

PGO: a Parallel Computing Platform for Global Optimization Based on Genetic Algorithm

Kejing He ^{a,e,*}, Li Zheng ^b, Shoubin Dong ^a, Liqun Tang ^c,
Jianfeng Wu ^d, Chunmiao Zheng ^e

^a*Guangdong Key Laboratory of Computer Network, South China University of
Technology, Guangzhou, China*

^b*Center for Agricultural Resources Research, IGDB, Chinese Academy of
Sciences, Shijiazhuang, China*

^c*College of Traffic and Communications, South China University of Technology,
Guangzhou, China*

^d*Department of Earth Sciences, Nanjing University, Nanjing, China*

^e*Department of Geological Sciences, University of Alabama, Tuscaloosa, AL,
United States*

Abstract

This paper presents the design, architecture and implementation of a general parallel computing platform, termed PGO, based on the Genetic Algorithm for global optimization. PGO provides an efficient and easy-to-use framework for parallelizing the global optimization procedure for general scientific modeling and simulation processes. Along with a core optimization kernel built on a modified Genetic Algorithm and a parallel computing technique, PGO also includes a general input generator

and an output extractor that can facilitate its easy integration with various scientific computing tasks. In current paper, we demonstrate the efficiency and versatility of PGO with two different applications: (1) the parallelization of a large scale parameter estimation problem associated with modeling water flow in a heterogeneous deep vadose zone; (2) the parallelization of a complex simulation-optimization procedure for searching for an optimal groundwater remediation design. PGO is developed as an open source code, and is independent of the computer operating system. It has been tested in a heterogeneous computing environment consisting of Solaris 9, Fedora Core 2 Linux, and Microsoft Windows machines, and is freely available for download from <http://grid.scut.edu.cn/PGO/>.

Key words: parallel computing, high performance computation platform, global optimization, the Genetic Algorithm

1 Introduction

In scientific and engineering computing, optimization is one of the most frequently encountered computational problems. Optimization refers to the process of identifying values for the unknown model parameters or control variables so that the given objective function achieves its optimum. When the application model is linear or the objective function is relatively simple, local gradient-based optimization techniques (such as Newton and Quasi-Newton methods) are effective and efficient. Local gradient-based optimization methods have been widely used in many popular tools for the optimization process. For example, PEST (Doherty, 2004) uses a nonlinear optimization technique

* Corresponding author. Tel.: +86-20-8711-0288; fax: +86-20-8711-0019.
Email address: kjhe@scut.edu.cn (Kejing He).

known as the Gauss-Marquardt-Levenberg method, and UCODE (Poeter and Hill, 1998) solves the nonlinear regression problem by minimizing a weighted least squares objective function with respect to the unknown parameter values using a modified Gauss-Newton method. However, if the application model is highly nonlinear, local search procedures will be less effective and optimization results will be very sensitive to the choices of initial values.

Many researchers have successfully applied global optimization methods, such as the Genetic Algorithm (GA), rather than local gradient-based optimization techniques to geoscientific computational problems. The Genetic Algorithm is a random search technique based on the theory of Darwinian evolution. It was first introduced by Holland (1975), and was later extended by De Jong (1985) and Goldberg (1989). In geoscientific computation, the GA has found applications in calibrating conceptual rainfall-runoff models (Wang, 1991), solving a multiple objective groundwater pollution containment problem (Ritzel et al., 1994), conducting the inverse modeling in hydrogeology (Karpouzou et al., 2001; Prasad and Rastogi, 2001), and optimizing the groundwater remediation and contamination control systems (Erickson et al., 2002; Bayer and Finkel, 2004).

