Advances in Water Resources 67 (2014) 1–15

Contents lists available at ScienceDirect

Advances in Water Resources

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

# A parallel dynamic programming algorithm for multi-reservoir system optimization

Xiang Li [a], Jiahua Wei [a], Tiejian Li [a], Guangqian Wang [a], William W.-G. Yeh [b,*]

[a] State Key Laboratory of Hydroscience & Engineering, Tsinghua University, Beijing 100084, China
[b] Department of Civil and Environmental Engineering, University of California, Los Angeles, CA 90095, USA

ARTICLE INFO

*Article history:*
Received 8 May 2013
Received in revised form 8 January 2014
Accepted 12 January 2014
Available online 30 January 2014

*Keywords:*
Dynamic programming
Multi-reservoir system optimization
Joint operation
Parallel computing

ABSTRACT

This paper develops a parallel dynamic programming algorithm to optimize the joint operation of a multi-reservoir system. First, a multi-dimensional dynamic programming (DP) model is formulated for a multi-reservoir system. Second, the DP algorithm is parallelized using a peer-to-peer parallel paradigm. The parallelization is based on the distributed memory architecture and the message passing interface (MPI) protocol. We consider both the distributed computing and distributed computer memory in the parallelization. The parallel paradigm aims at reducing the computation time as well as alleviating the computer memory requirement associated with running a multi-dimensional DP model. Next, we test the parallel DP algorithm on the classic, benchmark four-reservoir problem on a high-performance computing (HPC) system with up to 350 cores. Results indicate that the parallel DP algorithm exhibits good performance in parallel efficiency; the parallel DP algorithm is scalable and will not be restricted by the number of cores. Finally, the parallel DP algorithm is applied to a real-world, five-reservoir system in China. The results demonstrate the parallel efficiency and practical utility of the proposed methodology.
© 2014 Elsevier Ltd. All rights reserved.

## 1. Introduction

Dynamic programming (DP), an algorithm attributed largely to Bellman [3], is developed for optimizing a multi-stage (the term "stage" represents time step throughout the paper) decision process. If the return or cost at each stage is independent and satisfies the monotonicity and separability conditions [23], the original multi-stage problem can be decomposed into stages with decisions required at each stage. The decomposed problem then can be solved recursively, two stages at a time, using the recursive equation of DP. DP is particularly suited for optimizing reservoir management and operation as the structure of the optimization problem conforms to a multi-stage decision process. Over the past four decades, DP had been used extensively in the optimization of reservoir management and operation [4,6,8,13,22,35,37–40].

In the discrete form of DP, storage of each reservoir is discretized into a finite number of levels. By exhaustive enumeration over all possible combinations of discrete levels at each stage for all reservoirs in a system, global optimality can be assured in a discrete sense. However, the well-known "curse of dimensionality" [2] limits the application of DP to multi-state variable problems, as the state space increases exponentially with an increase in the number of state variables. This drastic increase in state space and the consequent random access memory (RAM) requirement quickly can exceed the hardware capacity of a modern computer [13]. A variety of DP variants, such as incremental dynamic programming (IDP) [15], dynamic programming successive approximations (DPSA) [14], incremental dynamic programming and successive approximations (IDPSA) [32] and discrete differential dynamic programming (DDDP) [9] have been proposed to alleviate the dimensionality problem. However these variants all require an initial trajectory for each state variable. For a non-convex problem, there is no assurance of convergence to the global optimum. Recently, Mousavi and Karamouz [22] reduced the computation time of a DP model for a multi-reservoir system by diagnosing infeasible storage combinations and removing them from further computations. Zhao et al. [40] proposed an improved DP model for optimizing reservoir operation by taking advantage of the monotonic relationship between reservoir storage and the optimal release decision. However, the model only can be applied to reservoir operation with a concave objective function.

