3 Genetic Algorithms and Their Applications to Water Resources Systems

Deepti Rani[1], *Sharad Kumar Jain*[2], *Dinesh Kumar Srivastava*[1] *and Muthiah Perumal*[1]

[1]Department of Hydrology, IIT Roorkee, Roorkee, India, [2]Water Resources Development and Management Department, IIT Roorkee, Roorkee, India

3.1 Introduction

According to Mitchell (1999), "In the 1950s and the 1960s several computer scientists independently studied evolutionary systems with the idea that evolution could be used as an optimization tool for engineering problems. The idea was to evolve a population of candidate solutions to a given problem, using operators inspired by natural genetic variation and natural selection." All these techniques are collectively referred to as *evolutionary computation (EC) techniques*. EC techniques, also known as heuristic search methods, mostly involve nature-inspired metaheuristic optimization algorithms such as evolutionary algorithms (EAs), comprising genetic algorithms (GAs); evolutionary programming, evolution strategy, and genetic programming; swarm intelligence, comprising ant colony optimization and particle swarm optimization; simulated annealing; and tabu search (Rani and Moreira, 2010).

GAs are a particular class of EA based on the mechanics of natural selection and natural genetics (Goldberg, 1989). GA uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. The method was invented by John Holland (1975) and was later popularized by one of his students, David Goldberg, who solved a difficult problem involving the control of gas-pipeline transmission for his dissertation. His book (Goldberg, 1989) provides GA methodology using both mathematical and computational aspects. He was the first to develop a theoretical basis for GAs through the schema theorem. The work of De Jong (1975) showed the usefulness of the GA for function optimization and made the first concerted effort to find optimized GA parameters. Unlike conventional optimization search methods based on gradients, GAs work on a population of possible solutions, attempting to find

Metaheuristics in Water, Geotechnical and Transport Engineering. DOI: http://dx.doi.org/10.1016/B978-0-12-398296-4.00003-9

a solution set that either maximizes or minimizes the value of a function of those solution values (Loucks and van Beek, 2005).

Like other optimization algorithms, a GA starts by defining decision variables and objective function. It terminates like other optimization algorithms too, by testing for convergence. Nevertheless, it is very different than the others with regard to the steps involved in the process. GAs are typically implemented as a computer simulation. GAs have a main generational process cycle. The GA process begins with a population of chromosomes, which is the set of possible solutions for the decision variables of an optimization problem, and moves toward achieving better solutions through evolution. The decision variables are encoded as binary or real-valued strings (genes) for a given search space. A chromosome is the set of these substrings (genes). The evolution starts from a population of completely random chromosomes and occurs in generations. In each generation, the fitness of the whole population is evaluated, and multiple chromosomes are stochastically selected from the current population (based on their fitness) and modified using genetic operators such as crossover and mutation to form a new population. The new population is then used in the next iteration (generation) of the algorithm (Davis, 1991). Population size depends on the nature of the problem, but typically there are hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, covering the entire range of possible solutions (the search space). This algorithm is repeated sequentially until the desired stopping criterion is achieved.

Advantageous features of GAs in solving large-scale, nonlinear optimization problems are that they can be used with continuous or discrete parameters, require no simplifying assumptions about the problem, and, unlike gradient methods, they do not require computation of derivative information during the optimization (Haupt and Haupt, 2004). Davis (1991) has identified three main advantages of GAs in optimization: "First, they generally find nearly global optima in complex spaces. This is important because the search spaces for our problems are highly multimodal, a property that leads hill-climbing algorithms to get stuck in local optima. Second, genetic algorithms do not require any form of smoothness, that is, they can handle nonlinearity and discontinuity and third, considering their ability to find global optima, genetic algorithms are fast, especially when tuned to the domain on which they are operating." Another advantage of GAs is their inherently parallel nature, i.e., the evaluation of individuals within a population can be conducted simultaneously, as in nature.

Most of the early works in GAs came in the fields of computer science and artificial intelligence. More recently, interest has extended to essentially all branches of science, engineering, economy, and research and development, where search and optimization are of interest. The widespread interest in GAs appears to be due to the success in solving many difficult optimization problems. Today, many applications of GAs in different fields can be found in literature. GAs have been applied to many real-life optimization problems by several researchers. Goldberg and Kuo (1987) developed a study for pipeline optimization by making use of GAs. Soh and Yang (1996) used GAs in combination with fuzzy logic for structural-shaped optimization problems. Feng et al. (1997) applied GAs to the problem of cost-time trade-offs in construction projects. Halhal et al. (1997) applied GAs to a network rehabilitation problem having multiple objectives. A methodology based on GAs has been developed by Li and Love (1998) for optimizing the layout of construction-site-level facilities. Wang and Zheng (2002) studied a job shop scheduling problem with a modified GA. Wei et al. (2005) employed GAs in their research, with the aim of optimization of truss size and shaping with frequency constraints. In water resources, GAs have been applied in many fields, for example, rainfall-runoff modeling (Wang, 1991), water supply network design (Dandy and Engelhardt, 2001; Simpson et al., 1994), and groundwater management problems (Cieniawski et al., 1995; McKinney and Lin, 1994; Ritzel et al., 1994). Davidson and Goulter (1995) used GAs to optimize the layout of rectilinear-branched distribution (natural gas/water) systems.

Theoretical aspects of GAs are already available in many textbooks, and this chapter does not aim to discuss them. It is intended here to give a simple presentation that can be helpful in understanding the basic GA procedure, and one can apply the GA to solve problems related to water resource development and management. The references cited within the text and provided at the end of this chapter should be able to guide the readers to more advanced topics in GAs. Overall, this chapter will provide enough material for anyone curious about GAs and their applications in water resources.

This chapter is organized as follows. Section 3.2 provides an overview of GA and the individual steps involved in a typical GA process. This is followed by Section 3.3 giving a review of applications of GAs in water resource problems, followed by an example of a reservoir operation problem and its solution, describing the steps involved in the GA procedure.

3.2 Genetic Algorithms

There are many publications that give excellent introductions to GAs: see, for example, Holland (1975), Goldberg (1989), Davis (1991), Michalewicz (1999), Mitchell (1999), Deb (2003), and Haupt and Haupt (2004). A GA is a mix of principles behind natural evolution in biology and artificial intelligence in computer science. Therefore, GA terminology uses both natural and artificial terms.

As stated earlier, GAs search for the optimum solution from one set of possible solutions that is an array of decision-variable values. This set of possible solutions is called a *population*. There are several populations in a GA run, and each of these populations is called a *generation*. Generally, at each new generation, better solutions (i.e., decision-variable values) that are closer to the optimum solution as compared to the previous generation are created. In the GA context, the set of possible solutions (array of decision-variable values) is defined as a *chromosome*, while each decision-variable value present in the chromosome is formed by genes.

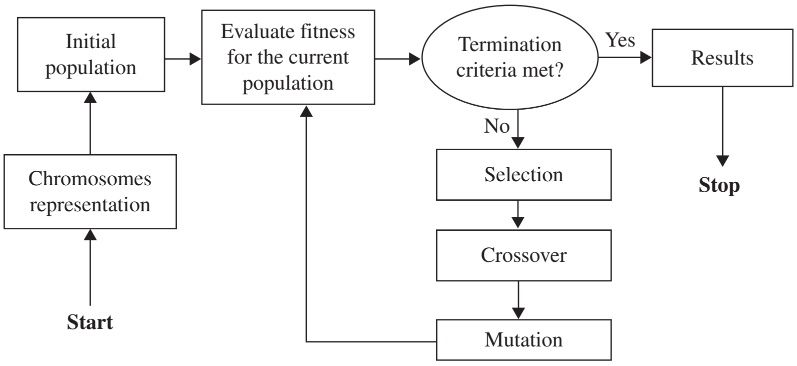
Initial population
Evaluate fitness for the current population
Termination criteria met?
Yes
Results
No
Selection
Stop
Chromosomes representation
Crossover
Start
Mutation

Figure 3.1 Overall GA process.

Population size is the number of chromosomes present in a population. The GA process is briefly described below, and the overall GA process is shown in Figure 3.1.

At the start of the GA optimization, the user has to define the GA operator, such as type of chromosome representation, population size, selection process, types of crossover and mutation, and crossover and mutation probabilities. The initial population is generated according to the selected chromosomal representation at random or using *a priori* knowledge of the search space. For example, given the upper and lower bounds for each decision variable, the chromosomes are created randomly so as to remain within their upper and lower limits. The initial population provides the set of all possible solutions for the first generation, according to the user-defined decision-variable ranges, which have been created randomly. The objective function is used to evaluate each chromosome in the population. Each chromosome in the population has an assigned fitness value, which is used to select the chromosomes from the current population. This process is known as *selection*. Genetic operators, such as crossover and mutation, are performed on the selected chromosomes to create a new set of chromosomes that make the population for the next generation. This algorithm is repeated sequentially until the stopping criterion is achieved. The stopping criterion of a GA is governed either by the number of generations or by the rate of change in the objective function value. Fitness values are expected to improve, indicating the creation of better individuals in new generations. Several generations are considered in the GA process until the user-defined termination criteria is reached.

3.2.1 GA Operators

The GA operators, namely chromosome representation, population size, selection type, and crossover and mutation, control the process of GAs. These operators play an important role in the efficiency of GA optimization in reaching the optimum solution. One of the challenging aspects of using GAs is to choose the optimum GA operator set for the relevant problem.

Representation of Chromosomes

Physical parameters in the search space constituting the phenotypes are encoded into genotypes. The genotype of an individual is the chromosome, and the potential solution to a problem corresponding to the chromosome is the phenotype. In GAs, genetic operators are applied to the genotype to generate better solutions until the optimum is obtained. Then the individual (genotype) representing the optimum solution is decoded to phenotypes. Chromosome representation or encoding is a process of representing the decision-variable values in GAs such that the computer can interact with these values. The decision variables, or phenotypes, in the GA are obtained by applying some mapping from the chromosome representation into the search space. Coding in GA is defined by the type of gene expression, which may be expressed using binary, gray, integers, or real coding. In general, a chromosome (genotype) is presented as

$$(x_1, x_2, \ldots, x_n) \text{ such that } x_1 \in X_1,\ x_2 \in X_2, \ldots, x_n \in X_n \tag{3.1}$$

where, $x_1, x_2, \ldots, x_n$ are bits, integers, real numbers or a mixture of these, and X_1, $X_2, \ldots, X_n$ are the respective search spaces for $x_1, x_2, \ldots, x_n$.

In principle, any character set and coding scheme can be used for chromosome representation. However, the initial GA work of Holland (1975) was done with binary representation, as it was computationally easy. Furthermore, the binary character set can yield the largest number of possible solutions for any given parameter representation, thereby giving more information to guide the genetic search. The GA operators work directly on this representation of the chromosomes to get the optimal solution.

