Environmental Modelling & Software, Volume 57, July 2014, Pages 152-164

Parallel Discrete Differential Dynamic Programming For Multireservoir Operation

Chuntian Cheng[1,*], Sen Wang[2], Kwok-Wing Chau[3] and Xinyu Wu[4]

ABSTRACT— The curse of dimensionality and computational time cost are a great challenge to operation of large-scale hydropower systems in China because computer memory and computing time increase exponentially with increasing number of reservoirs. Traditional DDDP algorithm, which is one of the most popular classical algorithms for alleviating the dimensionality problem and cutting down the computing time for operation of hydropower systems, can be naturally parallelized due to its requirement of discretization of the state variables. However, the computational time performed on DDDP still increases exponentially with increasing number of reservoirs. Therefore, a fine-grained parallel discrete differential dynamic programming (PDDDP) algorithm, which is based on Fork/Join parallel framework in a multi-core environment, is proposed to improve the computing efficiency for long-term operation of multireservoir hydropower systems. The proposed algorithm is tested using a huge cascaded hydropower system located on Lancang River which is one of the 13 largest hydropower bases in China. The results demonstrate that the PDDDP algorithm can enhance the computing efficiency significantly and take full advantage of multi-core resources, showing its potential practicability and validity for solution of optimal operation of large-scale hydropower systems in future.

(KEYWORDS: hydropower systems, discrete differential dynamic programming, parallel, long-term operation, optimization)

[1]Professor, corresponding author, Institute of Hydropower and Hydroinformatics, Dalian University of Technology, Dalian 116024 China. E-mail: ctcheng@dlut.edu.cn

[2]Ph.D. Student, Institute of Hydropower and Hydroinformatics, Dalian University of Technology, Dalian 116024 China. E-mail: wangsen19861227@163.com

[3]Professor, Department of Civil & Environmental Engineering, Hong Kong Polytechnic University, Hong Kong, China. E-mail: cekwchau@polyu.edu.hk

[4]Ph.D. Associate Professor, Institute of Hydropower and Hydroinformatics, Dalian University of Technology, Dalian 116024 China. E-mail: wuxinyu@dlut.edu.cn

INTRODUCTION

Multireservoir operations are one of complex and challenging tasks (Labadie, 2004) addressed by many researchers in past decades. Comprehensive methods and models that deal with a variety of problems about hydropower reservoirs operations are now available. Numerous classical algorithms, such as linear programming (Trezos, 1991; Reis et al., 2006; Azamathulla et al., 2008), nonlinear programming (Martin, 1983; Lund and Ferreira, 1996; Barros et al., 2003), network flow algorithm (Braga and Barbosa, 2001), dynamic programming (DP) or improved dynamic programming (Yakowitz, 1982; Kumar and Baliarsingh, 2003; Goor et al., 2011; Liu, 2011; Zhao et al., 2012a, b), and heuristic programming (Dariane and Momtahen, 2009; Malekmohammadi et al., 2009; Moeini and Afshar, 2013; Zhang et al., 2013), have been applied to the multiple reservoirs operation problems with nonlinear and non-convex objective functions. The advantages and disadvantages of these classical approaches were specified in the reported literatures (Yeh, 1985; Labadie, 2004). The choice of methods is dependent on the operation tasks, available data, objectives and constraints. In the past decade, large-scale hydropower system optimization operations had become very prominent with the fast development in China and Brazil (Barros et al., 2001; Barros, 2003; Zambon et al., 2012; Cheng et al, 2012a, b, c). Especially in China, one of the countries that are rich in water resources, with the gross theoretical hydropower potential of 694 GW and technically exploitable installed capacity of 542 GW, there has been a rapid rate of development of hydropower systems. Now, the total installed capacity of hydropower has now exceeded 200GW, and the number of hydropower plants is more than 45,000. The number of large and medium-size hydropower plants operated by a central dispatching center is more than 100 and its total installed capacity reaches 50GW. In the future 20 years, the number of hydropower plants operated by a regional dispatching center will be over 200 and its total installed capacity will surpass 140GW. The challenges to the operation management of large-scale hydropower systems are tremendous in China. The greatest obstacle faced in the optimal operation of hydropower system remains the curse of dimensionality with increasing numbers of reservoirs, resulting in exponential increase of computer memory and computational time. In solving problems on large and complex hydropower systems, a general idea is to adopt methods which can reduce or alleviate dimensions. Progressive optimality algorithms (POA) (Howson and Sancho, 1975; Turgeon, 1981; Cheng et al., 2012c), discrete differential dynamic programming (DDDP) (Heidari et al., 1971; Chow et al., 1975; Tospornsampan et al., 2005) and dynamic programming successive approximation (DPSA) (Erkmen et al., 1994; Opan, 2011) have been developed to overcome the incremental dimensionality of DP along with the distensible scale of the hydropower systems. POA is a computationally efficient method of reducing the dimensionality difficulties by decomposing a multi-stage decision problem into a series of two-stage problems. DDDP realizes the reduction of the discretized number of state variables by iterative search in the constantly

changing corrider. DPSA decomposes a multi-dimensional problem into a sequence of one-dimensional problems by optimizing over one state variable at a time. However, these approaches also have some disadvantages in the solution. POA and DDDP are sensitive to initial trajectories, and may converge to a local optimum in some situations. Global optimum is only attainable for convex problems by DPSA, but even local optimum is not assured for non-convex problems. In spite of these inherent weaknesses, the approaches mentioned above are still widely applied for the operation of large-scale hydropower systems because of their feasibility in the practical problems. Other approaches employ an integrated mix of formulations and solution methods to alleviate the curse of dimensionality. However, they can only be applied to optimal operation problems with relatively simple, small-scale constraints and objectives. For large and complex hydropower systems, it is very difficult to simplify both the objective function and constraints to either non-linear or linear optimization model format for direct mathematical solution. Therefore, the priority is given to the improved DPs such as POA, DDDP and DPSA for the large and complex hydropower systems.

