A Parallel Shuffled Complex Evolution Model Calibrating Algorithm to Reduce Computational Time

Muttil, N. [1,*], S.Y. Liong [2], O. Nesterov [2]

[1] School of Architectural, Civil and Mechanical Engineering and Institute for Sustainability & Innovation, Victoria University, P.O. Box 14428, Melbourne 8001, Vic., Australia
[2] Tropical Marine Science Institute (TMSI), National University of Singapore, Singapore
[*] Formerly, Research Fellow at TMSI, National University of Singapore, Singapore
Email: nitin.muttil@vu.edu.au

Keywords: *Model calibration, hydrologic models, parallel computing, message passing interface (MPI)*

EXTENDED ABSTRACT

The Shuffled Complex Evolution (SCE-UA) method has been widely applied for calibration of rainfall-runoff models and has been shown to be robust and efficient search algorithm. In spite of its superiority, since many commonly used rainfall-runoff models have large simulation times, the use of model calibrating algorithms may become impractical due to the high computational time involved. This would necessitate the use of superior parallel computing technologies in the calibrating algorithms, with the aim of reducing the computational times. This study aims to parallelize the SCE-UA to accelerate the model calibrating process.

In the field of evolutionary algorithms (EAs), the use of parallel computing within the standard Genetic Algorithm (GA) has been researched into for the past two to three decades. There are three main parallel paradigms in evolutionary algorithms: the master-slave model, the diffusion model and the multi-population (or island) model. In this study, the master-slave model (see Figure 1) is adopted, since it is the simplest potential parallelization strategy, which can be easily implemented on a cluster of PCs linked by a Local Area Network (LAN). Moreover, in calibrating rainfall-runoff models, since the model simulation times are significantly greater than the communication times (between master and slaves), very good speedups are possible using master-slave models.

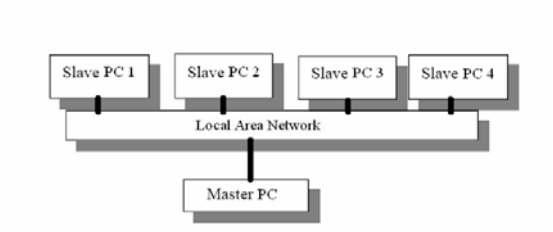

Figure 1. The master-slave parallelization strategy

In the SCE-UA, since the population of points is split into a number of complexes that evolve independent of each other, it is a fully parallel problem. Thus, in the proposed parallel version of the SCE-UA, model evaluations for each complex are executed in parallel on separate PCs, as compared to the sequential evaluation of complexes on a single PC in the original SCE-UA. The parallel computing capability is based on master-slave architecture, and the Message Passing Interface (MPI) is used to establish the master-slave interactions.

In this parallel SCE-UA, the master process is the SCE-UA, which is a lightweight process, as its computational cost is negligible in comparison to the slave processes, which actually do the model simulations. Therefore, the master process and the first slave (also the first complex in SCE-UA) are run on the same physical PC.

In this study, it is observed that the parallelization of SCE-UA leads to 'linear speedups' in the model calibrating process. This means that if a calibration run requires 20 hours to complete 1000 model simulations, then using 4 PCs in parallel would take 5 hours to complete the same number of model simulations. Moreover, parallelization of SCE-UA using PCs that are linked by a LAN is an easy and affordable alternative to achieve significant reduction of the computational time, without resorting to expensive high-end systems. Such savings in computational time in turn facilitates significantly more search of the parameter space during the calibration process.

1. INTRODUCTION

With the availability of high-performance computer technology, many complex hydrologic and hydrodynamic models are available, which are characterized by a multitude of parameters. Due to spatial and temporal variability, measurement errors, etc., the values of many of these parameters will not be exactly known, necessitating model calibration using a historical record of input-output data. The successful application of such models depends on how well the model is calibrated.