Despite its many applications in scientific research, the routine adoption of GAs for general modeling and simulations is not an easy task. First, the modelers need interfaces for data exchange between the simulation model and the chosen GA tool. These interfaces often include input generation, output extraction and the mapping of model's unknown parameters/variables to GA's abstract chromosomes. Second, the Genetic Algorithm tends to converge very slowly when applied to a highly nonlinear external model involved many parameters and subsequently an enormous search space. It may take days even

weeks to accomplish an optimization task. While the design of Genetic Algorithm is intrinsically suitable for the parallelization of its execution, the integration of a general model with a parallel computing environment is however not a trivial process. Modelers need to cope with the message transferring between different machines and the distributed data storage and access during optimization progress.

The purpose of this paper is to present a general parallel computing platform, termed PGO, based on the Genetic Algorithm for facilitating the global optimization in routine scientific computation. PGO provides an efficient and easy-to-use framework for parallelizing the global optimization procedure based on the GA. Along with a core optimization kernel built on a modified Genetic Algorithm and a parallel computing technique, PGO also includes a general input generator and a output extractor that can facilitate its easy integration with various scientific computing tasks. PGO is programmed in Perl script language and developed as an open source code. It is independent of the computer operating system and has been tested in a heterogeneous computing environment consisting of Solaris 9, Fedora Core 2 Linux, and Microsoft Windows machines.

PGO and its user manual are in public domain and available for download from <http://grid.scut.edu.cn/PGO/>. The presentation of this paper is organized as follows. Following the introduction in section 1, section 2 explains the theory of optimization, Genetic Algorithm, parallel technique, and their integration. Section 3 describes the architecture and configuration of PGO. In section 4, PGO is applied to (1) the parallelization of a large scale parameter estimation problem associated with modeling water flow in a heterogeneous deep vadose zone; and (2) the parallelization of a complex simulation-optimization pro-

cedure for searching for an optimal groundwater remediation design. Finally, some concluding remarks are made in section 5.

2 Optimization, Genetic Algorithm, and Parallel Technique

2.1 Problem Statement

Suppose there exists a general parametric mathematical model Ω with J inputs, K observable outputs and N variable parameters. The output \vec{y} of Ω is:

$$\vec{y} = \Omega(\vec{x}, \vec{p}) \quad \vec{x} \in \mathbb{R}^J, \vec{y} \in \mathbb{R}^K, \vec{p} \in \mathbb{R}^N \quad (1)$$

Then, the goal of optimization is to find suitable value \vec{p}^* so that the objective function $\mathcal{F} : \mathbb{R}^K \rightarrow \mathbb{R}$ achieves its optimum.

2.2 Nomenclature

\mathbf{N} population size, number of chromosomes in each generation

P_e elite probability

P_c crossover probability

P_m mutation probability

L_t number of generations tolerated for no improvement on the objective before the GA is terminated.

2.3 The Standard Genetic Algorithm

The Genetic Algorithm is a type of intelligent random search technique that mimics the process of Darwinian evolution. The algorithm begins by randomly generating a set of chromosomes and each of which is evaluated for its fitness for solving the given minimization or maximization task. After the fitness evaluation, a new population, or generation, arises from the previous generation through competition based on the individual's fitness evaluation. The new population, retaining the same size as that of the previous population, is expected to contain individuals with improved fitness values, because the use of probabilistic transition rules by the GA makes the fittest individuals have the highest probability of being selected to enter the next generation. Unlike the local gradient based methods, the Genetic Algorithm requires no calculation of the gradient and is not susceptible to the trapping of local minima. As the generation evolves constantly, it will approach the optimal or near-optimal solution gradually.

When implementing the GA, each parameter p_i is represented by a real number called a gene. Genes are cascaded to form a longer string \vec{p} called a chromosome. Chromosomes are used to represent the possible combinations of unknown parameter values. A collection of N chromosomes is called a population. In each generation, let $e_{i,j}$ be the objective of the j th chromosome at the i th generation, the GA would search for the optimal objective $e_{opt}^i = \text{Optimal}(e_{i,j})$ over the entire space of parameters and attempts to drive e_{opt}^i to optimum over the succeeding generations.

