions, and for selecting preferred ones. The search procedure is independent of the particular characteristics of the water resource system being analyzed. This fitness "function" can be a complex groundwater quality model, for example, the parameter values of which are being suggested by the outcome of the GA procedure. Thus in such an application, both simulation and optimization procedures are combined and there are no restrictions on the features of either. As might

Table 5.3 Several iterations for solving the allocation problem using genetic algorithms

	population	crossover		mutation	fitness	sel. prob.	cum. prob.	new pop.
first iteration	230	23|0	220	210	13.5	0.09	0.09	230
	220	22|0	230	230	15.5	0.10	0.19	201
	021	021	021	021	15.5	0.10	0.29	211
	201	201	201	201	15.5	0.10	0.39	132
	301	3|01	321	321	(became infeasible)			301
	221	2|21	201	101	12.5	0.08	0.47	132
	301	30|1	301	301	16.5	0.11	0.58	301
	211	21|1	211	211	21.0	0.14	0.72	021
	132	132	132	132*	26.5	0.19	0.91	230
	310	310	310	210	13.5	0.09	1.00	132
	total fitness				150.0			
second iteration	230	230	230	220	16.0	0.08	0.08	221
	201	201	201	201	15.5	0.08	0.16	132
	211	21|1	212	212*	27.5	0.14	0.30	230
	132	13|2	131	121	20.5	0.10	0.40	212
	301	301	301	301	16.5	0.08	0.48	201
	132	132	132	132	26.5	0.14	0.62	132
	301	3|01	001	001	7.5	0.04	0.66	301
	021	0|21	321	221	23.5	0.12	0.78	221
	230	230	230	230	15.5	0.08	0.86	212
	132	132	132	132	26.5	0.14	1.00	001
	total fitness				195.5			
third iteration	221	22|1	222	222*	30.0	0.15	0.15	221
	132	13|2	131	121	20.5	0.10	0.25	222
	230	230	230	230	15.5	0.07	0.32	230
	212	212	212	112	24.5	0.12	0.44	202
	201	20|1	202	202	22.0	0.09	0.53	131
	132	13|2	131	131	20.0	0.11	0.64	222
	301	301	301	201	15.5	0.08	0.72	012
	221	221	221	221	23.5	0.11	0.83	121
	212	2|12	201	201	15.5	0.08	0.91	202
	001	0|01	012	012	19.5	0.09	1.00	121
	total fitness				206.5			

E020820e

be expected, this has opened up a wide variety of planning and management problems that now can be analyzed in the search for effective solutions.

5.3.3 Differential Evolution

Differential evolution (DE) operates through similar computational steps as employed by a standard evolutionary algorithm (EA) such as genetic algorithms. DE is an optimization technique that iteratively modifies a population of candidate solutions to make it converge to an optimum value. However, unlike traditional EAs, DE programs create new-generation population members by adding a weighted difference between two population vectors to a third vector. To illustrate, after initializing multiple candidate solutions with random values, begin an iterative process where for each candidate solution x you produce a trial vector $v = a + (b - c)/2$, where a, b, c are three distinct candidate solutions picked randomly among the population of possible solutions. Next, you randomly swap vector components between x and v to produce v'. At least one component from v must be swapped. Finally, you replace x in your population with v' only if v' is a better candidate (i.e., it improves the value your objective or fitness function). This process is repeated until no better solution can be found. No separate probability distribution need be used for generating the offspring.

Since its inception in 1995, many variants of the basic algorithm have been developed with improved performance. Books and web pages are available that present detailed reviews of the basic concepts of DE and of its major variants, as well as its application to multiobjective, constrained, large-scale, and uncertain optimization problems. Numerous computer software packages are also available for solving problems using DE. For example, see Das and Suganthan (2011), Storn and Price (1997), Price et al. (2006), and Schwefel (1995) to mention a few.

5.3.4 Covariance Matrix Adaptation Evolution Strategy

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is another stochastic, derivative-free method for numerical solution of nonlinear or nonconvex continuous optimization problems. They belong to the class of evolutionary algorithms. Pairwise dependencies between the variables are represented by a covariance matrix. The covariance matrix adaptation (CMA) method updates the covariance matrix in a way that improves the value of the fitness function. Adaptation of the covariance matrix is similar to the approximation of the inverse Hessian matrix in calculus-based optimization. In contrast to most classical methods, fewer assumptions on the nature of the underlying objective function are made. Only the ranking between candidate solutions is exploited for learning the sample distribution and neither derivatives nor even the function values themselves are required by the method (Hansen 2006; Igel et al. 2007).

Some software programs for DE are at (http://www1.icsi.berkeley.edu/~storn/code.html), for CMA-ES at (https://www.lri.fr/~hansen/cmaesintro.html) and for multiobjective EAs at (http://moeaframework.org/).

5.4 Genetic Programming

One of the challenges in computer science is to program computers to perform tasks without telling them how. In other words, how to enable computers to learn to program themselves for solving particular problems? Since the 1950s, computer scientists have tried, with varying degrees of success, to give computers the ability to learn. The name for this field of study is “machine learning” (ML), a phrase used in 1959 by the first person to make a computer perform a serious learning task, Arthur Samuel. Originally, “machine learning” meant the ability of computers to program themselves. That goal has, for many years, proven very difficult. As a consequence, computer scientists have pursued more modest goals. A good present-day definition of machine learning is given by Mitchell (1997), who identifies machine learning as the study of computer algorithms that improve automatically through experience.

Genetic programming (GP) aspires to do just that: to induce a population of computer programs or models (objects that turn inputs to outputs) that improve automatically as they experience the data on which they are trained (Banzhaf et al. 1998). Genetic programming is one of the many machine-learning methods. Within the machine-learning community, it is common to use "genetic programming" as shorthand for any machine-learning system that evolves tree structures (Koza 1992).

While there is no GP today that will automatically generate a model to solve any problem, there are some examples where GP has evolved programs that are better than the best programs written by people to solve a number of difficult engineering problems. Some examples of these human-competitive GP achievements can be seen in Koza et al. (1999), as well as in a longer list on the Internet (www.genetic-programming.com/humancompetitive.html). Since Babovic (1996) introduced the GP paradigm in the field of water engineering, a number of researchers have used the technique to analyze a variety of water management problems.

The main distinctive feature of GP is that it conducts its search for a solution to a given problem by changing model structure rather than by finding better values of model parameters or variables. There is no guarantee, however, that the resulting structure (which could be as simple as regression Eqs. 5.1, 5.2, or 5.3) will give us any insight into the actual workings of the system.

The task of genetic programming is to find at the same time both a suitable functional form of a model and the numerical values of its parameters. To implement GP, the user must define the basic building blocks (mathematical operations and variables) that may be used; the algorithm then tries to build the model using sequences of the specified building blocks.

One of the successful applications of GP in automatic model building is that of symbolic regression. Here GP searches for a mathematical regression expression in symbolic form that best produces the observed output given the associated input. To perform this task GP uses a physical symbol system divided into two sets. The first set contains the symbols for independent variables as well as parameter constants as appropriate. The content of this set is based on the nature of the problem to be solved. The second set contains the basic operators used to form a function. For example, the second set can contain the arithmetic operators (+, −, *, /) and perhaps others such as log, square root, sine, and cosine as well, again based on the perceived degree of complexity of the regression.

To produce new expressions (individuals) GP requires that two "parent" expressions from the previous generation be divided and recombined into two offspring expressions. An example of this is the parse tree for the expression $\boldsymbol{a} + (\boldsymbol{b}/\boldsymbol{c})$ illustrated in Fig. 5.9. The crossover operation simply exchanges a branch of one parent with a branch of the other.