Because of the hardware limitations of a single computer as well as large-scale computing requirements, parallel computing has been applied in many fields [25]. In the water resources field, there are several successful examples. Bastian and Helmig [1]

* Corresponding author. Tel.: +1 3108252300; fax: +1 3108257581.
*E-mail addresses:* l-xiang09@mails.tsinghua.edu.cn, ideal.thu@gmail.com (X. Li), weijiahua@tsinghua.edu.cn (J. Wei), litiejian@tsinghua.edu.cn (T. Li), dhhwgq@tsinghua.edu.cn (G. Wang), williamy@seas.ucla.edu (W.W.-G. Yeh).

http://dx.doi.org/10.1016/j.advwatres.2014.01.002
0309-1708/© 2014 Elsevier Ltd. All rights reserved.

employed a data parallel implementation of a Newton-multi-grid algorithm for two-phase flow in porous media. Kollet and Maxwell [10] incorporated an efficient parallelism into an integrated hydrologic model – ParFlow. Tang et al. [31] used the master–slave and multi-population parallelization schemes for the Epsilon-Nondominated Sorted Genetic Algorithm-II and applied them to several water resources problems. Wang et al. [33] designed a common parallel framework, while Li et al. [16] introduced a dynamic parallel algorithm for hydrological model simulations. Rouholahnejad et al. [27] proposed a parallelization framework for hydrological model calibration. Wu et al. [36] parallelized the Soil and Water Assessment Tool. These previous studies demonstrated that by using parallel computing associated with proper parallelization strategies for optimization or simulation, solutions can be improved and computation times can be reduced dramatically. Although parallel computing has been used extensively in hydrologic models, the potential has not been explored fully in the field of reservoir operation [29].

Over the last several decades, with the rapid development of high-performance computing (HPC) environments, parallel dynamic programming algorithm has been studied from theories gradually to applications. From the theoretical point of view, Casti et al. [5] presented various forms of parallelism and the corresponding parallel algorithms for DP. Rytter [28] considered some DP problems and estimated the computation complexity and the number of computing processes based on a hypothetical parallel random access machine, namely the shared memory architecture. However, the shared memory architecture may not be supportable for large RAM requirement problems, since all parallel tasks access the finite RAM in the architecture (because of motherboard size restrictions). From the practical point of view, El Baz and Elkihel [7] designed several load-balancing strategies and applied a parallel DP algorithm with Open MP, a protocol based on the shared memory architecture, to the 0–1 knapsack problem. Martins et al. [19] and Tan et al. [30] parallelized DP algorithms for a class of problems, such as biological sequence comparisons. Piccardi and Soncini-Sessa [24] studied the discretization and inflow correlation on the solution reliability of a stochastic dynamic programming (SDP) and exploited parallel computing to a one-reservoir optimal control problem; the parallelization was attributed largely to a vectorizing compiler or parallelizing compiler. Li et al. [17] implemented a knowledge-based approach to accurately determine hydropower generation and developed a master–slave parallel DP algorithm on an HPC system to optimize the coordinated operation of the Three Gorges Project and Gezhouba Project cascade hydropower plants in China. The master–slave is a frequently-used parallel paradigm for parallelizing a DP algorithm, where the master process has a full version of the DP algorithm, and calls slave processes to evaluate and return objective values and saves all variables in the RAM of the master process. However, this parallel paradigm merely shortens computation time but does not alleviate the RAM requirement.

In this paper, we develop a parallel DP algorithm for multi-reservoir system optimization. The parallel DP algorithm aims at reducing the computation time and alleviating the RAM requirement, taking advantage of parallel computing on the distributed memory architecture. The paper is organized as follows: Section 2 formulates a DP model for a multi-reservoir system optimization problem; Section 3 analyzes the parallelism inherent to the DP algorithm and then proposes a peer-to-peer parallel paradigm for the DP algorithm; Section 4 considers the classic four-reservoir example and tests the parallel DP algorithm on the problem; Section 5 applies the parallel DP algorithm to a real-world five-reservoir system in China; finally, Section 6 presents the conclusions.

