Computers & Industrial Engineering 65 (2013) 310–321

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Use of parallel deterministic dynamic programming and hierarchical adaptive genetic algorithm for reservoir operation optimization

Zhongbo Zhang [a], Shuanghu Zhang [b,*], Yuhui Wang [c], Yunzhong Jiang [b], Hao Wang [a,b]

[a] State Key Laboratory of Hydraulic Engineering Simulation and Safety, Tianjin University, Tianjin 300072, China
[b] State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, China Institute of Water Resources and Hydropower Research, Beijing 100038, China
[c] College of Environmental Science and Engineering, Donghua University, Shanghai 201620, China

ARTICLE INFO

Article history:
Received 21 September 2012
Received in revised form 21 December 2012
Accepted 3 February 2013
Available online 26 February 2013

Keywords:
Parallel
Reservoir operation optimization
Hierarchy
Algorithm

ABSTRACT

Reservoir operation optimization (ROO) is a complicated dynamically constrained nonlinear problem that is important in the context of reservoir system operation. In this study, parallel deterministic dynamic programming (PDDP) and a hierarchical adaptive genetic algorithm (HAGA) are proposed to solve the problem, which involves many conflicting objectives and constraints. In the PDDP method, multi-threads are found to exhibit better speed-up than single threads and to perform well for up to four threads. In the HAGA, an adaptive dynamic parameter control mechanism is applied to determine parameter settings, and an elite individual is preserved in the archive from the first hierarchy to the second hierarchy. Compared with other methods, the HAGA provides a better operational result with greater effectiveness and robustness because of the population diversity created by the archive operator. Comparison of the results of the HAGA and PDDP shows two contradictory objectives in the ROO problem-economy and reliability. The simulation results reveal that: compared with proposed PDDP, the proposed HAGA integrated with parallel model appears to be better in terms of power generation benefit and computational efficiency.

1. Introduction

The water resource crisis is increasingly becoming challenging and complicated, posing a dilemma for stakeholders desiring effective water allocation. Reservoir operation optimization (ROO) facilitates not only water resource allocation that yields maximum benefits with respect to multiple objectives in areas such as agriculture, energy, and industry but also rational water exploitation and hydropower generation. Moreover, for obtaining solutions to ROO problems, advanced computational technology and improved algorithms are used for enhancing the computation efficiency. ROO can be used for formulating, analyzing, and solving operation optimization problems in water resource planning (Loucks, Stedinger, & Haith, 1981; Yeh, 1985).

Most methods used for ROO analysis involve conventional optimization algorithms and various metaheuristic algorithms. Over the past several decades, a wide range of methods have been proposed to solve ROO problems. Those reported to be effective are linear programming (LP) (Jabr, Coonick, & Cory, 2000), nonlinear programming (NLP) (Chen, 2007; Takriti & Krasenbrink, 1999), quadratic programming (QP) (Papageorgiou & Fraga, 2007), and Lagrangian relaxation (LR) (Hindi & Ghani, 1991; Keib, Ma, & Hart, 1994). Dynamic programming (DP) is a powerful optimization technique that is applied to ROO and is considered to be a conventional optimization algorithm in reservoir operation. Among the traditional optimization techniques for reservoir operation, DP (Travers & Kaye, 1998) boasts high popularity. In other methods, there may be difficulties in finding the optimal solution. In the case of LP, the nonlinear and unsmooth characteristics of ROO problems are often ignored during linearization, generating large errors in the optimal operation. In NLP and QP, the objective function should be continuous and differentiable. Moreover, some approximations are necessary in the formulation when NLP and QP are used, and they may lead to inaccurate solutions when an continuous and differentiable objective function is used. In LR, Lagrange multipliers with updating strategy are used, and therefore, the method suffers from oscillations in the optimal result. In contrast, DP imposes no restrictions on the unsmooth and nonconvex nature of the ROO problem, which makes it superior. The ROO problem is a highly constrained nonlinear discrete dynamic optimization problem solved by complete enumeration with a DP solver.

The theory of DP, first used by Bellman in the early 1950s (Bellman, 1957), resulted in the principle of optimality. This principle is useful for solving problems with a separated monotonic criterion function. DP is a method for efficiently solving a broad range of searching and optimization problems. It involves the use of the principle of optimality in the optimal region, and a recursive algorithm for dividing a decision process into several interrelated

* Corresponding author. Tel.: +86 13581819570.
E-mail address: zzbainiya@163.com (S. Zhang).

0360-8352/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cie.2013.02.003

stages in terms of time, space, etc. Generally, the major objective of the ROO problem at a hydropower station aims to maximize the total power generation by utilizing the limited water resource for future stages. The application of DP to ROO is popular since the problem has been formulated as a differential DP problem. The theory of DP for large-scale problems has been studied by Yakowitz (1982), who reviewed the use of the method for solving a water resource planning problem. DP models can be classified into two categories: stochastic dynamic programming models with uncertainty factors, and deterministic DP (DDP) models. For reservoir operating rule generation, Karamouz and Houck (1987) compared stochastic DP (SDP) and DDP with regression (DPR). They found that the DPR model performs better for large reservoirs with capacities exceeding 50% of the mean annual flow. DDP has also been applied to an ROO problem in which uncertain factor evaluation is not considered, and it may actually provide the optimal result. Although the DDP can solve the reservoir optimization problem, the high dimensionality of the problem poses difficulties and might not converge within a reasonable time, especially for large-scale hydropower systems. The practical limitation of DP is obvious as the number of possible decisions increases with the number of reservoirs. Further, the computational accuracy should be improved, which would result in a low computational efficiency. Therefore, the implementation of DDP in parallel, which would reduce the run-time, would significantly increase the efficiency of such parallel model.

Some hydraulic models, such as TRENT, CalTWiMS, TELEMAC, and RMA, use the message passing interface (MPI) (Gropp, Lusk, & Skjellum, 1994) and domain decomposition for simulations in parallel. Recently, the JFLOW diffusive wave model used graphics processing units (GPUs). However, these parallel tools hide a considerable amount of complexity. The computational strength of modern microprocessors has increased with the development of various commercial multi-threaded processors. Nowadays, the most popular parallel systems are based on shared memory. We can use the multi-threading feature of the hardware, which distinguishes these systems from traditional shared multi-processor systems, distributed memory systems, or hybrid distributed-shared memory systems. Consequently, increasing the thread-level parallelism becomes paramount for achieving the maximum potential performance from shared-memory multi-processor systems built with shared-memory threading processors. The parallel implementation of DDP involving complex data structures and memory allocation is used for ROO with a shared memory system that employs open multiprocessing (OpenMP). OpenMP (Chapman, Jost, & van der Pas, 2007) is a set of compiler directives with library routines that is used for creating an application program interface (API) for supporting multi-platform shared-memory parallel programming in FORTRAN, C, and C++ on all architectures, including UNIX and Windows NT platforms. Key advantages of OpenMP include easy implementation and the minimal recoding required for implementation in parallel. An OpenMP code can be developed on one platform and deployed on any other platform installing an OpenMP compiler. These parallel machines are built upon a set of processors that have access to a common memory by exchanging data between concurrent tasks without communicating with any processor and that communicate data by writing or reading shared data. We adopt a parallel DDP (PDDP) FORTRAN code that is implemented in parallel for ROO.

Owing to the lack of computational efficiency for ROO in the case of DP, modern heuristic stochastic search algorithms such as the genetic algorithm (GA) (Baskar, Subbaraj, & Rao, 2003; Chiang, 2007), evolutionary programming (EP) (Basu, 2004), simulated annealing (SA) (Basu, 2005), particle swarm optimization (PSO) (Cai, McKinney, & Lasdon, 2001), and the differential evolution algorithm (DE) have been extensively used to solve the ROO problem without ant restriction on the unsmooth and nonconvex characteristics of the problem. In recent decades, the GA has become widely used for the application of global optimization to ROO. Because of the increase in the complexity of ROO with the dimensionality, a large-scale ROO problem with an enormous number of variables and constraints must be decomposed into sub-problems to enhance the robustness of searching. Therefore, ROO problems can be alternatively solved by the GA. The genetic algorithm (Baskar et al., 2003) was formally introduced in the 1970s by John Holland (Holland, 1992). It has been proved to be effective in solving optimization and search problems in many fields of study (Chen, 1998; Goldberg, 1989; Karr, 1991; Kung & Chen, 1997; Lin, 1997). The GA is an adaptive heuristic search algorithm that mimics the principles laid down in Darwin's theory of evolution and successively applies genetic operators such as selection, crossover (recombination), and mutation to iteratively improve the fitness of a population and reach the global optimum in the search space. It is one of the most promising algorithms, and the population-by-population method is the most important characteristic of the GA when compared to the point-to-point method of DP (Goldberg & Kuo, 1987). In recent decades, the GA has become popular for the application of global optimization to reservoir planning and management for scientific research and in the field of engineering (Cai et al., 2001; Chaves & Chang, 2008; Chen & Chang, 2007, 2009; Huang, Yuang, & Lee, 2002). However, the GA has not really been suitable for complex nonlinear problems like ROO problems.