Software programs have been written to implement GP. For example, GPKernel developed by Babovic and Keijzer (2000) at the Danish Hydraulic Institute (DHI) has been used in applications such as: rainfall–runoff modeling (Babovic and Abbott 1997; Drecourt 1999; Liong et al. 2000), sediment transport modeling, salt intrusion in estuaries, and roughness estimation for a flow over a vegetation bed (Babovic and Abbott 1997). More details about GPKernel can be seen in Aguilera (2000).

The challenge in applying genetic programming for model development is not only getting a close fit between observed and predicted outputs, given a set of input data, but also of interpreting the model that is generated to obtain additional understanding of the actual processes taking place. There are also potential problems in creating a dimensionally correct model if the input data are not dimensionless. As a consequence, many applications using

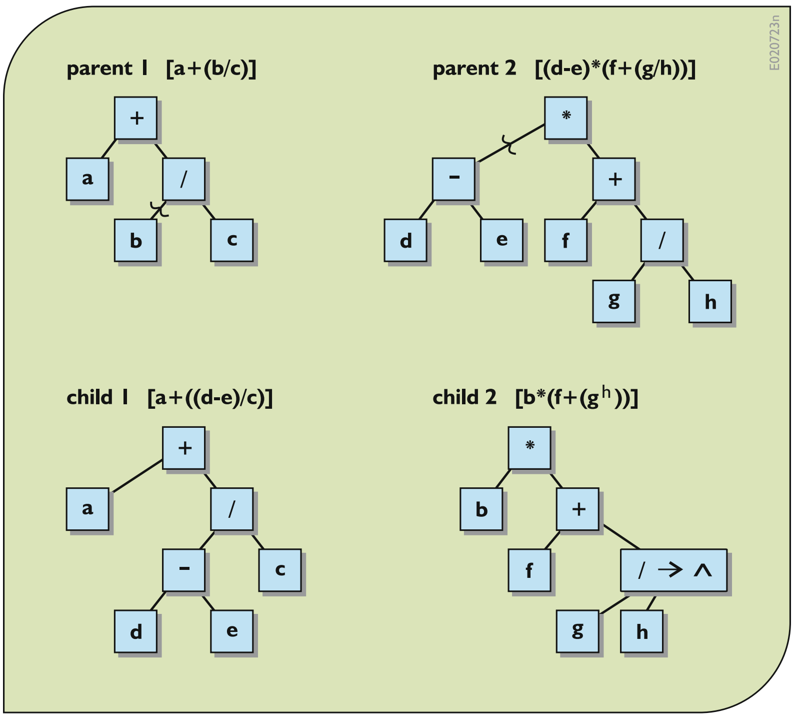


Fig. 5.9 An illustration of a crossover operation and mutation operation for genetic programming

GP seem to require some guidance based on a mix of both physically based and data-driven approaches.

5.5 Qualitative Functions and Modeling

So far the discussion in this chapter has been focused on quantitative data that have numerical values. The precise quantification of many system performance criteria and parameter and decision variables is not always possible, nor is it always necessary. When the values of variables cannot be precisely specified, they are said to be uncertain or fuzzy. If the values are uncertain, probability distributions may be used to quantify them. (The next chapter describes this approach in some detail.) Alternatively, if they are best described by qualitative adjectives, such as dry or wet, hot or cold, expensive or cheap, clean or dirty, and high or low, membership functions indicating the fraction of stakeholders who believe particular quantitative descriptions of parameter or decision variable values are indeed hot, or cold, or clean or dirty, etc., can be used to quantify these qualitative descriptions. Both probability distributions and membership functions of these uncertain or qualitative variables can be included in quantitative models. This section introduces how qualitative variables can be included within models used for the preliminary screening of alternative water resources plans and management policies.

5.5.1 Linguistic Functions

Large, small, pure, polluted, satisfactory, unsatisfactory, sufficient, insufficient, excellent, good, fair, poor, and so on are words often used to describe various attributes or performance measures of water resources systems. These descriptors do not have "crisp," well-defined boundaries that separate them from their opposites. A particular mix of economic and environmental impacts may be *more acceptable* to some and *less acceptable* to others. Plan *A* is *better* than Plan *B*. The quality and temperature of water is *good* for swimming. These qualitative, or so-called "fuzzy," statements convey information despite the imprecision of the italicized adjectives. The next section illustrates how these linguistic qualitative descriptors can be incorporated into optimization models using membership functions.

5.5.2 Membership Functions

Assume a set *A* of real or integer numbers ranging from 18 to 25. Thus $A = [18, 25]$. Any number x is either in or not in the set *A*. The statement "x belongs to A" is either true or false depending on the value of x. The set *A* is called a crisp set. If one is not able to say for certain whether or not any number x is in the set, then the set *A* could be referred to as *fuzzy*. The degree of truth attached to that statement is defined by a membership function. Membership functions range from 0 (completely false) to 1 (completely true).

Consider the statement, "The water temperature should be suitable for swimming." Just what temperatures are suitable will depend on the persons asked. It would be difficult for anyone to define precisely those temperatures that are suitable if it is understood that temperatures outside that range are absolutely not suitable.

A function defining the interval or range of water temperatures suitable for swimming is shown in Fig. 5.10. Such functions may be defined on the basis of the responses of many potential swimmers. There is a zone of imprecision or disagreement at both ends of the range. The form or shape of a function depends on the individual subjective feelings of the "members" or individuals who are asked their opinions. To define this particular function, each individual i could be asked to define his or her comfortable water temperature interval (T_{1i}, T_{2i}). The value associated with any temperature value T equals the number of individuals who place that T within their range (T_{1i}, T_{2i}), divided by the total number of individual opinions obtained. It is the fraction of the total number of individuals that consider the water temperature T suitable for swimming. For this reason such functions are often called membership functions (Figs. 5.10, 5.11 and 5.12).

The assignment of membership values is based on subjective judgments, but such judgments seem to be sufficient for much of human communication.

Now suppose the water temperature applied to a swimming pool where the temperature could be regulated. The hotter the temperature the more it will cost. If we could quantify the economic benefits associated with various temperatures we could perform a benefit–cost analysis by maximizing the net benefits. Alternatively, we could maximize the fraction of people who consider the temperature good for swimming subject to a cost constraint using a membership function such as in Fig. 5.10 in place of an economic benefit function (Chap. 4 discusses ways of doing this.).

Continuing with this example, assume you are asked to provide the desired temperature at a reasonable cost. Just what is reasonable can also be defined by another membership function, but this time the function applies to cost, not temperature. Both the objective and constraint of this

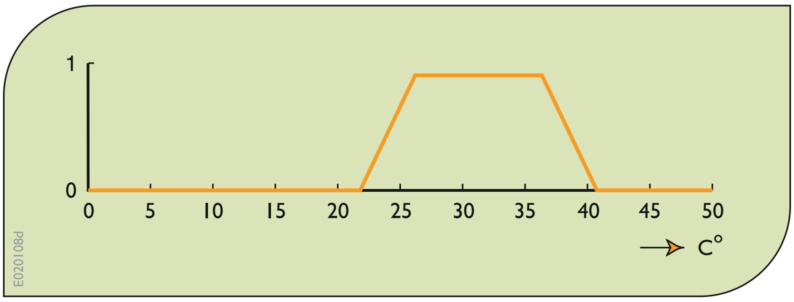


Fig. 5.10 A membership function for suitability of water temperature for swimming

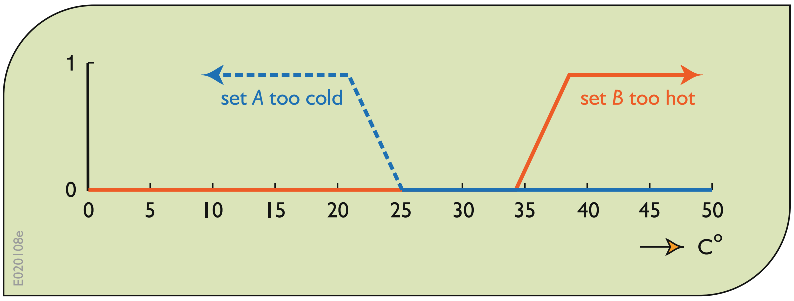


Fig. 5.11 Two membership functions relating to swimming water temperature. Set *A* is the set defining the fraction of all individuals who think the water temperature is too cold, and Set *B* defines the fraction of all individuals who think the water temperature is too hot