Some simplified algorithms mentioned above have successfully reduced the dimensions or alleviate the curse of dimensionality, but the computational time cost still enormously increases with the number of hydropower plants for the optimization solutions. Especially, when the scale of hydropower system reaches a certain large degree, computational time cost is absolutely intolerable. Therefore, the computational efficiency is a great challenge to the operation of large-scale hydropower systems. Generally, there are two basic approaches to improve computational efficiency. One is to improve the classical algorithms or seek for new and good algorithms. The other is to use new computer techniques including hardware and software. This paper will address the latter and our focus is on the parallelization of dimensionality reduction methods, especially how to utilize the current popular multi-core resources.

Since the release of the first batch multi-core processors by IBM in 2001, Sun in 2004, and AMD in 2005, more and more cores are built into a single processor with the development of multi-core technology. Moreover, the increasing popularity of multi-core processors provides the necessary hardware basis for the implementation of computational tasks with fine-grained parallel mechanism. Hence, multi-core parallel computing technology (Zhu, 2013; Morell-Gimenez, 2013) is always the major research field of computer science. Parallel computation in multi-core environment means that the whole task is decomposed into numerous subtasks, and then subtasks are assigned to different cores in which subtasks can be executed independently, for speeding up the computational process. Nowadays, improving computational efficiency by parallelization is confirmed to be successful in many fields (da Silva and Finardi, 2003; Rouholahnejad, 2012; Jordi and Wang, 2012; Bryan, 2013; Zhao et al., 2013; Joseph and Guillaume, 2013). Therefore, the research of parallel optimization algorithms is very significant in order to solve the

time-consuming problem for large-scale hydropower systems. For realizing the parallelization, a variety of parallel frameworks have been developed such as Fork/Join (Lea, 2000), Message Passing Interface (MPI) (Li et al., 2011; Wu et al., 2013) and Open Multi-Processing (OpenMP) (Innocenti et al., 2009; Neal et al., 2010). The choice of parallel frameworks is often dependent on the characteristics of the task, applicable conditions, compatibility of operating systems and the diversity of source languages between algorithms and frameworks. By testing rigorously, Fork/Join framework has now been packaged as standard programs in Java version 7, and the choice of Fork/Join framework is certainly appropriate to parallelize the algorithms of Java programmed. For this reason, this paper uses Fork/Join framework to realize the parallelization of our Java programmed algorithm.

Though Fork/Join framework is not the most effective technique with good parallel performance (Lea, 2000), it also has some conspicuous advantages for attractive choice. Firstly, Fork/Join framework can divide a task into lots of subtasks recursively with divide-and-conquer strategy, which is suitable to be applied to the problems with heavy computational tasks such as optimization of hydropower system operation. Secondly, Fork/Join framework can make full use of multi-core resources which enables it to be applied widely in a popular personal computer with multiple cores. Thirdly, it possesses lightweight scheduling mechanics to be independent of any container. At last, Fork/Join framework is an open source program so that it is much easier to design parallel algorithms for the given problems.

In this paper, due to our focus on realization of the parallelization, the stochasticity of inflows has not been considered in the model for simplifying calculations, and DDDP is selected to test the computational efficiency of parallelization. A fine-grained parallel discrete differential dynamic programming (PDDDP) algorithm, which is based on Fork/Join parallel framework (Lea, 2000) in a multi-core environment, is proposed to improve the computational efficiency for long-term operation of multireservoir hydropower systems. The parallelization of DDDP is thoroughly analyzed. A huge cascaded hydropower system, located in the Lancang River which is one of the 13 largest hydropower bases in China, is used to test the feasibility and validity of the proposed algorithm. The case studies show that high parallel efficiency and rational optimization solution can be obtained by the use of PDDDP, and much less computer time is expended on the computational process than the original DDDP. Therefore, with the fast development of hydropower systems in China, it is expected that the parallelization of the optimization algorithms will be an effective approach to improve the computational efficiency for the operation of large-scale hydropower systems in future.

1 PROBLEM FORMULATION

2 *Variable Definition*

3 Hydropower system operation is characterized by various objectives and constraints. The notations used in the
4 paper are introduced as follows:

E total sum of energy produced by all reservoirs in time horizon [MWh]

T number of time steps

M number of reservoirs in hydropower system

p_m^t power generation of reservoir m for period t [MW]

Δ_t duration [h]

k_m generation efficiency of reservoir m

q_m^t average turbine discharge of reservoir m for period t [m^3/s]

H_m^t net head of reservoir m for period t [m]

$H_{m,s}^t$ storage water level of reservoir m for period t [m]

$H_{m,b}^t$ tail water level of reservoir m for period t [m]

$H_{m,l}^t$ head loss of reservoir m for period t [m]

S_m^t initial storage of reservoir m for period t [m^3]

I_m^t inflow of reservoir m for period t [m^3/s]

R_m^t average discharge of reservoir m for period t [m^3/s]

hn number of seconds in an hour [s]

In_m^t local inflow of reservoir m for period t [m^3/s]

L_m total number of upstream reservoirs connected to reservoir m immediately

U_m array storing the sequence number of upstream reservoirs connected to reservoir m immediately

d_m^t average spilling discharge of reservoir m for period t [m^3/s]

S_m^B initial storage of reservoir m [m^3]

S_m^E specified end-of-period storage of reservoir m for final period [m^3]

$\overline{q}_m^t$ upper bound of average turbine discharge of reservoir m for period t [m^3/s]

$\underline{q}_m^t$ lower bound of average turbine discharge of reservoir m for period t [m^3/s]

$\underline{p}_m^t$ lower bound of average power generation of reservoir m for period t [MW]

$\overline{p}_m^t$ upper bound of average power generation of reservoir m for period t [MW]

$\underline{Z}_m^t$ lower bound of end-of-period water level of reservoir m for period t [m]

Z_m^t end-of-period water level of reservoir m for period t [m]

$\overline{Z}_m^t$ upper bound of end-of-period water level of reservoir m for period t [m]

$\underline{R}_m^t$ lower bound of average discharge of reservoir m for period t [m^3/s]

$\overline{R}_m^t$ upper bound of average discharge of reservoir m for period t [m^3/s]