Using the Genetic Algorithm for highly nonlinear optimization often requires

considerable computational time, especially when a large number of unknown parameters are involved. The computational time required by the Genetic Algorithm optimization can be roughly calculated by

$$Time = G \times N \times T \quad (2)$$

where G is the number of generations for achieving the convergence, N is the population size, and T is the time that a single run of the application model will take. For example, in the case of modeling water flow in deep vadose zone presented in section 4.1, the application of a standard GA will have a G of 600, N of 800 and T of 10 minutes, which amounts to a computational time of 9 years in total for achieving the optimal solutions. Such a computational cost is prohibitive for most users. Fortunately, the operations on individual chromosome are independent from each other within each generation, which makes the GA is inherently suitable for parallelization. In current study, we employ the parallel technique to distribute multiple chromosomes to parallel processors to speed up the computation. At the meant time, we also adopt a modified GA to reduce the population size N and the number of generations G , in order to increase the computational efficiency further.

2.4 The Modified Genetic Algorithm used in PGO

Before paralleling the GA optimization process, we provide two modifications on the standard GA to help speed up the convergence. First, we adopt an elite GA which will preserve the best chromosomes even thought their probability being selected is small. Second, we provide an interface in PGO to let users define their own decoding module, which will help to narrow down the search

space thus improve the convergent speed.

2.4.1 The Elite Genetic Algorithm

In the conventional GA, chromosomes are selected for mating based on the ratio of their fitness value to the sum of total fitness value of all chromosomes in one generation. For example, the probability of the j th chromosome being

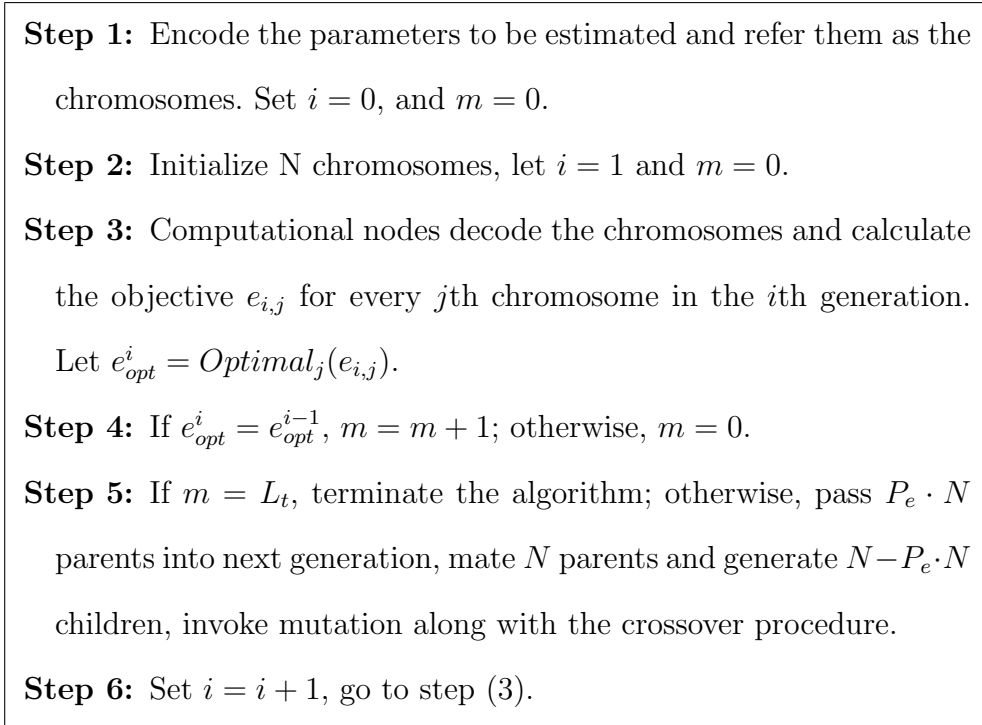


Fig. 1. The GA scheme used in PGO

selected for mating in the next generation is

$$P_j = \frac{f_j}{\sum_{i=1}^N (f_i)} \quad (3)$$

In the event of optimizing a highly nonlinear problem, N will be huge for covering all the search space, and P_j in (3) will be small even for chromosomes that are close to the ideal values. Using this fitness criterion, chromosomes representing parameter sets that are close to optimal values might be discarded,

resulting in slow convergence and poor performance.