The conventional GA operations and theory were developed on the basis of binary coding, which was used in many applications (Goldberg, 1989). The use of real-valued genes in GAs is claimed by Wright (1991) to offer a number of advantages in numerical function optimization over binary coding. Binary coding and real coding differ mainly in how the crossover and mutation operators are performed in the GA process. There has been growing interest in real-value coding for GAs. In real-value coding, each chromosome is coded as an array of real numbers, with the same length for the decision variable. Gray coding is another type of bit string coding, which uses adjacent variable values where the code occurs as only one binary digit. It was developed to overcome a problem called "Hamming Cliffs," which exists in binary coding and has been used in a number of studies in the water resources field (Dandy et al., 1996).

Binary Coding

The most commonly used representation of chromosomes in the GA is binary coding by using binary numbers 0 and 1. In this coding, each decision variable in

	String 1	String 2
Chromosome	10001	01111

Figure 3.2 Formation of chromosome.

the parameter set is encoded as a binary string, and these are concatenated to form a chromosome. The length of the binary substring (i.e., number of bits) for a variable depends on the size of the search space and the number of decimal places required for accuracy of the decoded variable values (Michalewicz, 1999). If each decision variable is given a string of length L, and there are n such variables, then the chromosome will have a total string length of nL. For example, let there be two decision variables, x_1 and x_2, and let the string length be 5 for each variable. Then the chromosome length is 10, as shown in Figure 3.2.

The search space is divided into 2^L intervals, each having a width equal to $(x_{i,\max} - x_{i,\min})/2^L$ for a binary string of length L, where $x_{i,\max}$ is the upper bound of the decision variable, and $x_{i,\min}$ is the lower bound of the decision variable:

$$d = (x_{i,\max} - x_{i,\min})/2^L \text{ defines the solution accuracy} \tag{3.2}$$

The binary numbers have a base of 2 and use only two characters, 0 and 1. A binary string, therefore, is decoded using Eq. (3.3):

$$N = a_n 2^n + a_{n-1} 2^{n-1} + \cdots + a_1 2^1 + a_0 2^0 \tag{3.3}$$

where

a_i is either 0 or 1 (ith bit in the string),
2^n represents the power of 2 of digit a_i,
n is the number of bits in binary-coded decision variable (i.e., $L - 1$),
N is the decoded integer value of the binary string,

and the corresponding actual value of the variables is obtained using Eq. (3.4):

$$x_i = x_{i,\min} + \frac{x_{i,\max} - x_{i,\min}}{2^L - 1} N \tag{3.4}$$

For this example, let the search space for decision variables x_1 and x_2 range from 0 to 5 and 1 to 10, respectively. For the chromosome shown in Figure 3.2, the decoded value for the substrings and the corresponding value of the decision variables will be as shown in Table 3.1.

Using string length $L = 5$, the entire search space for decision variable x_1 can be divided into 31 intervals of 0.16 width each, as shown in Table 3.2. The solution accuracy may be increased by increasing the length of the string. The lower and upper bounds of the real-value search space (i.e., 0 and 5) can be mapped into binary numbers using Eq. (3.2), and all the other intermediate values (i.e., 0–31) can also be easily expressed in binary numbers using Eq. (3.3). The entire search

Table 3.1 Coding and Decoding in GAs

	Decision Variables	
	x_1	x_2
Chromosomes represented as binary strings assuming string length, $L = 5$	10001	01111
Decoded integer value	$1.2^4 + 0.2^3 + 0.2^2 + 0.2^1 + 0.2^0 = 17$	$0.2^4 + 1.2^3 + 1.2^2 + 1.2^1 + 1.2^0 = 15$
Corresponding value of decision variable with solution accuracy $(x_{\max} - x_{\min})/(2^L - 1)$	2.74	4.84

space for x_1 in binary encoding and decoded real values are given in Table 3.2. Different ranges and accuracies can be considered in GAs through different binary substring lengths for different decision variables. All GA operators are performed on binary strings and once GA optimization is completed, the binary strings can be decoded into real values.

Gray Coding

Gray coding is an ordering of binary character sets such that all adjacent numerical numbers differ by only one bit, whereas in binary coding, adjacent numbers may differ in many bit positions. Gray coding representation has the property that any two points next to each other in the search space differ by one bit only (Haupt and Haupt, 2004). In other words, an increase of one step in the value of the decision variable corresponds to a change of only a single bit. The advantage of gray coding is that random bit flips in mutation are likely to make small changes and therefore result in a smooth mapping between the real search space and the encoded strings. To convert binary coding to gray coding, truth table conversion, as shown in Table 3.3, is followed.

When converting from binary to gray, the first bit of the binary code remains as it is, and the remaining bits follow the truth table conversion, two bits taken sequentially at a time, giving the next bit in gray coding. An example of representation of binary and gray coding of numeric numbers of 1−31 is shown in Table 3.4.

The number of bit positions that differ in two adjacent bit strings of equal length is known as *Hamming distance*. For example, the *Hamming distance* between 01111 and 10000 is 5, since all bit positions differ, and require alteration of 5 bits when converting the number 15 to 16 in binary representation. The Hamming distance associated with certain strings, such as 01111 and 10000, poses difficulty in transition to a neighboring solution in real space, as it requires the alteration of many bits. In gray coding, this distance between any two adjacent binary strings is always 1. Caruana and Schaffer (1988) reported that gray coding can eliminate

Table 3.2 Binary and Real Value Search Space for Decision Variable x_1

Binary encoding	**Decoded value**	**Corresponding real value of x_1, in the search space**
00000	0	0.00
00001	1	0.16
00010	2	0.32
00011	3	0.48
00100	4	0.65
00101	5	0.81
00110	6	0.97
00111	7	1.13
01000	8	1.29
01001	9	1.45
01010	10	1.61
01011	11	1.77
01100	12	1.94
01101	13	2.10
01110	14	2.26
01111	15	2.42
10000	16	2.58
10001	17	2.74
10010	18	2.90
10011	19	3.06
10100	20	3.23
10101	21	3.39
10110	22	3.55
10111	23	3.71
11000	24	3.87
11001	25	4.03
11010	26	4.19
11011	27	4.35
11100	28	4.52
11101	29	4.68
11110	30	4.84
11111	31	5.00

Table 3.3 Truth Table Conversions (B_1 and B_2 are adjacent bits in a binary string)

B_1	**1**	**0**
B_2		
1	0	1
0	1	0

Table 3.4 Representations of Integer Numbers in Binary and Gray Coding

Integers	Binary Coding	Gray Coding
0	00000	00000
1	00001	00001
2	00010	00011
3	00011	00010
4	00100	00110
5	00101	00111
6	00110	00101
7	00111	00100
8	01000	01100
9	01001	01101
10	01010	01111
11	01011	01110
12	01100	01010
13	01101	01011
14	01110	01001
15	01111	01000
16	10000	11000
17	10001	11001
18	10010	11011
19	10011	11010
20	10100	11110
21	10101	11111
22	10110	11101
23	10111	11100
24	11000	10100
25	11001	10101
26	11010	10111
27	11011	10110
28	11100	10010
29	11101	10011
30	11110	10001
31	11111	10000

the hidden bias in binary coding and that the large Hamming distances in the binary representation could result in the search process being deceived or unable to locate the global optimum efficiently. Gray coding has been preferred by several researchers while using GAs in water resource applications (Wardlaw and Sharif, 1999).

Real-Value Coding

For problems with a large number of decision variables, having large search spaces, and requiring a higher degree of precision, binary-coded GAs have performed poorly (Michalewicz, 1999). Wright (1991) claims that the use of real-valued genes in GAs overcomes a number of drawbacks of binary coding. In real coding, each variable is represented as a vector of real numbers with the same length as that of the solution vector. Efficiency of the GA is increased because genotype into phenotype conversion is not required. In addition, less memory is required because efficient floating-point internal computer representations can be used directly; there is no loss in precision due to formation of discreteness to binary or other values; and there is greater freedom to use different genetic operators. Nonetheless, real coding is more applicable and it seems to fit continuous optimization problems better than binary coding. Eshelman and Schaffer (1993) suggested choosing any of these coding mechanisms, whichever is most suitable for the fitness function. Other authors, such as Michalewicz (1999), justify the use of real coding, showing their advantages with respect to the efficiency and precision reached compared to the binary one. Real coding has been the preferred choice for variable representation in most of the applications found in water resources using GA.

Another form of real number representation is integer coding. In integer coding, the chromosomes are composed of integer values rather than real numbers. The only difference between real coding and integer coding is in the operation of mutation.

Population Size

The population size is the number of chromosomes in the population. The size of a population depends on the nature of the problem, but typically a population contains hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, covering the entire search space. Given upper and lower bounds for each chromosome (decision variable), chromosomes are created randomly so as to remain within the given limits. The principle is to maintain a population of chromosomes, which represents candidate solutions to the problem that evolve over time through a process of competition and controlled variation. Each chromosome in the population has an assigned fitness to determine which chromosomes are used to form new ones in the competition process, which is called *selection*. The new ones are created using genetic operators such as crossover and mutation.

Larger population sizes increase the amount of variation present in the population but require more fitness evaluations (Goldberg, 1989). Therefore, when the population size is too large, users tend to reduce the number of generations in order to reduce the computing effort, since the computing effort depends on the multiple of population size and number of generations. Reduction in the number of generations reduces the overall solution quality. On the other hand, a small population size can cause the GAs to converge prematurely to a suboptimal solution. Goldberg (1989) reported that a population size ranging from 30 to 200 was the general choice of many GA researchers. Furthermore, Goldberg pointed out that the population size was both application dependent and related to string length. For longer chromosomes and challenging optimization problems, larger population sizes were needed to maintain diversity because it allowed better exploration.

Selection

Selection is the survival of the fittest within the GA. The selection process determines which chromosomes are preferred for generating the next population, according to their fitness values in the current population. The key notion in selection is to give a higher priority or preference to better individuals. During each generation, a proportion of the existing population is selected to breed a new generation; therefore, the selection operator is also known as the *reproduction operator*. All chromosomes in the population, or in a proportion of the existing population, can undergo the selection process using a selection method. This percentage is known as the *generation gap*, which is defined by the user as an input in GAs. The selection process emphasizes to copy the chromosomes with better fitness for the next generation than those with lower fitness values. This may lose population diversity or the variation present in the population and could lead to a premature convergence. Therefore, the method used in the selection process should be able to maintain the balance between selection pressure and population diversity. There are several selection techniques available for GA optimization. Proportional selection, rank selection, and tournament selection (Goldberg and Deb, 1991) are among the most commonly used selection methods. These are briefly discussed below.