The GA starts with a randomly generated initial population and improves the fitness of solutions through iterations by implementing operators such as scheme selection (reproduction), crossover, and mutation. A crossover operation leads to a mixture of chromosomes in the offspring without creating new allelic material. This can lead to slow convergence toward the same sub-optimal solution. Srinivas and Deb (1995) put forward the adaptive genetic algorithm (AGA) with adaptive crossover and mutation. The probabilities of crossover and mutation vary depending on the fitness values of the solutions, thus improving the convergence rate and robustness (Chen, Chen, & Chiang, 2009). Solutions with high fitness are preserved, while those with sub-average fitness are completely disrupted. However, the AGA is still challenging without the consideration of the initial population diversity. Preserving population diversity is crucial for a successful and efficient search of complex multimodal response surfaces. Lack of population diversity often results in convergence to a sub-optimal solution, whereas excessive diversity results in the inability to further refine solutions for closer approximation. As the number of decision variables increases in ROO problems, the search process becomes ineffective within the vast amount of GA chromosomes and results in slow evolution between consecutive generations. Consequently, the probability of converging to the optimal solution decreases sharply. In this paper, an improved AGA was proposed for solving ROO problems. We used the hierarchical structure of the archive scheme to change population diversity, and we proposed a hierarchical adaptive genetic algorithm (HAGA).

Failure to improve the efficiency of and remove the deficiency in the algorithm used for solving the ROO problem may lead to either excessive computation or unsatisfactory solutions. In this study, we also discussed the prime characteristics of ROO problems and a DDP algorithm with a parallel model. We also apply improved conventional and metaheuristic algorithms to ROO. A new parallel DDP (PDDP) approach incorporating both computational efficiency and optimal decisions was presented for the Three Gorges Project (TGP). An improved AGA (HAGA) incorporating optimal decisions was applied to the TGP, and the parallel model was also applied to HAGA. Moreover, outlines of PDDP and HAGA optimization techniques applied to ROO problems were presented (Fig. 1).

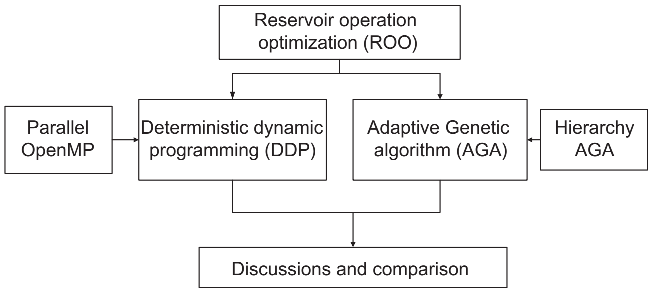
Reservoir operation optimization (ROO)
Parallel OpenMP
Deterministic dynamic programming (DDP)
Adaptive Genetic algorithm (AGA)
Hierarchy AGA
Discussions and comparison

Fig. 1. The flow diagram of ROO research.

The remainder of this paper was organized as follows. Section 2 presented the formulation of the ROO problem. Section 3 explained the improved optimal algorithms. In Section 4, the application of the proposed optimal algorithm was shown and the results were presented. Finally, in Section 5, the results of the study were discussed and conclusions were provided.

2. Formulation of problems

ROO is aimed at maximizing water resource benefits by determining an optimal plan for a hydropower station over the operation period, while satisfying all kinds of physical and operational constraints. Generally, the objective function and associated constraints of the ROO problem can be formulated as follows.

2.1. Objective functions

The major objective of ROO is to maximize the reservoir benefits, which are mainly related to hydropower generation and water supply. Hydropower generation is a significant benefit derived from a reservoir system, and it is related to a given operation of the hydropower station for T intervals as follows:

$$F = \max \sum_{t=1}^{T} AO_t H_t M_t \text{ or } F = \max \sum_{t=1}^{T} N_t M_t \tag{1}$$

where F is the total hydropower generation for all operation intervals, A is the comprehensive output coefficient, O_t is the rate of outflow from the reservoir in the tth operation interval, H_t is the average reservoir storage level in the tth operation interval, T is the number of intervals over the operating horizon, M_t is the period of the tth operation interval, and N_t is the reservoir output in the tth operation interval.

Recently, multi-objective ROO has become popular. This was developed as a technique to resolve conflicting objectives and it converts the ROO into a multi-objective problem. Besides hydropower generation, other objectives are taken into account, such as flood control, navigation, benefits from supply of water to downstream rivers, and the maximization of water supply for civil or agricultural uses. These objectives may be incompatible with each other. In multi-objective optimization, there is no simple optimal solution. Instead, the interaction of multiple objectives yields a set of nondominant solutions, which belong to the objective function space called optimal Pareto front (or Pareto set). The optimal Pareto set takes into account the existing relations among the different criteria for optimality. Starting from a solution within the Pareto set, the fulfillment of an objective can only be improved at the cost of diminishing the fulfillment of at least one of the other objectives (Collette & Siarry, 2003).

Traditionally, multi-objective optimization problems have been solved using weighting methods or ε-constraint methods. A single objective is obtained as the weighted sum of many objectives. The optimal Pareto set is obtained by varying the weights associated with each objective and solving the problem sequentially. DeJong (1985) used the weighting method to integrate multiple objectives into a single objective. In this approach, the need to compare incompatible objectives significantly complicates the selection of the weights. The ε-constraint method retains one objective as the primary and treats the others as constraints, so all but one of the objectives is incorporated into the constraint set. The objectives included in the constraint set are varied parametrically from the lower bound to the upper bound.

The main objective of this study was to maximize power generation under certain constraints. To easily implement the algorithm for solving an ROO problem, we adopted an external penalty function to convert a constrained optimization problem into an unconstrained one. A reservoir operation system involves complicated procedures. For simplification, we considered only the two most vital objectives, which were flood prevention and power generation (with irrigation, navigation, etc. ignored). Although reservoir operation should be cost-effective, it is important to ensure its reliability and safety. Generally, during the flood season, the reservoir operation maintains the storage at a fixed level (reservoir inflow is equal to reservoir discharge). Hence, reliable output from the hydropower station and maximum power generation during the non-flood season were considered in this study. We designed the fitness function with penalty terms, which augment the objective function with penalty values associated with infeasible solutions:

$$f(V) = F(V) + M \bullet \min\{N - N_{\inf}, 0\} \tag{2}$$

where $f(V)$ is the fitness value, $F(V)$ is the total hydropower generation for all operation intervals, M is a penalty weight, and $N_{\inf}$ is the minimum output.

2.2. Constraints

The reservoir operation problem is subjected to equality and inequality constraints.

A. The water volume balance equation is written as:

$$V_{t+1} = V_t + (I_t - O_t) \bullet M_t; \quad t = 1, 2, \ldots, T \tag{3}$$

where V_{t+1} is the reservoir storage volume in the tth operation interval, V_t is the reservoir storage volume in the tth operation interval, I_t is the inflow rate of the reservoir in the tth operation interval, O_t is the outflow rate in the tth operation interval, and M_t is the duration of the tth operation interval. The equation describes the water balance under the assumption that there assumes no water loss from bed leakage.

B. The other constraints can be written as

$$(V, Z, O, N)_t^l \le (V, Z, O, N)_t \le (V, Z, O, N)_t^u \quad t = 1, 2, \ldots, T \tag{4}$$

where V, Z, O, and N are the reservoir storage level, storage volume, discharge, and output capacity, respectively, while l and u are the lower and upper reservoir limits in the tth operation interval, respectively.

3. Methodologies

3.1. Parallel deterministic dynamic programming algorithm

3.1.1. Deterministic dynamic programming algorithm

Differential DDP can be used to solve the multistage optimization problem recursively. Dividing the reservoir operation into sub-operations on the basis of operation intervals, DDP yields an optimal solution to the whole problem by using the solutions for the sub-operations. The ROO problem can be solved by the following DDP steps described below.

3.1.1.1. Stage variables. Each stage is a dynamic problem that is featured in real time. ROO problems can be divided into several stages. Generally, the problem is considered recursive from one stage to the next, leading to a final solution called recursive optimal solution. ROO involves a recursive procedure based on a DDP model including a multistage decision process. A decision must be made in each interval, being a month, 10 days, or five days, which depends on the actual conditions of the ROO problem.