Objective Function

A number of objective functions (Barros et al., 2003, 2005) have been applied to long-term operation of hydropower systems, such as minimizing the loss of the stored potential energy, minimizing storage deviations from targets, and maximizing total energy production. In this paper, the maximization of the total energy production is used to test the feasibility and validity of the proposed algorithm due to its simplicity for long-term optimal operation of hydropower systems. The objective is to determine the optimum strategy for maximum energy production in a time horizon subject to various constraints, and the function is formulated as follows:

$$E = \text{Max} \sum_{t=1}^{T} \sum_{m=1}^{M} p_m^t \Delta_t \tag{1}$$

Where:

$$p_m^t = k_m q_m^t H_m^t \tag{2}$$

$$H_m^t = H_{m,s}^t - H_{m,b}^t - H_{m,l}^t \tag{3}$$

Constraints

Water balance

$$S_m^{t+1} = S_m^t + (I_m^t - R_m^t) \times \Delta_t \times hn \tag{4}$$

$$I_m^t = In_m^t + \sum_{l=1}^{L_m} R_{U_m[l]}^t \tag{5}$$

$$R_m^t = q_m^t + d_m^t \tag{6}$$

Initial and terminal storage limits

$$S_m^0 = S_m^B,\ S_m^T = S_m^E \tag{7}$$

Turbine flow limits

$$\underline{q}_m^t \le q_m^t \le \overline{q}_m^t \tag{8}$$

Power generating limits

$$\underline{p}_m^t \le p_m^t \le \overline{p}_m^t \tag{9}$$

Water level limits

$$\underline{Z}_m^t \le Z_m^t \le \overline{Z}_m^t \tag{10}$$

Release limits

$$\underline{R}_m^t \le R_m^t \le \overline{R}_m^t \tag{11}$$

PARALLEL DDDP METHODOLOGY

With the increase of the number of reservoirs in hydropower systems, the operational difficulty will become more prominent, especially that obtaining optimization solution will consume relatively long time. Therefore, it is crucial to develop highly efficient optimization algorithm for reducing computational time of planning and operation of hydropower systems.

In this section, DP will be first briefly described before the introduction of the DDDP procedure as DDDP is based on DP. Then, the detailed analysis for the parallelization of DDDP is given At last, the parallel framework Fork/Join is introduced and a description of the parallel algorithm for long-term optimal operation of hydropower systems is proposed.

Standard DP algorithm

DP, proposed by Bellman (1957), has been one of the most popular optimization approaches and widely applied in many fields for handling the optimization problems nowadays. It is an effectual optimization method to address the sequential-decision problem by decomposing a problem into several sub-problems which can be handled sequentially

over each stage. The difficulties for the functional relationships in the objective and constraints can be commendably solved by DP method. For any DP optimization problem, the recursive equation is the basis to describe the multi-stage decision problem. Assuming that the objective function of a DP problem is to obtain the maximum return, the forward recursive equation can be expressed in the following mathematical term:

$$F_n(\overline{S}_n) = Max[R_n(\overline{S}_n, \overline{D}_n) + F_{n-1}(\overline{S}_{n-1})]; n = 1, 2, 3, \cdots, N \tag{12}$$

$$\overline{S}_n = T_{n-1}(\overline{S}_{n-1}, \overline{D}_n); n = 1, 2, 3, \cdots, N \tag{13}$$

where n is the stage index; N is the number of stages; $\overline{S}_n$ is a state vector at stage n; $\overline{S}_0$ is the known initial state vector; $\overline{D}_n$ is the decision vector at stage n; $F_n(\cdot)$ is defined as the maximum return from initial stage to stage n; $F_0(\overline{S}_0)$ is a known value related to the state vector $\overline{S}_0$; $R_n(\cdot)$ is defined as the return for stage n; $T_{n-1}(\cdot)$ is defined as the transformation function for converting the state vector $\overline{S}_{n-1}$ to the state vector $\overline{S}_n$.

Supposing that the optimization problem involves N stages, a sequence of decision vectors $\overline{D}_1, \overline{D}_2, \cdots, \overline{D}_N$ is formed from the initial state vector $\overline{S}_0$ to the final state vector $\overline{S}_N$. At every stage, many admissible values are discretized in state space and each value can obtain a return in $R_n(\cdot)$. Generally, $\overline{S}_n$ and $\overline{D}_n$ respectively represent the sets of storage and release for period n in the optimal hydropower system problem. As DP algorithm approximates the Bellman function with a function defined on the discretized state space, more admissible values should be discretized in state space for obtaining the optimal solution in theory. However, increasing the number of discrete values will increase the computational complexity of the algorithm seriously. For instance, supposing that 10 admissible values are discretized for each stage in a one-dimensional problem (1-plant system), 10^2 possible combinations need to be computed in Eq. (12) at any stage. For an M-dimensional problem (M-plant system) the computation burden has an exponentially growth to $10^{2\times M}$ at any stage. Hence, with the expansive scale of problem, DP formulation suffers seriously from the curse of dimensionality.

Standard DDDP algorithm

Traditional dynamic programming will be faced with two problems inevitably while it is applied to high-dimensional hydropower system. One is the computer memory requirement, and the other is the computer time requirement. Discrete differential dynamic programming, proposed by Larson (1968), is an improved dynamic programming method for reducing computer memory requirement and cutting down computer time. It is an improved

iteration technique to alleviate the “curse of dimensionality” problem arising from the operation of high-dimensional hydropower system. The DDDP procedure starts with a feasible initial trajectory which satisfies all constraints imposed on the system. Generally speaking, the feasible initial trajectory may be acquired by engineering judgment such as from the knowledge on the occurrence of dry and wet periods of a year, or by other simplified methods such as system decomposition. In the DDDP procedure, a corridor, which is composed of a restricted set of discrete values of the state variables in the admissible domain, is used to restrict the state space for searching for an improved trajectory around the current given trajectory. These discrete values of the state variables must be assumed in the feasible range at every stage for converging to the optimal policy and only a small number of values, usually recommended to 3 values, should be allowed for reducing the computer memory requirement. The corridor width is treated as a constant throughout a cycle, though it may be different to any stage theoretically. A schematic description of a 3-valued corridor is shown in Figure 1, in which $\Delta_1 = \Delta_2$ and the corridor width $cw = \Delta_1 + \Delta_2$ can be seen.