Proportional Selection Method

The proportional selection method selects chromosomes for reproduction of the next generation with a probability proportional to the fitness of the chromosomes. In this method, the probability (P) of selecting a chromosome for reproduction can be expressed as

$$P = \frac{\mathrm{ft}_i}{\sum_{i=1}^{N} \mathrm{ft}_i} \tag{3.5}$$

where ft_i is the fitness value of the ith chromosome in the current population of size N, and $\sum_{i=1}^{N} \mathrm{ft}_i$ is the total fitness, which is the sum of fitness values of all chromosomes in the current population.

This method provides noninteger copies of chromosomes for reproduction. Therefore, various methods have been suggested to select the integer number of copies of selected chromosomes for the next generation, including Monte Carlo, roulette wheel, and stochastic universal selection. The roulette wheel selection method is discussed next.

Roulette Wheel Selection The most common selection method is roulette wheel selection. Goldberg (1989) reported that it is also the simplest method. The basic implementation of the roulette wheel selection method assigns each chromosome a “slice” of the wheel, with the size of the slice proportional to the fitness value of the chromosome. In other words, the fitter a member is, the bigger slice of the wheel it gets. To select a chromosome for selection, the roulette wheel is “spun,” and the chromosome corresponding to the slice at the point where the wheel stops is grabbed as the one to survive in the offspring generation.

The main steps for the roulette wheel selection algorithm may be generalized as follows:

1. The fitness of each chromosome, ft_i, and their sum $\sum_{i=1}^{N}\mathrm{ft}_i$ are calculated, where the population size is N.
2. A real random number, rand ( ), within the range [0,1] is generated, and s is set to be equal to the multiplication of this random number by the sum of the fitness values, $s = \mathrm{rand}\,(\cdot) \times \sum_{i=1}^{N} \mathrm{ft}_i$.
3. A minimal k is determined such that $s \le \sum_{i=1}^{k} \mathrm{ft}_i$, and the kth chromosome is selected for the next generation.
4. Steps 2 and 3 are repeated until the number of selected chromosomes becomes equal to the population size, N.

Tournament Selection

Another selection technique is tournament selection, where randomly selected pairs of chromosomes "fight" to become parents in the mating pool through their fitness function value (Goldberg, 1989). Tournament selection runs a "tournament" among two or more chromosomes chosen at random from the population, and selects the winner in accordance with their fitness values, such that the one with the best fitness is selected for crossover. This process is continued until the required number of chromosomes is selected for the next generation. Selection pressure can be easily adjusted by changing the tournament size. If the tournament size is larger, weak chromosomes have less chance to be selected. In general, in tournament selection, N chromosomes are selected at random and the fittest is selected. The most common type of tournament selection is binary tournament selection, where just two chromosomes are selected.

Rank Selection

In the rank-selection approach, each population is sorted in order of fitness, assigning a numerical rank to each chromosome based on fitness, and the chromosomes are selected based on this ranking rather than the fitness value using the proportionate selection operator. The advantage of this method is that it can prevent very fit chromosomes from gaining dominance early at the expense of less fit ones, thereby increasing the population's genetic diversity (Goldberg and Deb, 1991).

Roulette wheel selection, tournament selection, and rank selection are considered to be the most common and popular selection techniques and have been used frequently in many studies. However, there are many other selection techniques, namely, elitist selection, generational selection, steady-state selection, and hierarchical selection. These techniques may be used independently or in combination. Brief introduction of those selection techniques are given next. A detailed review of selection techniques used in GAs is presented by Shivraj and Ravichandran (2011).

Elitist Selection

The fittest chromosomes from each generation are selected for the next generation, a process known as *elitism*. Most GAs do not use pure elitism, but instead use a modified form where a single chromosome or a few of the best chromosomes from each generation are copied into the next generation. Elitism can be combined with any other selection technique.

Generational Selection
The offspring of the chromosomes selected from each generation become the entire next generation. No chromosomes are retained between generations.

Steady-State Selection
The offspring of the chromosomes selected from each generation go back into the previous generation and replaces some of the less fit members. This process helps to keep some chromosomes between generations.

Hierarchical Selection
Chromosomes go through multiple rounds of selection each generation. Lower-level evaluations are faster and less discriminating, while those that survive to higher levels are evaluated more rigorously. The advantage of this method is that it reduces overall computation time by using faster, less selective evaluation to weed out the majority of chromosomes that show little or no promise, and subjecting only those who survive this initial test to more rigorous and more computationally expensive fitness evaluation.

Crossover

The crossover operator is used to create new chromosomes for the next generation by combining randomly two selected chromosomes from the current generation. Crossover helps to transfer the information between successful candidates—chromosomes can benefit from what others have learned, and schemata can be mixed and combined, with the potential to produce an offspring that has the strengths of both its parents and the weaknesses of neither. However, some algorithms use an elitist selection strategy, which ensures that the fittest chromosome from one generation is propagated into the next generation without any disturbance. The crossover rate is the probability that crossover reproduction will be performed and is an input to GAs. For example, a crossover rate of 0.9 means that 90% of the population is undergoing the crossover operation. A higher crossover rate encourages better mixing of the chromosomes.

There are several crossover methods available for reproducing the next generation. In general, crossover methods can be classified into two groups depending on the chromosomes representation (i.e., binary coding or real-value coding). A number of crossover methods are discussed by Herrera et al. (1998) for binary coding and real coding. The choice of crossover method primarily depends on the application.

Crossover Operators for Binary Coding
In bit string coding, crossover is performed by simply swapping bits between the crossover points. Different types of bit string crossover methods (Davis, 1991; Goldberg, 1989) are discussed next.

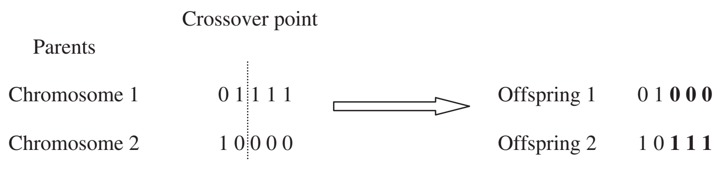

Figure 3.3 Single-point crossover.

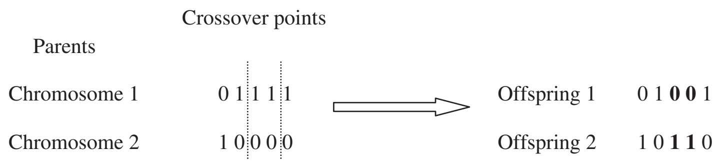

Figure 3.4 Multipoint crossover.

Single-Point Crossover Two parent chromosomes are combined randomly at a randomly selected crossover point somewhere along the length of the chromosome, and the sections on either side are swapped. For example, consider the following two chromosomes, each having 6 binary bits. After crossover, the new chromosomes (i.e., referred as offspring or children) are created as follows if the randomly chosen crossover point is 2 (Figure 3.3).

Multipoint Crossover In multipoint crossover, the number of crossover points are chosen at random, with no duplicates, and sorted in ascending order. Then, the bits between successive crossover points are exchanged between the two parents to produce two new chromosomes. The section between the first bit and the first crossover point is not exchanged between chromosomes. For example, consider the same example of two chromosomes used in a single crossover. If the randomly chosen crossover points are 2 and 4, the new chromosomes are created as shown in Figure 3.4.

The two-point crossover is a subset of the multipoint crossover. The disruptive nature of multipoint crossover appears to encourage the exploration of the search space, rather than favoring the convergence to highly fit chromosomes early in the search, thus making the search more robust.

Uniform Crossover Single-point and multipoint crossover define crossover points between the first and last bit of two chromosomes to exchange the bits between them. Uniform crossover generalizes this scheme to make every bit position a potential crossover point. In uniform crossover, one offspring is constructed by choosing every bit with a probability *P* from either parent, as shown next using the same example, by exchanging bits at the first, third, and fifth position between the parents (Figure 3.5).

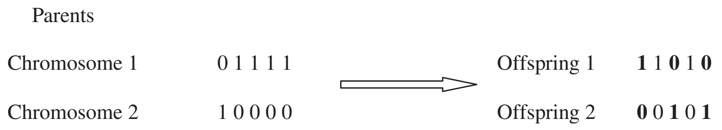
Parents
Chromosome 1 0 1 1 1 1
Chromosome 2 1 0 0 0 0
Offspring 1 **1** 1 **0** 1 **0**
Offspring 2 **0** 0 **1** 0 **1**

Figure 3.5 Uniform crossover.

Crossover Operators for Real Coding

In real coding, crossover is simply performed by swapping real values of the genes between the crossover points. Different types of real-value crossover methods have been used. Assume that $x_1 = (x_1^1, x_2^1, \ldots, x_n^1)$ and $x_2 = (x_1^2, x_2^2, \ldots, x_n^2)$ are the two chromosomes selected for crossover operation from the current population. Different crossover operators that can be used in real-coded GAs are discussed next.

Two-Point Crossover Two points of crossover $i,j \in (1,2,\ldots,n-1)$ are randomly selected, provided that $i < j$ and the segments of the parent, defined by them, are exchanged for generating two offspring (Eshelman et al., 1989), y_1 and y_2, such that:

$$y_1 = (x_1^1, x_2^1, \ldots, x_i^2, x_{i+1}^2, \ldots, x_j^2, x_{j+1}^1, \ldots, x_n^1) \tag{3.6}$$

$$y_2 = (x_1^2, x_2^2, \ldots, x_i^1, x_{i+1}^1, \ldots, x_j^1, x_{j+1}^2, \ldots, x_n^2) \tag{3.7}$$

Random Crossover Two offspring are created:

$$y_1 = (y_1^1, y_2^1, \ldots, y_n^1) \quad \text{and} \quad y_2 = (y_1^2, y_2^2, \ldots, y_n^2)$$

The value of each gene in the offspring is determined by the random uniform choice of the values of this gene in the parents:

$$y_i^k = \left\{ \begin{array}{ll} x_i^1 & \text{if } u = 0 \\ x_i^2 & \text{if } u = 1 \end{array} \right\}, \quad k = 1, 2 \tag{3.8}$$

where u is a random number that can have a value 0 or 1 (Syswerda, 1989).

Arithmetic Crossover Two offspring, $y_1 = (y_1^1, y_2^1, \ldots, y_n^1)$ and $y_2 = (y_1^2, y_2^2, \ldots, y_n^2)$, are produced, such that

$$y_i^1 = \lambda \cdot x_i^1 + (1 - \lambda) \cdot x_i^2 \tag{3.9}$$

$$y_i^2 = \lambda \cdot x_i^2 + (1 - \lambda) \cdot x_i^1 \tag{3.10}$$

where $\lambda \in [0,1]$.

