International Conference "Water, Environment, Energy and Society" (WEES–2009)
New Delhi, 12–16 January 2009

Evolutionary Computing in Optimal Reservoir Operation

M. Janga Reddy
Department of Civil Engineering
IIT Bombay, Mumbai - 400 076, INDIA

D. Nagesh Kumar
Department of Civil Engineering
Indian Institute of Science, Bangalore - 560 012, INDIA
E-mail: nagesh@civil.iisc.ernet.in

ABSTRACT: Efficient design and operation of water resource systems is a challenging task in many real world applications. Many issues related to water resources require the solutions of optimization. As computers have become more powerful, the size and complexity of problems which can be simulated and solved by optimization techniques have correspondingly expanded. Real life water management problems involve nonlinear optimization and often associated with complexities of non-convex objective functions and multimodal solutions. If the objective function is not known analytically, traditional methods are not applicable. Consequently these difficulties lead to go for non-conventional optimization techniques. Recently, evolutionary computation techniques have been receiving increased attention in view of their potential as global optimization techniques for complex problems. This popularity is mainly due to the robustness, ease of use and wide applicability of evolutionary algorithms. This paper aims to discuss some of the issues of evolutionary algorithms and summarizes applications of evolutionary computing techniques such as Genetic Algorithms (GA), Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) for effective water resources management, especially in the context of reservoir operation.

INTRODUCTION

In general, the system modeling in water resources aims to reduce the total system cost and failure risk, then tries to maximize the benefits by providing a robust design and/or operation policy. One of the most important engineering tools that can be employed in such activities is optimization. The general objective in optimization is to choose a set of values of the variables subject to the various constraints that will produce the desired optimum response for the chosen objective function. Today, there are a variety of optimization techniques existing, but no single optimization method or algorithm can be applied efficiently to all problems. The method chosen for any particular case will depend primarily on: (1) the character of the objective function and whether it is known explicitly, (2) the nature of the constraints, and (3) the number of independent and dependent variables etc.

Many water resources applications generally involve non-linear optimization in problem solving. So the Linear Programming (LP) cannot work in such cases. The enumerative based DP technique poses severe computational problems for a multi-purpose multi-reservoir system due to increase in the number of state variables and the corresponding discrete states, since in this method, linear increase in number of state variables (dimensions) causes exponential increase in computational time requirement. So, when DP is applied to larger-dimensional problems, it has a major hurdle of *the curse of dimensionality*. The gradient-based Nonlinear Programming (NLP) methods can solve problems with smooth nonlinear objectives and constraints. However, in large and highly nonlinear models, these algorithms may fail to find feasible solutions, or converge to local optimum depending upon the degree of non-linearity and initial guess. Hence, these traditional optimization techniques do not ensure global optimum and also have limited applications. Lack of ability to obtain a global optimum in the case of traditional nonlinear-optimization techniques and intensity of computational requirements in the case of dynamic programming motivated the search for new approaches, which would combine efficiency and ability to find the global optimum (Janga Reddy and Nagesh Kumar, 2005a). In the recent past, non-traditional search and optimization methods based on natural phenomenon also called Evolutionary Computation (EC) techniques, have been developed and applied to many practical problems. In

the following section, first a brief description on evolutionary computing principles and then some of the major types of evolutionary algorithms are presented.

EVOLUTIONARY COMPUTING

During the last two decades, there has been a growing interest in algorithms, which are based on the principle of natural evolution (i.e., survival of the fittest). A common term, accepted recently, refers to such techniques as Evolutionary Algorithms. (EAs). Evolutionary programs as probabilistic optimization algorithms based on the similarities with the biological evolutionary process are especially promising for complicated engineering optimization problems, where other traditional optimization methods cannot be easily applied. In EAs, a population of individuals, each representing a search point in the space of feasible solutions, is exposed to a collective learning process which proceeds from generation to generation. The population is arbitrarily initialized and subjected to the process of selection, recombination and mutation through stages known as generations, such that the newly created generations evolve towards more favorable regions of the search space. The progress in the search is achieved by evaluating the fitness of all individuals in the population, selecting the individuals with the highest fitness value and combining them to create new individuals with increased likelihood of improved fitness. After some generations, the program converges, and the best individual represents the optimum or near-optimum solution. The two most important issues in the evolution process are population diversity and selective pressure. These factors are strongly related, i.e., an increase in the selective pressure decreases the diversity of the population, and vice versa. In other words, strong selective pressure "supports" the premature convergence of the search and a weak selective pressure can make the search ineffective. Different evolutionary techniques use different scaling methods and different selection schemes (e.g., proportional selection, ranking, tournament) to strike a balance between these two factors. However, the structure of any evolutionary computation algorithm is very much the same. A sample structure is shown in Figure 1.

The most well-known paradigm of EAs is the Genetic Algorithms (GA) that is used widely, especially in engineering and industrial applications. Several other ideas, such as Evolutionary Programming (Holland, 1975; Goldberg, 1985; Michaelwicz, 1999), Evolution Strategies (Fogel *et al.*, 1966; Fogel, 1994; Schewel, 1994), Genetic Programming (Koza, 1992) inspired by the GAs are exhibiting significant results in several scientific fields. Apart from these techniques, there are many hybrid systems, which incorporate various features of the above paradigms and consequently are hard to classify, which can be referred just as Evolutionary Computing (EC) methods (Dasgupta & Michalewicz, 1997). Table 1 shows the important characteristics and similarities of different EC methods.

Procedure of evolutionary algorithm:

```
begin
        t ← 0
        initialize P(t)
        evaluate P(t)
        while (not termination-condition) do
        begin
                t ← t + 1
                select P(t) from P(t - 1)
                alter P(t)
                evaluate P(t)
        end
end
```

Fig. 1: The structure of an evolutionary algorithm

Table 1: Characteristics of Different Evolutionary Algorithms (Back and Schwefel, 1993)

	Evolution Strategies (ES)	*Evolutionary Programming (EP)*	*Genetic Algorithms (GA)*
Representation	Real-valued	Real-valued	Binary/real valued
Fitness is	Objective function value	Scaled objective value	Scaled objective value
Mutation	Main operator	Only operator	Background operator
Selection	Deterministic, extinctive	Probabilistic, extinctive	Probabilistic, preservative
Recombination	Different variants important for self adaptation	none	Main operator
Self-adaptation	Standard deviations and covariances	Variances (in meta-EP)	none

Although a complete review of EC is beyond the scope of this paper an overview of different types of EAs is given with an emphasis on showing how the various types of algorithm differ and the stages involved in defining each one.