3.1.1.2. State variables. When making a decision at any stage, state variables are updated simultaneously. State variable discretization is necessary when using DDP models to solve ROO problems. The reservoir storage volume is an essential state variable and the discretization has a pronounced effect on the computational efforts. In this approach, the point-to-point transitions between storage volumes at the beginning and end of each period are replaced by equivalent storage intervals. If the reservoir storage level is adopted as a state variable, and make the equal storage level change, the storage volumes show larger variations for high storage levels compared to low storage levels. Alternatively, we adopt the reservoir storage volume at the beginning of each period as a state variable that can represent the evolution of the ROO, for conformance with the requirement of no following effects. Smooth transitions between storage intervals are allowed as the storage volume interval is actually a continuous variable, and the range of the state variables is from the storage volume at the normal storage level to the dead storage level.

3.1.1.3. Decision variables. The general form of an ROO problem is such that every locally optimal solution should be calculated to obtain the global optimal solution. At each step, we make a decision only about the current best solution, and this decision might be changed later on the basis of previous and current information available at that time. Therefore, the run-time of DDP is reduced; the decision is only made locally as needed to traverse the list once. As the reservoir storage volume is known at the beginning stage, any discrete reservoir storage volume at the end of a stage prompts one decision, which can be written as

$$O_i = q + (V^{begin} - V_i^{end})t \qquad i = 1, 2, \ldots, n \tag{5}$$

where q indicates the reservoir inflow at the t stage, n is the number of state variable discretizations at the present stage, and O_i is the reservoir release corresponding to the ith decision; V^{begin} and V_i^{end} are known as beginning reservoir storage volume and the end reservoir storage volume corresponding to the ith decision.

The fitness of the present stage can be written as

$$F_i^t = AO_iH_it \quad i = 1, 2, \ldots, n \tag{6}$$

where H_i indicates the reservoir storage level corresponding to the ith decision at the present stage and F_i^i indicates the fitness of the ith decision at the present stage.

3.1.1.4. Recursive equation and its solution. The application of DDP to the ROO problem involves a recursive procedure. In this study, reverse recursion is adopted. The DDP method for obtaining the solution of the optimization problem is based on the fitness formula at a given time. Assume that t denotes discrete time steps, where $t = 0, 1, \ldots, T$; t would also represent the stage. At time interval t, the optimization problem over the remaining time depends on the state i and the decision variable D_t, which constitute a complete set of decisions. In general, the decision depends on the current state i. The fitness function is used to find the decision that maximizes the optimization problem. The t stage fitness function, which represents a transition to state j during the next time interval, depends on i, j, and the current decision D_t, given that the optimal operation system is in the current state i. Thereafter, D_t is replaced by the function $D_{i,j}^t$, and the recursive equation for the tth discrete point is

$$f_t = \max_{D_t}\{D_{i,j}^t + f_{t-1}^i\} \quad j = 1, 2, \ldots, n \tag{7}$$

where f_t is the optimal decision at the current stage (t) and f_{t-1}^i is the optimal decision at stage ($t - 1$).

In the reverse recursion procedure, starting from the last stage, the fitness is calculated up to the first stage, and the line of optimal operation is obtained at last.

3.1.2. Parallel deterministic dynamic programming

Parallel computing distributes sub-problems to sub-units. Parallelization is the process of analyzing sequential codes for parallelism and restructuring them to run efficiently on multiprocessor systems by distributing the computational workload (sub-problems) among the processors. Every OpenMP program begins as a single process, the master thread, which executes sequentially until the first parallel region (parallel sections) construct is encountered. In a parallel region construct, the master thread creates a team of parallel threads. The workload is then enclosed by a parallel region construct and executed in parallel among the various team threads. When the team threads complete the work by computing in the parallel region construct, they synchronize and terminate, leaving only the master thread. By running (forking) multiple concurrent threads (processors) for parallel processing when a parallel region is encountered, the computations are combined into one thread for serial processing. This type of parallelism is called fork-join parallelism (Fig. 2).

The fork/join framework (Lea, 2000) is designed for task that can be recursively broken into smaller pieces by using a work-stealing algorithm and is based on expressing parallelism by means of two basic primitives: a fork, which starts the parallel execution of a code fragment (commonly a procedure or a method), and a join, which blocks the main application thread until the execution of these code fragments is complete. PDDP algorithms solve problems by breaking them down into several sub-problems of the same type until trivial problems are obtained; these trivial problems are solved directly. Like divide-and-conquer algorithms, PDDP algorithms are recursive; they repeatedly generate subtasks (forks) for each sub-problem, whose solutions are combined (joined) to obtain a solution of the original problem. Small sub-problems are commonly solved by calling a fragment of a sequential code.

The operation of a reservoir system involves a complex decision-making process in which many variables and objectives are integrated, requiring a considerable amount of computation and high efficiency. When DDP is applied to every stage of an ROO problem, there are many possible decisions that entail similar computations repeatedly. At every stage in Eqs. (5)–(7), there are n discretization points for state variables with n^2 groups between the beginning and end of storage levels. Hence, there are n^2 possible decisions at each stage, and they are computed for the corresponding fitness values. If the beginning and end of storage levels in the operation period are fixed and known (flood control level), then for all T stages, there are $2n + n^2(T - 2)$ possible decisions, which scale as $O(k^2T)$ for time. Before recursion, computing the fitness of each possible decision at each stage is not affected by any other possible decision at any stage and has no effect on any other decision, which reflects a characteristic of parallelism. In PDDP, all possible decisions are initially distributed evenly across the p processors. Each processor is assigned $[2k + k^2(T - 2)]/p$ particles, and the processors are responsible for computing the fitness only for possible decisions, which scale as

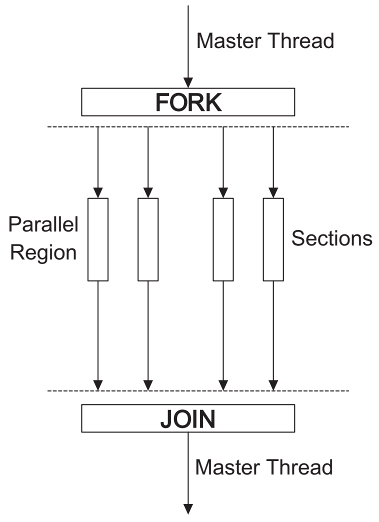

Fig. 2. The mode of fork-join scheme.

$O((k^2T/p)+p)$ for time. Therefore, more tasks can be completed within the same time.

3.2. Hierarchical adaptive genetic algorithm

3.2.1. HAGA formulation

In this approach, we aim to achieve a trade-off between exploration and exploitation by varying p_c and p_m adaptively in response to the fitness values of the solutions and by using the hierarchical structure. The HAGA uses an archive based on hierarchy to improve the optimization capability and convergence property of the AGA for solving the ROO problem.

The HAGA designed for solving the ROO problem is based on a reservoir objective function, which is described in Section 2.1. The constraints in the ROO problem are discussed in Section 2.2, and the procedures in the HAGA are listed below.

A. First Hierarchy: This uses the breadth mutation model to generate the next generation.

Step 1: Set related parameters, including the population size and gene length.

Step 2: Generate the initial population randomly as follows:

$$V_t = V_t^l + r \times (V_t^u - V_t^l) \tag{8}$$

where $r \in [0,1]$ is a random number. By this method, feasible solutions are generated in the form $(V_1^1, V_2^1, \ldots, V_n^1), (V_1^2, V_2^2, \ldots, V_n^2), \ldots, (V_1^{popsize}, V_2^{popsize}, \ldots, V_n^{popsize})$, where *popsize* is the population size and n is the chromosome length corresponding to the operation period.

Step 3: Calculate the fitness value by using Eq. (2) and select the elite for the second hierarchy. We use the breadth mutation model to reduce the probability of generating an infeasible solution. By using the random operator, the next generation is generated nearby the best individual, and this can be expressed as follows:

$$\begin{cases} (V_1^i,\ldots,V_n^i) = (V_1^{best},\ldots,V_n^{best}) - r \times ((V_1^{best},\ldots,V_n^{best}) - (V_1^l,\ldots,V_n^l)) & i \leqslant popsize/2 \\ (V_1^i,\ldots,V_n^i) = (V_1^{best},\ldots,V_n^{best}) + r \times ((V_1^u,\ldots,V_n^u) - (V_1^{best},\ldots,V_n^{best})) & i > popsize/2 \end{cases} \tag{9}$$

where V^i denotes an individual of the next generation and V^{best} denotes the elite individual corresponding to the best solution found so far.

A. Second Hierarchy: This initializes the individuals of the population by using the archive of the first hierarchy. The other procedures are the same as those of the improved AGA.

Step 1: Set the number of generations $k_{\max}$.

Step 2: Calculate the fitness value by using Eq. (2) for the population from the first hierarchy and select individuals for the next generation. On the basis of the individual fitness and selection operator, use elite roulette selection (Song, 2009) to determine which individual in the parent population should be selected for the next generation. On the basis of the random number, the probability of each individual being selected is determined, and by using the elite principle, the best individual in the last generation is directly admitted to the next generation. Each evolution generates the best individual for the next generation.