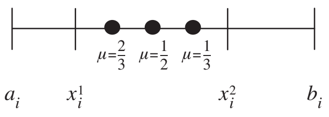

Figure 3.6 Geometrical crossover with different values for $\mu \in [0,1]$.

Geometrical Crossover Two offspring, $y_1 = (y_1^1, y_2^1, \ldots, y_n^1)$ and $y_2 = (y_1^2, y_2^2, y_n^2)$, are created, where

$$y_i^1 = x_i^{1\mu} \cdot x_i^{2(1-\mu)} \tag{3.11}$$

$$y_i^2 = x_i^{2\mu} \cdot x_i^{1(1-\mu)} \tag{3.12}$$

where $\mu \in [0,1]$.

Geometric crossover in shown in Figure 3.6 (Michalewicz, 1999).

BLX-α Crossover Two offspring, $y_1 = (y_1^1, y_2^1, \ldots, y_n^1)$ and $y_2 = (y_1^2, y_2^2, \ldots, y_n^2)$ are generated. where, y_i^k is a randomly, uniformly chosen number from the interval $[X_{\min} - I\alpha, X_{\max} + I\alpha]$ and $X_{\max}$, $X_{\min}$, and I are defined as shown here:

$$X_{\max} = \max\{x_i^1, x_i^2\} \tag{3.13}$$

$$X_{\min} = \min\{x_i^1, x_i^2\} \tag{3.14}$$

$$I = X_{\max} - X_{\min} \tag{3.15}$$

Generally, BLX-α crossover gives the best results. And it is observed that the higher value of α results in better solutions. As α increases the exploration level increases, since the relaxed exploitation zones are spread over exploration zones, thereby increasing the diversity levels in the population (Herrera et al., 1998).

For detailed descriptions of these and other crossover operators (e.g., Fuzzy, SBX (Simulated binary crossover), UNDX (Unimodal normally distributed crossover), and simplex crossover), real-coding readers are referred to Deb (2003) and Herrera et al. (1998).

Mutation

One further operator in GA is the mutation operator, which works on the level of chromosome genes by randomly altering a gene value (Deb, 2003). Mutation introduces innovation into the population by randomly modifying the chromosomes. The operation is designed to prevent GA from premature termination, since it prevents the population from becoming saturated with chromosomes that look alike. Usually considered as a background operator, the role of the mutation operator is often seen as guaranteeing that the probability of searching any given chromosome will never be zero. In GAs, mutation is randomly applied with low probability and modifies elements in the chromosomes. Large mutation rates increase the probability of destroying good chromosomes but prevent premature convergence. The mutation rate determines the probability that mutation will occur. For example, if the population size is 200, string length is 10, and mutation rate is 0.005, then only 10-bit positions will mutate in the whole population (i.e., $200 \times 10 \times 0.005 = 10$).

Similar to crossover techniques, mutation methods can be classified according to the binary coding or real-value coding of the GA.

Mutation for Binary and Gray Coding

In binary and gray coding systems, a chromosome mutation is performed at randomly chosen genes by flipping bit 0 to 1 and vice versa (Goldberg, 1989; Holland, 1975).

Mutation for Real Coding

Mutation in real-coded GA is performed, either by disarranging the gene values or by randomly selecting the new values. For example, let $x = (x_1, x_2, \ldots, x_n)$ be a chromosome and x_i be a gene to be mutated. Then a random number x'_i may be chosen from a given search space of x_i and will replace x_i. Mutation for integer coding is performed analogous to real-value coding, except that after mutation, the value for that gene is rounded to the nearest integer.

A detailed discussion of other mutation operators may be found in related textbooks and publications, for instance, see Herrera et al. (1998) and Deb (2003).

3.3 Review of GA Applications to Water Resource Problems

GAs can successfully deal with a wide range of problem areas. Briefly, the reasons for this success, according to Goldberg (1994), are "(1) GAs can solve hard problems quickly and reliably, (2) GAs are easy to interface to existing simulations and models, (3) GAs are extensible, and (4) GAs are easy to hybridize. All these reasons may be summed up in only one statement: GAs are *robust*." GAs are more powerful in difficult environments where the search space usually is large, discontinuous, complex, and poorly understood. They are not guaranteed to find the global optimum solution to a problem, but they are generally efficient at finding acceptable solutions to many real-life problems.

Goldberg (1989) gives a comprehensive review of GA applications before 1989. During the recent years, GA applications have grown enormously in many fields. GAs have been the most commonly applied nature-inspired metaheuristic algorithms in the water resource planning and management literature. Reviews of their application in different fields of water resources are reported in Nicklow et al. (2010), Rani and Moreira (2010), Labadie (2004), and Cunha (2002).

In this section, applications of GAs in the field of water resources have been classified in different groups.

3.3.1 Water Distribution Systems and Pump Scheduling Problems

Over the past two decades, considerable investment has been made in developing and applying GAs to improve the design and performance of water distribution systems. Interestingly, one of the earlier applications of GAs in water engineering was the optimization of pump schedules for a serial liquid pipeline (Goldberg and Kuo, 1987). Since then, there has been increasing interest in the application of GAs to a wide variety of water distribution system problems, such as calibration of water distribution models, optimal system design, and operation and pump scheduling.

Simpson et al. (1994) were the first to use GAs for water distribution systems. They applied and compared a GA solution to enumeration and to nonlinear programming. Vairavamoorthy and Ali (2005) presented a GA for the least-cost pipe network design problem that discards the regions of the search space where impractical or infeasible solutions are likely to exist, therefore improving search efficiency.

Dandy and Engelhardt (2001) demonstrated the use of a GA to find a near-optimal pipe replacement schedule so as to minimize the present value of capital, repair, and damage costs. Mackle et al. (1995) were among the first to apply a binary GA to pump scheduling problems by minimizing energy costs, subject to reservoir filling and emptying constraints. Subsequently, Savic et al. (1997) developed a multiobjective GA (MOGA) approach to determine pump scheduling. To reduce the excessive run times required by the GA, van Zyl et al. (2004) developed a hybrid optimization approach, in which they combined a steady-state GA with the Hooke and Jeeves hill-climbing method. Rao and Salomons (2007) developed a process based on the combined use of an artificial neural network (ANN) for predicting the consequences of different pump and valve control settings and a GA for selecting the best combination of those settings. The methodology has successfully been demonstrated on the distribution systems of Valencia (Spain) and Haifa (Israel). Munavalli and Mohan-Kumar (2003) and Prasad et al. (2004) used GA for optimal scheduling of multiple chlorine sources.

Besides the above-mentioned papers, many other applications of MOGAs have appeared in the water distribution system literature (Savic and Walter, 1997). Prasad and Park (2004) and Vamvakeridou-Lyroudia et al. (2005) employed the MOGA approach for optimal design of water distribution networks.

3.3.2 Sewer System Design Optimization

The optimal design of a sewer network aims to minimize construction costs while ensuring adequate system performance under specified design criteria. GAs have been the most popular and successful optimization techniques for the design of sewer systems (Afshar et al., 2006; Farmani et al., 2006). Hybrid GAs and MOGAs are becoming attractive in this field of study as well. Farmani et al. (2006) and Guo et al. (2006) employed local search techniques to seed an NSGA II (Non-dominated sorting genetic algorithm II) in the design of sewer networks.

3.3.3 Water Quality and Waste Management

GAs have been applied successfully in the design and operation of water and wastewater treatment plants and to other water quality management problems. GA was applied by Suggala and Bhattacharya (2003) to identify process parameters to remove organics from wastewater cost-effectively to meet pollutant removal standards. For operation of a domestic wastewater treatment plant, Chen et al. (2003) investigated the use of a GA to identify real-time control strategies, such as pH and nutrient levels, electricity consumption, and effluent flow rates, for meeting cost goals and effluent standards. Chen and Chang (1998) introduced a GA to solve a nonlinear fuzzy multiobjective programming model, considering biochemical oxygen demand and dissolved oxygen as water quality parameters, where the water quality calculation was based on the Streeter–Phelps equation. Burn and Yulianti (2001) explored waste load allocation problems using GAs. Yandamuri et al. (2006) similarly proposed optimal waste load allocation models for rivers using NSGA II. Kerachian and Karamouz (2005) extended some of the classical waste load allocation models for river water quality management for determining the monthly treatment or removal fraction to evaporation ponds. The high dimensionality of the problem (large number of decision variables) was handled by using a sequential dynamic GA.

3.3.4 Watershed Planning and Management

Yeh and Labadie (1997) introduced the application of GAs to watershed planning and presented a multiobjective watershed-level planning of stormwater detention systems using MOGAs to generate nondominated solutions for the system cost and detention effect for a watershed-level detention system. Harrell and Ranjithan (2003) applied a GA-based methodology to identify detention pond designs and land-use allocations within subbasins to manage water quality at a watershed scale. Combined use of GA and simulation models can be seen in many watershed management studies. Muleta and Nicklow (2005) linked a GA with the Soil and Water Assessment Tool to identify land-use patterns to meet water quality and cost objectives. Perez-Pedini et al. (2005) combined a distributed hydrologic model with GA for an urban watershed to determine the optimal location of infiltration-based best management practices for stormwater management.

3.3.5 Groundwater System Optimization

Groundwater optimization problems include groundwater remediation design, monitoring network design, groundwater and coastal aquifer management, parameter estimation, and source identification. Cunha (2002) and Mayer et al. (2002) presented reviews of design optimization problems that apply traditional and heuristic solution approaches to solving groundwater flow and contaminant transport processes and remediation problems, while Qin et al. (2009) also reviewed both simulation and optimization approaches used in groundwater systems.

Rogers et al. (1995) was the first to apply a GA to a field-scale remediation problem by using an ANN in place of numerical groundwater flow and the contaminant transport simulation model. Yan and Minsker (2006) proposed an adaptive neural network GA that incorporates an ANN as an approximation model that is adaptively and automatically trained within a GA, to reduce the computational requirement of groundwater remediation problems. Zheng and Wang (2002) applied a GA with response functions to solve a field-scale remediation problem at the Massachusetts Military Reservation that included 500,000 nodes in the simulation model and a 30-year planning horizon. Espinoza et al. (2005) proposed the self-adaptive hybrid GA and demonstrated its ability to reduce computation cost for groundwater remediation problems. Sinha and Minsker (2007) proposed multiscale island injection GAs, which includes multiple population functions at different spatial scales, to reduce the computational time to solve a field-scale pump-and-treat remediation optimization problem.

Many studies have considered parameter uncertainty in solving groundwater remediation optimization problems. Smalley et al. (2000) applied a noisy GA to bioremediation design, with health risk included in the formulation. Wu et al. (2006) compared a Monte Carlo simple GA (SGA) with a noisy GA to solve a sampling network problem with uncertainty in the hydraulic conductivity. Hu et al. (2007) presented an application of two-objective optimization of an *in situ* bioremediation system for a hypothetical site under uncertainty. Singh and Minsker (2008) developed a probabilistic MOGA, which combines a method similar to the noisy GA, with an additional archiving step with the NSGA II, and applied it to two pump-and-treat problems—a hypothetical and a field-scale case study.