EVOLUTIONARY PROGRAMMING

Evolutionary Programming (EP) models evolution as a process of adaptive species (Michaelwicz, 1999). A typical EP works as follows. An initial population is created using a random number generator and evaluated using the problem specific pay-off function. Each chromosome is then mutated to create a new population of off-springs. Mutation involves either, the addition, deletion, change of output, change of transition of a node, or a change of starting node. The off-springs are then evaluated and the better half of the combined set of parents and off-springs is used as the next population. This evolutionary process is repeated until an acceptable solution is found. A chromosome in EP encodes the behavior of an individual. Mutation is the only mating operator that is used and it is applied to every individual irrespective of their evaluated pay-off. Selection is made from the combined set of parents plus offspring.

EVOLUTION STRATEGIES

Evolution Strategies (ESs) models evolution as a process of the adaptive behaviour of individuals. Evolution strategies (Fogel, 1994; Schewel, 1994) use real variables and aim at numerical optimization. Because of that, the individuals incorporated can also a set of strategic parameters. Evolutionary strategies rely mainly on mutation operator (Gaussian noise with zero mean). The ESs evolve by making a series of discrete adjustments (i.e. mutations) to an experimental structure. After each adjustment, the new structure, i.e. the off-spring, is evaluated and compared to the previous structure, i.e. the parent. The better of the two is then chosen and used in the next cycle. As selection in this evolutionary cycle is made from one parent and one off-spring, the algorithm is known as a "(1 + 1)" ES.

These two-membered ESs modify (i.e. mutate) an n-dimensional real-valued vector $x \in \Re^n$ of object variables by adding a normally distributed random variable with expectation zero and standard deviation σ to each of the object variables x_i. The standard deviation is the same for all components of x, i.e. $\forall i \in \{1, 2..., n\} : x_i' = x_i + \sigma N_i(0, 1)$, where x' is the off-spring of x and $N_i(0; 1)$ is the realization of a normally distributed random variable with expectation 0 and standard deviation 1.

Since the introduction of ESs, two additional strategies have been developed: $(\mu + \lambda)$ and (μ, λ). Both of these ESs work on populations rather than single individuals and are referred to as multi-membered ESs. A $(\mu+\lambda)$ ES creates λ off-springs from μ parents and selects the best μ individuals from the combined set of μ parents plus λ off-springs to make the next population. A (μ, λ) ES, on the other hand, creates λ off-springs and selects the best μ individuals from the off-springs alone (in general, $1 \leq \mu \leq \lambda$).

GENETIC ALGORITHMS

Genetic Algorithm (GA) models evolution at the level of genetic chromosomes (i.e., the basic instructions for making things). The original version of a GA by John Holland (1975) is based on binary encoding of the solution parameters, and utilizes bit-flip mutation and n-point crossover. A sample structure of GAs is given in Figure 2. A popular selection operator for GAs has been proportional (or Roulette-wheel) selection, but because of its known drawbacks tournament selection and ranking selection are commonly used now-a-days. For numerical optimization, floating point encoding with Gaussian mutation, arithmetic crossover and tournament selection is a common choice. Moreover, an operation called elitism is remarkably important for the performance of a GA (Michalewicz and Fogel, 2000). The idea of elitism is to leave a certain proportion of the best individuals in every generation untouched by the variation operators. This idea is somewhat similar to Evolution Strategies and Evolutionary Programming (Fogel *et al.*, 1994), where a population of parents generates a new offspring by mutation in each iteration. The population of the next generation is created by selection from the elite parents and newly created off-springs.

Since the classic binary encoding GAs are extensively discussed in many studies (e.g. Goldberg, 1989; Deb, 1996), in this paper, we explain more on floating point or real value GA. There are many ways of implementing real coded GAs. However, this study presents a selective procedure, which appears to be efficient (Paterlini and Krink, 2006).