Traditionally, systematic manual methods have been used for the calibration of conceptual rainfall-runoff models. However, in order to obtain reliable results this type of calibration requires that the user be an expert and it is usually a very long time-consuming process. Because of this, there has been a great deal of research into the development of automatic methods for parameter estimation, which utilize the speed and power of digital computers. These methods may be divided into two categories: local and global methods. Local methods are susceptible to getting trapped in local optima, since the shape of the response surface (the objective function mapped out in the parameter space) is known to be complex, with the existence of many regions of attraction and multiple local optima in each region. To deal with this problem of multiple local minima, global optimization methods have been applied extensively. These methods are global in the sense that they constitute a parallel search of the search space (as opposed to a point by point search) by using a population of potential solutions. Among the global model calibrating algorithms, the Genetic Algorithms (Wang, 1991) and Shuffled Complex Evolution (SCE-UA, Duan et al., 1992) have been popular.

In various studies, SCE-UA based algorithms have been demonstrated to be robust and efficient in calibrating rainfall-runoff models (Kuczera, 1997; Franchini et al., 1998; Muttil and Liong, 2004). A further consideration in assessing its performance is that of the computational time required for the calibrations. Application of global optimization methods like the SCE-UA to high-dimensional parameter estimation problems requires the solution of a large number of model runs. The computational burden of these model runs often renders the use of such advanced global optimization algorithms for calibrating parameters in complex hydrologic models impractical. In such a scenario, it may be imperative to resort to super- or/and parallel computing.

In the field of water resources and hydrology, parallel computing has been used only in the very recent past. Cheng et al. (2005) used a parallel genetic algorithm (PGA) for watershed model calibration in order to speed up the calibration procedure. In calibrating the Xinanjiang conceptual rainfall-runoff model, they demonstrated that the PGA was superior to the serial GA with respect to overall optimization time and also the stability of the solution. Vrugt et al. (2006) present a parallel version of the Shuffled Complex Evolution Metropolis (SCEM-UA) global optimization algorithm for stochastic estimation of parameters in environmental models. Using three case studies, which include calibration of the SAC-SMA conceptual rainfall-runoff model, they also demonstrate that parallel parameter estimation results in considerable time savings when compared with traditional sequential optimization runs. In yet another study, Tang et al. (2007) demonstrate and compare the master-slave and the multi-population parallelization strategies (described later in Section 2) for the Epsilon-Nondominated Sorted Genetic Algorithm-II (ε-NSGAII) on a hydrologic model calibration test case and also on a discrete, constrained groundwater monitoring application. They conclude that the master-slave approach is superior to the multi-population approach on both these water resources applications, especially considering its simplicity and ease of implementation.

This study aims to speed up the model calibration process using the SCE-UA algorithm. Parallel computing capability is implemented into the SCE-UA by evaluating each complex in parallel on separate personal computers (PCs). The parallel computing capability is based on master-slave architecture, and the Message Passing Interface (MPI) is used to establish the master-slave interactions. Traditionally, this speeding up would require expensive and high-end systems. Recently, due to the decreasing hardware cost and the increasing computation power of workstations, using a cluster of PCs has become an affordable and attractive alternative to high-end systems. Clusters of PCs have the following advantages: they have a better price/performance ratio than high-end systems; they may be upgraded more frequently; they can employ different kinds of machines; and, their aggregate power scales up with the increase in the number of PCs (Cheng et al., 2005). Moreover, a cluster of PCs may now be easily linked by a Local Area Network (LAN) to provide researchers with more powerful integrated PC-LAN systems. In this study, it is observed that the parallel evaluation of the complexes in the SCE-UA on separate PCs is an easy and affordable

alternative to achieve significant reduction of the computational time, thus making it possible for an exhaustive search of the parameter space. The next section briefly describes the different ways in which parallel computing has been used in the field of evolutionary algorithms (EAs). This is followed by the description of how parallel computing is incorporated into the SCE-UA. Finally, conclusions that can be drawn from this study are presented.