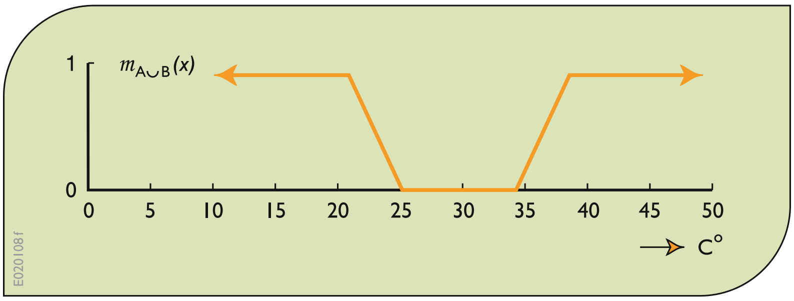


Fig. 5.12 Membership function for water temperatures that are considered too cold or too hot

problem are described qualitatively. In this case one could consider there are in fact two objectives, suitable temperature and acceptable cost. A model that maximizes the minimum value of both membership functions is one approach for finding an acceptable policy for controlling the water temperature at this swimming pool.

5.5.3 Illustrations of Qualitative Modeling

5.5.3.1 Water Allocation

Consider the application of qualitative modeling to the water allocation problem illustrated in Fig. 5.7. Assume, as in the previous uses of this example, the problem is to find the allocations of water to each firm that maximize the total benefits TB(*X*):

$$\text{Maximize TB}(X) = \left(6x_1 - x_1^2\right) + \left(7x_2 - 1.5x_2^2\right) + \left(8x_3 - 0.5x_3^2\right) \tag{5.10}$$

These allocations cannot exceed the amount of water available, *Q*, less any that must remain in the river, *R*. Assuming the available flow for allocations, *Q* − *R*, as 6, the crisp optimization problem is to maximize Eq. (5.10) subject to the resource constraint:

$$x_1 + x_2 + x_3 \leq 6 \tag{5.11}$$

The optimal solution is $x_1 = 1$, $x_2 = 1$, and $x_3 = 4$ as previously obtained in Chap. 4 using any of several different optimization methods. The maximum total benefit, TB(***X***), from Eq. (5.10), equals 34.5.

To create a qualitative equivalent of this crisp model, the objective can be expressed as a membership function of the set of all possible objective values. The higher the objective value the greater the membership function value. Since membership functions range from 0 to 1, the objective needs to be scaled so that it also ranges from 0 to 1.

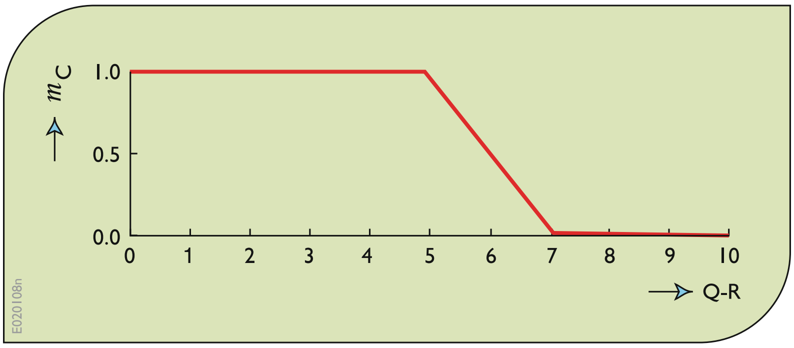


Fig. 5.13 Membership function for "about 6 units more or less"

The highest value of the objective occurs when there is sufficient water to maximize each firm's benefits. This unconstrained solution would result in a total benefit of 49.17 and this happens when $x_1 = 3$, $x_2 = 2.33$, and $x_3 = 8$.

Thus, the objective membership function can be expressed by

$$m(\boldsymbol{X}) = \left[\left(6x_1 - x_1^2\right) + \left(7x_2 - 1.5x_2^2\right) + \left(8x_3 - 0.5x_3^2\right)\right]/49.17 \tag{5.12}$$

It is obvious that the two functions (Eqs. 5.10 and 5.12) are equivalent. However, the goal of maximizing objective function 5.10 is changed to that of maximizing the degree of reaching the objective target. The optimization problem becomes

$$\text{maximize } m(\boldsymbol{X}) = \left[\left(6x_1 - x_1^2\right) + \left(7x_2 - 1.5x_2^2\right) + \left(8x_3 - 0.5x_3^2\right)\right]/49.17 \tag{5.13}$$

subject to

$$x_1 + x_2 + x_3 \le 6 \tag{5.14}$$

The optimal solution of (5.13) and (5.14) results in the same values of each allocation as do Eqs. (5.10) and (5.11). The optimal degree of satisfaction is $m(\boldsymbol{X}) = 0.70$.

Next, assume the total amount of resources available to be allocated is limited to "about 6 units more or less," which is a qualitative constraint. Assume the membership function describing this constraint is defined by Eq. (5.14) and is shown in Fig. 5.13.

$$\begin{aligned}
&m_C(\boldsymbol{X}) = 1 \quad \text{if} \quad x_1 + x_2 + x_3 \le 5\\
&m_C(\boldsymbol{X}) = [7 - (x_1 + x_2 + x_3)]/2 \quad \text{if} \quad 5 \le x_1 + x_2 + x_3 \le 7\\
&m_C(\boldsymbol{X}) = 0 \quad \text{if} \quad x_1 + x_2 + x_3 \ge 7
\end{aligned} \tag{5.15}$$

Let the membership function of (5.12) be called $m_G(\boldsymbol{X})$. The qualitative optimization problem becomes one of maximizing the minimum value of the two membership functions ($m_G(\boldsymbol{X})$, $m_C(\boldsymbol{X})$) subject to their definitions in Eqs. (5.12) and (5.15).

This yields $x_1 = 0.91$, $x_2 = 0.94$, $x_3 = 3.81$, $m_G(X) = m_C(X) = 0.67$, and the total net benefit, Eq. (5.10), is TB($\boldsymbol{X}$) = 33.1. Compare this with the crisp solution of $x_1 = 1$, $x_2 = 1$, $x_3 = 4$, and the total net benefit of 34.5.

5.5.3.2 Qualitative Reservoir Storage and Release Targets

Consider the problem of trying to identify a reservoir storage volume target, T^S, for recreation facilities given a known minimum release target, T^R, and reservoir capacity K. Assume, in this simple example, these known release and unknown storage targets must apply in each of the three seasons in a year. The objective will be to find the highest value of the storage target, T^S, that minimizes the sum of squared deviations from actual storage volumes and releases that are less than the minimum release target.

Table 5.4 The solution to the reservoir optimization problem

variable	value	remarks
T^S	15.6	target storage for each period
S_1	19.4	reservoir storage volume at beginning of period 1
S_2	7.5	reservoir storage volume at beginning of period 2
S_3	20.0	reservoir storage volume at beginning of period 3
R_1	16.9	reservoir release during period 1
R_2	37.5	reservoir release during period 2
R_3	20.6	reservoir release during period 3

Given a sequence of inflows, Q_t, the optimization model is

$$\text{Minimize } D = \sum_t \left[(T^{\mathrm{s}} - S_t)^2 + \mathrm{DR}^2\right] - 0.001T^{\mathrm{S}} \tag{5.16}$$

subject to

$$S_t + Q_t - R_t = S_{t+1} \quad t = 1, 2, 3; \\ \text{if } \; t = 3, t+1 = 1 \tag{5.17}$$

$$S_t \le K \quad t = 1, 2, 3 \tag{5.18}$$

$$R_t \ge T^{\mathrm{R}} - \mathrm{DR}_t \quad t = 1, 2, 3 \tag{5.19}$$

Assume K = 20, T^{R} = 25 and the inflows Q_t are 5, 50, and 20 for periods t = 1, 2, and 3. The optimal solution, yielding an objective value of 184.4, is listed in Table 5.4.