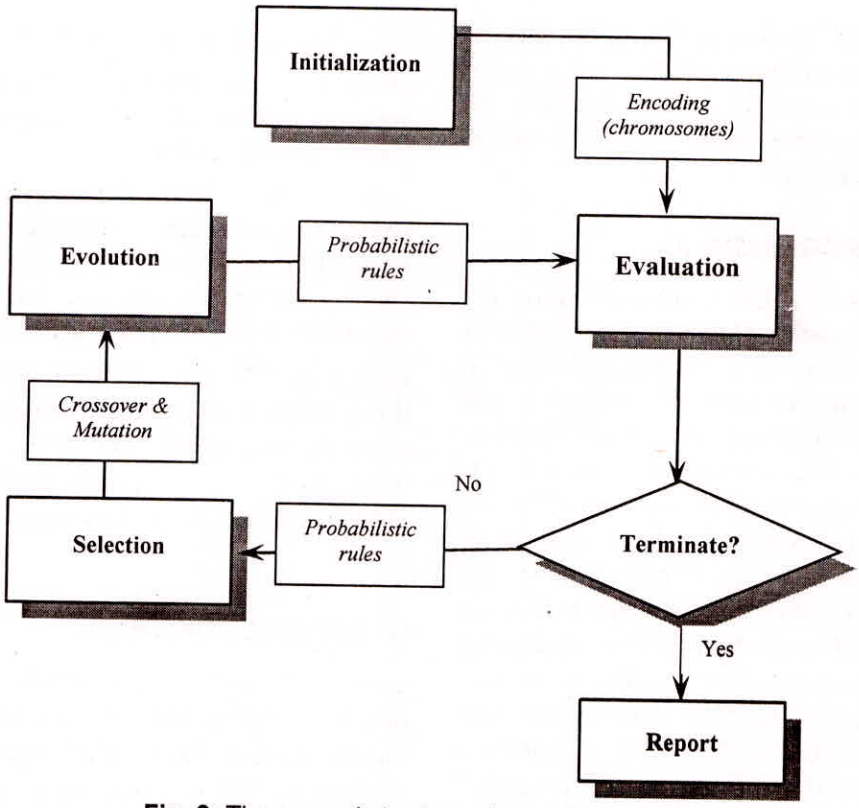

Fig. 2: The general structure of genetic algorithms

In real coded GA implementation, first a population of individuals containing the candidate solutions (encoded in floating point numbers) is created (initialization) and the fitness of each individual is evaluated by the fitness function (evaluation). For the initialization of the population, the GA uses randomly chosen object feature vectors from the data set. After initialization, the individuals stored in the variable *pop* are evaluated according to the fitness function and to determine the elite members. For elitism, the population is ranked to determine and mark the best individuals, which are left unchanged by selection, mutation and crossover during the next iteration. The population is iteratively refined by selection of individuals, application of mutation and crossover operators, re-evaluation of the new population according to the fitness function and updating of the elite. For selection, tournament selection (mostly of size 2) is used. The current population *pop* is saved as *oldpop* and for each individual *j* another individual *k* is randomly chosen from *oldpop*, and then the fitnesses compared. Substitute *j* by *k* in population *pop,* if *k*'s fitness is better. Before applying crossover the current population *pop is* saved again as *oldpop* and then arithmetic crossover is applied as follows,

$$pop(j)\_x_n = cw \times oldpop(j)\_x_n + (1 - cw) \times oldpop(k)\_x_n$$

where $pop(j)\_x_n$ is the nth solution parameter of individual j , $pop(j)$ is the offspring of the parents $oldpop(j)$ and $oldpop(k)$, cw is a uniform random weight of $U(0, 1)$, which is generated for each problem parameter n. For mutation, Gaussian mutation was chosen, such that,

$$pop(j)\_x_n = pop(j)\_x_n + \mathrm{N}(0,1) \times \sigma_m \times (xMax_n - xMin_n)$$

where $pop(j)\_x_n$ is again the nth solution parameter of individual j, $N(0, 1)$ is the Gaussian normal distribution with mean 0.0 and variance 1.0, and σ_m is the variance parameter of the mutation operator. $xMax_n$ and $xMin_n$ are the maximum and minimum search space bounds. The crossover and mutation operators are applied to each individual in the population, which is not in the elite and with a probability p_m for mutation and p_c for crossover, respectively. The algorithm terminates after a fixed number of iterations. The optimization result is the candidate solution and the fitness of the best individual in the last generation. The pseudo-code of this real coding GA is presented in Figure 3.

The other class of meta-heuristic techniques that are gaining more focus for optimization are Swarm

Intelligence (SI) techniques. Swarm Intelligence techniques that have been used for optimization of water resources were Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), etc. A brief note on them is given below.

SWARM INTELLIGENCE

Recently, a new research field arose called Swarm Intelligence (SI). SI argues that intelligent human cognition derives from the interaction of individuals in a social environment. The main ideas of socio-cognition can be effectively applied to develop stable and efficient algorithms for optimization tasks (Kennedy and Eberhart, 2001). Ant Colony Optimization (ACO) is the most well-known SI algorithm and is basically inspired from the foraging search behavior of real ants and their ability in finding shortest paths. They are mainly used for discrete combinatorial optimization tasks, exhibiting very interesting results in experiments as well as in real life applications (Dorigo and Di Caro, 1999). The Particle Swarm Optimization (PSO) technique is another SI technique, and has been originally proposed by Eberhart and Kennedy (1995) for continuous optimization tasks. PSO has been proved to be very efficient algorithm in solving hard optimization problems and engineering applications (Kennedy and Eberhart, 2001). In the following sections a brief description of basic principles and working of these two swarm intelligence techniques is given.

Initialize population *pop* "Create population from randomly chosen object vectors"
Evaluate population *pop* "Evaluate all candidate solutions"
Determine elite in *pop* "Rank population by fitness and mark the elite"

For a fixed number of iterations

"Apply selection: tournament selection of size 2"
oldpop = *pop*
for all individuals *j* in pop
    if individual *j*, i.e., *pop*(*j*), is not in the elite then
    select another individual *k* randomly, i.e., $k = U_{int}(0, popsize)$
    if individual *k* in *oldpop* is better than individual *j* in *pop* then
        pop(*j*) = *oldpop*(*k*)

"Apply crossover: arithmetic crossover"
oldpop = *pop*
for all individual *j* in pop
    if individual *j* is not in the elite and $U(0,1) < p_c$ then
    select another individual *k* randomly, i.e., $k = U_{int}(0, popsize)$
    for all candidate solution parameters *n* in *x*
        cw = U(0,1)
        $pop(j)\_x_n = cw.\ oldpop(j)\_x_n + (1- cw).\ oldpop(k)\_x_n$

"Apply mutation and check bounds: Gaussian mutation with fixed mutation rate"
for all individuals *j* in *pop*
    if individual *j* is not in the elite and $U(0,1) < p_m$ then
    for all candidate solution parameters *n* in *x*
    $pop(j)\_x_n = pop(j)\_x_n + N(0,1).\ \sigma_m\ .\ (xMax_n - xMin_n)$
        if $pop(j)\_x_n > xMax_n$ then $pop(j)\_x_n = xMax_n$
        if $pop(j)\_x_n < xMin_n$ then $pop(j)\_x_n = xMin_n$

evaluate population pop " Evaluate all candidate solutions"
determine elite in pop "Rank population by fitness and mark the elite"

Report the best recorded solution

$pop(j)\_x$ = candidate solution of individual j; $pop(j)\_x_n$ = nth problem parameter in the candidate solution of the j^{th} individual. $xMax_n$, $xMin_n$ = upper and lower search space bounds for problem parameter n. U(a, b) = uniform pseudo-random number in the interval [a, b]. U_{int}(a, b) = uniform pseudo-random integer number in the interval [a, b]. $N(\mu, \sigma)$ = normal (Gaussian) distributed pseudo random number with mean μ and variance σ.GA parameters: p_c = probability of crossover, p_m = probability of mutation, σ_m = mutation variance parameter.