2. PARALLEL COMPUTING IN EVOLUTIONARY ALGORITHMS

During the past decade there has been considerable progress in the development of distributed computer systems using the power of multiple processors to efficiently solve complex, high-dimensional computational problems (Eklund, 2004). Parallel computing offers the possibility of solving computationally challenging optimization problems in less time than is possible using ordinary serial computing (Goldberg et al., 1995). In the field of evolutionary algorithms, the use of parallel computing within the standard Genetic Algorithm (GA) has been researched into for the past two to three decades. Bethke (1976) made one of the first investigations of parallel GA models. He described a global population with a partial exchange of individuals in successive generations. His analysis showed that, implemented on parallel hardware, near-linear speedup could be achieved. One of the first real implementations of parallel GA was made by Tanese (1989). She conducted studies of different topologies and migration rates on a distributed population model on a 64 processor N-CUBE system. In some experiments she reported super-linear speedup compared to sequential GA. Such parallel GAs (PGAs) has been subsequently applied in many fields (Abramson et al., 1993; Pereira and Lapa, 2003).

In this section, we briefly review the key parallelization strategies in genetic algorithms. PGAs may be categorized into three different basic approaches: master–slave GAs, cellular GAs (fine-grained) and multi-population GAs (island or distributed) (Cantu-Paz, 1997; Pereira and Lapa, 2003; Cheng et al., 2005).

The master–slave GA, as shown in Figure 1, is the parallel version of the simple GA. Consequently, it does not alter nor restrict the genetic operations. Only the fitness evaluation is distributed among the available machines. Generation control, selection and genetic operations are not paralleled. The search-space exploration of this PGA paradigm is conceptually identical to that of GAs. It is important to note that, in order to realize any computational speedup, computations of the objective function should be fairly complex and time consuming. Otherwise, the communication time might overwhelm the computation time and hence poor speedup results. Thus, this method should only be used when the effort on fitness evaluation is substantial.

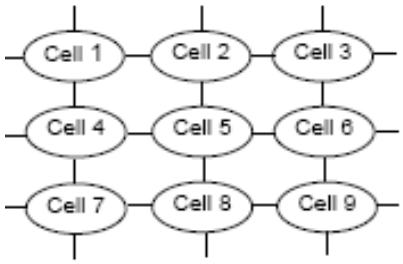

Figure 2. The cellular GA parallelization strategy

In cellular GAs, each individual member of the population is put into a processor (cell). The cells are geographically arranged so that neighborhood restrictions will be imposed in the crossover operations, as shown in Figure 2. This paradigm requires a number of processors and is usually running on massive parallel computers with single instruction multiple data stream (SIMD). The platform is not easily available for ordinary users and therefore this method is rarely used unless a large-scale SIMD parallel computer is available.

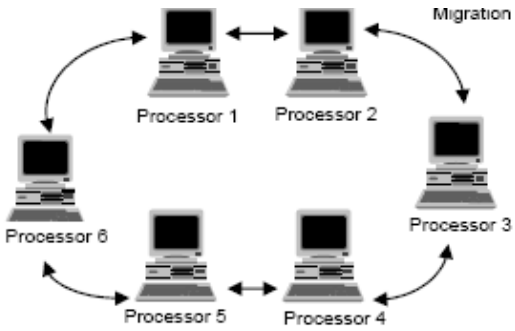

Figure 3. The multi-population parallelization strategy

The multi-population GA is an 'island' paradigm, which is sometimes termed distributed or coarse-grained approach. The paradigm is based on the phenomenon of natural populations evolving in relative isolation, such as those that might occur within some ocean island chains with limited migration. Communication backbones can connect processors in logical or physical geometric structures such as rings, meshes, triangles and hypercubes. Each sub-population is located in a processor (island) and evolved by a separate process. In order to promote cooperation between

processors, a new operator, called migration is created. According to some predefined strategy, individuals migrate from one processor to another. As migration occurs, information about different regions of the search space is exchanged between processors, thus providing more diversity in the search. The paradigm can therefore be implemented on a PC-LAN with a relatively small number of workstations. Figure 3 shows a typical ring topology of this paradigm.