A number of works have proposed GA approaches to groundwater monitoring network design (Chadalavada and Datta, 2008).

3.3.6 Parameter Identification

Parameter identification can be defined as a generalized term that denotes any practice, including field or experimental work, to identify parameters for a model. The parameter identification problem for most hydrologic applications is ill-posed, multimodal, nonlinear, and nonconvex (Yeh, 1986). Wang (1991) was among the first to apply the “simple” GA to the calibration problem (Nicklow et al., 2010). Subsequently, many other studies have applied GAs and their variants to watershed calibration. Zechman and Ranjithan (2007) developed a combined GA and genetic programming methodology to address the difficulties associated with models used for parameter estimation.

Tsai et al. (2003) and Mahinthakumar and Sayeed (2005) presented a similar global–local optimization approach, where a GA was used as a global optimizer to provide approximately optimal solutions that were fed in local optimization approaches. Other applications of GAs for groundwater calibration can be found in Wang and Zheng (1998).

NSGA II (Deb et al., 2000) and its variants have been widely used for multiobjective parameter identification in watershed modeling (Khu and Madsen, 2005;

Tang et al., 2006). NSGA II, used for watershed calibration, has employed the Pareto ranking scheme (used in Goldberg (1989)) to deal with multiple objectives.

3.3.7 Optimization of Reservoir System Operation

Optimization of reservoir operations involves allocation of resources, development of stream flow regulation strategies, formulating operating rules, and making real-time release decisions. A reservoir regulation plan, which is also referred to as an *operating procedure* or a *release policy*, is a group of rules quantifying the amount of water to be stored, released, or withdrawn from a reservoir or system of reservoirs under various conditions.

In a multireservoir system, input to a reservoir includes natural inflows, including all other inflows from surface runoffs, streams, and undammed rivers and all releases from adjacent upstream reservoirs on the same river or its tributaries. The output from a reservoir may be through diversion (for irrigation or other uses), spillways (for flood management), release to maintain ecological flow required in the river, and penstocks (to generate power). Also, some water is lost due to evaporation from the water surface and seepage into the ground.

A typical reservoir operation optimization model deals with constraints such as the continuity equation, maximum and minimum storage in the reservoirs, maximum and minimum releases from the reservoirs, and some case-specific obligations. The most commonly accepted objectives are the optimality of the water supply for irrigation, industrial and domestic use, hydropower generation, water quality improvement, recreation, fish and wildlife enhancement, flood control, and navigation.

The reservoir operation rule is commonly defined by a function in which the release of water from a reservoir for the given time interval is computed by using the values of current reservoir storage and current and expected demands and inflows. Generally speaking, the optimization problem takes the following form.

Maximization or minimization of the objective function, subject to the following constraints:

- The continuity equation is satisfied.
- Storage is within the upper and lower bounds.
- Releases are within the upper and lower bounds.
- Final storages are satisfied.

Several approaches have been developed for the optimization of reservoir operations, defining reservoir operating rules, and many different techniques have been studied with regard to this optimization problem. Numerous optimization models have been proposed and reviewed by many scientists (Labadie, 2004).

Esat and Hall (1994) applied a GA to the four-reservoir problem. The objective of this problem was to maximize the benefits from power generation and irrigation water supply, having constraints on both storage and release from the reservoirs. They concluded that GAs have a significant potential in reservoir operation optimization, and GAs are superior over standard dynamic programming (SDP) techniques in many aspects.

Oliveira and Loucks (1997) used a GA to evaluate operating rules for multireservoir systems and indicated that optimum reservoir operating policies can be determined by means of GAs. Wardlaw and Sharif (1999) evaluated GA formulation to different reservoir operation problems, along with a range of sensitivity analysis using different combinations of chromosome representation (binary, gray, and real coding) and crossover and mutation probabilities. Further, they applied GAs to the optimization of multireservoir systems (Sharif and Wardlaw, 2000), and the results were found to be comparable with DDDP (Discrete differential dynamic programming). Jothiprakash and Shanthi (2006) developed a GA model to derive optimal operational strategies for a single reservoir and concluded that GA can be a good alternative for real-time operation. It is important to highlight here that most researchers have agreed that GA could be a potential alternative to SDP.

A number of researchers have come to advocate that a real-coded (or floating-point) GA has a definite advantage over a binary-coded GA (Michalewicz, 1999). In real-value coding, there is no discretization of decision-variable space. Attempts have been made by many researchers to compare the performance of both GA approaches in the context of reservoir systems optimization. Chang et al. (2005) and Jian-Xia et al. (2005) compared the two approaches and found that real-coded GAs were more efficient and faster than binary-coded GAs. Chen and Chang (2007) proposed a real-coded, hypercubic-distributed GA (HDGA). Application of this method to a multireservoir system in northern Taiwan showed that HDGAs can provide much better performance than conventional GAs.

To reduce the computational requirements of the GA, it has been applied in combination with other optimization methods. Cai et al. (2001) presented a combined genetic algorithm−linear programming (GA−LP) strategy to solve the large nonlinear reservoir systems optimization model. GA was used to linearize the original problem in each time period, which is later solved sequentially using LP. The hybrid GA−LP approach was able to find good approximate solutions to the nonlinear models. In view of the computational advantages of combined GA−LP strategies to deal with nonlinearities, Reis et al. (2006) proposed and evaluated a stochastic hybrid GA−LP approach to the operation of reservoir systems, which admits a variety of future inflow variability through a treelike structure of synthetically generated inflows.

Huang et al. (2002) presented a GA-based SDP model to cope with the dimensionality problem of a multiple-reservoir system. A combination of GA and DDDP was proposed by Tospornsampan et al. (2005) for the irrigation reservoir operation problem. The main advantage of the hybrid approach is to save computational resources for optimizing parameters. Also, the good solutions obtained from the GA are used as the initial policy for DDDP, therefore reducing the probability of DDDP to trap in the local optima. Kuo et al. (2006) used a hybrid-neural GA for water quality management of the Feitsui Reservoir in Taiwan.

Ganji et al. (2007) developed a modified version of the SGA, for application to a reservoir operation problem. The SGA reduces the overall run time compared to the SGA through dynamically updating the length of chromosomes. Karamouz et al. (2007) solved a similar problem using a GA-K nearest neighborhood

(GA-KNN) based optimization model. In this methodology, the lengths of chromosomes are increased based on the results of a K-nearest neighborhood (KNN) forecasting model. Nagesh Kumar et al. (2006) developed a GA model for obtaining an optimal reservoir operating policy, but focusing on optimal crop water allocations from an irrigation reservoir in the state of Karnataka, India. The objective of the study was to maximize the relative yield from a specified cropping pattern. Kerachian and Karamouz (2006, 2007) used an algorithm combining a water quality simulation model and a stochastic conflict resolution GA-based optimization technique for determining optimal reservoir operation rules. Zahraie et al. (2008) solved a similar problem using a GA KNN-based optimization model. The KNN method is a nonparametric regression methodology that uses the similarity between observations of predictors and K similar sets of historical observations to obtain the best estimate for a dependent variable. K vectors of the past observations are obtained based on the minimum Euclidean norm from the present condition among all candidates.

Recently, Hinçal et al. (2011) applied GAs to a multireservoir system operation to maximize the energy production in the system by using two different approaches: the conventional (monthly) approach and the real-time approach. Comparison of the results revealed that the energy amounts optimized by using the conventional approach were higher than the energy produced in a real-time operation. However, by using the real-time approach, a close approximation to the real operational data had been achieved.

3.4 The GA Process for a Reservoir Operation Problem

The purpose of optimal reservoir operation is to obtain a policy to specify how water in a reservoir is regulated to satisfy the desired objectives. The optimal operating policy serves to reap the maximum benefit from the reservoir system satisfying the system demands. Here, we assume that the operating policy is composed of a decision variable, which is the release from the reservoir at each time period. The benefit is the return from release of water, and the benefit function is supposed to be given for each time period. Figure 3.7 shows a single reservoir system and the variables associated with a reservoir operation problem.

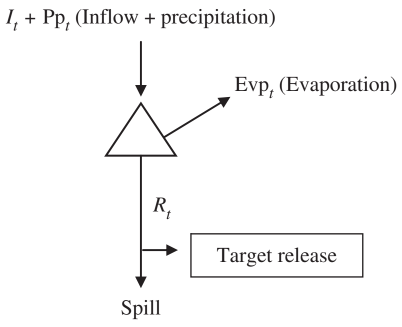
I_t + Pp_t (Inflow + precipitation)
Evp_t (Evaporation)
R_t
Target release
Spill

Figure 3.7 Variables associated with a reservoir operation problem.

Optimization aims to find the optimum combination of releases that will maximize the return for the system. There are upper and lower limits for releases and storages. These limitations form the constraints of the problem. Another constraint of the problem is that the continuity equation is to be satisfied for each time period. In general, a reservoir operation optimization problem may be expressed as follows:

The objective function is:

Maximization of net benefit

$$g_t(R_t) = \text{Max} \sum\nolimits_{t=1}^{N} [\text{NB}_t(R_t)] \tag{3.16}$$

where $\text{NB}_t(R_t)$ is the benefit function, which is a function of the release at time period t. R_t is the release for period t.

The objective function is subject to:

The continuity equation being satisfied, which is stated as:

$$S_{t+1} = S_t + I_t - R_t + \text{Pp}_t - \text{Evp}_t \quad \forall t = 1, \ldots, N \tag{3.17}$$

where S_t, I_t, and R_t are the storage, inflow, and releases for the given reservoir at time period t, and N is the time horizon for the problem under consideration. Pp_t and Evp_t are precipitation over reservoir surface and evaporation from reservoir surface during time period t, respectively.

Limits on storage impose constraints are of the form,

$$S_{\min} \le S_t \le S_{\max} \quad \forall t = 1, \ldots, N \tag{3.18}$$

which ensures that storage (S_t) will be within specified minimum and maximum values.

Limits on release are as follows:

$$R_{\min} \le R_t \le R_{\max} \quad \forall\ t = 1, \ldots, N \tag{3.19}$$

and release (R_t) should be within specified minimum and maximum ranges.

Releases are the decision variables in the problem. Decision variables exist in the composition of the chromosomes of the population in the GA. Constraints of releases are identified during the generation of the initial population, and as a matter of fact, they are satisfied. The continuity equation is readily satisfied since the storages are computed by using the continuity equation given in Eq. (3.17).