Now consider changing the objective function into maximizing the weighted degrees of "satisfying" the reservoir storage volume and release targets.

$$\text{Maximize} \quad \sum_t (w_{\mathrm{S}} m_{\mathrm{S}t} + w_{\mathrm{R}} m_{\mathrm{R}t}) \tag{5.20}$$

where w_{S} and w_{R} are weights indicating the relative importance of storage volume targets and release targets, respectively. The variables $m_{\mathrm{S}t}$ are the degrees of satisfying the storage volume target in the three periods t, expressed by Eq. (5.21). The variables $m_{\mathrm{R}t}$ are the degrees of satisfying the release target in periods t, expressed by Eq. (5.22).

$$m_{\mathrm{S}t} = S_t/\mathrm{TS} \quad \text{for} \quad S_t \le \mathrm{TS} \quad \text{and} \\ (K - S_t)/(K - \mathrm{TS}) \quad \text{for} \quad \mathrm{TS} \le S_t \tag{5.21}$$

$$m_{\mathrm{R}t} = R_t/\mathrm{TR} \;\; \text{for} \;\; R_t \le \mathrm{TR} \quad \text{and} \\ 1 \;\; \text{for} \;\; R_t > \mathrm{TR} \tag{5.22}$$

Equations (5.21) and (5.22) are shown in Figs. 5.14 and 5.15, respectively.

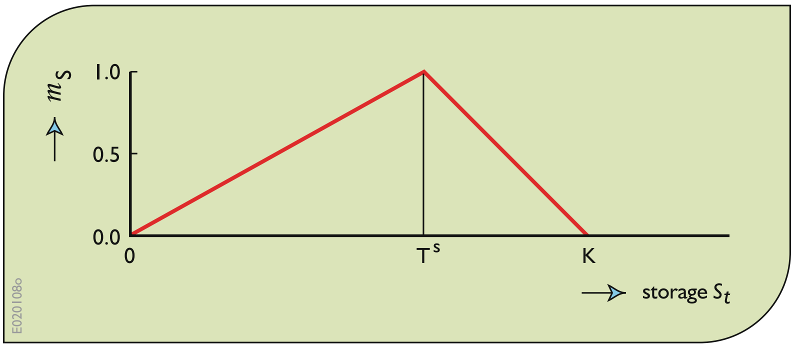


Fig. 5.14 Membership function for storage volumes

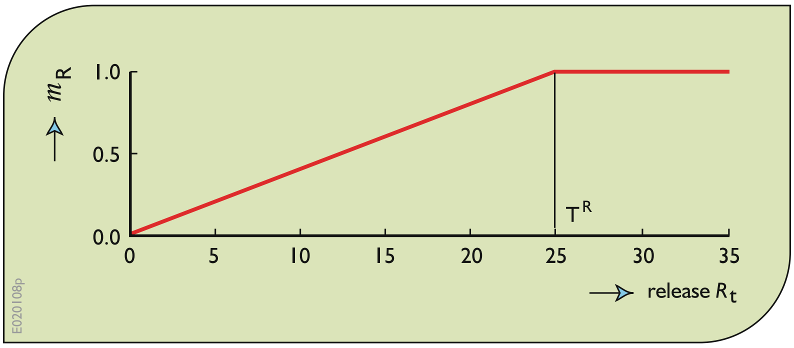


Fig. 5.15 Membership function for releases

Box 5.1. Reservoir model written for solution using LINGO.

```
SETS:
PERIODS /1..3/: I, R, m, ms, mr, s1, s2, ms1, ms2;
NUMBERS /1..4/: S;
ENDSETS
!*** OBJECTIVE ***; max = degree + 0.001*TS;
!Initial conditions; s(1) = s(TN + 1);
!Total degree of satisfaction; degree = @SUM(PERIODS(t): m(t));
!Weighted degree in period t; @FOR (PERIODS(t):
m(t) = ws*ms(t) + wr*mr(t);
S(t) = s1(t) + s2(t);
s1(t) < TS ;  s2(t) < K - TS  ;
!ms(t) = (s1(t)/TS) - (s2(t)/(K-TS)) =  rewritten in case dividing by 0;
ms1(t)*TS = s1(t);   ms2(t)*(K-TS) = s2(t);  ms(t) = ms1(t) - ms2(t);
  mr(t) < R(t)/TR ;    mr(t) < 1;  S(t+1) = S(t) + I(t) - R(t);     );
DATA:
TN = 3; K = 20; ws = ?; wr = ?;     I = 5, 50, 20;  TR = 25;
ENDDATA
```

E020903d

Table 5.5 Solution of qualitative model for reservoir storage volumes and releases based on objective (5.20)

variable	value	remarks
degree	2.48	total weighted sum membership function values
T^{S}	20.00	target storage volume
S_1	20.00	storage volume at beginning of period 1
S_2	0.00	storage volume at beginning of period 2
S_3	20.00	storage volume at beginning of period 3
R_1	25.00	reservoir release in period 1
R_2	30.00	reservoir release in period 2
R_3	20.00	reservoir release in period 3
M_1	1.00	sum weighted membership values period 1
M_2	0.60	sum weighted membership values period 2
M_3	0.88	sum weighted membership values period 3
M_1^{S}	1.00	storage volume membership value period 1
M_2^{S}	0.00	storage volume membership value period 2
M_3^{S}	1.00	storage volume membership value period 3
M_1^{R}	1.00	reservoir release membership value period 1
M_2^{R}	1.00	reservoir release membership value period 2
M_3^{R}	0.80	reservoir release membership value period 3

This optimization model written for solution using LINGO® is as shown in Box 5.1.

Given weights $w_s = 0.4$ and $w_R = 0.6$, the optimal solution obtained from solving the model shown in Box 5.1 using LINGO® is listed in Table 5.5.

If the objective Eq. 5.20 is changed to one of maximizing the minimum membership function value, the objective becomes

Maximize

$$m_{\min} = \text{maximize minimum}(m_{St}, m_{Rt}) \tag{5.23}$$

To include the objective Eq. 5.23 in the optimization model a common lower bound, m_{min}, is set on each membership function, m_{St} and m_{Rt}, and this variable is maximized. The optimal solution changes somewhat and is as shown in Table 5.6.

Table 5.6 Optimal solution of reservoir operation model based on objective (5.23)

variable	value	remarks
MMF	0.556	minimum membership function value
T^S	20.00	target storage volume
S_1	20.00	storage volume at beginning of period 1
S_2	11.11	storage volume at beginning of period 2
S_3	20.00	storage volume at beginning of period 3
R_1	13.88	reservoir release in period 1
R_2	41.11	reservoir release in period 2
R_3	20.00	reservoir release in period 3
M_1^S	0.556	storage volume membership value period 1
M_2^S	0.556	storage volume membership value period 2
M_3^S	0.800	storage volume membership value period 3
M_1^R	0.556	reservoir release membership value period 1
M_2^R	1.000	reservoir release membership value period 2
M_3^R	0.556	reservoir release membership value period 3

This solution differs from that shown in Table 5.5 primarily in the values of the membership functions. The target storage volume operating variable value, T^s, stays the same value in this example.