Fig. 3: Pseudo-code of the Genetic Algorithm (GA)

ANT COLONY OPTIMIZATION

Ant Colony Optimization (ACO) is a population-based, general search technique for solution of difficult combinatorial and complex problems, which is inspired by the pheromone trail laying and training behavior of real ant colonies. Colorni and Dorigo (1991) developed the first ant system algorithm based on the foraging behavior exhibited by ant colonies in their search for food. Ant Colony Optimization (ACO) algorithms have been successfully applied to a number of benchmark combinatorial optimization problems, such as the traveling salesman and quadratic assignment problems (Dorigo *et al.*, 2000). The main features of ant colony optimization algorithm are pheromone trail and heuristic information. A pseudo-code of ACO algorithm is given in Figure 4 (Nagesh Kumar and Janga Reddy, 2006).

```
Begin
 Initialize
 While stopping criterion not satisfied do
                Position each ant in a starting node
                Repeat
                        For each ant do
                        Choose next node by applying
                        the state transition rule
                        Apply step by step pheromone
                        update
                        End for
                Until every ant has built a solution
                Update best solution
                Apply offline pheromone update
        End While
End
```

Fig. 4: Pseudo-code of Ant Colony Optimization (ACO) algorithm

Ant Colony Optimization has many features, which are similar to Genetic Algorithms (GAs) (Dorigo and Gambardella, 1997). Both ACO and GA are population based stochastic search techniques. GA works on the principle of natural evolution or survival of the fittest, where as ACO works on pheromone trail laying behaviour of ant colonies. GA uses crossover and mutation as prime operators in its evolution for next generation, where as ACO uses pheromone trail and heuristic information. The most important difference between GA and ACO algorithms is the way the trial solutions are generated. In ACO algorithms, trial solutions are constructed incrementally based on the information contained in the environment and the solutions are improved by modifying the environment through a form of indirect communication called stigmergy (Dorigo *et al.*, 2000). On the other hand, in GAs the trial solutions are in the form of strings of genetic materials and new solutions are obtained through modification of the previous solutions (Maier *et al.*, 2003). Thus, in GAs the memory of the system is embedded in the trial solutions, whereas in ACO algorithms the system memory is contained in the environment itself.

As any direct search method like GA, the ACO model is also quite sensitive to setup parameters and so it is important to fine-tune the parameters for a particular problem of interest, before actually applying the same to the problem (Dorigo *et al.*, 2000).

PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) technique is a Swarm Intelligence (SI) method inspired by social behavior of bird flocking or fish schooling. PSO, originally proposed by Eberhart and Kennedy (1995), is a population based heuristic search technique for solving continuous optimization problems. PSO shares many similarities with evolutionary computation techniques such as GA. PSOs are initialized with a population of random solutions and searches for optima by updating generations. However, in contrast to methods like GA's, in PSO, no operators inspired by natural evolution are applied to extract a new generation of candidate solutions. Instead, PSO relies on the exchange of information between individuals (particles) of the population (swarm). In affect, each particle adjusts its trajectory towards its own previous best position and towards the best previous position attained by any other member of its neighborhood (usually the entire swarm) (Kennedy and Eaberhart, 2001).

Suppose the search space is D-dimensional, then the i^{th} individual (*particle*), of the population (swarm), can be represented by a D-dimensional vector, $X_i = (x_{i1}, x_{i2}, ..., x_{iD})^T$. The *velocity* (position change) of this particle, can be represented by another D-dimensional vector $V_i = (v_{i1}, v_{i2,..}, v_{iD})^T$. The best previously visited position of the i^{th} particle is denoted as $P_i = (p_{i1}, p_{i2, ...}, p_{iD})^T$. Defining g as the index of the best particle in the swarm (i.e. the g^{th} particle is the best), and superscripts denoting the iteration number, the swarm is manipulated according to the following two equations,

$$v_{id}^{n+1} = \omega\, v_{id}^{n} + c_1 r_1^n (p_{id}^n - x_{id}^n) + c_2 r_2^n (p_{gd}^n - x_{id}^n) \quad \ldots (1)$$

$$x_{id}^{n+1} = x_{id}^{n} + v_{id}^{n+1} \quad \ldots (2)$$

where, $d = 1, 2, \ldots, D$; $i = 1, 2, \ldots, N$, and N is the size of the swarm; ω = inertial weight; c_1 and c_2 are positive constants; r_1, r_2 are random numbers, uniformly distributed in [0, 1]; and $n = 1, 2, \ldots$, the iteration number.

In PSO algorithm each particle is initialized with a random swarm of particles, and random velocity vectors. Then the fitness of each particle is evaluated by the fitness function. Two 'best' values are defined, the global and the personal best. The global best is the highest fitness value in an entire run (best solution so far), and the personal best is the highest fitness value of a specific particle. Each particle is attracted towards the location of the 'best fitness achieved so far' across the whole population. In order to achieve this, a particle stores the previously reached 'best' positions in a cognitive memory. The relative 'pull' of the global and the personal best is determined by the acceleration constants c_1 and c_2. After this update each particle is then revaluated. If any fitness is greater than the global best, then the new position becomes the new global best. If the particle's fitness value is greater than the personal best, then the current value becomes the new personal best. This procedure is repeated till the termination criteria is satisfied. The pseudo-code of the PSO algorithm is given Figure 5 (Janga Reddy and Nagesh Kumar, 2007).

```
Begin
 Initialize swarm position X(0) and velocities V(0)
 Set iteration counter, n = 0
 Repeat
        Compute fitness function for each individual of
        swarm
         Compute PBest(n) and GBest
         Begin (Perform PSO operations)
         Compute V(n +1) from Eq. (1)
         Compute X(n +1) from Eq. (2)
         End
         Set n = n + 1
 Until termination criteria is satisfied
End
```

Fig. 5: Pseudo-Code of the Particle Swarm Optimization (PSO) algorithm

APPLICATIONS OF EVOLUTIONARY ALGORITHMS IN RESERVOIR OPERATION

The EC methods have emerged as a powerful tool for optimization and management of water resources problems. There are numerous applications of Evolutionary Algorithms for water related problems, viz., reservoir operation (e.g., Oliveira and Loucks, 1997), water distribution systems design (e.g., Savic and Walters, 1997; Babayan *et al.*, 2005), ground water remediation (e.g., Maskey *et al.*, 2002; Hilton and Culver, 2005), parameter estimation in hydrological modeling (e.g., Vrugt *et al.*, 2003 & 2005), Watershed Management (e.g., Muleta and Nicklow, 2005) etc. However, this paper mainly focuses on reviewing EAs applications to reservoir operation problems only. Recently various exciting techniques were proposed for optimization of reservoir management problems viz., GAs, PSO, and ACO techniques.