INSERT HERE FIGURE 1

Before the DDDP procedure starts, several corridor widths are prescribed. In general, the larger corridor widths are used in earlier cycles to search for coarse optimization solutions in entire state space, and then the corridor widths should be decreased gradually in latter cycles for converging to the optimization solution. Each corridor width corresponds to a cycle, in which the corresponding corridor width is used uniquely for constructing the corridor for multiple iterations. At the beginning, a feasible trial trajectory and a corridor with the given width should be determined, and then the DP recursive equation is used to examine the states in the neighborhood of the trial trajectory within the boundaries of the corridor. An improved trajectory can be found if it gives a better value of the objective function than the trial trajectory. Next, the improved trajectory is treated as the trial trajectory in the next iteration. Figure 2 shows the optimization description of an iteration process.

INSERT HERE FIGURE 2

For each iteration in a cycle, the corridor is reconstructed to search for new improved trajectory. Once the obtained new trajectory is as same as the trial trajectory, adjust the corridor width and enter the next cycle with the previous obtained trajectory treated as the trial trajectory. For next cycle, smaller corridor width is considered to repeat iterations.

The general process of the DDDP procedure can mainly be described in the following aspects:

Step 1: Several corridor widths are prescribed in descending order, and each corridor width corresponds to a cycle.

Step 2: Establish a feasible trajectory by engineering judgment or other simplified methods as a trial trajectory.

Step 3: On the basis of the current cycle index, the corresponding corridor width is selected from the prescrbied widths to form the boundaries of the trial trajectory what is called a 'corridor'.

Step 4: Seek a new improved trajectory by recursion equation of DP within the corridor, and then the improved trajectory is used as the trial trajectory for the next iteration and the corridor is reconstructed.

Step 5: Repeat the iteration procedure (Step 4) for some times in a cycle until the obtained new trajectory is as same as the trial trajectory.

Step 6: Enter the next cycle, and return to Step 3.

Step 7: Repeat the cycle procedure (Step 3 to Step 6) until the final cycle terminates and the current improved trajectory represents the optimization solution.

Analysis for the parallelization of DDDP

Parallelization is that computer system can execute two or more tasks at the same time. Namely, the different tasks, in multi-core environment, can be executed in different cores simultaneously with less computational time than single one. Therefore, parallelization is an effective way to reduce computation time of the procedures to improve the computational efficiency. In the optimization procedure of DDDP, the cycles of iteration process consume the majority of computation time. Furthermore, the number of discrete states in the state space at every stage directly influences the quality of solution and the amount of computational time. In general, the more the number of admissible values in the state space is discretized, the better the quality of solution could be, but simultaneously the more computational time will be consumed because of more computations of decision combinations. Supposing if 3 state values are discretized in a 2-plant hydropower system involving N stages, with the initial storage and final storage specified, an illustration for describing the decision combinations is shown in Figure 3, in which storage status is defined as state variable and water release as decision variable. As can be seen in Figure 3, $3^2+(N-2)3^4+3^2$ decisions, each of which is independent to acquire a return from Eq. (12), can be generated per iteration, namely there are actually $3^2+(N-2)3^4+3^2$ computations per iteration for obtaining an optimization policy. Similarly, assuming that a hydropower system involves M plants, N stages and y state values for each reservoir per stage, the total number of decisions per iteration will be $y^M+(N-2)y^{2M}+y^M$. Obviously, there is an exponential increase in the total number of computations as well, with the increasing state values and the expansive scale of hydropower system. It is the major reason that a small number of the state values, usually 3 values, should be allowed in DDDP for alleviating the problem of the curse of dimensionality. On the basis of independence of decision combinations per

iteration, let all of decision combinations be parallelized into P cores in each of which $[y^{M}+(N-2)y^{2M}+y^{M}]/P$ calculations will be performed independently. Therefore, parallel technology can be applied to iterations within DDDP procedure in reducing significantly the computational time to improve the computational efficiency. In this article, the number of the state values is set as 3 for optimization solution. Moreover, fork/join parallel framework is adopted for the parallelization design of DDDP.

INSERT HERE FIGURE 3

Fork/Join framework

Fork/Join algorithms (Lea, 2000) are parallel versions of divide-and-conquer algorithms, which are to repeatedly divide the current problem into a number of smaller subtasks recursively until the split subtasks are small enough to be solved by simple methods, and then the final execution result of the problem is obtained after all of the subtasks have been completed. Figure 4 shows the schematic diagram of divide-and-conquer strategy. During the running operation, a threshold is defined as the scale of subtasks for deciding whether the current subtask needs to be split. The selection of the appropriate threshold, which has direct impact on the performance of parallelization, is a very important step in the parallelization design. Experiments demonstrated that too small threshold values will cause excessive consumption on the management overhead of Fork/Join tasks, and too large values will lead to inadequately utilizing parallel resources. For obtaining the best threshold value for different-size tasks, some tests are necessary, but in general an appropriate threshold value for preferable parallelization performance can be set as following:

$$T_v = \lceil S_t / P \rceil \tag{14}$$

Where T_v is defined as the threshold value; S_t is defined as the size of the computational father task; P is defined as the number of the cores. $\lceil\ \rceil$ represents that the threshold value rounds up to an integer.

Under Fork/Join framework, a thread pool, in which the total number of worker threads is the same as the number of cores with default set, is created at the beginning of parallel processing for avoiding the enormous cost caused by creating and closing a thread repeatedly to save system resources. Moreover, a high performance task scheduling algorithm, named work-stealing, is performed in Fork/Join framework. It consists of several worker threads in each of which only a deque is used to operate tasks and the worker threads process their own deques by popping tasks. When a task forks a new worker thread, it will be pushed onto the head of that worker's own deque and be performed preferentially. By the use of the work-stealing algorithm, the issue of queue contention is well addressed. Furthermore, a worker thread with no local tasks running will try to steal a task from the base of deque in another worker thread. If

a worker thread completes all tasks and fails to steal one from other threads, it backs off and tries again later unless all worker threads are idle. Therefore, the work-stealing algorithm can reduce the occurrence of several worker threads in contention, and take full advantage of multi-core resources. A schematic description of work-stealing can be seen in Figure 5.