In order to avoid slow convergence and not to miss global optimum, in PGO the standard GA is modified to ensure that these chromosomes are saved and passed on to the next generation. In each generation, the best $P_e \cdot N$ elite chromosomes are allowed to survive into the next generation. Because the elite parents are allowed to live onto the next generation, the objective in the next generation will be always better than or equal to the objective in the present generation. This is termed the elite Genetic Algorithm, and its implementation steps are illustrated in Fig. 1. Generally speaking, elite selection improves algorithm performance (De Jong, 1975; Goldberg, 1989).

2.4.2 Search Space Mapping Module

Theoretically speaking, any real numbers in the allowed range could be eligible for being chosen as the parameter values. In a specific application, however, practitioners can specify a much narrower range for each parameter to be estimated. PGO provides an interface to let users define their own decoding module (Fig. 2 and section 3.1). The decoding module is a function to map one space to another. For example, if the parameter values are in \mathbb{R}^N space, but the most reasonable combination of these parameter values can be classed to some categories, the decoding module will be used to map category indicators to actual parameter values, and the search space will then decrease from \mathbb{R}^N to \mathbb{N} . In the Genetic Algorithm search, a smaller search space will lead to a smaller population size and quicker convergence.

2.5 *Coupling with Parallel Techniques*

As described at the beginning of section 2.4, using the Genetic Algorithm for optimization is a time-consuming work. On the other hand, the inherent design of the Genetic Algorithm is ideally suitable for parallelization. Chromosomes in the same generation are independent of each other and there is no interaction between genetic operations performed on each chromosome. Therefore, much of the computational time will be saved if we carry these operations on each chromosome in parallel rather than sequentially. In PGO, the parallel computing framework is organized as a master-slave system and uses a central database management system (DBMS) for storing all the data during optimization progress. The server distributes chromosomes to computational nodes which are responsible for decoding the chromosome into real values, running the external application model, calculating error and returning error to the server. As communication overhead can be neglected among independent model runs, a near-linear speedup with the numbers of CPUs used can be achieved. Although the MPI (Message Passing Interface) (Karonis et al., 2003) has been a popular choice in many parallel computing applications, we choose the central DBMS approach in PGO based on following considerations. First, many modelers may not have access to Unix-like servers, and have to run their optimization jobs on windows based PCs. Using MPI on windows may complicate the installation, configuration, and the use. Second, the central DBMS approach is functional enough in handling data exchange between master and slave machines. The well-organized data in DBMS are valuable for further development. For example, the PGO with DBMS data structure is very suitable for integrating with a wider Grid environment to enable the service-oriented

Grid computing (Foster and Kesselman, 1999). The Grid-enabled version of PGO is in fact currently under development in our laboratory. In the following sections, we will explain in details the architecture and configuration of this parallel computing platform.

3 The Software

3.1 Architecture

PGO consists of two parts: server module and computational module as illustrated in Fig. 2. The server module is mainly the optimization kernel which controls the evolution of whole optimization progress. The computational module runs on many computational nodes, each of which gets a chromosome from the server and decodes the abstract chromosome into real parameter values, and store them into template files so as to generate input files for external application model. When all the input files are ready, the computational module will run the external application model, extract the results from the output files produced by the model, and return the errors to the server for further evaluation.