3. IMPLEMENTING PARALLELIZATION IN SCE-UA

In this section, we present the implementation of parallel computing capability into the original serial SCE-UA. Of the three types of parallelization approaches discussed in the previous section, in this study, a master-slave approach is adopted since it is the simplest potential parallelization strategy, which can be easily implemented on a cluster of PCs linked by a LAN. Moreover, it should be noted that for many hydrologic and hydraulic models, since the model simulation times are significantly larger than the communication times (between master and slaves), very good speedups are possible using the master-slave approach. The Message Passing Interface (MPI) is used to establish the master-slave interactions.

The authors would like to point out that in a previous study, a parallel version of SCE-UA was developed by Sharma et al. (2006), who also used a master-slave approach. They had implemented the parallel SCE-UA on a Idra and Deeppurple cluster, running Red Hat Linux 7.2 and Tru64 UNIX operating systems, respectively with Compaq Alpha architecture. These operating systems and high end servers are not as popular as the Windows based PCs, both in terms of ease of use and cost. Thus, the current implementation of parallel SCE-UA is much more affordable and can be easily implemented on the PC-LAN system. The following sub-sections describe the MPI library used and details of the parallel SCE-UA.

3.1. The MPI Library

The Message-Passing Interface (MPI) is a specification for the user interface to message-passing libraries for parallel computers. MPI can be used to write programs for efficient execution on a wide variety of parallel machines, including massively parallel supercomputers, shared-memory multiprocessors and networks of workstations. MPI allows the coordination of a program running as multiple processes in a distributed memory environment, yet is flexible enough to also be used in a shared memory system. MPI programs always work with processes, although commonly people talk about processors. When one tries to get maximum performance, one process per processor is selected as part of the mapping activity; this mapping activity happens at runtime, through the agent that starts the MPI program, normally called 'mpirun'. The standardization of the MPI library is one of its most powerful features. What it means is the parallel programmer can write code containing MPI subroutine and function calls that will work on *any* machine on which the MPI library is installed without having to make changes in his code. A complete detail of the MPI is provided in Gropp et al. (1994) and Pacheco (1997).

The MPICH, an implementation of the full MPI-1.2 specification, is used in this study. MPICH is a freely available, portable implementation of MPI, a standard for message-passing for distributed-memory applications used in parallel computing. MPICH is available for Microsoft Windows and for most flavours of UNIX (including Linux and Mac OS X). Moreover, MPICH is a developed program library. More information including tutorials can be found on the MPICH web site at: http://www-unix.mcs.anl.gov/mpi/mpich1/.

3.2. The original SCE-UA

The shuffled complex evolution (SCE-UA) algorithm was developed at the University of Arizona (Duan et al. 1992, 1993) to deal with the difficult problems encountered in the calibration of conceptual rainfall-runoff models. It incorporates the best features from several existing methods, including competitive evolution, the combination of random and deterministic strategies, the concepts of controlled random search, and complex shuffling.

In essence, the SCE-UA begins with an initial population of points sampled randomly from the feasible space. The population is partitioned into one or more complexes, each containing a fixed number of points. Each complex is allowed to evolve based on a competitive evolution technique that uses the simplex search method (Nelder and Mead, 1965) to direct the search in the correct direction. Periodically, the entire population is shuffled and points are reassigned to new complexes to enable information sharing. This shuffling strategy reduces the chance of complexes being trapped on flat regions and thus converging prematurely. As the search progresses, the entire population tends to converge toward the neighbourhood of the global optimum, provided the initial population size is sufficiently large. For

a lucid explanation on the details of the algorithm, the reader is referred to Duan et al. (1994).

The original SCE-UA is a serial algorithm in the sense that the evaluation of the objective function is done in a sequential manner on a single PC. The next sub-section presents the proposed parallel version of the SCE-UA.

3.3. The parallel SCE-UA

A feasible way of solving a problem as quickly as possible is to partition the problem into smaller independent pieces, so that all the pieces can be solved simultaneously (or in parallel). In the field of parallel computing, problems for which no particular effort is needed to segment it into a very large number of parallel tasks, and there is no essential dependency (or communication) between those parallel tasks are called 'embarrassingly parallel' problems. In other words, each task can be computed independently from every other task, thus each task could be made to run on a separate processor to achieve quicker results. In the SCE-UA, since the population of points is partitioned into a number of complexes and the complexes evolve independent of each other, this problem falls within the category of an embarrassingly parallel one. Thus, it seems natural to assign the model simulations within each complex to a slave PC. Hence, the number of complexes and the number of slave PCs used are the same.