Genetic Algorithms

Esat and Hall (1994) applied a GA to the four-reservoir problem, suggesting that GAs have potential in water resources optimization with significant savings in computer memory and execution times.

Oliveira and Loucks (1997) used genetic algorithms to derive the multi-reservoir operating policies. The genetic algorithms use real-valued vectors containing information needed to define both system release and individual reservoir storage volume targets as functions of total storage in each of multiple within-year periods. Genetic operators are used to generate successive sets of possible operating policies. Each policy is then evaluated using simulation to compute a performance index for a given flow series. The better performing policies are then used as a basis for generating new sets of possible policies. The process of improved policy generation and evaluation is repeated until no further improvement in performance is obtained. They demonstrated the methodology application through an example reservoir system for water supply and hydropower.

Chang and Chen (1998) examined two types of genetic algorithms, real-coded and binary-coded, for function optimization and applied to the optimization of a flood control reservoir model. They found that both genetic algorithms are more efficient and robust than the random search method; and the real-coded GA was performing better in terms of efficiency and precision than the binary-coded GA.

Wardlaw and Sharif (1999) evaluated several alternative formulations of a genetic algorithm for four-reservoir, deterministic, finite-horizon problem. They found that the most promising genetic algorithm approach for the four-reservoir problem comprises real-value coding, tournament selection, uniform crossover, and modified uniform mutation; and the

real-value coding operates significantly faster than binary coding and produces better results. They also suggested that a genetic algorithm can be satisfactorily used in real time operations with stochastically generated inflows and has potential as an alternative to stochastic dynamic programming approaches.

Sharif and Wardlaw (2000) used genetic algorithm model for the optimization of reservoir systems in Indonesia. They compared the genetic algorithm results with those produced by discrete differential dynamic programming and found that the genetic algorithm results are very close to the optimum, and the technique appears to be robust. Contrary to methods based on dynamic programming, discretization of state variables is not required; and there is no requirement for trial state trajectories to initiate the search using a genetic algorithm. But they observed that as the number of decision variables (chromosome length) increases with the number of reservoirs and planning periods, it makes increasingly difficult to satisfy the problem constraints using GAs.

Kuo *et al.* (2000) applied Genetic Algorithm (GA) to irrigation planning through a case study for optimizing economic profits, simulating the water demand, crop yields, and estimating the related crop area percentages with specified water supply and crop area constraints. They suggested that GAs can be used as an effective tool in decision support systems for irrigation project planning.

Chang and Chang (2001) presented an approach based on the Genetic Algorithm (GA) and the Adaptive Network-based Fuzzy Inference System (ANFIS) to improve real-time reservoir operation. The GA is used to search the optimal reservoir operating histogram based on a given inflow series, which can be recognized as the base of input-output training patterns in the next step. The ANFIS is then built to create the fuzzy inference system, to construct the suitable structure and parameters, and to estimate the optimal water release according to the reservoir depth and inflow situation. They demonstrated its applicability for operation of the Shihmen reservoir, Taiwan and found that the approach gave superior performance with regard to the prediction of total water deficit and generalized shortage index compared to the existing policies.

Cai *et al.* (2001) presented a combined GA and Linear Programming (LP) strategy for solving large nonlinear problems. They used GA to optimize reservoir surface levels, called "complicating variables", for linearizing the operation problem in each time period to be later solved sequentially for different time periods. They suggested that, if careful choices of the complicating variables are made, a fairly standard GA is capable of finding high quality solutions to reservoir problems in reasonable computing times. Merabtene (2002) developed a risk assessment model for optimal drought management of an integrated water resources system using a genetic algorithm and found that GAs can be used as an effective tool for drought management.

Ponnambalam *et al.* (2003) employed soft computing based tools viz., Fuzzy Inference Systems (FIS), Artificial Neural Networks (ANN), and Genetic Algorithms (GA) for the optimization of reservoir operation, considering the objective of minimization of variance of benefits. First, general operating rules were developed by both ANFIS-based fuzzy rules and compared with multiple regression methods. It was found that the ANFIS rules perform much better than the regression rules for different levels of uncertainty of inflows. However, ANFIS requires a large number of parameters and a sophisticated fitting method than the regression method, and also it requires a larger set of training data for parameter estimation. This was overcome by simulating the optimal solution of a non-linear optimization scheme over a long-term horizon. For considering the risk aversion characteristic of the operating rules, a parameterized T-norm operator was employed and GA was used to optimize the value of those parameters while simulating the ANFIS rules.

Chaves (2003) proposed a methodology for the assessment of planning operations of a storage reservoir considering both water quantity and quality. For optimization purpose, dynamic programming combined with stochastic techniques to handle the probabilistic characteristics of inflow quantity and quality, and then genetic algorithm model was used to carryout the sensitivity analysis.

Chang *et al.* (2003) developed two models combining the reservoir simulation model and the sediment flushing, to satisfy the water demand and water consumed in the flushing operation of a reservoir system. In the reservoir simulation model, the Genetic Algorithm (GA) is used to optimize and determine the flushing operation rule curves. The sediment-flushing model estimates the amount of the flushed sediment volume, and the simulated results update the elevation-storage curve, which can be taken into account in the reservoir simulation model. By applying the approach to Tapu reservoir, they found that the developed models can provide significant benefits over the current practicing methods.