INSERT HERE FIGURE 4

INSERT HERE FIGURE 5

Another prominent advantage of Fork/Join framework lies in its lightweight scheduling mechanics in that it is independent of any container in different operation systems. In addition, Fork/Join framework program is a kind of open source program. It provides a convenient Application Programming Interface (API) for programmers who can exploit the framework easily by only a few simple design rules and patterns (Lea, 2000), but without knowing in advance how much parallelization the target system will offer. Therefore, the main advantage in creating such a Java lightweight execution framework is to enable Fork/Join programs to be written in a more portable fashion and to run on the wide range of operating systems supporting Java Virtual Machine (JVM).

Time Complexity of PDDDP

According to complexity theory, an important indicator in measuring efficiency of algorithms is the computational time used for the computational tasks. However, the computational time must be test in computer but cannot be obtained in theory. Hence, the number of the computations or the size of computational task is used to describe the time complexity of algorithms (Rudek, 2013). For the DDDP algorithm, the computational effort is mainly spent on the computation of all operation decisions in the iterations, and the execution time $T_E(M)$ consumed to solve DDDP problem involving 3 state values may be roughly estimated as

$$T_E(M) = O(f(M)) \; ; \; f(M) = 3^M + (N-2)3^{2M} + 3^M \tag{15}$$

where $O(\cdot)$ is defined as the time complexity of the DDDP; $f(M)$ is defined as the size of computational tasks per iteration in an M-dimensional system. From Eq. (15), it can be seen that the time complexity of the DDDP algorithm per iteration can be simplified as $O(3^{2M})$. Since the time complexity of DDDP grows exponentially with the scale of hydropower system, the computational time spent on solving DDDP problems is still unbearable. Thus, a parallel discrete differentiation dynamic programming running in the multi-core environment is presented to cut down the computational time. During the design of the parallel processing, the key is to treat the problem as a collection of numerous independent subtasks, and then assign these subtasks to different cores for simultaneous computations in

order to attain parallelization. It can be readily observed from the analysis of DDDP that calculations of all operation decisions per iteration can be parallelized. Considering that data processing and computation of parallelization algorithm are conducted via multiple cores of CPU, an important design consideration is the communication between different cores. Since the computational father task is divided into several subtasks by the divide-and-conquer method, this algorithm of chip-based parallel processing is termed fine-grained methodology.

Parallel Algorithm Design

For implementation of parallelization, the computation of all operation decisions per iteration is treated as a task and then the divide-and-conquer mode is used to divide the task into several subtasks for simultaneous operation by multiple cores. Each core is only responsible for the computation of the allocated decisions for returns. Once the operations of all subtasks are completed, parallel processing is terminated and it returns to the main thread environment, as illustrated in Figure 6. During the execution, parallel computing for the decision combinations per iteration are performed with results stored in a result set. When all operations have been completed, Eq. (12) is used in the main thread for recursively determining the optimization solution at the current iteration. For DDDP problem, parallel processing is repeatedly executed until all iterations are completed.

INSERT HERE FIGURE 6

As shown in Figure 7, all steps of the PDDDP procedure are described. Differing from DDDP, the realization of parallel processing is added and some details for the realization deserve attention.

1) Multiple iterations are repeated for solution in PDDDP procedure and the parallel processing is performed in iteration. Therefore, a thread pool should be created at the beginning of the procedure for avoiding the enormous cost caused by creating and destroying the pool repeatedly.

2) The number of computations of decision combinations (the size of the farther task) is confirmed with the number of reservoirs in the hydropower system. Then, the appropriate threshold value can be decided by Eq. (14) with the consideration on the number of cores in the current configuration. This selected value cannot guarantee the best parallel efficiency, but can guarantee sufficient utilization of CPU resources without idle status.

3) The main thread transmits the required data for computation to each child thread, such as basic attributes of power plants, characteristic curves and constraints. During the computational process, the condition that two different child threads have pertinent data is not allowed. Thus, it may purposely allow a certain degree of data redundancy in design to avoid any mistakes.

INSERT HERE FIGURE 7

The above parallel design model adopts task decomposition method, with the realization of the parallel design

assisted by the Fork/Join parallel framework. When PDDDP is performed in a multi-core environment, P cores can execute the tasks at the same time. Therefore, the number of calculations of decisions per iteration in one core is $[3^M + (N-2)y^{2M} + 3^M]/P$ and the time complexity of the PDDDP algorithm is $O(3^{2M}/P)$. It is thus obvious that PDDDP can improve substantially the computational efficiency in comparison with DDDP. The scale of problems that can be addressed by task decomposition increases with the number of cores. Namely, the implementation of such a decomposition method is able to complete more tasks in the same time period and can adequately solve the difficult scalability problems in parallel processing.

CASE STUDY

Application Background

The Lancang River, which originates from the Qinghai Province and flows through Tibet to the Yunnan Province, is one of the 13 hydropower bases in China. The overall length of the river is about 2000 km and the drainage area is 91,000 km^2 in its main stream with an annual average flow of 2180 m^3/s. Furthermore, its economical exploitable capacity has been determined to be approximately 25,450 MW, ranked the third in all hydropower bases. 14 hydropower plants have been planned on Lancang River in the main stream, all of which have a total installed capacity of approximately 24800 MW. It contains 6 dominated hydropower plants with a total installed capacity over 17,000 MW, and other small-size hydropower plants with a total installed capacity over 7000 MW. The cascaded hydropower system consists of 6 dominated plants, among which the Gushui is the leader plant, in the main stream as shown in Figure 8. Moreover, the important characteristics of the 6 dominated hydropower plants are described in Table 1. The storage rates of cascaded reservoirs are shown in Figure 9. The Xiaowan plant and the Nuozhadu plant possess multi-year regulation performance, so they can play an important role in the operation of hydropower system by distributing the flows to make maximum benefit.