The parallel SCE-UA is same as the original serial algorithm, except that the complexes are evolved in parallel on multiple slave PCs rather than on a single PC. The slave PCs are controlled by the master process (on the master PC) and information is passed to and fro between the master process and the slaves. The master process is the SCE-UA, which is a lightweight process, as the computational cost of SCE-UA is negligible in comparison to the slave processes, which actually do the computationally heavy model simulations. Therefore, the master process and the first slave (also the first complex in SCE-UA) are run on the same physical processor. The evolved complexes from multiple slave PCs are sent back to the master PC, where the complexes are combined and shuffled. If the stopping criteria are not met, the population is again partitioned into complexes and a new loop starts. The working of the parallel SCE-UA is demonstrated in Figure 4.

The efficacy of a parallel processing application can be judged using the concept of 'speedup', S_p, defined in Eqn. (1) below:

$$S_p = T_s / T_p \qquad (1)$$

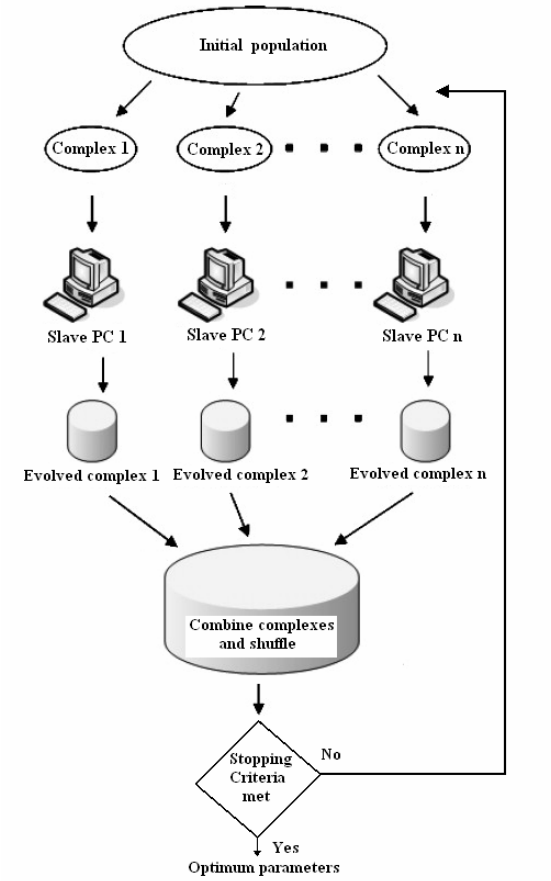

Figure 4. Simplified working of the parallel SCE-UA algorithm

Speedup compares the clock time required to solve an application in serial (i.e. on one processor), T_s, with the clock time required using multiple processors, T_p. The proposed parallel SCE-UA is observed to attain 'linear speedups', which means that when P processors are used to solve an application, the parallel computing time, T_p, will equal (T_s / P), i.e. speedup is equal to the number of processors used. For example, if a calibration run requires 20 hours to complete 1000 model simulations, then using 5 PCs in parallel would take 4 hours to complete the same number of model simulations.

4. CONCLUSION

This study proposes a parallel version of the SCE-UA algorithm with the aim of reducing the computational time for calibrating rainfall-runoff models. In the SCE-UA, since the model simulation in the partitioned complexes are independent of each other, it seems natural to assign the simulations within each complex to a slave PC, making the master-slave parallelization strategy an obvious choice. Moreover, the master-slave strategy is found to be affordable and easy to

implement on a low-cost PC-LAN system and thus there is no need to resort to expensive high-end systems. It is also observed that the parallelization of SCE-UA leads to 'linear speedups' in the model calibrating process, resulting in substantial reduction in computational time for calibrating rainfall-runoff models. This in turn facilitates significantly more search of the parameter space during the calibration process.