Labadie (2004) in his comprehensive in-depth review for multi-reservoir system operation scrutinized

the ability of GAs and suggested that GAs can be linked with trusted simulation models to gain advantage in complex reservoir system operations. Srinivasa Raju and Nagesh Kumar (2004) applied Genetic Algorithms (GA) to develop efficient cropping pattern for maximizing benefits for an irrigation project in India. Results obtained by GA are compared with Linear Programming solution and found that they are reasonably close.

Ahmed and Sharma (2005) used Genetic Algorithms (GA) model for finding the optimal operating policy of a multi-purpose reservoir, located on the river Pagladia, a major tributary of the river Brahmaputra. The policies derived by the GA model are compared with those of the Stochastic Dynamic Programming (SDP) model on the basis of their performance in reservoir simulation for 20 years of historic monthly streamflow, and they observed that GA-derived policies are promising and competitive and can be effectively used for reservoir operation.

Reis *et al.*, (2005) proposed an approach using Genetic Algorithm (GA) and Linear Programming (LP) to determine operational decisions for reservoirs of a hydro system throughout a planning period, with the possibility of considering a variety of equally likely hydrologic sequences representing inflows. The GA-LP approach permits the evaluation of a reduced number of parameters by GA and operational variables by LP, and also provides easiness in implementation and helps to extract useful parameters for future operational decisions.

Jian-Xia *et al.* (2005) successfully applied two forms of genetic algorithms viz., binary-coded and real-coded to optimal reservoir dispatching problem and found that the real-value coding is proved to be significantly faster than binary coding, and is producing better results.

Chang *et al.* (2005a) investigated the efficiency and effectiveness of two Genetic Algorithms (GAs), i.e., binary-coded and real-coded, to derive multipurpose reservoir operating rule curves. The applicability and effectiveness of the GA methods are tested on the operation of the Shih-Men reservoir in Taiwan. They observed that the GAs provide an adequate, effective and robust way for searching the rule curves and the real-coded GA is more efficient than the binary-coded GA.

Chang *et al.* (2005b) developed an intelligent control system for reservoir operation. The methodology includes two major processes, the knowledge acquired and implemented, and the inference system. A Genetic Algorithm (GA) was employed to extract knowledge based on the historical inflow data with a design objective function and on the operating rule curves respectively. The Adaptive Network-based Fuzzy Inference System (ANFIS) is then used to implement the knowledge, to create the fuzzy inference system, and then to estimate the optimal reservoir operation policies. They demonstrated the applicability of the methodology to a case study of Shihmen reservoir, Taiwan. They concluded that (1) the GA is an efficient way to search the optimal input-output patterns, (2) the Fuzzy Rule Base (FRB) can extract the knowledge from the operating rule curves, and (3) the ANFIS models built on different types of knowledge can produce much better performance than the traditional rule curves in real-time reservoir operation.

Apart from single objective optimization, evolutionary algorithms (MOEAs) have also been applied for multi-objective reservoir operation problems. Kim *et al.* (2005) applied NSGA-II to multi-reservoir system optimization in the Han River basin. Two objective functions and three cases having different constraint conditions were used to achieve non-dominated solutions and found that multi-objective genetic algorithms can be very much useful in decision making for multi-reservoir systems.

Janga Reddy and Nagesh Kumar (2006) employed a Multi-Objective Evolutionary Algorithm (MOEA) to derive optimal operation policies for a multipurpose reservoir system. They demonstrated its applicability through a case study of Bhadra reservoir system, India, having multiple objectives of maximizing benefits from irrigation, hydropower generation and water quality requirements. The model is applied for three different inflow scenarios and the corresponding Pareto optimal fronts were obtained. Then, three kinds of priorities of the multiple objectives were analyzed and the respective operating policies obtained. It was found that MOEAs are able to find a set of well distributed optimal solutions along the Pareto front in a single simulation run and can overcome some of the limitations of traditional multi-objective optimization techniques.

By integrating Pareto optimality principles into Differential Evolution (DE) algorithm, Janga Reddy and Nagesh Kumar (2007a) proposed an efficient and effective approach for multi-objective optimization of water resource problems, namely Multi-Objective Differential Evolution (MODE) algorithm. This approach uses, non-dominated sorting, ranking, and crowding distance assignment procedures, and also maintains an external archive to maintain the best non-inferior solutions explored over the generations. By

applying to a case study in multi-objective reservoir operation, it is found that differential evolution algorithm can be a suitable algorithm for problems having interdependence among the decision variables and MODE can be a viable alternative for generating optimal trade-offs in multi-objective optimization of water resources systems.

SWARM INTELLIGENCE APPLICATIONS IN RESERVOIR OPERATION

Nagesh Kumar and Janga Reddy (2006) proposed Ant Colony Optimization (ACO) procedure to derive operating policies for a multi-purpose reservoir system and demonstrated its applicability through a case study of Hirakud reservoir, India. The ACO model formulation for reservoir operation was approached by considering a finite time series of inflows, classifying the reservoir volume into several class intervals, and determining the reservoir release for each period with respect to a predefined optimality criterion. The model is formulated to maximize the hydropower production from the reservoir by considering the given priorities and constraints. The ACO procedure was employed for monthly reservoir operation, and consists of two models viz., short-time horizon operation (yearly operation model) and long-time horizon operation model (36 years at a time). To evaluate the performance of ACO, the developed models are also solved using real-coded Genetic Algorithm (GA). They found that ACO model outperforms GA model, especially in the case of long-time horizon reservoir operation, consequently which might help to evolve better operation policies by relaxing the over year storage requirements.

Janga Reddy and Nagesh Kumar (2005a) applied Particle Swarm Optimization (PSO) technique to derive operation policy for a four reservoir system operation problem. They compared PSO performance with dynamic programming and GA results, and found that PSO provides quick convergence to near optimal solutions and can be used as an efficient alternative for non-linear optimization. Janga Reddy and Nagesh Kumar (2005b) also presented a multi-objective PSO algorithm for multi-objective reservoir operation problems and by applying it to a case study, they demonstrated its utility for water resources management. The developed MOPSO uses the non-dominated sorting concept, and parameter free-niching scheme to promote solution diversity.