Step 3: Adopt the adaptive crossover and mutation operators to generate the next generation. In the course of crossover and mutation, the probabilities of crossover and mutation for a selected individual are adjusted according to the following formulas:

$$P_c = \begin{cases} P_{c1} - (P_{c1} - P_{c2})(f' - f_{avg})/(f_{\max} - f_{avg}) & f' \geqslant f_{avg} \\ P_{c1} & f' < f_{avg} \end{cases} \tag{10}$$

$$P_m = \begin{cases} P_{m1} - (P_{m1} - P_{m2})(f' - f_{avg})/(f_{\max} - f_{avg}) & f' \geqslant f_{avg} \\ P_{m2} & f' < f_{avg} \end{cases} \tag{11}$$

where $P_{c1}, P_{c2}, P_{m1},$ and P_{m2} represent adaptive coefficients, f_{avg} represents the average fitness value, and f' represents the fitness of the selected individual, which is the higher of the fitness values of the two individuals selected in crossover and the individual selected in mutation. By above-mentioned operator, we can produce offspring for the next generation. An elite-preserving approach is adopted, and the choice of offspring is made by comparing them with the best solutions found so far. The best solution is preserved for the next generation.

There are some conditions under which the HAGA reverts to the breadth mutation model. To prevent the AGA from becoming trapped by a local optimum, the model guides the AGA to avoid the local optimum and obtain the global optimum to some extent. The HAGA can use the breadth mutation model if the fitness value meets the following condition:

$$Abs(f_{best}^i - f_{best}^{i-1}) \leqslant \varepsilon \tag{12}$$

where f_{best}^i is the fitness value of the elite individual in the ith generation, f_{best}^{i-1} is the fitness value of the elite individual in the $(i-1)$th generation, and ε is the control region. Accordingly, we can consider the AGA falling into a local optimum if the fitness value of the elite individual barely changes between two generations.

Step 4: If the maximum number of iterations $k_{\max}$ is reached for the HAGA, the procedure will proceed to the next iteration from Step 2. Otherwise, the elite will be taken as the new solution and the AGA procedure will be terminated. A sketch of the hierarchy is shown in Fig. 3.

4. Case study

4.1. Description of the Three Gorges Project (TGP)

Located in the middle of the Xiling Gorges, the Three Gorges Project (TGP) is one of the major water conservancy projects in China. The Yangtze River has a drainage area of 1.8 million square kilometers and includes agricultural and industrial terrain. It is the third largest river in the world and the largest river in China in terms of channel length and water flow. As the river flows east through the mountains, it encounters a narrow constriction at the Three Gorges of Qutang, Wu, and Xiling (or Sanxia) (Fig. 4).

With two hydropower gravity dams, the TGP is built 40 km upstream from the Gezhouba dam, and it has a height of 175 m and a length of 2335 m. The spillway dam, located in the middle of the river's course, is 483 m long with 23 bottom outlets and 22 surface sluice gates. Salient features of the TGP are presented in Table 1.

The operation of the TGP reservoir is managed in a way that comprehensively meets the requirements of flood control, power generation, river navigation, and sediment discharge. From the end of May to the beginning of June, the reservoir storage level is maintained at 145 m, and during the flood season from June to September, the reservoir is generally operated at a lower storage level. (For operation in the early stage, the reservoir storage level is 156 m and the flood control level is 135 m.) Flow exceeding the discharge capacity of the power station is released. The reservoir is used to retain floodwater if the reservoir inflow surpasses the safe discharge in the downstream reaches. However, the reservoir storage level is still lowered to 145 m after the flood peak has passed. In October, the reservoir storage level rises gradually to 175 m. From November to the end of April of the following year, the storage level of the reservoir is kept as high as possible to allow the peak load of the electrical grid to regulate the operation of the power station. Then, the reservoir storage level is lowered, but it should not fall below 155 m before the end of April in order to ensure adequate navigational conditions. January to April is the wet season and the storage level decreases gradually, resulting in a low flow rate for the TGP. Only through ROO may the TGP satisfy the multiple purposes of water supply, including irrigation, hydropower generation, industrial and domestic uses, flood control, and recreation.

The Yichang station is nearby the Three Gorges dam, and the station runoff varies: it is over 79% between May and October (flood season) and less than 7.5% between January and March. There is less runoff during the non-flood period of a low-flow year, and therefore, water resource utilization can be enhanced through optimal operation decisions. Many researchers have focused on the non-flood period, for which ROO can be useful owing to diminished water resources. An optimal algorithm can be used for the ROO problem in this case.

4.2. Results and discussion

During the flood season from June to September, the Three Gorges reservoir storage level is kept at the flood control level of 145 m, so the optimal operation period is from September 1 to June 10 of the following year. There are 28 time intervals (considering a period of ten days as one operation interval). Only for dry year is the ROO useful for application to water resource utilizations, which emphasizes the importance of reasonable allocation of water resources. For example, 1972 was a dry year with a reliability of 95% based on TGP inflow data for 121 years from 1882 to 2002, where the reliability is obtained after the statistical analysis of 121 yearly inflow data in each year.

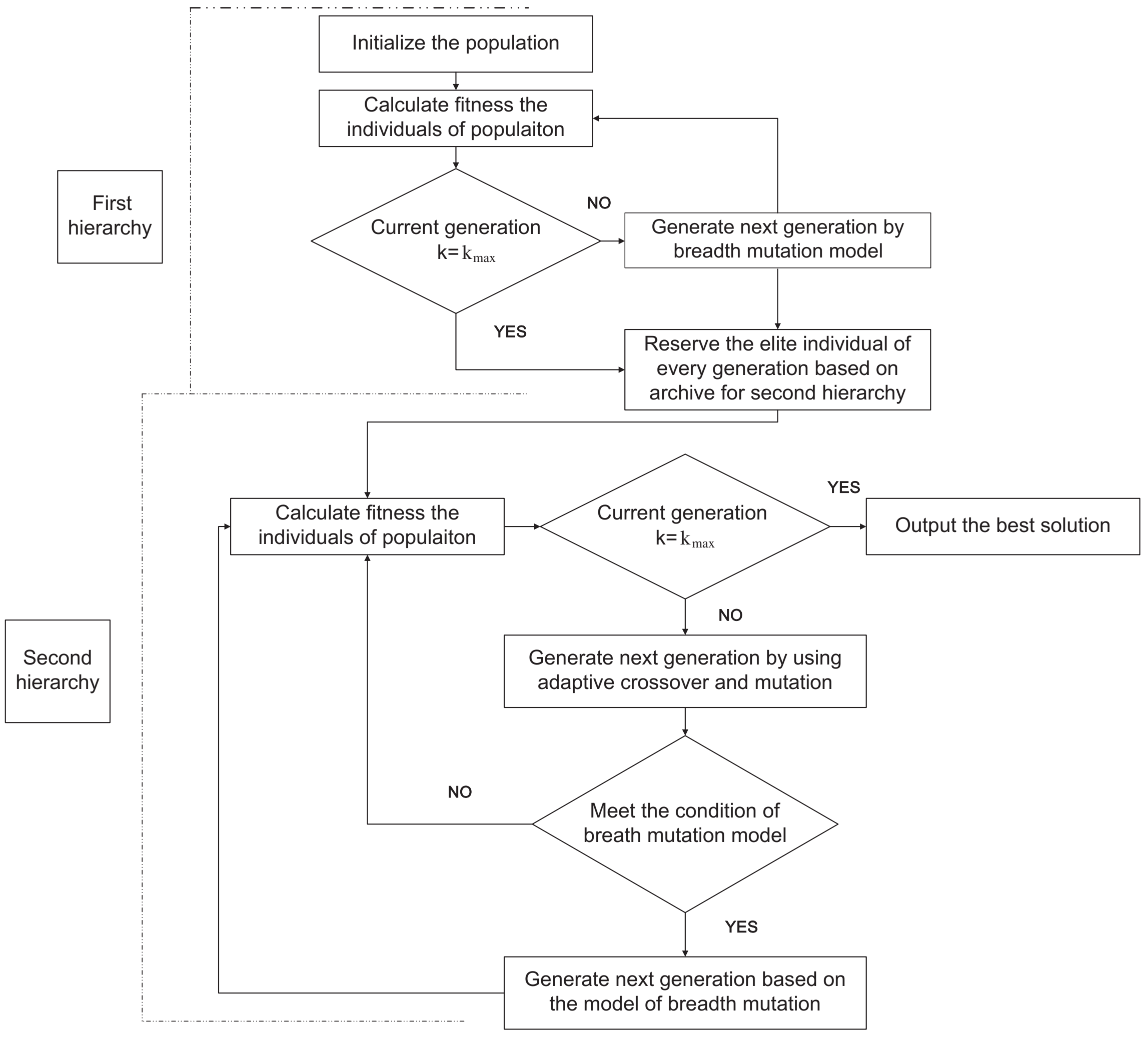
First hierarchy
Initialize the population
Calculate fitness the individuals of populaiton
Current generation k= k_{max}
NO
Generate next generation by breadth mutation model
YES
Reserve the elite individual of every generation based on archive for second hierarchy
Second hierarchy
Calculate fitness the individuals of populaiton
Current generation k= k_{max}
YES
Output the best solution
NO
Generate next generation by using adaptive crossover and mutation
NO
Meet the condition of breath mutation model
YES
Generate next generation based on the model of breadth mutation

Fig. 3. The flow diagram of hierarchy algorithm.