In this paper, the 6 dominated hydropower plants are selected to take part in operation. For testing the validity and efficiency of PDDDP algorithm, three case studies, which compose of different combinations of hydropower plants, are taken into consideration for optimal long-term operation of hydropower system in the following subsections. The three cases is respectively 4-plant, 5-plant and 6-plant hydropower system, and the detailed information about these systems can be seen in Table 2. In order to simplify the test, deterministic inflows are considered for yearly operation on a monthly basis from January to December in 3 cases. Multi-year mean monthly local inflows of hydropower plants are shown in Figure 10. Moreover, the initial and terminated storage of each hydropower plant are both fixed for testing.

INSERT HERE TABLE 1

INSERT HERE TABLE 2

INSERT HERE FIGURE 8

INSERT HERE FIGURE 9

INSERT HERE FIGURE 10

Computer Configuration

The proposed algorithm is implemented by adopting Java language and Fork/Join framework in the Java 2 platform Enterprise Edition (J2EE). In order to verify the efficiency of the design method, computations are performed in solving different-scale hydropower optimal operation problems in multi-core environment on two diverse configurations.

Configuration 1: The operating system is Windows server 2003 and the CPU consists of four Intel Xeon MP 7120M@3.0 GHz (2 cores).

Configuration 2: The operating system is Windows XP professional and the CPU consists of one Intel Xeon E3-1245@3.30GHz (4 cores).

Metrics

Two significant indicators for evaluating parallel computation performance are speedup and efficiency (Zhang et al., 2013; Tesfa et al., 2011), respectively defined as S_p and E_p which are shown in Eqs. (16) and (17).

$$S_p = T_1 / T_p \tag{16}$$

$$E_p = S_p / P \tag{17}$$

where T_1 is defined as the execution time for the task in serial way with a single core and T_p is defined as the execution time for the task in parallel way with P cores.

Results Analysis

As mentioned in the section "Analysis for the parallelization of DDDP", the time complexity of DDDP grows exponentially with the increasing scale of hydropower system. In this paper, the exponentially growth of time complexity in the three cases is obvious. In case 1, the number of decision combinations per iteration is $3^4 + (12-2) \times 3^{2\times4} + 3^4 = 65,772$. In a similar way, the number reaches to $590,976$ in case 2, as well $5,315,868$ in case 3. Therefore, the execution time required for returns of all decision combinations per iteration is

rapidly increased with an exponentially growth when the scale of hydropower system is expanding.

In order to test the effect of multi-core environment on the computation efficiency, multiple comparisons of 3 cases in different multi-core environments were performed in the prescribed different configurations as shown in Table 3. From Table 3, the number of cores has no effect on the computation in serial situation. Moreover, the execution times corresponding to 3 cases by DDDP are rapidly increased with the increase of the number of hydropower plants, exhibiting exponential growth. Theoretically, the increase of computation time will be increase 9 times when the number of plants is increased by one as mentioned above. Notice that the increase of computation time is greater than the theoretical value in our case studies. For Configuration 1, the time consumption for Case 2 increases to 12.3 times in comparison with Case 1 but not 9 times. Similarly, the time consumption for Case 3 increases 12.1 times in comparison with Case 2, also not 9 times. The same performance appears for Configuration 2 as well. The reason for this is that, the larger number of plants the hydropower systems possess, the larger number of iterations DDDP performs for convergence in a real application. As a result, the increased computation time will be greater than 3^2 times and the "curse of dimensionality" problem will even be more serious. In addition, the execution times by DDDP have differences between two different configurations, because the performance of different configurations is affected by various factors such as frequency and cache.

The computation times using PDDDP are also summarized with the three cases for the two configurations in Table 3. Compared with DDDP, the maximum of decrement in the execution time is obtained in the 8-core environment for the prescribed configurations and the time reductions are 5303s and 4215s respectively for Case 3. As can be seen from Table 3, the time reductions are not obvious for the small number of plants using few CPU cores, because the realization of parallelization has few influences on the computational time reduction of small-size computational task. But it is remarkable for a greater number of plants using more CPU cores. Furthermore, the speedup values using different CPU cores for the given configurations are also listed in Table 3. From Table 3, the speed of the three cases performed on Configuration 2 is larger than Configuration 1 in the same multi-core environments. Hence, different configurations have an important impact on the parallel efficiency of Fork/Join framework. Furthermore, the three cases demonstrated that the speedup is increased with the number of CPU cores but not proportional to their number of CPU cores. The reason is that the parallelization program is only performed in iteration process of the algorithm, but other program is running in serial way. In addition, communicating time consumed among threads and memory usages on system for running program both have an effect on the computational efficiency. On the other hand, the parallel computational efficiencies are also compared, as shown in Table 3, Figure 11 and Figure 12. The results show that the more number of CPU cores for the same task, the lower

efficiency. The results also indicate that the more number of plants for the same number of CPU cores, the greater efficiency. The facts mentioned above imply that the parallel technologies can be used to significantly enhance the computational efficiency and take full advantage of multi-core resources for large-scale hydropower system operations.

INSERT HERE TABLE 3

INSERT HERE FIGURE 11

INSERT HERE FIGURE 12

The choice of the threshold of Fork/Join framework applied in PDDDP is very important to obtain the best speedup. In this paper, the recursive technique, divide-and-conquer, is used to split a problem in half by some method, and then those halves are split in half recursively until the scale of the problem is sufficiently small, and as an illustration, division for Case 3 is shown in Figure 13. Some different threshold values in three cases are selected to test the parallel performance for Configuration 1. As can be seen in Table 4, an excessively large threshold value will lead to the indivisibility of the problem so that it cannot be solved in the parallel mode, such as setting threshold value at 100×10^4 in Case 2 for any cores. On the contrary, an excessively small threshold value will cause the problem to be divided into lots of parts, resulting in more communication and thread-switching among threads such as setting threshold value at 100 in Case 1. Moreover, unreasonable threshold value may only take advantage of some parts of system resources. For instance, as shown division for Case 3 in Figure 13, when the threshold value is set at 300×10^4, the problem is divided into only two parts which are assigned into two worker threads for execution, so good speedup cannot be acquired distinctly in the 8-core environment because other worker threads are idle, as well the threshold value set at 150×10^4 with the problem divided into only four parts. In addition, the choice of the best threshold value in different-scale problem is entirely different and should be obtained by multiple tests. Generally speaking, for ensuring generality of the parallel program, an appropriate threshold value can be set with Eq. (14) to make full use of system resources for good speedup at least.