Nagesh Kumar and Janga Reddy (2007) improved the standard PSO algorithm by incorporating a special operator called elitist-mutation and developed Elitist-Mutated Particle Swarm Optimization (EMPSO) technique. They first tested a hypothetical multi-reservoir system and compared the results with other techniques, it was found that EMPSO is quite promising, saving significant computational time and function evaluations by quickly converging to global optimal solutions. To show practical utility, EMPSO was then applied to a realistic case study, Bhadra reservoir system in India, and it was found that EMPSO is consistently performing better than the standard PSO and Genetic Algorithm (GA) techniques.

Later by utilizing the strengths of PSO, quick convergence and yielding efficient solutions for single objective optimization, by integrating Pareto dominance principles into Particle Swarm Optimization (PSO) algorithm, Janga Reddy and Nagesh Kumar (2007b) developed an efficient approach for multi-objective optimization namely, elitist-mutated multi-objective PSO (EM-MOPSO) algorithm and evaluated its performance for several standard test problems and also for a multi-objective reservoir operation problem. This method uses special operators of variable size, external repository and an efficient Elitist-Mutation (EM) operator to properly maintain diversity and explore efficient Pareto frontiers. By applying EM-MOPSO for a case study of multi-objective reservoir operation, they found that the approach is fast and reliable, and is yielding wide spread of Pareto optimal solutions in a single run. Overall the stochastic search techniques are gaining more and more popularity for optimization of real world problems and have wider scope to solve complex systems.

CONCLUDING REMARKS

Evolutionary Computation (EC) is a rapidly expanding area of artificial intelligence research. The use of evolutionary algorithms for solving optimization problems has become very extensive in the last few years. The main advantage of EC algorithms is the usage of a population of potential solutions that explore the search space simultaneously, exchanging information among them and uses only objective function values and not derivatives of the objective function. Also EAs are stochastic search algorithms, can move to any complicated search space and locate global optimal solutions. Evolutionary Computation (EC) methods provide solutions to many complex optimization problems that are difficult to cope with using the traditional gradient based methods, due to their nature that may imply discontinuities of the

search space, non-differentiable objective functions, imprecise arguments and function values.

Some of the remarks on the current state of the art in Evolutionary Computation:

- There is no general algorithm that can be applicable to all problems, as the efficiency varies as a function of problem size and complexity. However, incorporation of problem specific knowledge and heuristics will help to achieve faster and efficient solutions to real world problems.
- Most EAs converge to an optimal point starting either from inside or outside the feasible region.
- EAs may require calibration of the search parameters to ensure efficient convergence.
- EAs do not take into account shape or gradient of the objective function.
- For complex problems, EAs may require large number of simulations to find an optimal/near optimum solution.

However, recently several new ideas/algorithms are being proposed, which showed improved performance. Therefore, EC is increasingly gaining interest among the research community to employ EC methods for many practical and industrial applications.

REFERENCES

Abbaspour, K.C., Schulin, R. and van Genuchten, M.T. (2001). Estimating unsaturated soil hydraulic parameters using ant colony optimization. *Adv. Water Resour.* 24(8), 827–933.

Ahmed, J.A. and Sharma, A.K. (2005). Genetic algorithm for optimal operating policy of a multi-purpose reservoir, 19(2), 145–161.

Babayan, A., Kapelan, Z. Savic, D. and Walters, G.A. (2005). Least-Cost Design of Water Distribution Networks under Demand Uncertainty, J. Water Resour. Plng. and Mgmt., 131(5), 2005.

Back, T. and Schwefel, H.P. (1993). An overview of evolutionary algorithms for parameter optimization, Evolutionary computation 1: 1–23.

Cai, X., McKinney, D.C. and Lasdonc, L.S. (2001). Solving non-linear water management models using a combined genetic algorithm and linear programming approach, Advances in Water Resources, 24(6), 667–676.

Chang, L.C. and Chang, F.L. (2001). Intelligent control for modelling of real-time reservoir operation, Hydrological Processes, 2001: 15(9), 1621–1634.

Chang, F.J., Chen, L. and Chang, L.C. (2005a). Optimizing the reservoir operating rule curves by genetic algorithms. Hydrological Processes, 19 (11), 2277–2289.

Chang, Y.T., Chang, L.C. and Chang, F.J. (2005b). Intelligent control for modeling of real-time reservoir operation, part II: artificial neural network with operating rule curves, Hydrological Processes, 19(7), 1431–1444.

Chang, F.J. and Chen L. (1998). Real-coded genetic algorithm for rule-based flood control reservoir management. *Water Resour. Manage.*, 12(3): 185–198

Chang, F.J., Lai, J.S. and Kao, L.S. (2003). Optimization of operation rule curves and flushing schedule in a reservoir, Hydrological Processes, 2003: 17(8), 1623–1640.

Chen, Y.M. (1997). Management of water resources using improved genetic algorithms, Computers and Electronics in Agriculture, 18(2–3), 117–127.

Chaves, P., Kojiri, T. and Yamashiki, Y. (2003). Optimization of storage reservoir considering water quantity and quality, Hydrological Processes, 17 (14), 2769–2793.

Colorni, A., Dorigo, M. and Maniezzo, V. (1991) Distributed optimization by ant colonies. *Proc., 1st European Conf. on Artificial Life*, Cambridge, MIT Press, 134–142.

Dasgupta, D. and Michalewicz, Z. (1997). Evolutionary algorithms in Engineering Applications. Germany, Springer.

Deb, K. (1996). Optimization for engineering design: Algorithms and examples. New Delhi, Prentice-Hall.

Dorigo, M. and Di Caro, G. (1999). The Ant Colony Optimization Meta-Heuristic, in D. Corne, M. Dorigo, and F. Glover (Eds.), *New Ideas in Optimization*, 1999.

Dorigo, M. and Gambardella, L.M. (1997) Ant colony system: A co-operative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* 1(1), 53–66.

Eberhart, R.C. and Kennedy, J. (1995). A New Optimizer Using Particle Swarm Theory, Proceedings Sixth Symposium on Micro Machine and Human Science, IEEE Service Center, Piscataway, NJ, 39–43.