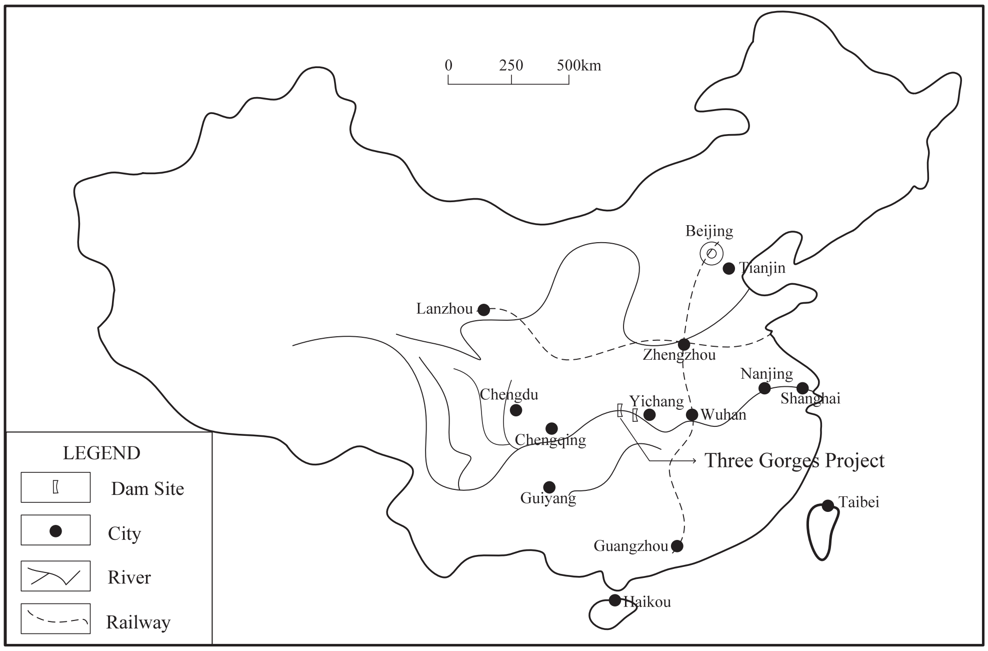

Fig. 4. The location of the Three Gorges Project, China.

Table 1
The salient features of TGP.

Items	Value	Units
Total capacity	39.3	billion m^3
Area	1084	km^2
Machine units	26 s	sets
Installed capacity	1.82×10^4	MW
Mean depth	70	m
Maximum depth	170	m
Average width	1100	m
Flood control storage level	145	m
Maximum storage level	175	m

4.2.1. Results of PDDP

The data available for an optimal algorithm analysis include reservoir properties, rule curves, inflow over ten-day intervals, and expected water demand.

For PDDP, we used a CPU with four processors and a clock speed of 2.40 GHz. All programs were implemented in FORTRAN using the OpenMP interface. With the parameter settings listed in Table 2, the proposed PDDP was implemented to solve ROO problems for the period from September 1 to June 10 of the following year.

For the OpenMP implementation, an input file allows the user to input the desired number of threads. On the basis of this number, the ROO problem is solved statically or dynamically among the threads by using the OpenMP library commands. We considered 2, 4, 8, and 16 threads.

We tested different numbers of threads on the quad-core multiprocessor system. Fig. 5 showed the OpenMP parallel computation time versus the number of threads for the PDDP algorithm discussed in Section 3.1. The performance of the DDP, with and without the OpenMP parallelization, was presented in Table 3.

Using parallel computation could obviously speed up the DDP algorithm. Serial computation in DDP was considered as a single thread, so setting 2, 4, 8, and 16 threads for DDP can decrease the computation time, as shown in Table 3. It was notable that (a) the computation time was much less for the parallel computation than for the sequential computation and (b) the CPU utilization was much greater for the parallel computation than for the sequential computation. The better performance of the OpenMP parallel computation was attributed to its memory localization, which facilitates faster data access and less communication contention among processors. The CPU usage for the parallel computation was 100%, which implied a full workload and full use of the computational resource. In the OpenMP parallel computation, the PDDP used all threads of the quad-core computer (default: four threads). The serial computation was only 25% resident in the CPU, which implied significant wastage of the parallel resource.

The following two results indicated that the OpenMP parallel performance was notable: (1) multi-threads produced a good speedup compared to single threads and (2) while the performance increased with an increase in the number of threads from 4 to 16, the performance did not improve further upon the addition of more threads. The computer had a CPU with four processors. When 8 or 16 threads were set for PDDP, there were some sub-tasks in a queue since the number of threads exceeded the number of processors. Hence, 8 or 16 threads made the PDDP more complicated than four threads for such a computer. The best parallel performance was obtained with four threads, for which the data access was much faster and there was less communication contention among processors.

4.2.2. Results of HAGA

The HAGA methods were also implemented through FORTRAN programming, solving the ROO problems mentioned above. The mutation parameter and crossover parameter are generally key factors that considerably affect the performance of algorithms (Lu & Zhou 2010). Suitable values of these two parameters were estimated by using Eqs. (10) and (11) for implementing HAGA. The Corresponding parameters used for adaptive mutation and crossover were listed in Table 4. In order to verify the effectiveness of the algorithm presented in this paper, two other algorithms were implemented to solve the same problem: algorithm 1 was the simple AGA and algorithm 2 was the AGA with the breadth mutation model in Eq. (12).

The proposed HAGA was implemented for $k_{\max} = 200$ with two hierarchies, each hierarchy corresponding to 100 iterations. Preserving population diversity was crucial for a successful and

Table 2
The parameters of the PDDP.

Parameter of the PDDP	Setting value
The number of stage	28
The number of discretization points at every state (decision variables)	500
The number of thread	2, 4(default) and 8, 16

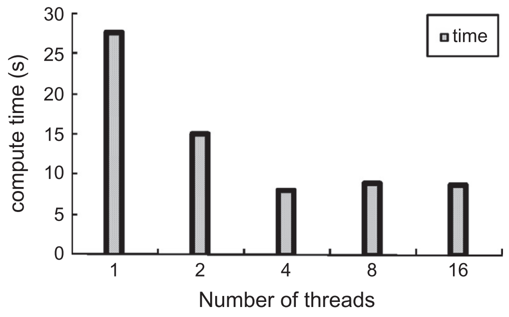

Fig. 5. PDDP algorithm performance with different threads.

Table 3
The computational performance of DDP and PDDP.

Parameters	No.	Serial (Thread = 1)	Parallel (threads)			
			2	4	8	16
Compute time/s	1	27.79	15.09	7.98	8.96	8.45
	2	27.79	15.09	7.98	8.78	9.06
	3	27.79	15.09	7.98	9.06	8.54
Average time/s	1	27.79	15.09	7.98	8.93	8.68
CPU usage/%	1	25	50	100	100	100

Table 4
Parameters of three methods.

Method	*Popsize*	$k_{\max}$	ε	p_{c1}	p_{c2}	p_{m1}	p_{m2}
Algorithm 1	100	100	0.1	0.9	0.6	0.1	0.001
Algorithm 2	100	100	0.1	0.9	0.6	0.1	0.001
HAGA	100	200	0.1	0.9	0.6	0.1	0.001

efficient search in the case of an ROO problem. The proposed HAGA improved population diversity by using the archive from the first hierarchy, and the elite chromosome of every generation was chosen as the initial individual for the second hierarchy, as shown in Fig. 6. It was clearly seen that the fitness function value was evenly distributed based on breadth mutation model, providing a better diversity of initial population by the strategy of archive.

A comparison of the operation results and convergence performances of the proposed HAGA methods was performed. Table 5 listed the best solutions of those methods, and the corresponding convergence property comparisons were shown in Fig. 7.

Table 5 clearly showed that the proposed HAGA could provide a better operation result with the maximum hydropower. Moreover, it can be seen clearly that the proposed HAGA methods could obtain better operation results than algorithm 1, algorithm 2, and PDDP, and the course of operation was shown in Table 6. The best hydropower generation over the operation horizon obtained by the hierarchy methods for the operation period in the ROO problems was 49.634 bkW. Fig. 7 provided details of the power generation for the best operation results obtained by the proposed HAGA and the other two methods.