INSERT HERE FIGURE 13

INSERT HERE TABLE 4

CONCLUSION

In this study, a new efficient technique employing PDDDP is presented to address the optimal long-term operation of a hydroelectric power system. The simplified objective and fixed inflows are used to test the proposed algorithm because the main objective of paper is to demonstrate the potential application of parallel technologies in large scale

hydropower system operation of China. The hydropower systems in the Lancang River, one of the 13 hydropower bases in China, are used to test the potential ability of PDDDP for large-scale hydropower systems in future. The simulation results of different number of cascaded hydropower plants in the Lancang River basin show that the use of the PDDDP method can substantially improve the computation efficiency and obtain the same results with DDDP. Users can complete an optimization computation in a shorter period of time, furnishing an easy and fast simulation of operation schemes. It facilitates comparison and analysis of various schemes, thus enhancing the scientific ground and feasibility of the optimal operation scheme. The case study also demonstrates that: 1) it has high computation speed and search efficiency, thus improving computational efficiency; 2) with the expansion of the plant number, the advantages of the parallel processing method will become more notable; 3) it facilitates the use of idle computational resources, thus avoiding wastage. The case study results imply that parallel processing is a viable approach to improve the computation efficiency for power generation of large-scale hydropower systems and it is a very useful technique for hydropower systems of China in future, especially with the rapid growth and scale of China's hydropower and the ever increasing level of computer hardware and software.

ACKNOWLEDGEMENTS

This study is supported by the National Outstanding Youth Foundation of China (51025934), the Major International Joint Research Project from the National Nature Science Foundation of China (51210014), Central Research Grant of Hong Kong Polytechnic University (G-U861) and the National Basic Research Program of China (973 Program) (No. 2013CB035906).

REFERENCES

Azamathulla, H.M., Wu, F.C., Ghani, A.A, Narulkar, S.M., Zakaria, N.A.,Chang, C.K., 2008. Comparison between genetic algorithm and linear programming approach for real time operation. Journal of Hydro-Environment Research 2, 172-181.

Barros, M.T.L., Lopes, J.E.G., Yang, S.L.,Yeh, W.W.G., 2001. Large-scale hydropower system optimization. In: Marino, M.A., Simonovic, S.P (Eds), Proceedings of International Symposium on Integrated Water Resources Management. Univ Calif, Davis, CA, pp. 263-288.

Barros, M.T.L., Tsai, F.T.C., Yang, S.L., Lopes, J.E.G.,Yeh, W.W.G., 2003. Optimization of large-scale hydropower system operations. Journal of Water Resources Planning and Management 129, 178-188.

Barros, M.T.L., Ros, D.A.,Lopes, J.E.G., 2005. Objective functions for hydropower system operation. In: Moglen, G.E (Ed), Proceedings of Managing Watersheds for Human and Natural Impacts. Williamsburg, Virginia, USA,

pp. 1-11.

Braga, B., Barbosa, P.S.F., 2001. Multiobjective real-time reservoir operation with a network flow algorithm. Journal of The American Water Resources Association 37, 837-852.

Bellman R.E., 1957. Dynamic programming. Princeton University Press, Princeton, N.J.

Bryan, B.A., 2013. High-performance computing tools for the integrated assessment and modelling of social-ecological systems. Environmental Modelling & Software 39, 295-303.

Cheng, C.T., Shen, J.J., Wu, X.Y., Chau, K.W., 2012a. Operation challenges for fast-growing China's hydropower systems and respondence to energy saving and emission reduction. Renewable & Sustainable Energy Reviews 16, 2386-2393.

Cheng, C.T., Shen, J.J., Wu, X.Y., 2012b. Short-term scheduling for large-scale cascaded hydropower systems with multi-vibration zones of high head. Journal of Water Resources Planning and Management 138, 257-267.

Cheng, C.T., Shen, J.J., Wu, X.Y., Chau, K.W., 2012c. Short-term hydroscheduling with discrepant objectives using multi-step progressive optimality algorithm. Journal of The American Water Resources Association. 48, 464-479.

Chow, V.T., Maidment, D.R., Tauxe, G.W., 1975. Computer time and memory requirements for DP and DDDP in water resource systems analysis. Water Resources Research 11, 621-628.

da Silva, E.L., Finardi, E.C., 2003. Parallel processing applied to the planning of hydrothermal systems. IEEE Transactions on Parallel and Distributed Systems 14, 721-729.

Dariane, A.,Momtahen, S., 2009. Optimization of multireservoir systems operation using modified direct search genetic algorithm. Journal of Water Resources Planning and Management 135, 141-148.

Erkmen, I., Karatas, B., 1994. Short-term hydrothermal coordination by using multi-pass dynamic programming with successive approximation. In: 7th Mediterranean Electrotechnical Conference, Antalya, Turkey, pp. 925-928

Goor, Q., Kelman, R.,Tilmant, A., 2011. Optimal multipurpose-multireservoir operation model with variable productivity of hydropower plants. Journal of Water Resources Planning and Management 137, 258-267.

Heidari, M., Chow, V.T., Kokotovi, P.V, Meredith, D.D., 1971. Discrete differential dynamic programming approach to water resources systems optimization. Water Resources Research 7, 273-282.

Howson, H.R., Sancho, N.G.F., 1975. A new algorithm for the solution of multi-state dynamic programming problems. Mathematical Programming 8, 104-116.