Esat, V., Hall, M.J. (1994). Water resources system optimization using genetic algorithms. In *Proceedings of the 1st International Conference on Hydroinformatics*. Balkema: Rotterdam; 225–231.

Fogel, L.J. (1994). Evolutionary programming in perspective: the topdown view, in J. Zurada, R. Marks II, and C. Robinson (Eds.), *Computational Intelligence: Imitating Life*, 1994, pp. 135–146.

Fogel, L.J., Owens, A.J. and Walsh, M.J. (1966). *Artificial Intelligence through Simulated Evolution.* New York: Wiley.

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

Hilton, A.B.C. and Culver, T.B. (2005). Groundwater remediation design under uncertainty using genetic algorithm. *J. Water Resour. Plng. and Mgmt*, 131(1), 25–34.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*, The MIT Press, 1975.

Janga Reddy, M. and Nagesh Kumar, D. (2005a). Multi-reservoir operation using particle swarm optimization, Proc. of Intl. Conf. on Hydrological Perspectives for Sustainable Development (HYPESD-2005), IIT Roorkee, India, 2, pp. 556–564.

Janga Reddy, M. and Nagesh Kumar, D. (2005b). Multi-objective particle swarm optimization for optimal reservoir operation, Proceedings of 2nd Indian International Conference on Artificial Intelligence (IICAI-2005), Pune, India, pp. 1183–1192.

Janga Reddy, M. and Nagesh Kumar, D. (2006). Optimal reservoir operation using multi objective evolutionary algorithm. *Water Resour. Manage.*, 20(6), 861–878.

Janga Reddy, M. and Nagesh Kumar, D. (2007a) Multi-objective Differential Evolution with application to reservoir system optimization. J. Comp. Civil Engrg., ASCE, 21(2), 136–146.

Janga Reddy, M. and Nagesh Kumar, D. (2007b). Multi-objective particle swarm optimization for generating optimal trade-offs in reservoir operation, Hydrological Processes, Wiley InterScience, in print, available online.

Jian-Xia, C., Qiang, H. and Yi-min, W. (2005). Genetic Algorithms for Optimal Reservoir Dispatching, *Water Resour. Manage.*, 19(4), 2005, 321–331.

Kennedy, J. and Eberhart, R.C. (2001). *Swarm Intelligence*, Morgan Kaufmann, 2001.

Kim, T., Heo, J.H. and Jeong, C.S. (2005). Multi-reservoir system optimization in the Han River basin using multi-objective genetic algorithms, Hydrological Processes, 2005, 10.1002/hyp.6047.

Koza, J.R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT Press, 1992.

Kuo, S.F., Merkley, G.P. and Liu, C.W. (2000). Decision support for irrigation project planning using a genetic algorithm, Agricultural Water Management, Volume 45, Issue 3, August 2000, pp. 243–266.

Labadie, J.W. (2004). Optimal Operation of Multi-reservoir Systems: State-of-the-Art Review, J. Water Resour. Plng. and Mgmt., 130(2), 93–111 .

Maskey, S., Jonoski, A. and Solomatine, D.P. (2002). Groundwater Remediation Strategy Using Global Optimization Algorithms, J. Water Resour. Plng. and Mgmt., 128(6), 431–440.

Maier, H.R *et al.* (2003). Ant colony optimization for design of water distribution systems. *J. Water Resour. Plng. and Mgmt.* 129(3), 200–209.

Merabtene, T., Kawamura, A., Jinno, K. and Olsson, J. (2002). Risk assessment for optimal drought management of an integrated water resources system using a genetic algorithm, Hydrological Processes, 16 (11), 2189–2208.

Michalewicz, Z. (1999). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, 1999.

Muleta, M.K. and Nicklow, J.W. (2005). Decision Support for Watershed Management Using Evolutionary Algorithms. *J. Water Resour. Plng. and Mgmt,* 131(1), 35–44.

Nagesh Kumar, D. and Janga Reddy, M. (2006). Ant colony optimization for multi-purpose reservoir operation. *Water Resour. Manage.*, 20(6), 879–898.

Nagesh Kumar, D. and Janga Reddy, M. (2007). Multi-purpose reservoir operation using particle swarm optimization. J. Water Res. Plan. Manage., ASCE, 133 (3), 1–10.

Oliveira, R. and Loucks, D.P. (1997). Operating rules for multi-reservoir systems, Water Res. Res. 33(4), 839–852.

Paterlini, S. and Krink, T. (2006). Differential evolution and particle swarm optimization in partitional clustering, Comp. Stat. & Data Analysis, 50, 1220–1247.

Ponnambalam, K., Karray, F. and Mousavi, S.J. (2003). Minimizing variance of reservoir systems operations benefits using soft computing tools, Fuzzy Sets and Systems, 2003:139(2), 451–461.

Reis, L.F.R., Walters, G.A., Savic, D. and Chaudhry, F.H. (2005). Multi-reservoir operation planning using hybrid genetic algorithm and linear programming (GA-LP): an alternative stochastic approach, *Water Resour. Manage.*, 19(6), 831–848.

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

Schwefel, H.-P. (1994). On the evolution of evolutionary computation, in J. Zurada, R. Marks II, and C. Robinson (Eds.), *Computational Intelligence: Imitating Life.*

Sharif, M. and Wardlaw, R. (2000). 'Multi-reservoir systems optimization using genetic algorithm: Case study', J. Comp. Civil Engrg., ASCE, 14(4), 255–263.

Srinivasa Raju, K. and Nagesh Kumar, D. (2004). Irrigation Planning using Genetic Algorithms, *Water Resour. Manage.*, 18(2), 2004, 163–176 .

Wardlaw, R. and Sharif, M. (1999). Evaluation of genetic algorithms for optimal reservoir system operation, J. Water Res. Plan. Manage. ASCE 125(1), 25–33.

Wardlaw, R. and Bhaktikul, K. (2004). Comparison of Genetic Algorithm and Linear Programming Approaches for Lateral Canal Scheduling. J. Irrig. and Drain. Engrg., 130 (4), 311–317.