3.2 Integration

Since PGO has provided most components for a general optimization task, integrating PGO with a specific geoscientific modeling is as easy as installing and configuring a regular software. The common steps are:

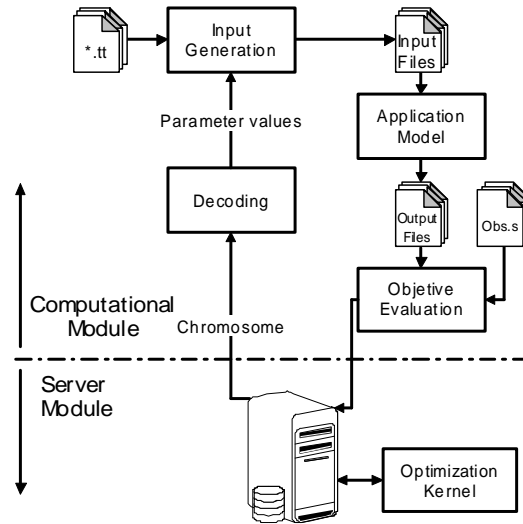


Fig. 2. Architecture of PGO

Step 1: Install prerequisite softwares, including Perl and the MySQL DBMS.

PGO comes a script that will help making the installation a very smooth process.

Step 2: Configurate PGO, edit the configuration file to suit your particular case. In section 3.3, we will explain the configuration process in detail.

Step 3: Copy the computational module to every computational nodes. Start the server module on the server, and start the computational module on each computational node.

3.3 Configuration

To facilitate the easy use of PGO, we provide a flexible XML file named `config.xml` to store all the configurations. `Config.xml` contains settings for the Genetic Algorithm employed (e.g. population size, crossover probability, and mutation probability etc.), the lower bound, upper bound and the precision of model parameters, rules for making model input files, extracting observations from model output files and evaluating the objective function. The graphical

view of the XML schema used in PGO is illustrated in Figure 3.

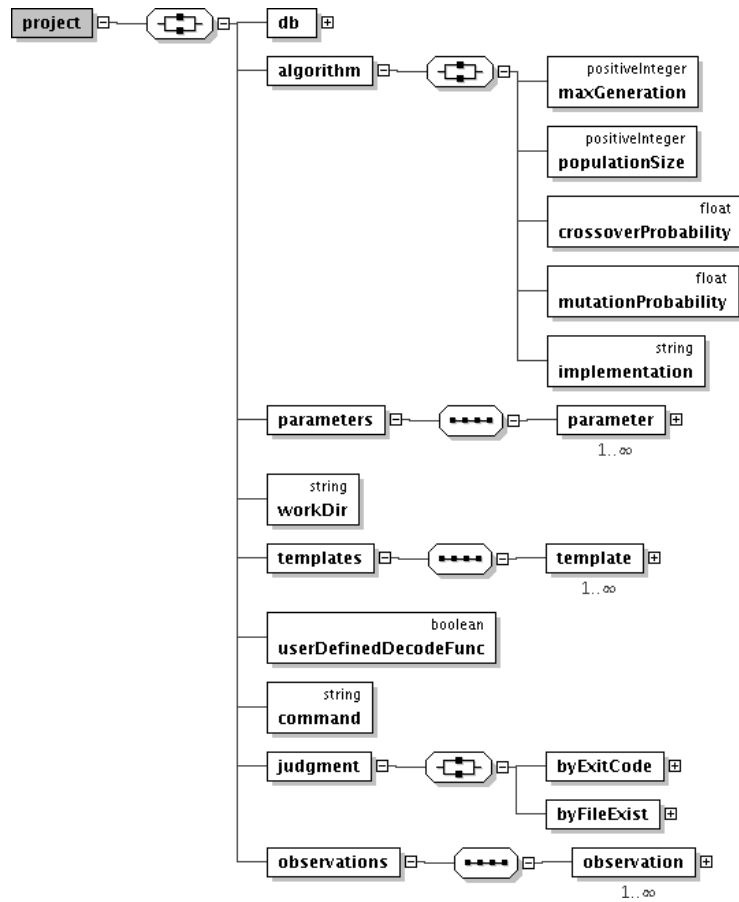


Fig. 3. The graphical view of the XML schema used for configuration

4 Applications

4.1 A large scale parameter estimation for modeling water flow in a heterogeneous deep vadose zone

The first application involves a simulation based on the SWAP model (Kroes and van Dam, 2003) of the dynamic interactions between soil, water, atmosphere and plant in a deep vadose zone of a typical area in North China Plain, where years of intensive ground water pumping for irrigation have caused the