Innocenti, E., Silvani, X., Muzy, A.,Hill, D.R., 2009. A software framework for fine grain parallelization of cellular models with OpenMP: Application to fire spread. Environmental Modelling & Software 24, 819-31.

Jordi, A., Wang, D.P., 2012. sbPOM: A parallel implementation of Princenton Ocean Model. Environmental Modelling & Software 38, 59-61.

Joseph, J.,Guillaume, J., 2013. Using a parallelized MCMC algorithm in R to identify appropriate likelihood functions for SWAT. Environmental Modelling & Software 46, 292-298.

Kumar, D.N., Baliarsingh, F., 2003. Folded dynamic programming for optimal operation of multireservoir system. Water Resources Management 17, 337-353.

Labadie, J.W., 2004. Optimal operation of multireservoir systems: state-of-the-art review. Journal of Water Resources Planning and Management 130, 93-111.

Larson R.E., 1968. State increment dynamic programming. American Elsevier Pub. Co., New York.

Lea, D., 2000. A Java fork/join framework. In: Proceedings of the ACM 2000 conference on Java Grande. San Francisco, California, United States, pp. 36-43.

Li, T., Wang, G., Chen, J.,Wang, H., 2011. Dynamic parallelization of hydrological model simulations. Environmental Modelling & Software 26, 1736-46.

Liu, P., Cai, X.M., Guo, S.L., 2011. Deriving multiple near-optimal solutions to deterministic reservoir operation problems. Water Resources Research 47.

Lund, J.R., Ferreira, I., 1996. Operating rule optimization for Missouri River reservoir system. Journal of Water Resources Planning and Management 122, 287-295.

Malekmohammadi, B., Kerachian, R.,Zahraie, B., 2009. Developing monthly operating rules for a cascade system of reservoirs: application of bayesian networks. Environmental Modelling & Software 24, 1420-1432.

Martin, Q., 1983. Optimal operation of multiple reservoir systems. Journal of Water Resources Planning and Management 109, 58-74.

Moeini, R., Afshar, M.H., 2013. Extension of the constrained ant colony optimization algorithms for the optimal operation of multi-reservoir systems. Journal of Hydroinformatics 15, 155-173.

Morell-Gimenez, V., Jimeno-Morenilla, A., Garcia-Rodriguez, J., 2013. Efficient tool path computation using multi-core GPUs. Computers In Industry 64, 50-56.

Neal, J.C., Fewtrell, T.J., Bates, P.D.,Wright, N.G., 2010. A comparison of three parallelisation methods for 2D flood inundation models. Environmental Modelling & Software 25, 398-411.

Opan, M., 2011. Real-Time Optimal Operation of Multiple Reservoirs System. Teknik Dergi. 22, 5359-5385.

Reis, L.F.R., Bessler, F.T., Walters, G.A., Savic, D., 2006. Water supply reservoir operation by combined genetic algorithm-linear programming (GA-LP) Approach. Water Resources Management 20, 227-255.

Rouholahnejad, E., Abbaspour, K.C., Vejdani, M., Srinivasan, R., Schulin, R., Lehmann, A., 2012. A parallelization framework for calibration of hydrological models. Environmental Modelling & Software 31, 28-36.

Rudek, R., 2013. The computational complexity analysis of the two-processor flowshop problems with position dependent job processing times. Applied Mathematics and Computation 221, 819-32.

Tesfa, T.K., Tarboton, D.G., Watson, D.W., Schreuders, K.A.T., Baker, M.E.,Wallace, R.M., 2011. Extraction of hydrological proximity measures from DEMs using parallel processing. Environmental Modelling & Software 26, 1696-709.

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

Trezos, T., 1991. Integer Programming Application for Planning of Hydropower Production. Journal of Water Resources Planning and Management 117, 340-351.

Turgeon, A., 1981. Optimal short-term hydro scheduling from the principle of progressive optimality. Water Resources Research 17, 481-486.

Wu, Y., Li, T., Sun, L.,Chen, J., 2013. Parallelization of a hydrological model using the message passing interface. Environmental Modelling & Software 43, 124-132.

Yakowitz, S., 1982. Dynamic programming applications in water resources. Water Resources Research 18, 673-696.

Yeh, W., 1985. Reservoir management and operations models: A state-of-the-art review. Water Resources Research 12, 1797–1818.

Zambon, R.C., Barros, M.T.L., Lopes, J.E.G., Barbosa, P.S.F., Francato, A.L., Yeh, W.W.G., 2012. Optimization of large-scale hydrothermal system operation. Journal of Water Resources Planning and Management 138, 135-143.

Zhang, R., Zhou, J.Z., Ouyang, S., Wang, X.M., Zhang, H.F., 2013. Optimal operation of multi-reservoir system by multi-elite guide particle swarm optimization. International Journal of Electrical Power & Energy Systems 48, 58-68.

Zhang, X., Beeson, P., Link, R., Manowitz, D., Izaurralde, R.C., Sadeghi, A., Thomson, A.M., Sahajpal, R., Srinivasan, R.,Arnold, J.G., 2013. Efficient multi-objective calibration of a computationally intensive hydrologic model with parallel computing software in Python. Environmental Modelling & Software 46, 208-18.

Zhao, T.T.G., Cai, X.M., Lei, X.H., Wang, H., 2012a. Improved dynamic programming for reservoir operation optimization with a concave objective function. Journal of Water Resources Planning and Management 138,

590-596.

Zhao, T.T.G., Zhao, J.S., Yang, D.W., 2012b. Improved dynamic programming for hydropower reservoir operation. Journal of Water Resources Planning and Management. doi: 10.1061/(ASCE)WR.1943-5452.0000343.

Zhao, G., Bryan, B.A., King, D., Luo, Z.K., Wang, E.L., Bende-Michl, U., Song, X.D., Yu, Q., 2013. Large-scale, high-resolution agricultural systems modeling using a hybrid approach combining grid computing and parallel processing. Environmental Modelling & Software 41, 231-238.

Zhu, X.Y., Li, K.L., Salah, A., 2013. A data parallel strategy for aligning multiple biological sequences on multi-core computers. Computers in Biology and Medicine 43, 350-361.