5.5.3.3 Qualitative Water Quality Management Objectives and Constraints

Consider the stream pollution problem illustrated in Fig. 5.12. The stream receives waste, W_i from sources located at sites $i = 1$ and 2. Without some waste treatment at these sites, the pollutant concentrations at sites 2 and 3 will exceed the maximum desired concentration. The problem is to find the fraction of wastewater removal, x_i, at sites $i = 1$ and 2 required to meet the quality standards at sites 2 and 3 at a minimum total cost. The data used for the problem shown in Fig. 5.16 are defined and listed in Table 5.7.

Using the notation defined in Table 5.7, the crisp model for this problem, as discussed in the previous chapter, is

$$\text{Minimize} \quad C_1(X_1) + C_2(X_2) \tag{5.24}$$

subject to

Water quality constraint at site 2:

$$[P_1Q_1 + W_1(1 - X_1)]a_{12}/Q_2 \le P_2^{\max}$$

$$[(32)(10) + 250{,}000(1 - X_1)/86.4]\,0.25/12 \le 20 \text{ which, when simplified, is}: X_1 \ge 0.78 \tag{5.25}$$

Table 5.7 Parameter values selected for the water quality management problem illustrated in Fig. 5.12

	parameter	unit	value	remark
flow	Q_1	m³/s	10	flow just upstream of site 1
	Q_2	m³/s	12	flow just upstream of site 2
	Q_3	m³/s	13	flow at park
waste	W_1	kg/day	250,000	pollutant mass produced at site 1
	W_2	kg/day	80,000	pollutant mass produced at site 2
pollutant conc.	P_1	mg/l	32	concentration just upstream of site 1
	P_2	mg/l	20	maximum allowable concentration upstream of 2
	P_3	mg/l	20	maximum allowable concentration at site 3
decay fraction	a_{12}	--	0.25	fraction of site 1 pollutant mass at site 2
	a_{13}	--	0.15	fraction of site 1 pollutant mass at site 3
	a_{23}	--	0.60	fraction of site 2 pollutant mass at site 3

Water quality constraint at site 3:

$$\{[P_1Q_1 + W_1(1 - X_1)]a_{13} + [W_2(1 - X_2)]a_{23})/Q_3 \le P_3^{\max}$$

$$\{[(32)(10) + 250{,}000(1 - X_1)/86.4]\,0.15 + [80{,}000(1 - X_2)/86.4]0.60\}/13 \le 20 \tag{5.26}$$

which, when simplified, is: $X_1 + 1.28X_2 \ge 1.79$

Restrictions on fractions of waste removal:

$$0 \le X_i \le 1.0 \quad \text{for sites } i = 1 \text{ and } 2 \tag{5.27}$$

For a wide range of reasonable costs, the optimal solution found using linear programming was 0.78 and 0.79, or essentially 80% removal efficiencies at sites 1 and 2. Compare this solution with that of the following qualitative model.

To consider a more qualitative version of this problem, suppose the maximum allowable pollutant concentrations in the stream at sites 2 and 3 were expressed as "about 20 mg/l more or less." Obtaining opinions of individuals of what they consider to be "20 mg/l more or less," a membership function can be defined. Assume it is as shown in Fig. 5.17.

Next, assume that the government environmental agency expects each polluter to install best available technology (BAT) or to carry out best management practices (BMP) regardless of whether or not this is required to meet stream quality standards. Asking experts just what BAT or BMP means with respect to treatment efficiencies could result in a variety of answers. These responses can be used to define membership functions for each of the two firms in this example. Assume these membership functions for both firms are as shown in Fig. 5.18.

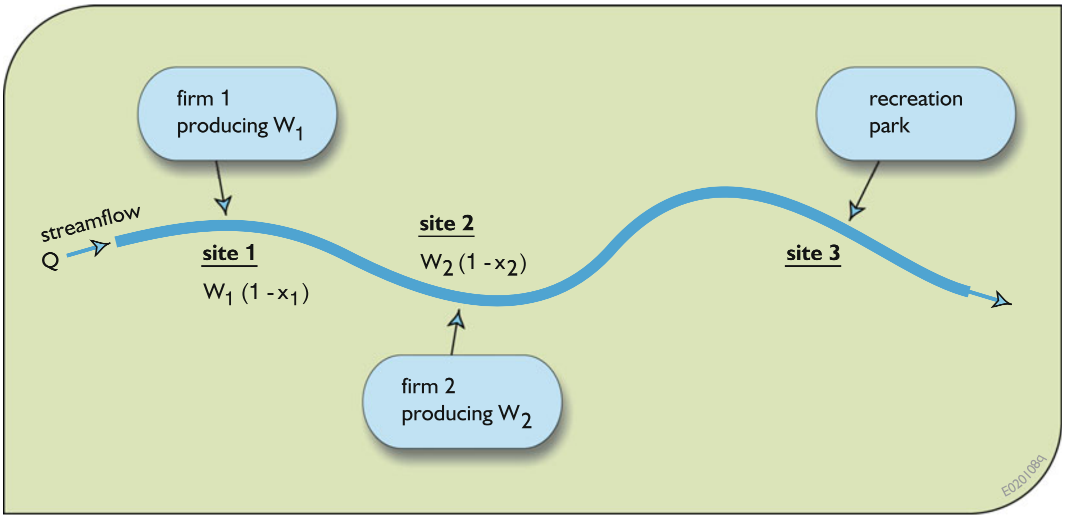
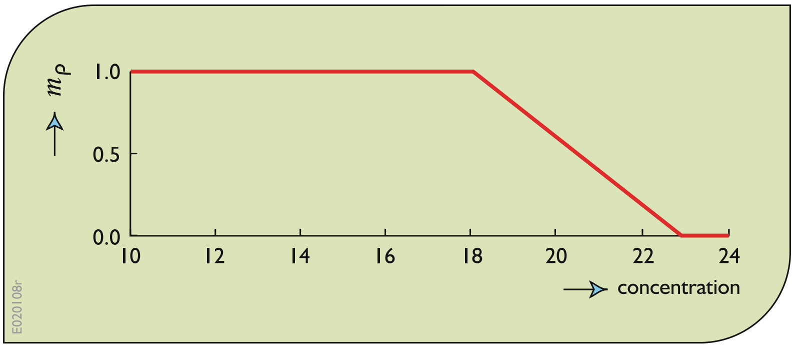


Fig. 5.16 A stream pollution problem of finding the waste removal efficiencies (x_1, x_2) that meet the stream quality standards at least cost

Fig. 5.17 Membership function for "about 20 mg/l more or less"



Finally, assume there is a third concern that has to do with equity. It is expected that no polluter should be required to treat at a much higher efficiency than any other polluter. A membership function defining just what differences are acceptable or equitable could quantify this concern. Assume such a membership function is as shown in Fig. 5.19.