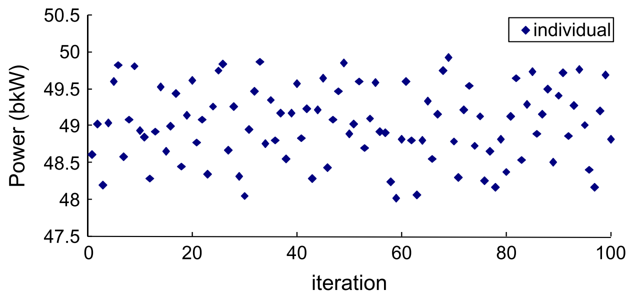

Fig. 6. The elite individual of first hierarchy.

Table 5
Comparison of the result obtained from different methods.

Method	Maximum hydropower (billion Kw h) (bkW)
PDDP	48.77
Algorithm 1	48.419
Algorithm 2	49.379
HAGA	49.634

It was evident that algorithm 2 and the HAGA can achieve better power generation than algorithm 1 for the ROO problems. Algorithm 1 might be premature, finding a local optimum easily. In contrast, algorithm 2 and the HAGA were enhanced with the breadth mutation model, which prevented them from being trapped by a local optimum, provided that certain conditions were met between two generations. In particular, the HAGA can obtain better solutions than the other two algorithms. The proposed HAGA has computed the adaptively and converges to obtain the maximum value at the end of the procedure. At the beginning of the procedure, the HAGA used the archive from the first hierarchy as the initial population and evolution started from the second hierarchy. The archive from the first hierarchy led to a high objective value but a low fitness, indicating that there were some infeasible solutions with large penalty values. With evolution, the infeasible individuals could be eliminated by increasing the value of penalty items of the above-mentioned constraint handling methods. The HAGA found the best solutions after eliminating the infeasible individuals at the end of the procedure, and it satisfied all the constraints of the ROO problem while increasing the power effectively. Thus, the HAGA was robust for solving the ROO problem.

4.2.3. Comparisons between HAGA and PDDP and discussions

To further define the operation strategy for the ROO problem, the HAGA and PDDP were implemented for the same case and a comparison was made between them to test the optimal result of the proposed algorithm. Table 7 listed the solutions obtained for the ROO problem by using the following algorithms: two optimal algorithms (using different optimal mechanisms), PDDP (using a recursive strategy with a parallel model), and the HAGA (adopting a heuristic mechanism as the evolution strategy).

The operation results obtained by the PDDP methods could satisfy the reservoir output constraints, basically meeting the firm output demand. However, the operation results obtained by the HAGA method could not satisfy the firm output constraints but only the minimum output demand. For some operation intervals, the output was lower than the firm output because of the random mechanism of the evolution algorithm.

It can be seen that the HAGA results in more power generation compared to the PDDP method. Although some operation intervals had low yields, the HAGA had an optimal effectiveness for the ROO

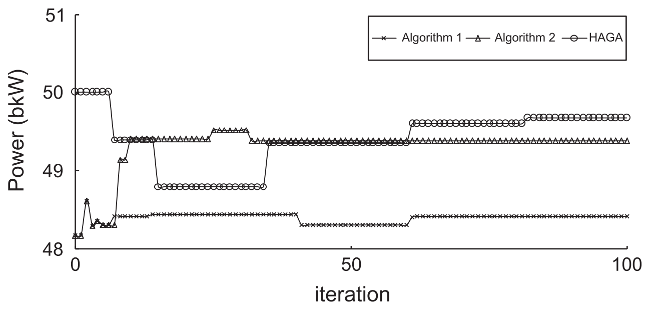

Fig. 7. Power generation comparison of the three methods.

problem. Fig. 8 gave details of the operating procedure, including the storage level and output at every stage.

Fig. 8 clearly showed that the two algorithms differ with respect to variations in the storage level and output with the operation stage. The PDDP method gave significant changes in the storage level and meets the firm output, while the HAGA results in moderate changes in the storage level and did not meet the firm output at some stages. Because the GA is stochastic, its minimum output was below the firm power and thus could not meet the power generation demand; however, it still had an advantage over PDDP with regard to power generation.

The net available head (H_t) in formula (1) was created in following way, being the most known type: building a dam in order to increase the storage water level just above the plant. Accordingly, H_t could be formulated as following:

$$H_t = H_{t-up} - H_{t-down} \tag{13}$$

where $H_{t\text{-}up}$ is the reservoir storage water level at the interval t; and $H_{t\text{-}down}$ represents tail water level, which can be considered as a constant elevation for TGP over the whole operation period. So the TGP storage level was the key factor associated with H_t in an operation period. The TGP has the functions of flood control. In order to guarantee flood control requirements, flood-control limiting level of 145 m shall be generally guaranteed during the period from the first ten days of June to the end of August. So the reservoir storage level was kept to 145 m at both begin and end of the operation period under aforementioned principle indicating:

$$V_{T+1} = V_1 \tag{14}$$

Based on formula (3)

$$\sum_{t=1}^{T} V_{t+1} = \sum_{t=1}^{T} V_t + \sum_{t=1}^{T} (I_t - O_t) M_t \tag{15}$$

The total water volume of inflow $\sum_{t=1}^{T} I_t M_t = \sum_{t=1}^{T} O_t M_t$, which is the total volume of water discharged as the result of hydropower generation over the whole operation period. The total water volume of inflow is deterministic in one year, so $\sum_{t=1}^{T} O_t M_t$ is a fixed value over one operation period; H_t is the key factor playing important role in power generation F over the whole operation period. Seen from Table 7, the average water level of HAGA (166.8 m) was higher than that of PDDP (163.3 m). HAGA got accordingly more hydropower generation than PDDP shown in Table 5.

In an operation period, when natural inflow was abundant and power output was large and steady, the hydropower plant could provide firm output for electric power system. However, during low-flow operation interval, due to different operate strategy adopted by two improved algorithm, the PDDP could satisfy the electric power system demand. For HAGA, there were some time intervals, when the power output was lower than firm output but satisfying the minimum output demand. Therefore, the PDDP could provide more steady power output for electric power system than the HAGA. When natural inflow was less than conveying firm output to meet electric power system requirements, the strategy of optimization algorithms was increasing discharge from the generating sets by regulating reservoir pondage, and resulted in

Table 6
The optimization of three algorithms.

t	I_t	Storage level (m)			Outflow (m^3/s)			Output (MW)			Power (bkW)		
		GA1	GA2	HAGA	GA1	GA2	HAGA	GA1	GA2	HAGA	GA1	GA2	HAGA
1	14,560	148.5	150.6	148.9	12,512	11,212	12,271	8927	8123	8781	2.142	1.949	2.107
2	19,180	156.8	160.6	161.5	13,282	11,481	9674	10,171	9100	7669	2.441	2.184	1.84
3	18,200	161.5	163.6	163.4	14,378	15,655	16,552	11,771	13,185	13,965	2.825	3.164	3.351
4	13,970	164.4	165.8	165.1	11,410	11,823	12,347	9744	10,271	10,673	2.338	2.465	2.561
5	15,020	168.5	170.5	170.4	10,999	10,273	9644	9729	9251	8661	2.335	2.22	2.078
6	11,628	171.0	172.0	174.3	9144	10,001	7574	8361	9270	7106	2.207	2.447	1.876
7	8681	170.3	174.6	174.2	9424	5711	8777	8689	5412	8372	2.085	1.299	2.009
8	11,300	170.3	173.6	174.3	11,298	12,522	11,202	10,365	11,880	10,661	2.487	2.851	2.558
9	6926	171.8	174.3	174.4	5239	6120	6862	4864	5828	6558	1.167	1.398	1.574
10	5133	170.8	174.9	174.7	6235	4427	4808	5799	4245	4608	1.391	1.018	1.106
11	4462	171.0	174.8	174.7	4298	4561	4316	3986	4384	4145	0.956	1.052	0.994
12	3916	171.8	174.5	174.7	3110	4196	3952	2897	4027	3797	0.764	1.063	1.002
13	3778	170.5	174.1	174.0	5198	4302	4675	4829	4115	4475	1.159	0.987	1.074
14	3500	168.4	172.6	174.0	5798	5151	3454	5300	4884	3296	1.272	1.172	0.791
15	3653	169.0	171.5	172.3	3077	4843	5410	2795	4539	5121	0.738	1.198	1.351
16	3509	169.1	171.1	171.2	3330	3945	4773	3036	3672	4462	0.728	0.881	1.07
17	3558	167.8	169.4	170.5	4900	5457	4374	4441	5026	4054	1.066	1.206	0.973
18	3772	167.1	169.9	169.8	4734	3076	4658	4250	2818	4290	0.816	0.541	0.823
19	3272	164.6	168.9	168.0	5659	4273	5207	4999	3907	4738	1.199	0.937	1.137
20	3585	164.1	166.8	166.4	4049	5745	5159	3528	5175	4619	0.846	1.242	1.108
21	3231	160.9	165.2	164.5	5790	4625	4895	4949	4095	4311	1.306	1.081	1.138
22	5752	159.7	163.2	164.5	6777	7627	5715	5659	6622	4983	1.358	1.589	1.196
23	7709	161.4	161.6	163.5	6324	9121	8713	5294	7768	7541	1.27	1.864	1.81
24	7440	158.0	160.5	161.4	10,214	8391	9230	8450	7052	7863	2.028	1.692	1.887
25	10,594	156.8	159.6	159.9	11,482	11,333	11,888	9258	9397	9915	2.221	2.255	2.379
26	15,320	156.1	152.8	158.1	15,931	20,632	16,728	12,631	16,173	13,615	3.031	3.881	3.267
27	11,561	154.5	154.6	154.1	12,673	10,284	14,387	9967	7976	11,403	2.631	2.105	3.01
28	14,350	145.0	145.0	145.0	20,495	20,629	20,257	14,998	15,111	14,799	3.599	3.626	3.551
Total											48.42	49.38	49.63