5. REFERENCES

Abramson, D., G. Mills and S. Perkins (1993), Parallelization of a genetic algorithm for the computation of efficient train schedules, Proceedings of the 1993 Parallel Computing and Transputers Conference, 139–149.

Bethke, A.D. (1976), Comparison of genetic algorithms and gradient-based optimizers on parallel processors: efficiency of use of processing capacity, Technical Report No. 197, University of Michigan, Logic of Computers Group, Ann Arbor, MI.

Cantu-Paz, E. (1997), A survey of parallel genetic algorithms, Illinois Genetic Algorithms Laboratory Report No. 97003, University of Illinois at Urbana-Champaign, IL.

Cheng, C.T., X.Y. Wu and K.W. Chau (2005), Multiple criteria rainfall-runoff model calibration using a parallel genetic algorithm in a cluster of computers, *Hydrological Sciences Journal*, 50(6), 1069-1087.

Duan, Q., S. Sorooshian and V.K. Gupta (1992), Effective and efficient global optimization for conceptual rainfall-runoff models, *Water Resources Research*, 28(4), 1015-1031.

Duan, Q.A., V.K. Gupta and S. Sorooshian (1993), Shuffled complex evolution approach for effective and efficient global minimization, *Journal of Optimization Theory and Applications*, 76(3), 501-521.

Duan, Q.A., S. Sorooshian and V.K. Gupta (1994), Optimal use of the SCE-UA global optimization method for calibrating watershed models, *Journal of Hydrology*, 158, 265-284.

Eklund, S. E. (2004), A massively parallel architecture for distributed genetic algorithms, *Parallel Computing*, 30, 647-676.

Franchini, M., G. Galeati and S. Berra (1998), Global optimisation techniques for the calibration of conceptual rainfall-runoff models, *Journal of Hydrologic Science*, 43(3), 443-458.

Goldberg, D.E., H. Kargupta, J. Horn, E. Cantu-Paz (1995), Critical deme size for serial and parallel genetic algorithms, Technical Report 95002, GA Laboratory, University of Illinois, Urbana-Campaign, IL.

Gropp, W., E. Lusk and A. Skjellum (1994), Using MPI: Potable parallel Programming with the Message-Passing Interface, MIT Press, Cambridge, Mass.

Kuczera, G. (1997), Efficient subspace probabilistic parameter optimization for catchment models, *Water Resources Research*, 33(1), 177-185.

Muttil, N. and S.Y. Liong (2004), A superior exploration-exploitation balance in shuffled complex evolution, *Journal of Hydraulic Engineering*, ASCE, 130(12), 1202-1205.

Nelder, J. A., and R. Mead (1965), A simplex method for function minimization, *Journal of Computing*, 7, 308-313.

Pacheco, P.S. (1997), Parallel Programming with MPI, Morgan Kaufmann Publishers, San Francisco, CA.

Pereira, C. M. N. A. and C. M. F. Lapa (2003), Coarse-grained parallel genetic algorithm applied to a nuclear reactor core design optimization problem, *Annals of Nuclear Energy*, 30, 555–565.

Sharma, V., D. A. Swayne, D. Lam and W. Schertzer (2006), Parallel shuffled complex evolution algorithm for calibration of hydrological models, Proceedings of the 20th International Symposium on High-Performance Computing Symposium (HPCS'06), 30-35.

Tanese, R. (1989), Distributed genetic algorithm, Proceedings of the 3rd International Conference on Genetic algorithms, 434–439.

Tang, Y., P.M. Reed and J.B. Kollat (2007), Parallelization strategies for rapid and

robust evolutionary multiobjective optimization in water resources applications, *Advances in Water Resources*, 30, 335-353.

Vrugt, J. A., B. O. Nuallain, B. A. Robinson, W. Bouten, S.C. Dekker and P. M. A. Sloot (2006), Application of parallel computing to stochastic parameter estimation in environmental models, *Computers & Geosciences*, 32 (8), 1139-1155.

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