## 2. Dynamic programming model

### 2.1. Objective function

A typical objective function for the optimal operation of a multi-reservoir system is either to maximize benefit or minimize operational cost. If more than one objective is involved, they can be combined by the weighting method to form a composite objective function. Without loss of generality, let us consider the maximization problem. According to Bellman's principle of optimality [3], the traditional forward recursive equation of an $n$-dimensional DP model (the term "dimension" here refers to the number of reservoirs) can be represented as:

$$F^*_{t+1}(\mathbf{S}(t+1)) = \max\{f_t(\mathbf{S}(t),\mathbf{S}(t+1)) + F^*_t(\mathbf{S}(t))\} \quad (1)$$

where $t$ is the time index, $t \in [1,T]$; $\mathbf{S}(t)$ is the storage vector at the beginning of time step $t$; $\mathbf{S}(t) = [S_1(t),\ldots,S_i(t),\ldots,S_n(t)]^T$; $\mathbf{S}(t+1)$ is the storage vector at the end of time step $t$; $i$ is the reservoir index, $i \in [1,n]$; $F^*_t(\bullet)$ is the maximum cumulative return from the first time step to the beginning of the $t$th time step resulting from the joint operation of $n$ reservoirs; initially, $F^*_1(\bullet) = 0$; $F^*_{t+1}(\bullet)$ is the maximum cumulative return from the first time step to the end of the $t$th time step resulting from the joint operation of $n$ reservoirs; and $f_t(\bullet)$ is the objective function to be maximized during time step $t$. Note that Eq. (1) is the inverted form of a DP model with reservoir storages as the decision variables. In the non-inverted DP model, the releases are the decision variables. For a deterministic DP model, the ending storage is related to the beginning storage by the continuity equation.

### 2.2. Constraints

In the operation of a multi-reservoir system, each individual reservoir is subject to its own set of constraints, while the reservoir system is subject to system constraints brought by the interconnection of reservoirs. Specifically, we consider the following constraints:

- Continuity equation:

$$\mathbf{S}(t+1) = \mathbf{S}(t) + \mathbf{I}(t) - \mathbf{M} \bullet \mathbf{R}(t) \quad \forall t \quad (2)$$

where $\mathbf{I}(t)$ is the vector of inflows to reservoirs ($i = 1,\ldots,n$) during time step $t$; $\mathbf{R}(t)$ is the vector of total releases from reservoirs ($i = 1,\ldots,n$) during time step $t$; $\mathbf{R}(t) = [R_1(t),\ldots,R_i(t)\ldots,R_n(t)]^T$; and $\mathbf{M}$ is the $n \times n$ reservoir system connectivity matrix. Without loss of generality, we assume that evaporation loss is balanced by precipitation.

- Initial and final reservoir storages:

$$\mathbf{S}(1) = \mathbf{S}^{\text{initial}} \quad (3)$$

$$\mathbf{S}(T+1) \geqslant \mathbf{S}^{\text{final}} \quad (4)$$

where $\mathbf{S}^{\text{initial}}$ and $\mathbf{S}^{\text{final}}$ are the vectors of initial storages and final expected storages of reservoirs ($i = 1,\ldots,n$).

- Lower and upper bounds on storages:

$$\mathbf{S}^{\min}(t+1) \leqslant \mathbf{S}(t+1) \leqslant \mathbf{S}^{\max}(t+1) \quad \forall t \quad (5)$$

where $\mathbf{S}^{\min}(t+1)$ and $\mathbf{S}^{\max}(t+1)$ are the vectors of minimum and maximum storages of reservoirs ($i = 1,\ldots,n$) at the end of time step $t$.

- Lower and upper bounds on releases:
  For all reservoirs:

$$\mathbf{R}^{\min}(t) \leqslant \mathbf{R}(t) \leqslant \mathbf{R}^{\max}(t) \quad \forall t \quad (6)$$

where $\mathbf{R}^{\min}(t)$ is the vector of the minimum required releases from reservoirs ($i = 1,\ldots,n$) during time step $t$; and $\mathbf{R}^{\max}(t)$ is the vector

Download English Version:

https://daneshyari.com/en/article/4525573

Download Persian Version:

https://daneshyari.com/article/4525573

Daneshyari.com