Table 7
The optimization of ROO.

Stage	Inflow	Storage level (m)		Outflow (m^3/s)		Output (MW)		Power (bkW)	
		PDDP	HAGA	PDDP	HAGA	PDDP	HAGA	PDDP	HAGA
1	14,560	150.0	148.9	11,620	12,271	8378	8781	2.01	2.107
2	19,180	162.0	161.5	9950	9674	7943	7669	1.906	1.84
3	18,200	162.0	163.4	18,200	16,552	15,228	13,965	3.654	3.351
4	13,970	161.9	165.2	14,044	12,347	11,830	10,673	2.839	2.561
5	15,020	171.4	170.4	5617	9644	5013	8661	1.203	2.078
6	11,628	175.0	174.3	7865	7574	7431	7106	1.961	1.876
7	8681	175.0	174.3	8681	8777	8331	8372	1.999	2.009
8	11,300	175.0	174.3	11,300	11,202	10,818	10,661	2.596	2.558
9	6926	175.0	174.4	6926	6862	6655	6558	1.597	1.574
10	5133	174.9	174.7	5197	4808	4998	4608	1.199	1.106
11	4462	174.3	174.7	5232	4316	5015	4145	1.203	0.994
12	3916	173.0	174.8	5258	3952	4996	3797	1.319	1.002
13	3778	171.6	174.0	5318	4675	4993	4475	1.198	1.074
14	3500	169.9	174.1	5393	3454	4992	3296	1.198	0.791
15	3653	168.0	172.4	5491	5410	4997	5121	1.319	1.351
16	3509	165.9	171.2	5594	4773	4994	4462	1.198	1.07
17	3558	163.6	170.5	5739	4374	5015	4054	1.203	0.973
18	3772	161.7	169.8	5858	4658	5011	4290	0.962	0.823
19	3272	158.4	168.0	5999	5207	4999	4738	1.199	1.137
20	3585	155.0	166.5	6216	5159	5000	4619	1.2	1.108
21	3231	150.1	164.5	6498	4895	4998	4311	1.319	1.138
22	5752	148.4	164.6	6746	5715	4989	4983	1.197	1.196
23	7709	149.9	163.5	6778	8713	5010	7541	1.202	1.81
24	7440	151.1	161.5	6669	9230	5008	7863	1.202	1.887
25	10,594	156.7	159.9	6422	11,888	5013	9915	1.203	2.379
26	15,320	167.5	158.2	5854	16,728	4999	13,615	1.199	3.267
27	11,561	159.2	154.2	18,387	14,387	15,628	11,403	4.125	3.01
28	14,350	145.0	145.0	24,168	20,257	18,098	14,799	4.343	3.551
Average		163.3	166.8	Total				48.77	49.634

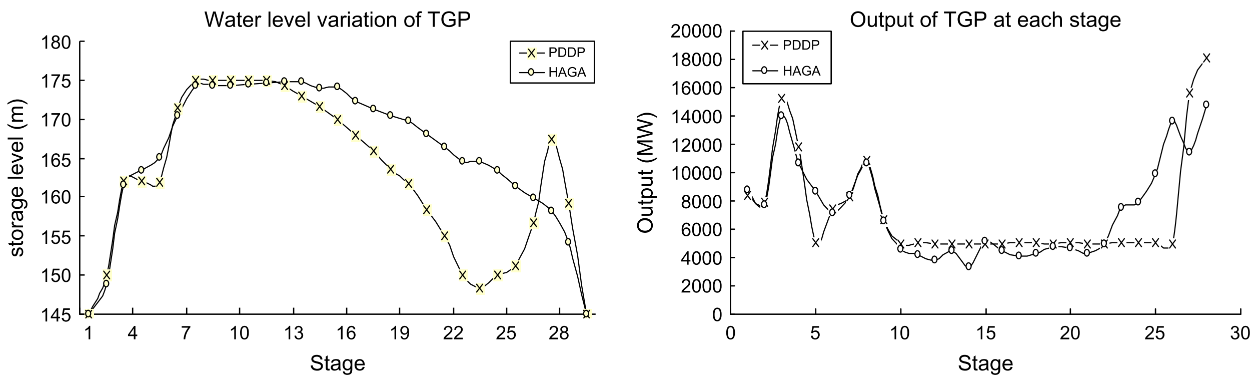

Fig. 8. The operation results of storage level and output.

decreasing operation water level of TGP. In order to meet firm power output, the operation storage level of PDDP was changing more largely than that of HAGA after 13th time interval as shown in Fig. 8. Although the operation result of HAGA showed that the power output did not meet electric power system, more generation benefit was obtained than that of PDDP over all operate intervals. In a word, increasing discharge from the reservoir lead to depress the storage water level. Total power generation from all operation intervals obtained from PDDP decreased based on formula (13), (14) and (15). Otherwise, the storage water level of HAGA would not decrease quickly failing to meet firm output demand and electric power system requirements, obtaining more power generation benefit.

Moreover, during low-flow interval, a combination operation of hydropower and thermal power station was adopted to meet electric power system demand; the hydropower station generally served for peak shaving, frequency modulation and emergency reserve, etc. in electric power system, so as to give full play of advantages of hydropower plant. The large fluctuation water level of PDDP did not comply with the principle of navigation. In revision, operation code provided by (The Ministry of Communications, 2005) has been incorporated into operation and regulation rules of TGP requiring that daily fluctuation amplitude of water level shall not exceed 3.0 m. So the HAGA was more suitable for some kind situation of ROO problem. Moreover, parallel OpenMp model can be applied to HAGA by distributing computational workload. The best parallel performance was obtained with four threads, so setting four threads (as shown in Fig. 5) for HAGA, the computational performance as shown in Table 8. From Table 8, it was clearly seen that (a) the HAGA with parallel can decrease the computational time compare to HAGA without parallel. (b) HAGA with parallel can solve the ROO problem more efficiently than PDDP. Thus, the successful application of HAGA with parallel for solving ROO problem was well demonstrated.

Thus, the PDDP was a little more reliable for electric power system but less generation benefit than HAGA. So the HAGA was more

Table 8
The computational performance of ROO.

Index	PDDP/s	HAHA/s	HAGA with parallel/s
1	7.98	1.78	1.03
2	7.98	1.79	1.03
3	7.98	1.85	1.02
Average time/s	7.98	1.81	1.03

economical. This showed that two important objectives in the ROO problem, economy and reliability, were contradictory for decision makers who want safety in the flood season, although there are more choices in the non-flood season. The simulate results obtained from two proposed methods revealed the characteristics of the ROO problem. Better power generation benefit was shown in Table 5 and better computational performance was shown in Table 8. Using HAGA was more suitable for some kind situation of ROO problem than PDDP. Moreover, the results presented herein demonstrated that HAGA optimization uncovered new ways to further improve the robustness and efficiency of evolutionary search for the ROO problem.

The two methods mentioned above gave different results to decision makers, but more decisions can be provided by a Pareto set. For designers making choices according to different requirements, there was also a multi-objective optimization problem with regard to the TGP. With a multi-objective optimization problem, there were many Pareto optimal solutions that form a so-called Pareto optimal front (Srinivas & Deb, 1995). In the absence of any additional information, none of these Pareto optimal solutions can be determined to be better than any other solution. This compels a user to find as many Pareto optimal solutions as possible. To solve a multi-objective problem, decision makers should find the optimal Pareto set. The solution preferred by the decision maker always reflects a compromise between the different objectives. The consideration of the existing relations among the different objectives allows the decision maker more flexibility in the selection of a suitable alternative solution. Moreover, a Pareto front facilitates the desegregation of decisions for a decision maker, and it can help in achieving optimal trade-offs for the ROO problem.