rapid decline of ground water table. The one-dimensional SWAP model employs the Richard's equation, which combines Darcy's law and the continuity equation, to simulate the water flow process in the deep vadose zone. Assuming strong heterogeneity exists in the vadose zone, the soil column under study will be divided into 25 layers with each layer having 7 unknown soil hydraulic property parameters as defined in the Van Genuchten relationship and Mualem's $K(\theta)$ function (Kroes and van Dam, 2003). We will then need to identify the 175 unknown parameters in order to perform the simulation. As we have explained earlier, the search for the best values for such a large number of parameter will be prohibitively expensive. Here we will apply PGO for this task for demonstrating the efficiency and utility of PGO.

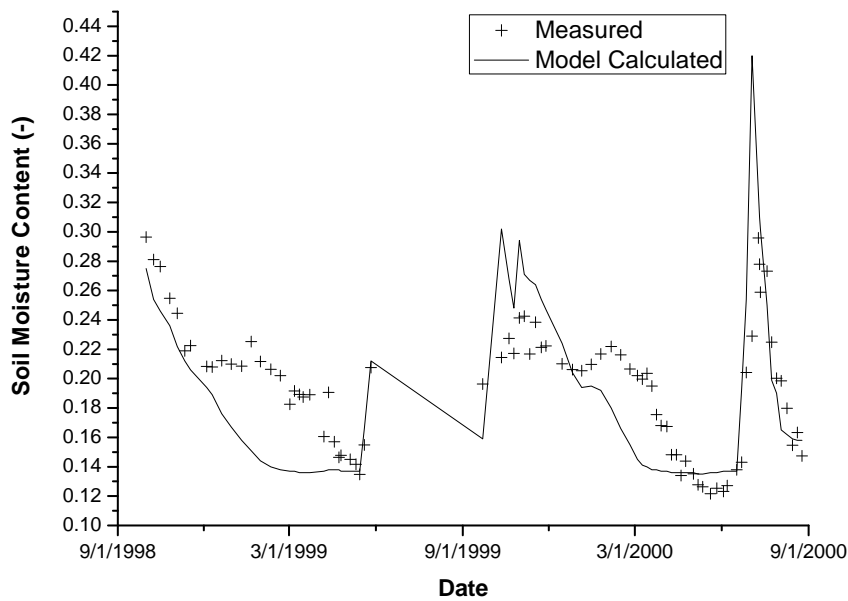
Field data used for model calibration were collected at Luancheng Agro-Ecological Research Station (Chinese Academy of Sciences), Luancheng County, Hebei Province, from October 1998 through September 2001. The data include meteorological records, crop properties, and soil classifications that serve as the inputs to the model, and also include the soil moisture change over 3 years and the ground water table fluctuations that are the targets of model calibration. The soil moisture contents are measured by neutron probes approximately every five days at nine to ten depth intervals between 0 and 180 cm. The ground water table elevation is also measured every five days. Detailed descriptions of experimental set up and procedures can be found in Liu et al. (2002).