Other constraints are embedded into the objective function as a penalty function (Chang et al., 2010; Hinçal et al., 2011). Thus, the constrained optimization problem is converted to an unconstrained optimization problem. The reason why a constraint problem is transformed into an unconstrained problem is to be able to handle the problem by means of the GA. This is done as follows:

If $S_t > S_{\max}$, then the penalty term $\sum_{t=1}^{N} \{c_1(S_{\max} - S_t)^2\}$ is introduced in an objective function, i.e., Eq. (3.16).

If $S_t < S_{\min}$, then the penalty term $\sum_{t=1}^{N}\{c_2(S_{\min} - S_t)^2\}$ is introduced in an objective function, i.e., Eq. (3.16), where the deviations from maximum and minimum storage are penalized by the square of deviation from constraints. Constants c_1, and c_2 are defined as the weight of the penalty term in order for them to be in the order of the benefit terms. The optimization problem, the objective function, and constraints of which are given above are adapted into the GA. Forthcoming steps will show how GAs are used to solve this problem.

3.4.1 Generation of Initial Population

A chromosome representing search space will be

$$C_j = \{R_1, R_2, \ldots, R_t, \ldots, R_N\} \tag{3.20}$$

Each gene within the chromosome represents a release made from a reservoir at a specific time period and can take up any value between the upper and lower bounds of releases. Since the decision variables are releases (R_t), and the maximum and minimum releases are known for the reservoir, the number of chromosomes generated within these upper and lower limits represent the entire search space for the problem. The population may be generated using binary or real coding. In real coding, randomly generated numbers within the upper and lower limits of the releases will constitute chromosomes of the population. The number of chromosomes generated will depend on the assumed size of the population (population size $j = 1, \ldots, M$).

3.4.2 Calculation of State Variables

After the generation of the initial population, which is composed of chromosomes containing releases (decision variables), calculation of storages (state variables) comes next. Storage for every gene of the individuals is computed making use of continuity Eq. (3.17), which is the equality constraint of the problem. Usage of Eq. (3.17) in calculation of storage ensures that the continuity equation is satisfied for every gene created. The inequality constraints ensure that the storages remain within their limits and are satisfied by incorporating the related penalty terms into the objective function.

3.4.3 Calculation of Fitness Values

In the next step, the fitness value for each chromosome is calculated. The fitness assigned to each gene has direct influence on the eligibility for each chromosome to live in the next generation. Penalty terms originating from violation of the constraints will make sure that the chromosomes violating the storage constraints will not be selected in the next population.

3.4.4 GA Operators

GA operators and selection, crossover, and mutation operators are implemented on the population to get the best solution, as already discussed in Section 3.3. The choice of selection technique depends on the nature of the problem, and various techniques may be applied and compared to choose the best solution for a particular problem.

Crossover operation is performed using a predefined crossover probability. Crossover probability leads to deciding whether to put the parent chromosomes under the process of crossover. A random number is generated and compared with the crossover probability to specify whether to apply the crossover operators. The selected chromosomes undergo crossover operation only if the random number generated is greater than the probability of crossover. There is no rule to define crossover probability, and usually a sensitivity analysis is carried out to get the best value for crossover probability for a particular problem. The crossover operator chosen also depends on the problem, and different crossover techniques may be compared to select the best one for the problem chosen.

Mutation is randomly applied with low probability, typically in the range 0.001 and 0.02, to modify the genes of some chromosomes. The role of mutation is often seen as a safety net to recover good genetic material that may be lost during selection and crossover operations. The mutation operator has been constructed to alter the gene randomly with consideration to the predefined probability of mutation. Ifthe random number generated is greater than the probability of mutation, the gene is reproduced using a suitable mutation operator; otherwise, it remains the same.

3.4.5 Example: A Four-Time-Period Reservoir Operation Problem

To illustrate the main features of GAs, let us consider a reservoir operation problem. The reservoir has an active storage capacity of 20 Million Cubic Meter (MCM). The active storage volume, S_t, in the reservoir can vary from 0 to 20. Let R_t be the release or discharge from the reservoir in time period t. Each variable is expressed as a volume unit for the period, $t = 1, 2, 3, 4$. In these time periods, the inflows to the reservoir are $I_t = 14$, 12, 6, and 8, respectively. The net benefit function for each time period for unit release from the reservoir is defined by $f_t = 11.5 + 1.5R_t^2$. Suppose that only integer solutions are to be considered and the maximum release from the reservoir cannot exceed 9, which is fixed as the target demand for each time period. What is the optimal release R_t for each time period? Evaporation losses and precipitation may be ignored.

Solution

Here, the objective of the problem is to maximize the net benefit from the released water; therefore, the overall objective function may be written as

$$\text{Max} \sum\nolimits_{t=1}^{4} (11.5 + 1.5R_t^2) - c(R_t - 9)^2 \qquad (3.21)$$

where the first term defines the benefit from releases and the second term minimizes the deviation from target demand. In this problem, c is a coefficient to be chosen such that the objective function remains positive, and the value of c is supposed to be 0.1.

The following are the constraints for this function:

The continuity equation:

$$S_{t+1} = S_t + I_t - R_t \quad \forall t \tag{3.22}$$

Limits on storages are as follows:

$$0 \leq S_t \leq 20 \quad \forall t \tag{3.23}$$

Limits on releases are as follows:

$$0 \leq R_t \leq 9 \quad \forall t \tag{3.24}$$

Since GA cannot explicitly handle constraints, these are taken care of by penalty functions. If $S_t \geq 20$, then penalty term $\sum_{t=1}^{4}\{c_1(20-S_t)^2\}$ will be introduced in the objective function, and if $S_t \leq 0$, then penalty term $\sum_{t=1}^{4}\{c_1(0-S_t)^2\}$ will be introduced instead.

Each individual solution set contains the values of all the decision variables whose best values are being sought. For example, if there are four decision variables x_1, x_2, x_3, and x_4 to be obtained, these variables are arranged into a string or chromosome, $x_1x_2x_3x_4$. If each decision variable is expressed using three digits, then the chromosome 005021050279 would represent $x_1 = 5$, $x_2 = 21$, $x_3 = 50$, and $x_4 = 279$.

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.

A population of possible feasible solutions is generated randomly. Each chromosome contains the values of all the decision variables whose best values are being sought. In this example, we are using numbers to the base 10; therefore, a sample chromosome 8376 will represent the releases $R_1 = 8$, $R_2 = 3$, $R_3 = 7$, and $R_4 = 6$. Another chromosome representing the solution to this problem, chosen randomly, would be 2769. These two chromosomes, each containing four genes, can pair up and have two children. Population size is a GA parameter—that is, the number of solutions being considered. To show the iterations for this example, we assume a population of 10 individuals. However, the best values of GA parameters are usually decided by trial and error.

The GA process begins with the random generation of an initial population of feasible solutions, proceeds with selection, random crossover, and mutation operations, and then randomly generates the new population for the next iteration. This process repeats itself with the new population and continues until there is no significant improvement in the best solution found.

For this example, the process includes the following steps:

1. Initial population is randomly generated, which is a set of solutions/chromosomes having randomly generated decision-variable values within the range 0–9. The release cannot exceed 9, therefore satisfying the release constraint.
2. The corresponding storages in the reservoir are calculated using the continuity Eq. (3.22), assuming that the storage at the beginning of operation is 0. This ensures that the continuity equation is satisfied. If the value of storage for any chromosome is bigger than 20 or less than 0, then the penalty terms are added in the objective function. A negative value of penalty coefficients is considered here, which will decrease the value of objective function and the chromosome will not appear in the next generation. This will prevent the violation of storage constraints.
3. The initial population undergoes the selection operation, and best decision variables are selected using the roulette wheel selection method, as discussed in section "Selection" earlier in this chapter.
4. The selected chromosomes are paired to determine whether a crossover is to be performed on each pair, assuming that the probability of a crossover is 60% ($P_c = 0.6$). If a crossover is to occur, we find the crossing site randomly, by creating a random number between 1 and 3. Note that not all five pairs will undergo crossover operation. With 60% probability, in iteration 1, it was seen that only the first, second, and fifth pairs (shown in bold in Table 3.5) will crossover at randomly chosen site 3. The single-point crossover operation is used in this example.
5. Next, determine if any chromosome in the resulting intermediate population is to be mutated. For mutation, we assume the probability of mutation ($P_m = 0.05$) for each gene. For this example, we assume that mutation increases the value of the number by 1, or if the original number is 9, mutation does not change it to 10; rather, it keeps it as it is. With this probability ($10 \times 4 \times 0.05 = 2$), two chromosomes will undergo mutation randomly. In iteration 1, chromosomes 3 and 6 are randomly chosen for the mutation operation, and for these chromosomes, the genes to be mutated are also chosen randomly by generating a random number between 1 and 4. The mutated genes are shown in bold and italics for iteration 1 in Table 3.5.
6. The last step creates a new population, and steps 2–5 are repeated for a predefined number of generations or until the best solution is obtained.

These steps are performed for two iterations (see Table 3.5). The solution found in the second iteration increases the sum of the fitness value from 2949.2 to 3747.2. This process can continue till the process has converged to the best solution it can find. The whole process may be repeated for different probabilities of crossover and mutation to find out the optimal parameters for the GA process for this problem.