5. Conclusions

The ROO problem consists of many conflicting objectives that must be optimized simultaneously under a series of equality and inequality constraints. The problem is difficult to solve efficiently not only because of its large scale but also because it is a multi-objective, dynamic, nonlinear, and constrained problem. A set in objective function space was identified using ε-constraint methods, and two objectives of the ROO problem were comprehensively optimized using a set of penalty items, to convert the original problem into a single-objective problem. PDDP and the HAGA were used to solve this problem.

The PDDP method could be applied to the ROO problem, which can then be parallelized to improve the computational efficiency. The proposed algorithm used multi-threading through OpenMP showing good performance for up to four threads on a quad-core computer system. The HAGA was proposed to avoid premature convergence through the use of an elitist archive and to obtain a solution of the ROO problem. Compared to PDDP and two other methods, this gave the best result for the TGP. The results showed that the HAGA in conjunction with a hierarchical strategy, the use of the breadth mutation model, showed the best performance in terms of convergence and final solution. Moreover, parallel model was integrated into the proposed HAGA to improve the computational efficiency of algorithm, and could provide better power generation benefit result in a shorter computational time for solving the ROO problem than PDDP. Although the output values of TGP obtained from HAGA for some time intervals were a little less than those of PDDP, which could not meet electric power system demand, the hydropower plants could be operated combined with thermal power station, and then provided enough power generation. Moreover, during low-flow season, in order to give full play of advantages of hydropower plants, the hydropower stations could serve for peak shaving, frequency modulation and emergency reserve, etc. in electric power system. The results proved the superiority of the proposed HAGA with parallel method regarding the convergence properties, power generation benefit, and computational efficiency.

Furthermore, two contradictory objectives, economy and reliability, were reconciled by comparing the results of the HAGA and PDDP. To secure an optimal trade-off solution to the ROO problem, our study on solving a multi-objective problem will continue in the future to find the optimal Pareto set that can help a decision maker choose a good solution. Thus, decision makers will be able to make choices according to requirements.

Acknowledgement

The authors gratefully acknowledge the financial supports from National Key Basic Research Program of China under No. 2013CB036406.

References

Baskar, S., Subbaraj, P., & Rao, M. (2003). Hybrid real coded genetic algorithm solution to economic dispatch problem. *Computers & Electrical Engineering, 29*, 407–419.
Basu, M. (2004). An interactive fuzzy satisfying method based on evolutionary programming technique for multiobjective short-term hydrothermal scheduling. *Electric Power Systems Research, 69*(2–3), 277–285.
Basu, M. (2005). A simulated annealing-based goal-attainment method for economic emission load dispatch of fixed head hydrothermal power systems. *International Journal of Electrical Power & Energy Systems, 27*(2), 147–153.
Bellman, R. (1957). *Dynamic programming* (1st ed.). NJ: Princeton University Press.
Cai, X., McKinney, D. C., & Lasdon, L. S. (2001). Solving nonlinear water management models using a combined genetic algorithm and linear programming approach. *Advances in Water Resources, 24*, 667–676.
Chapman, B., Jost, G., & van der Pas, R. (2007). *Using OpenMP: Portable shared memory parallel programming*. Cambridge (Massachusetts): The MIT Press.
Chaves, P., & Chang, F. J. (2008). Intelligent reservoir operation system based on evolving artificial neural networks. *Advances in Water Resources, 31*, 926–936.
Chen, P. C. (1998). *Genetic algorithm for control of structure system*, Master's Thesis, Department of Civil Engineering, Chung Yuan University, Taiwan.
Chen, C. (2007). Economic dispatch: A direct search approach. *Energy Conversion and Management, 48*, 219–225.
Chen, L., & Chang, F. J. (2007). Applying a real-coded multi-population genetic algorithm to multi-reservoir operation. *Hydrological Processes, 21*, 688–698.
Chen, Y. H., & Chang, F. J. (2009). Evolutionary artificial neural networks for hydrological systems forecasting. *Journal of Hydrology, 367*, 125–137.
Chen, P. C., Chen, C. W., & Chiang, W. L. (2009). GA-based modified adaptive fuzzy sliding mode controller for nonlinear systems. *Expert Systems with Applications, 36*, 5872–5879.
Chiang, C. (2007). Optimal economic emission dispatch of hydrothermal power systems. *International Journal of Electrical Power & Energy Systems, 29*(6), 462–469.
Collette, Y., & Siarry, P. (2003). *Multi-objective optimization-principle and case study*. Berlin: Springer Verlag.
DeJong, K. A. (1985). Genetic algorithms: A 10 year perspective. In *Proceedings of the international conference of genetic algorithms and their applications* (pp. 167–177). Pittsburgh, July 24–26.
Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. USA: Boston: Addison-Wesley.
Goldberg, D. E., & Kuo, C. H. (1987). Genetic algorithms in pipeline optimization. *Journal of Computing in Civil Engineering, 1*, 128–141.
Gropp, W., Lusk, E., & Skjellum, A. (1994). *Using MPI: Portable parallel programming with the message-passing interface*. Cambridge (Massachusetts): The MIT Press.
Hindi, K., & Ghani, M. (1991). Dynamic economic dispatch for large-scale power systems: A Lagrangian relaxation approach. *Electrical Power System Research, 13*, 51–56.

Holland, J. H. (1992). *Adaption in natural and artificial systems*. USA: Boston: The MIT Press.

Huang, W. C., Yuang, L. C., & Lee, C. M. (2002). Linking genetic algorithm with stochastic dynamic programming to the long-term operation of multi-reservoir system. *Water Resources Research, 38*, 1304–1312.

Jabr, R., Coonick, A., & Cory, B. (2000). A homogeneous linear programming algorithm for the security constrained economic dispatch problem. *IEEE Transactions on Power Systems, 15*, 930–937.

Karamouz, M., & Houck, M. H. (1987). Comparison of stochastic and deterministic dynamic programming for reservoir operating rule generation. *Water Resources Bulletin, 23*, 1–9.

Karr, C. L. (1991). Genetic algorithms for fuzzy controller. *AI Expert*, 26–33 February.

Keib, A., Ma, H., & Hart, J. (1994). Environmentally constrained economic dispatch using the Lagrangian relaxation method. *IEEE Transactions on Power System, 9*(4), 1723–1727.

Kung, C. C., & Chen, C. C. (1997). Grey fuzzy sliding mode controller design with genetic algorithm. In *Proceedings of 36th IEEE conference decision and control* (pp. 2748–2753). USA: California.

Lea, D. (2000). *A java fork/join framework[C]//proceedings of the ACM 2000 conference on java Grande*. San Francisco, California: ACM, pp. 36–43.

Lin, S. C. (1997). *Stable self-learning optimal fuzzy control system design and application*. Ph.D. Dissertation, Department of Electrical Engineering, National Taiwan University, Taiwan.

Loucks, D. P., Stedinger, J. R., & Haith, D. A. (1981). *Water resources systems planning and management*. New Jersey: Prentice-Hall. 320–324.

Lu, Y., & Zhou, J. (2010). An adaptive hybrid differential evolution algorithm for dynamic economic dispatch with valve-point effects. *Expert Systems with Applications, 37*, 4842–4849.

Papageorgiou, L., & Fraga, E. (2007). A mixed integer quadratic programming formulation for the economic dispatch of generators with prohibited operating zones. *Electrical Power System Research, 77*, 1292–1296.

Song, H. S. (2009). Hybrid genetic algorithm of solving multidimensional 0–1 knapsack problem. *Computational and Engineering Applications, 35*, 4–7.

Srinivas, N., & Deb, K. (1995). Multi-objective function optimization using non-dominated sorting genetic algorithms. *Evolution Computation, 2*, 221–248.

Takriti, S., & Krasenbrink, B. (1999). A decomposition approach for the fuel constrained economic power-dispatch problem. *European Journal of Operational Research, 112*, 460–466.

The Ministry of Communications (2005). *Test and research report of impact upon shipping between the two dams of Three Gorges hydropower station and Gezhouba Hydropower station by unsteady flow generated by daily regulation and water discharge at Three Gorges Hydropower station*. China: Southwest Waterway Transportation Engineering Research Institute of Ministry of Communications (pp. 57–62).

Travers, D., & Kaye, R. (1998). Dynamic dispatch by constructive dynamic programming. *IEEE Transactions on Power Systems, 13*, 72–78.

Yakowitz, S. (1982). Dynamic programming applications in water resource. *Water Resources Research, 18*, 673–696.

Yeh, W. W.-G. (1985). Reservoir management and operation models: A state-of-the-art review. *Water Resources Research, 21*, 1797–1818.