Considering each of these membership functions as objectives, a number of fuzzy optimization models can be defined. One is to find the treatment efficiencies that maximize the minimum value of each of these membership functions.

$$\text{Maximize} \quad m = \max \min\{m_P, m_T, m_E\} \tag{5.28}$$

If we assume that the pollutant concentrations at sites j = 2 and 3 will not exceed 23 mg/l, the

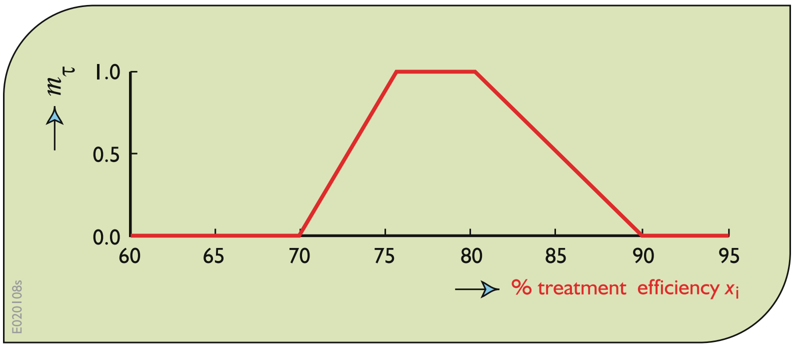


Fig. 5.18 Membership function for best available treatment technology

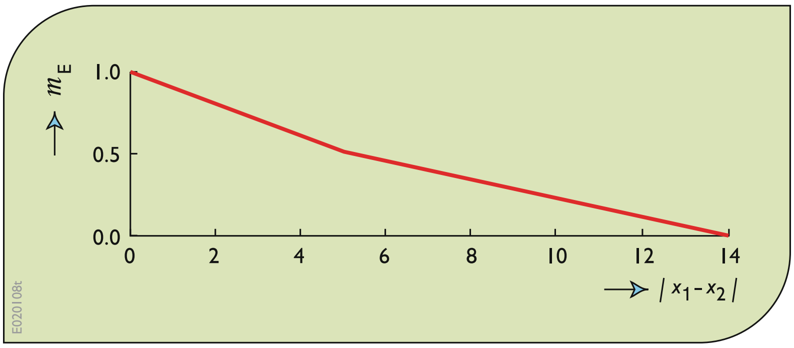


Fig. 5.19 Equity membership function in terms of the absolute difference between the two treatment efficiencies expressed as a percent

pollutant concentration membership functions m_{Pj} are

$$m_{Pj} = 1 - P_{2j}/5 \tag{5.29}$$

The pollutant concentration at each site j is the sum of two components:

$$P_j = P_{1j} + P_{2j} \tag{5.30}$$

where

$$P_{1j} \leq 18 \tag{5.31}$$

$$P_{2j} \leq (23 - 18) \tag{5.32}$$

If we assume the treatment plant efficiencies will be between 70 and 90% at both sites i = 1 and 2, the treatment technology membership functions m_{Ti} are

$$m_{Ti} = (x_{2i}/0.05) - (x_{4i}/0.10) \tag{5.33}$$

and the treatment efficiencies, expressed as fractions, are

$$X_i = 0.70 + x_{2i} + x_{3i} + x_{4i} \tag{5.34}$$

where

$$x_{2i} \leq 0.05 \tag{5.35}$$

$$x_{3i} \leq 0.05 \tag{5.36}$$

$$x_{4i} \leq 0.10 \tag{5.37}$$

Finally, assuming the difference between treatment efficiencies will be no greater than 14, the equity membership function, m_{E}, is

$$m_{\mathrm{E}} = Z - (0.5/0.05)D_1 + 0.5(1 - Z) - (0.5/(0.14 - 0.05))D_2 \qquad (5.38)$$

where

$$D_1 \le 0.05Z \qquad (5.39)$$

$$D_2 \le (0.14 - 0.05)(1 - Z) \qquad (5.40)$$

$$X_1 - X_2 = \mathrm{DP} - \mathrm{DM} \qquad (5.41)$$

$$\mathrm{DP} + \mathrm{DM} = D_1 + 0.05(1 - Z) + D_2 \qquad (5.42)$$

$$Z \text{ is a binary } 0, 1 \text{ variable.} \qquad (5.43)$$

The remainder of the water quality model remains the same: Water quality constraint at site 2:

$$\begin{aligned}&[P_1Q_1 + W_1(1 - X_1)]a_{12}/Q_2 = P_2\\&[(32)(10) + 250{,}000(1 - X_1)/86.4]0.25/12 = P_2\end{aligned} \qquad (5.44)$$

Water quality constraint at site 3:

$$\begin{aligned}&\{[P_1Q_1 + W_1(1 - X_1)]a_{13} + [W_2(1 - X_2)]\,a_{23}\}/Q_3 = P_3\\&\{[(32)(10) + 250{,}000(1 - X_1)/86.4]\,0.15\\&+ [80{,}000(1 - X_2)/86.4]0.60\}/13 = P_3\end{aligned} \qquad (5.45)$$

Restrictions on fractions of waste removal:

$$0 \le X_i \le 1.0 \quad \text{for sites} \quad i = 1 \text{ and } 2. \qquad (5.46)$$

Solving this model using LINGO® yields the results shown in Table 5.8.

This solution confirms the assumptions made when constructing the representations of the membership functions in the model. It is also very similar to the least-cost solution found from solving the crisp linear programming (LP) model containing no membership functions.

Table 5.8 Solution to water quality management model Eqs. 5.28 to 5.46

variable	value	remarks
M	0.93	minimum membership value
X_1	0.81	treatment efficiency at site 1
X_2	0.81	treatment efficiency at site 2
P_2	18.28	pollutant concentration just upstream of site 2
P_3	18.36	pollutant concentration just upstream of site 3
$M^{P}{}_2$	0.94	membership value for pollutant concentration site 2
$M^{P}{}_3$	0.93	membership value for pollutant concentration site 3
$M^{T}{}_1$	0.93	membership value for treatment level site 1
$M^{T}{}_2$	0.93	membership value for treatment level site 2
M^{E}	1.00	membership value for difference in treatment

E020827v

5.6 Conclusions

Most computer-based models used for water resources planning and management are physical, mechanistic, or process-based quantitative models. Builders of such models attempt to incorporate the important physical, biological, chemical, geomorphological, hydrological, and other types of interactions among all system components, as appropriate for the problem being addressed and the system being modeled. This is done in order to be able to predict possible economic, ecologic, environmental, or social impacts that might result from the implementation of some plan or policy. These types of models almost always contain parameters. These need values, and the values of the parameters affect the accuracy of the impact predictions.

This chapter has outlined some data-fitting methods of modeling that do not attempt to model natural, economic, or social processes. These have included ANN and two evolutionary search approaches: genetic algorithms (GA) for estimating the parameter and decision variable values, and genetic programming for finding models that replicate the real system. In some situations, these biologically motivated search methods, which are independent of the particular system model, provide the most practical way to calibrate model parameters.

Fortunately for potential users of GA, GP, and ANN methods, software programs implementing many of these methods are available on the Internet. Applications of such methods to groundwater modeling, sedimentation processes along coasts and in harbors, rainfall runoff prediction, reservoir operation, data classification, and predicting surge water levels for navigation represent only a small sample of what can be found in the current literature.

Not all data are quantitative. In many cases objectives and constraints are expressed as qualitative expressions. Optimization models incorporating such expressions or functions are sometimes appropriate when only qualitative statements apply to a particular water management problem or issue. This chapter concludes by showing how this can be done using some simple example problems associated with water allocations, reservoir operation, and pollution control.

The information presented in this chapter serves only as an introduction. Those interested in more detailed and complete explanations and applications may refer to any of the additional references listed in the next section.

References

Aguilera, D. R. (2000). *Genetic programming with GPKER-NEL*. DHI, Copenhagen: Unpublished.

Babovic, V. (1996). Emergence, evolution, intelligence: Hydroinformatics. Delft: Delft University of Technology.

Babovic, V., & Abbott, M. B. (1997). The evolution of equations from hydraulic data. Part II: Applications. *Journal of Hydraulic Research, 35*(3), 411–427.

Babovic, V., & Keijzer, M. (2000). Genetic programming as a model induction engine. *Journal of Hydroinformatics, 2*(1), 35–60.

Banzhaf, W., Nordin, P., Keller, R. E., & Frannkone, F. D. (1998). *Genetic programming: An introduction: On the automatic evolution of computer programs and its applications*. San Francisco, California: Morgan Kaufmann Publishers, Inc. and dpunkt—Verlag fur Digitale Technologie GmbH.

Das, S., & Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, *15*(1), 4–31.

Drecourt, J.-P. (1999). *Application of neural networks and genetic programming to rainfall-runoff modeling, D2K Technical Report 0699-1*. Copenhagen: DHI.