3.5 Conclusions

The GA has become a popular tool for researchers to solve a wide variety of water resource management problems. This chapter has presented a brief review of the theory of GAs and their applications to reservoir operation, groundwater management, water quality, parameter estimation, and other problems related to water resource management. GA has its own advantages and limitations when applied to

Table 3.5 GA Iteration for Reservoir Operation Example

Index for Chromosomes	**Initial Population (R_1 R_2 R_3 R_4)**	**Storages**				**Fitness (ft_i)**	**Probability of Selection (*P*)**	**R and ( )**	***s***	**Cumulative Fitness**	**Selected Index**	**Selected Chromosome**	**Crossover ($P_c = 0.5$)**	**Mutation ($P_m = 0.05$)**
		S_1	S_2	S_3	S_4									
Iteration 1														
1	2695	12	18	15	18	257.6	0.09	0.35	1039.33	257.6	4	8279	**8279**	8279
2	3595	11	18	15	18	249.2	0.08	0.41	1199.69	506.8	4	8279	**8279**	8279
3	9849	5	9	11	10	406.4	0.14	0.56	1653.02	913.2	6	3497	**3499**	3***5***99
4	8279	6	16	15	14	337.6	0.11	0.93	2734.76	1250.8	10	6799	**6797**	6797
5	6538	8	15	18	18	240.8	0.08	0.20	580.40	1491.6	3	9849	9849	9849
6	3497	11	19	16	17	272	0.09	0.01	20.63	1763.6	1	2695	2695	***3***695
7	2676	12	18	17	19	226.4	0.08	0.92	2720.55	1990	10	6799	6799	6799
8	5882	9	13	11	17	274.8	0.09	0.04	121.27	2264.8	1	2695	2695	2695
9	2687	12	18	16	17	269.2	0.09	0.62	1837.25	2534	7	2676	**2679**	2679
10	6799	8	13	10	9	415.2	0.14	0.97	2868.57	2949.2	10	6799	**6796**	6796
Total						$\sum_{i=1}^{N} \mathrm{ft}_i = 2949.2$								
Iteration 2														
1	8279	6	16	15	14	337.6	0.10	0.15	502.67	337.6	2	8279	**8299**	8299
2	8279	6	16	15	14	337.6	0.10	0.66	2203.79	675.2	7	6799	**6779**	6779
3	3599	11	18	15	14	334.8	0.10	0.62	2077.63	1010	7	6799	**6779**	6779
4	6797	8	13	10	11	366.8	0.11	0.87	2910.91	1376.8	9	2679	**2699**	2699

(*Continued*)

Table 3.5 (Continued)

Index for Chromosomes	Initial Population (R_1 R_2 R_3 R_4)	Storages				Fitness (ft_i)	Probability of Selection (P)	R and ( )	s	Cumulative Fitness	Selected Index	Selected Chromosome	Crossover ($P_c = 0.5$)	Mutation ($P_m = 0.05$)
		S_1	S_2	S_3	S_4									
5	9849	5	9	11	10	406.4	0.12	0.63	2135.69	1783.2	7	6799	**6749**	6749
6	3695	11	17	14	17	266.4	0.08	0.43	1457.26	2049.6	5	9849	**9899**	98***9***9
7	6799	8	13	10	9	415.2	0.12	0.40	1354.91	2464.8	4	6797	6797	6797
8	2695	12	18	15	18	257.6	0.08	0.29	970.45	2722.4	3	3599	3599	3***6***99
9	2679	12	18	17	16	294.8	0.09	0.52	1747.39	3017.2	5	9849	9849	9849
10	6796	8	13	10	12	346.8	0.10	0.09	311.00	3364	1	8279	8279	8279
Total						$\sum_{i=1}^{N} ft_i = 3364$								
Iteration 3														
1	8299	6	12	5	5	386								
2	6779	8	7	7	5	366.8								
3	6779	8	7	7	5	366.8								
4	2699	12	8	5	5	343.2								
5	6749	8	7	10	5	315.2								
6	9899	5	6	5	5	506.4								
7	6797	8	7	5	7	366.8								
8	3699	11	8	5	5	352								
9	9849	5	6	10	5	406.4								
10	8279	6	12	7	5	337.6								
Total						$\sum_{i=1}^{N} ft_i = 3747.2$								

these complex problems, and researchers continue to modify the algorithm itself or combine the use of the algorithm with other techniques. The description of the GA procedure, along with the illustrative example given at the end of the chapter, will be helpful for understanding the basics of the algorithm and its application to a water resource problem.

References

Afshar, M.H., Afshar, A., Mariño, M.A., Darbandi, A.A.S., 2006. Hydrograph-based storm sewer design optimisation by genetic algorithm. Can. J. Civ. Eng. 33 (3), 319–325.

Burn, D.H., Yulianti, J.S., 2001. Waste-load allocation using genetic algorithms. J. Water Resour. Planning Manage. 127 (2), 121–129.

Cai, X.M., McKinney, D.C., Lasdon, L.S., 2001. Solving nonlinear water management models using a combined genetic algorithm and linear programming approach. Adv. Water Resour. 24 (6), 667–676.

Caruana, R.A., Schaffer, J.D., 1988. Representation and hidden bias: gray vs. binary coding for genetic algorithms. In: Laird J. (Ed.), Proceedings of Fifth International Conference on Machine Learning. Morgan Kaufmann, Los Altos, CA, pp. 153–161.

Chadalavada, S., Datta, B., 2008. Dynamic optimal monitoring network design for transient transport of pollutants in groundwater aquifers. Water Resour. Manage. 22 (6), 651–670.

Chang, F.-J., Chen, L., Chang, L.-C., 2005. Optimizing the reservoir operating rule curves by genetic algorithms. Hydrolog. Process. 19 (11), 2277–2289.

Chang, L.-C., Chang, F.-J., Wang, K.-W., Dai, S.-Y., 2010. Constrained genetic algorithms for optimizing multi-use reservoir operation. J. Hydrol. 390, 66–74.

Chen, H.W., Chang, N.B., 1998. Water pollution control in the river basin by genetic algorithm-based fuzzy multi-objective programming modeling. Water Sci. Tech. 37 (8), 55–63.

Chen, L., Chang, F.J., 2007. Applying a real-coded multi-population genetic algorithm to multi-reservoir operation. Hydrolog. Process. 21 (5), 688–698.

Chen, W.C., Chang, N.B., Chen, J.C., 2003. Rough set-based hybrid fuzzy-neural controller design for industrial wastewater treatment. Water Res. 37 (1), 95–107.

Cieniawski, S.E., Eheart, J.W., Ranjithan, S., 1995. Using genetic algorithms to solve a multiobjective groundwater monitoring problem. Water Resour. Res. 31 (2), 399–409.

Cunha, M.D., 2002. Groundwater cleanup: the optimization perspective a literature review. Eng. Optim. 34 (6), 689–702.

Dandy, G.C., Engelhardt, M., 2001. Optimal scheduling of water pipe replacement using genetic algorithms. J. Water Resour. Planning Manage. 127 (4), 214–223.

Dandy, G.C., Simpson, A.R., Murphy, L.J., 1996. An improved genetic algorithm for pipe network optimization. Water Resour. Res. 32 (2), 449–458.

Davidson, J.W., Goulter, I.C., 1995. Evolution program for design of rectilinear branched networks. J. Comput. Civ. Eng. 9 (2), 112–121.

Davis, L., 1991. Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York, NY.

De Jong, K.A., 1975. Analysis of the behavior of a class of genetic adaptive systems. Ph.D. Dissertation. University of Michigan, Ann Arbor.

Deb, K., 2003. Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Singapore.

Deb, K., Agrawal, S., Pratap, A., Meyarivan, T., 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Schoenauer M., Deb K., Rudolph G., Yao X., Lutton E., Merelo J.J., Schwefel H.-P. (Eds.), Parallel Problem Solving From Nature, vol. VI (PPSN-VI). Springer, Berlin/Heidelberg, pp. 849−858.

Esat, V., Hall, M.J., 1994. Water resources system optimization using genetic algorithms. In: Verwey A., Minns A.W., Babovic V., Maksimovic C. (Eds.), Proceedings of First International Conference on Hydroinformatics. Balkema Rotterdam, The Netherlands, pp. 225−231.

Eshelman, L.J., Schaffer, J.D., 1993. Real coded genetic algorithms and interval schemata. In: Whitley, L.D. (Ed.), Foundation of Genetic Algorithms 2. Morgan Kaufmann, San Mateo, CA, pp. 187−202.

Eshelman, L.J., Caruana, A., Schaffer, J.D., 1989. Biases in the crossover landscape. In: Schaffer, J.D. (Ed.), Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp. 86−91.

Espinoza, F., Minsker, B., Goldberg, D.E., 2005. Adaptive hybrid genetic algorithm for groundwater remediation design. J. Water Resour. Planning Manage. 131 (1), 14−24.

Farmani, R., Savic, D.A., Walters, G.A., 2006. A hybrid technique for optimisation of branched urban water systems. In: Gourbesville P., Cunge J., Guinot V., Liong S.-Y. (Eds.), Proceedings of Seventh Hydroinformatics Conference, vol. 1. Research Publishing, Chennai, India, pp. 985−992.

Feng, C.W., Liu, L.A., Burns, S.A., 1997. Using genetic algorithms to solve construction time-cost trade-off problems. J. Comput. Civ. Eng. 11 (3), 184−189.

Ganji, A., Karamouz, M., Khalili, D., 2007. Development of stochastic conflict resolution models for reservoir operation. II. The value of players' information availability and cooperative behavior. Adv. Water Resour. 30, 528−542.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

Goldberg, D.E., Deb, K., 1991. A comparative analysis of selection schemes used in genetic algorithms. In: Rawlins G.J.E. (Ed.), Foundations of Genetic Algorithms. Morgan Kaufman, San Mateo, CA, pp. 69−93.

Goldberg, D.E., 1994. Genetic and evolutionary algorithms come of age. Communication of the Association for Computing Machinery (ACM). 37 (3), 113−119.

Goldberg, D.E., Kuo, C.H., 1987. Genetic algorithms in pipeline optimization. J. Comput. Civ. Eng. 1 (2), 128−141.

Guo, Y., Walters, G.A., Khu, S.T., Keedwell, E.C., 2006. Optimal design of sewer networks using hybrid cellular automata and genetic algorithm. In: Proceedings of Fifth IWA World Water Congress. IWA Publication, London.

Halhal, D., Walters, G.A., Ouazar, D., Savic, D.A., 1997. Water network rehabilitation with structured messy genetic algorithm. J. Water Resour. Planning Manage. 123 (3), 137−146.

Harrell, L.J., Ranjithan, S., 2003. Integrated detention pond design and land use planning for watershed management. J. Water Resour. Planning Manage. 129 (2), 98−106.

Haupt, R.L., Haupt, S.E., 2004. Practical Genetic Algorithms. John Wiley & Sons, Hoboken, NJ.

Herrera, F., Lozano, M., Verdegay, J.L., 1998. Tackling realcoded genetic algorithms: operators and tools for behavioural analysis. Artif. Intell. Rev. 12, 265−319.

Hınçal, O., Altan-Sakarya, A.B., Ger, A.M., 2011. Optimization of multireservoir systems by genetic algorithm. Water Resour. Manage. 25, 1465−1487.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI.

Hu, Z.Y., Chan, C.W., Huang, G.H., 2007. Multi-objective optimization for process control of the *in-situ* bioremediation system under uncertainty. Eng. Appl. Artif. Intell. 20 (2), 225−237.

Huang, W.C., Yuan, L.C., Lee, C.M., 2002. Linking genetic algorithms with stochastic dynamic programming to the long-term operation of a multireservoir system. Water Resour. Res. 38 (12), 9.

Jian-Xia, C., Qiang, H., Yi-min, W., 2005. Genetic algorithms for optimal reservoir dispatching. Water Resour. Manage. 19 (4), 321−331.

Jothiprakash, V., Shanthi, G., 2006. Single reservoir operating policies using genetic algorithm. Water Resour. Manage. 20 (6), 917−929.

Karamouz, M., Mojahedi, A., Ahmadi, A., 2007. Economic assessment of operational policies of inter-basin water transfer. Water Resour. Res. 3 (2), 86−101.

Kerachian, R., Karamouz, M., 2005. Waste-load allocation for seasonal river water quality management: application of sequential dynamic genetic algorithms. J. Sci. Iran. 12 (2), 117−130.

Kerachian, R., Karamouz, M., 2006. Optimal reservoir operation considering the water quality issues: a stochastic conflict resolution approach. Water Resour. Res. 42, W12401.

Kerachian, R., Karamouz, M., 2007. A stochastic conflict resolution model for water quality management in reservoir−river system. Adv. Water Resour. 30 (4), 866−882.

Khu, S.-T., Madsen, H., 2005. Multiobjective calibration with Pareto preference ordering: an application to rainfall-runoff model calibration. Water Resour. Res. 41, 1367−1376.

Kuo, J.-T., Wang, Y.Y., Lung, W.S., 2006. A hybrid neural-genetic algorithm for reservoir water quality management. Water Res. 40 (7), 1367−1376.

Labadie, J.W., 2004. Optimal operation of multireservoir systems: state-of the-art-review. J. Water Resour. Planning Manage. 130 (2), 93−111.

Li, H., Love, P.E.D., 1998. Site-level facilities layout using genetic algorithms. J. Comput. Civ. Eng. 12 (4), 227−231.

Loucks, D.P., van Beek, E., 2005. Water Resources Systems Planning and Management: An Introduction to Methods, Models and Applications. UNESCO, Paris.

Mackle, G., Savic, D.A., Walters, G.A., 1995. Application of genetic algorithms to pump scheduling for water supply. In: Proceedings of Genetic Algorithms in Engineering Systems: Innovations and Applications. GALESIA ‘95, IEE, London, pp. 400−405.

Mahinthakumar, G., Sayeed, M., 2005. Hybrid genetic algorithm: local search methods for solving groundwater source identification inverse problems. J. Water Resour. Planning Manage. 131 (1), 45−57.

Mayer, A.S., Kelley, C.T., Miller, C.T., 2002. Optimal design for problems involving flow and transport phenomena in saturated subsurface systems. Adv. Water Resour. 25 (8−12), 1233−1256.

McKinney, D.C., Lin, M.D., 1994. Genetic algorithm solution of groundwater management models. Water Resour. Res. 30 (6), 1897−1906.

Michalewicz, Z., 1999. Genetic Algorithms + Data Structures = Evolution Programs. third rev. and extended ed. Springer, New York, NY.

Mitchell, M., 1999. An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA.

Muleta, M.K., Nicklow, J.W., 2005. Decision support for watershed management using evolutionary algorithms. J. Water Resour. Planning Manage. 131 (1), 35−44.

Munavalli, G.R., Mohan, M.S., 2003. Optimal scheduling of multiple chlorine sources in water distribution systems. J. Water Resour. Planning Manage. 129 (6), 493–504.

Nagesh Kumar, D., Srinivasa Raju, K., Ashok, B., 2006. Optimal reservoir operation for irrigation of multiple crops using genetic algorithms. J. Irrigat. Drain. Eng. 132 (2), 123–129.

Nicklow, J., Reed, P., 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. J. Water Resour. Planning Manage. 136 (4), 412–432.

Oliveira, R., Loucks, D.P., 1997. Operating rules for multireservoir systems. Water Resour. Res. 33 (4), 839–852.

Perez-Pedini, C., Limbrunner, J.F., Vogel, R.M., 2005. Optimal location of infiltration-based best management practices for storm water management. J. Water Resour. Planning Manage. 131 (6), 441–448.

Prasad, T.D., Park, N.S., 2004. Multiobjective genetic algorithms for design of water distribution networks. J. Water Resour. Planning Manage. 130 (1), 73–82.

Prasad, T.D., Walters, G.A., Savic, D.A., 2004. Booster disinfection of water supply networks: a multi-objective approach. J. Water Resour. Planning Manage. 130 (5), 367–376.

Qin, X.S., Huang, G.H., He, L., 2009. Simulation and optimization technologies for petroleum waste management and remediation process control. J. Environ. Manage. 90 (1), 54–76.

Rani, D., Moreira, M.M., 2010. Simulation–optimization modeling: a survey and potential application in reservoir systems operation. Water Resour. Manage. 24 (6), 1107–1138.

Rao, Z., Salomons, E., 2007. Development of a real-time, near optimal control process for water-distribution networks. J. Hydroinform. 9 (1), 25–37.

Reis, L., Bessler, F., Walters, G., Savic, D., 2006. Water supply reservoir operation by combined genetic algorithm–linear programming (GA–LP) approach. Water Resour. Manage. 20 (2), 227–255.

Ritzel, B.J., Eheart, J.W., Ranjithan, S., 1994. Using genetic algorithms to solve a multiple-objective groundwater pollution containment-problem. Water Resour. Res. 30 (5), 1589–1603.

Rogers, L.L., Dowla, F.U., Johnson, V.M., 1995. Optimal fieldscale groundwater remediation using neural networks and the genetic algorithm. Environ. Sci. Tech. 29 (5), 1145–1155.

Savic, D., Walters, G., 1997. Genetic algorithms for least cost design of water distribution networks. J. Water Resour. Planning Manage. 123 (2), 67–77.

Savic, D.A., Walters, G.A., Schwab, M., 1997. Multiobjective genetic algorithms for pump scheduling in water supply. In: Corne, D., Shapiro, J.L. (Eds.), AISB '97, Lecture Notes in Computer Science, 1305. Springer, Berlin, pp. 227–236.

Sharif, M., Wardlaw, R., 2000. Multireservoir systems optimization using genetic algorithms: case study. J. Comput. Civ. Eng. 14 (4), 255–263.

Shivraj, R., Ravichandran, T., 2011. A review of selection methods in genetic algorithm. Int. J. Eng. Sci. Technol. 3 (5), 3792–3797.

Simpson, A.R., Dandy, G.C., Murphy, L.J., 1994. Genetic algorithms compared to other techniques for pipe optimization. J. Water Resour. Planning Manage. 120 (4), 423–443.

Singh, A., Minsker, B.S., 2008. Uncertainty-based multiobjective optimization of groundwater remediation design. Water Resour. Res. 44, W02404.

Sinha, E., Minsker, B.S., 2007. Multiscale island injection genetic algorithms for groundwater remediation. Adv. Water Resour. 30 (9), 1933–1942.

Smalley, J.B., Minsker, B.S., Goldberg, D.E., 2000. Risk-based *in situ* bioremediation design using a noisy genetic algorithm. Water Resour. Res. 36 (10), 3043–3052.

Soh, C.K., Yang, J.P., 1996. Fuzzy controlled genetic algorithm search for shape optimization. J. Comput. Civ. Eng. 10 (2), 143−150.

Suggala, S.V., Bhattacharya, P.K., 2003. Real coded genetic algorithm for optimization of pervaporation process parameters for removal of volatile organics from water. Ind. Eng. Chem. Res. 42 (13), 3118−3128.

Syswerda, G., 1989. Uniform crossover in genetic algorithms. In: Schaffer, J.D. (Ed.), Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp. 2−9.

Tang, Y., Reed, P., Wagener, T., 2006. How efficient and effective are evolutionary multiobjective algorithms at hydrologic model calibration? Hydrol. Earth Syst. Sci. 10, 289−307.

Tospornsampan, J., Kita, I., Ishii, M., Kitamura, Y., 2005. Optimization of a multiple reservoir system operation using a combination of genetic algorithm and discrete differential dynamic programming: a case study in Mae Klong system, Thailand. Paddy Water Environ. 3, 29−38.

Tsai, F.T.C., Sun, N.Z., Yeh, W.W.G., 2003. A combinatorial optimization scheme for parameter structure identification in groundwater modeling. Ground Water. 41 (2), 156−169.

Vairavamoorthy, K., Ali, M., 2005. Pipe index vector: a method to improve genetic-algorithm-based pipe optimization. J. Hydraul. Eng. 131 (12), 1117−1125.

Vamvakeridou-Lyroudia, L.S., Walters, G.A., Savic, D.A., 2005. Fuzzy multiobjective optimization of water distribution networks. J. Water Resour. Planning Manage. 131 (6), 467−476.

van Zyl, J., Savic, D.A., Walters, G.A., 2004. Operational optimization of water distribution systems using a hybrid genetic algorithm method. J. Water Resour. Planning Manage. 130 (2), 160−170.

Wang, Q.J., 1991. The genetic algorithm and its application to calibrating conceptual rainfall-runoff models. Water Resour. Res. 27 (9), 2467−2471.

Wang, L., Zheng, D.Z., 2002. A modified genetic algorithm for job shop scheduling. Int. J. Adv. Manufact. Technol. 20 (1), 72−76.

Wang, M., Zheng, C., 1998. Groundwater management optimization using genetic algorithms and simulated annealing: formulation and comparison. J. Am. Water Resour. Assoc. 34 (3), 519−530.

Wardlaw, R., Sharif, M., 1999. Evaluation of genetic algorithms for optimal reservoir system operation. J. Water Resour. Planning Manage. 125 (1), 25−33.

Wei, L.Y., Zhao, M., Wu, G.M., Meng, G., 2005. Truss optimization on shape and sizing with frequency constraints based on genetic algorithm. Comput. Mech. 35 (5), 361−368.

Wright, A., 1991. Genetic algorithms for real parameter optimization. Foundation of Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp. 205−218.

Wu, J., Zheng, C., Chien, C., Zheng, L., 2006. A comparative study of Monte Carlo simple genetic algorithm and noisy genetic algorithm for cost-effective sampling network design under uncertainty. Adv. Water Resour. 29, 899−911.

Yan, S., Minsker, B., 2006. Optimal groundwater remediation design using an adaptive neural network genetic algorithm. Water Resour. Res. 42, W05407.

Yandamuri, S.R.M., Srinivasan, K., Bhallamudi, S.M., 2006. Multiobjective optimal waste load allocation models for rivers using nondominated sorting genetic algorithm-II. J. Water Resour. Planning Manage. 132 (3), 133−143.

Yeh, W.W.-G., 1986. Review of parameter identification procedures in groundwater hydrology: the inverse problem. Water Resour. Res. 22 (2), 95−108.

Yeh, C.-H., Labadie, J.W., 1997. Multiobjective watershed-level planning of storm-water detention basins. J. Water Resour. Planning Manage. 123 (6), 336−343.

Zahraie, B., Kerachian, R., Malekmohammadi, B., 2008. Reservoir operation optimization using adaptive varying chromosome length genetic algorithm. Water Int. 33 (3), 380−391.

Zechman, E.M., Ranjithan, S., 2007. Evolutionary computation based approach for model error correction and calibration. Adv. Water Resour. 30 (5), 1360−1370.

Zheng, C.M., Wang, P.P., 2002. A field demonstration of the simulation optimization approach for remediation system design. Ground Water. 40 (3), 258−266.