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial intelligence through simulated evolution*. Hoboken: Wiley.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Westley: Reading.

Hagan, M. T., Demuth, H. B., & Beale, M. (1996). *Neural network design*. Boston, Massachusetts: PWS.

Hansen, N. (2006). The CMA evolution strategy: A comparing review. In *Towards a new evolutionary computation. Advances on estimation of distribution algorithms* (pp. 1769–1776). Berlin: Springer.

Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2nd ed.). N.J., Prentice-Hall: Upper Saddle River.

Hertz, J., Krogh, A., & Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Reading, Massachusetts: Addison Wesley.

Holland, J. H. (1975). Adaptation in natural and artificial systems. Ann Arbor, Michigan: University of Michigan Press.

Igel, C., Hansen, N., & Roth, S. (2007). Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation, 15*(1), 1–28.

Koza, J. R. (1992). Genetic programming: On the programming of computers by means of natural selection (complex adaptive systems). Cambridge, Massachusetts: MIT Press.

Koza, J. R., Bennett, F. H., III, Andre, D., & Keane, M. A. (1999). *Genetic programming III: Darwinian invention and problem solving*. San Francisco, California: Morgan Kaufmann.

Liong, S.-Y., Khu, S. T., Babovic, V., Havnoe, K., Chan, W. T., & Phoon, K. K. (2000). *Construction of non-linear models in water resources with evolutionary computation*. Singapore: National University of Singapore.

Mitchell, T. M. (1997). *Machine learning*. New York: McGraw-Hill.

Nicklow, J. W., Reed, P. M., Savic, D., Dessalegne, T., Harrell, L., Chan-Hilton, A., et al. (2010). State of the art for genetic algorithms and beyond in water resources planning and management. *Journal of Water Resources Planning and Management, 136*(4), 412–432.

Price, K. V., Storn, R. M., & Lampinen, J. A. (2006). *Differential evolution—A practical approach to global optimization*. Berlin Heidelberg: Springer-Verlag. ISBN 3540209506.

Principe, J. C., Euliano, N. R., & Lefebvre, W. C. (2000). *Neural and adaptive systems: Fundamentals through simulation*. New York: Wiley.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Frommann-Holzboog.

Schwefel, H. P. (1981). *Numerical optimization of computer models*. Hoboken: Wiley. ISBN 10: 471099880 | ISBN-13: 9780471099888.

Schwefel, H. P. (1995). *Evolution and optimum seeking*. Hoboken: Wiley.

See, L., & Openshaw, S. (1999). Applying soft computing approaches to river level forecasting. *Hydrological Sciences Journal, 44*(5), 763–778.

Simon, D. (2013). *Evolutionary optimization algorithms*. Hoboken: Wiley.

Smith, M. (1993). *Neural networks for statistical modeling*. New York: Van Nostrand Reinhold.

Storn, R., & Price, K. (1997). Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. *J Global Optimiz, 11*, 341–359.

Additional References (Further Reading)

Abrahart, R. J., Kneale, P. E., & See, L. M. (Eds.). (2004). *Neural networks for hydrological modeling*. Leiden, The Netherlands: AA Balkema.

Bardossy, A., & Duckstein, L. (1995). Fuzzy rule- based modeling with applications to geoPhysical, biological, and engineering systems. Boca Raton, Florida: CRC Press.

Babovic, V. (1998). A data mining approach to time series modeling and forecasting. In V. Babovic & L. C. Larsen (Eds.), *Proceedings of the Third International Conference on Hydroinformatics (Hydroinformatics '98)* (pp. 847–856). Rotterdam: Balkema.

Babovic, V., & Bojkov, V. H. (2001). *D2K technical report D2K TR 0401-1*. Copenhagen: DHI.

Beale, R., & Jackson, T. (1990). *Neural computing: An introduction*. Bristol: IOP.

Chen, S. Y. (1994). *Theory and application of fuzzy system decision-making*. Dalian, China: Dalian University of Technology Press. (In Chinese.).

Clair, T. A., & Ehrman, J. M. (1998). Using neural networks to assess the influence of changing seasonal climates in modifying discharge, dissolved organic carbon, and nitrogen export in eastern Canadian rivers. *Water Resources Research, 34*(3), 447–455.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems, 2*, 303–314.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*, 359–366.

Hsieh, W. W., & Tang, B. (1998). Applying neural network models to prediction and data analysis in meteorology and oceanography. *Bulletin of the American Meteorological Society, 79*, 1855–1870.

Kindler, J. (1992). Rationalizing water requirements with aid of fuzzy allocation model. *Journal of Water Resources Planning and Management, ASCE, 118*(3), 308–323.

Kundzewicz, W. (Ed.). (1995). *New uncertainty concepts in hydrology and water resources.* Cambridge, UK: Cambridge University Press.

Lange, N. T. G. (1998). Advantages of unit hydrograph derivation by neural networks. In V. Babovic & L. C. Larsen (Eds.), *Hydroinformatics '98* (pp. 783–789). Rotterdam: Balkema.

Lootsma, F. A. (1997). *Fuzzy logic for planning and decision-making*. Boston, Massachusetts: Kluwer Academic.

McKinney, D. C., & Lin, M. D. (1994). Genetic algorithm solution of groundwater management models. *Water Resources Research, 30*(6), 1897–1906.

Minns, A. W. (1996). Extended rainfall-runoff modeling using artificial neural networks. In A. Müller (Ed.), *Hydroinformatics '96* (pp. 207–213). Rotterdam: Balkema.

Minns, A. W., & Hall, M. J. (1996). Artificial neural networks as rainfall-runoff models. *Hydrological Sciences Journal, 41*(3), 399–417.

Olivera, R., & Loucks, D. P. (1997). Operating rules for multi-reservoir operation. *Water Resources Research, 33*(4), 839–852.

Price, R. K., Samedov, J. N., & Solomatine, D. P. (1998). An artificial neural network model of a generalized channel network. In V. Babovic & L. C. Larsen (Eds.), *Hydroinformatics '98* (pp. 813–818). Rotterdam: Balkema.

Proano, C. O., Verwey, A., van den Boogaard, H. F. P., & Minns, A. W. (1998). Emulation of a sewerage system computational model for the statistical processing of large number of simulations. In V. Babovic & L. C. Larsen (Eds.), *Hydroinformatics '98* (Vol. 2, pp. 1145–1152). Rotterdam: Balkema.

Recknagel, F., French, M., Harkonen P., & Yanunaka, K. I. (1997). Artificial neural network approach for modeling and prediction of algal blooms. *Ecological Modeling, 96*, 11–28.

Reed, P. M., Hadka, D., Herman, J. D., Kasprzyk, J. R., & Kollat, J. B. (2013). Evolutionary multiobjective optimization in water resources: The past, present, and future. *Advances in Water Resources, 51*, 438–456, doi:10.1016/j.advwatres.2012.01.005

Sanchez, L., Arroyo, V., Garcia, J., Koev, K., & Revilla, J. (1998). Use of neural networks in design of coastal sewage systems. *Journal of Hydraulic Engineering, 124*(5), 457–464.

Scardi, M. (1996). Artificial neural networks as empirical models for estimating phytoplankton production. *Marine Ecology Progress Series, 139*, 289–299.

Shen, Y., Solomatine, D. P., & van den Boogaard, H. F. P. (1998). Improving performance of chlorophyl concentration time series simulation with artificial neural networks. *Annual Journal of Hydraulic Engineering, JSCE, 42*, 751–756.

Simonovic, S. P. (2009). *Managing water resources: Methods and tools for a systems approach* (pp. 576). Paris; London: UNESCO; Earthscan James & James. ISBN 978-1-84407-554-6.

Solomatine, D. P., & Avila Torres, L. A. (1996). Neural network application of a hydrodynamic model in optimizing reservoir operation. In: A. Müller (Ed.), *Hydroinformatics '96* (pp. 201–206). Rotterdam: Balkema.

Stelling, G. S. (2000). A numerical method for inundation simulations. In: Y. N. Yoon, B. H. Jun, B. H. Seoh, & G. W. Choi (Eds.), *Proceedings of the 4th International Conference on Hydro-Science and Engineering, Seoul*, September 26–26, 2000. Seoul: Korea Water Resources Association.

Terano, T., Asai, K., & Sugeno, M. (1992). *Fuzzy systems theory and its application.* San Diego, California: Academic Press.

Tilmant, A., Vancloster, M., Duckstein, L., & Persoons, E. (2002). Comparison of fuzzy and nonfuzzy optimal reservoir operating policies. *Journal of Water Resources Planning and Management, ASCE, 128*(6), 390–398.

van den Boogaard, H. F. P., Gautam, D. K., & Mynett, A. E. (1998). Auto-regressive neural networks for the modeling of time series. In V. Babovic & L. C. Larsen (Eds.), *Hydroinformatics '98* (pp. 741–748). Rotterdam: Balkema.

van den Boogaard, H. F. P., Ten Brummelhuis, P. G. J., & Mynett, A. E. (2000). On-line data assimilation in auto-regressive neural networks. In *Proceedings of the Hydroinformatics 2000 Conference, The University of Engineering, Iowa, USA*, July 23–27, 2000. Iowa City: University of Iowa.

Van Gent, M. R. A., & Van Den Boogaard, H. F. P. (1998). Neural network modeling of forces on vertical structures. In: *Proceedings of the 27th International Conference on Coastal Engineering,* pp. 2096–109. Reston, Virginia: ASCE press.

Wen, C.-G., & Lee, C.-S. (1998). A neural network approach to multiobjective optimization for water quality management in a river basin. *Water Resources Research, 34*(3), 427–436.

Wüst, J. C. (1995). Current prediction for shipping guidance. In B. Kappen & S. Gielen (Eds.), *Neural networks: Artificial intelligence and industrial applications* (pp. 366-73). *Proceedings of the 3rd Annual SNN Symposium on Neural Networks*, Nijmegen, The Netherlands, September 14–15, 1995. London: Springer-Verlag.

Zhou, H.-C. (2000). *Notes on fuzzy optimization and applications*. Dalian, China: Dalian University of Technology Press.

Zimmermann, H.-J. (1987). *Fuzzy sets, decision-making, and expert systems.* Boston, Massachusetts: Kluwer Academic.

Exercises

5.1 An upstream reservoir serves as a recreation site for swimmers, wind surfers, and boaters. It also serves as a flood storage reservoir in the second of four seasons or time periods in a year. The reservoir's releases can be diverted to an irrigation area. A wetland area further downstream receives the unallocated portion of the reservoir release plus the return flow from the irrigation area. The irrigation return flow contains a salinity concentration that can damage the ecosystem.

(a) Assume there exist recreation lake level targets, irrigation allocation targets, and wetland flow and salinity targets. The challenge is to determine the reservoir releases and irrigation allocations so as to "best" meet these targets. This is the crisp' problem.

Data:
Reservoir storage capacity: 30 mcm;
During period 2 the flood storage capacity is 5 mcm;
Irrigation return flow fraction: 0.3 (i.e., 30% of that diverted for irrigation);
Salinity concentration of reservoir water: 1 ppt;
Salinity concentration of irrigation return flow: 20 ppt;
Reservoir average inflows for four seasons, respectively: 5, 50, 20, 10 mcm;

Targets for part (a):
Target maximum salinity concentration in wetland: 3 ppt;
Target storage target for all four seasons: 20 mcm;
Minimum flow target in wetland in each season, respectively: 10, 20, 15, 15 mcm;
Maximum flow target in wetland in each season, respectively: 20, 30, 25, 25 mcm;
Target irrigation allocations: 0, 20, 15, 5 mcm;

(b) Next create fuzzy membership functions to replace the targets and solve the problem. Assume that the targets used in (a) above are expressed in qualitative terms as membership functions. The membership functions indicate the degree of satisfaction for these targets. Solve for the "best" reservoir release and allocation policy that maximizes the minimum membership function value. Each membership function defines the relative level of satisfaction, where a value of 1 indicates complete stakeholder satisfaction. This is the qualitative problem.

5.2 Develop a flow chart showing how you would apply genetic algorithms to finding the parameters, a_{ij}, of a water quality prediction model, such as the one we have used to find the concentration downstream of an upstream discharge site i. This will be based on observed values of mass inputs, W_i, and concentrations, C_j, and flows, Q_j, at a downstream site j.

$$C_j = \sum_i W_i a_{ij} / Q_j$$

The objective to be used for fitness is to minimize the sum of the differences between the observed C_j and the computed C_j. To convert this to a maximization objective you could use something like the following:

$$\text{Max} 1/(1 + D)$$

where

$$D \geq \left(C_j\text{obs} - C_j\text{calculated}\right)$$
$$D \geq \left(C_j\text{calculated} - C_j\text{obs.}\right)$$

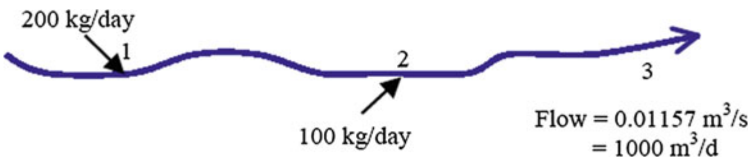
5.3 Use a genetic algorithm program to predict the parameter values asked for in problem 5.2, and then an artificial neural network ANN to obtain a predictor of downstream water quality based on the values of these parameters. You may use the model and data presented in Sect. 5.2 of Chap. 4 if you wish.

5.4. Using a genetic algorithm program to findthe allocations X_i that maximize the total benefits to the three water users i along a stream, whose individual benefits are

$$\begin{aligned} &\text{Use 1:} \quad 6X_1 - X_1^2 \\ &\text{Use 2:} \quad 7X_2 - X_2^2 \\ &\text{Use 3:} \quad 8X_3 - X_3^2 \end{aligned}$$

Assume the available stream flow is some known value (ranging from 0 to 20).
Determine the effect of different genetic algorithm parameter values on the ability to find the best solution.

5.5 Consider the wastewater treatment problem illustrated in the drawing below.



The initial stream concentration just upstream of site 1 is 32. The maximum concentration of the pollutant just upstream of site 2 is 20 mg/l (g/m^3), and at site 3 it is 25 mg/l. Assume the marginal cost per fraction (or percentage) of the waste load removed at site 1 is no less than that cost at site 2, regardless of the amount removed.
Using a suitable genetic algorithm program, solve for the least-cost wastewater treatment at sites 1 and 2 that will satisfy the quality constraints at sites 2 and 3, respectively.
Discuss the sensitivity of the GA parameter values in finding the best solution. You can get the exact solution using LINGO as discussed in Sect. 4.5.3.

5.6 Develop an artificial neural network for flow routing given the following two sets of upstream and downstream flows. Use one set of 5-periods for training (finding the unknown weights and other variables) and the other set for validation of the calculated parameter values (weights and bias constants).
Develop the simplest artificial neural network you can that does an adequate job of prediction.

Time period	Upstream flow	Downstream flow
1	450	366
2	685	593
3	830	755
4	580	636
5	200	325
1	550	439
2	255	304
3	830	678
4	680	679
5	470	534

[These outflows come from the following model, assuming an initial storage in period 1 of 50, the detention storage that will remain in the reach even if the inflows go to 0. For each period t:

$$\text{Outflow}(t) = 1.5(-50 + \text{initial storage}(t) + \text{inflow}(t))^{0.9},$$

where the outflow is the downstream flow and inflow is the upstream flow.]