The main input of SWAP is *.swp file, which contains general section, crop section, soil water section etc. We assume that in this simulation all other parameters are fixed and known except the soil hydraulic properties contained in soil water section. When conducting the parameter estimation by PGO,

each computational node gets a chromosome from the server, decodes the chromosome to 175 soil hydraulic parameters and writes these values in *.swp file. PGO has an interface available for user to specify the range of each parameter if desired. Then, it will run the SWAP model to produce a series of output files including *.wba file for groundwater level data and *.vap file for soil moisture content distribution. The computational node will then extract values at the observation points from *.wba file and *.vap file, compare simulation results to corresponding measured values and return the error to server. When the server has distributed all the chromosomes in current generation to all available computational nodes to accomplish all associated computations, the optimization kernel of PGO will breed next generation based on the scheme presented in section 1. Iterating previous steps, PGO will drive error to minimum and obtain the optimal or near optimal parameter values. In current application, we carry out the parameter estimation task in 82 heterogeneous servers with total 208 computational nodes. The entire parameter optimization process lasts half a month. Fig. 4 gives a sample of the comparisons between the measured and model calculated soil moisture changes at the interval of 15cm below ground surface. Fig. 5 illustrates the convergence of RMSEs (Root Mean Square Errors) of soil moisture content and groundwater level during the parameter estimation process. Over all, the RMSE between measured and model calculated soil moisture content of top 180cm was 4.9cm, or 12.5% of the averaged total water content. The RMSE between measured and model calculated groundwater level was 1.44m, or 6.0% of the averaged groundwater level.

In actual simulation of water flow in vadose zone, it is normally possible to group the soil profile layers into a few soil types thus dramatically reduce the

number of soil hydraulic parameters needed to be estimated. Here for the purpose of demonstrating the capability of PGO, we used a large number of soil layers and assumed each layer is associated with a different set of unknown parameters. This created an exceedingly large number of unknown parameters in the search space. Without the employment of a distributed computing resources and the framework we developed, this parameter estimation job as it is formulated may take the conventional GA up to 10 years to accomplish. The application of a local gradient based search method, PEST (Doherty, 2004), has also been tried and failed to produce desirable results.



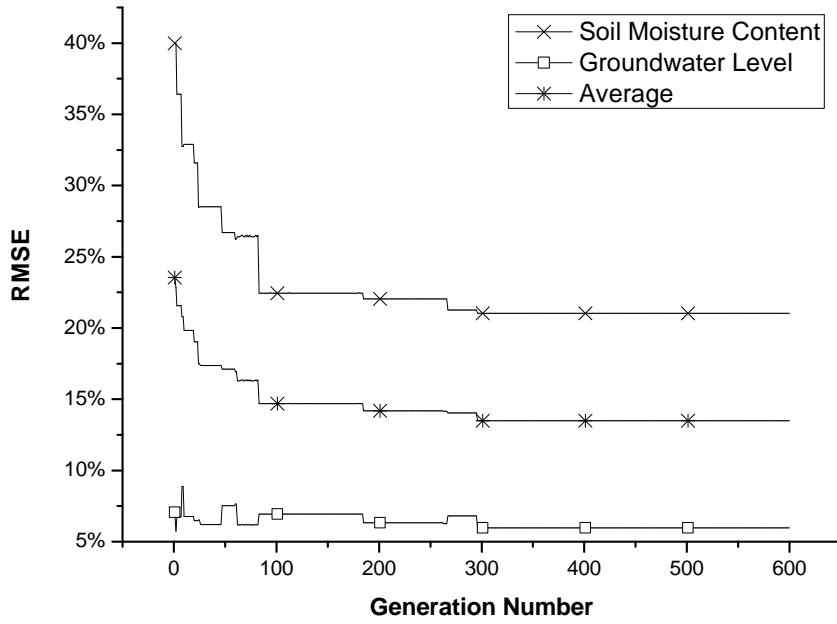


Fig. 5. Convergence of relative RMSEs during optimization process

4.2 Optimization of remediation system design

In our second application, we use PGO to implement a serial simulation-optimization code called 'Modular Groundwater Optimizer (MGO)' by Zheng and Wang (2003) to parallel environment. The MGO code couples the widely used groundwater flow simulator MODFLOW (McDonald and Harbaugh, 1988) and solute transport simulator MT3DMS (Zheng and Wang, 1999) with a general optimization package for formulating the most cost-effective groundwater remediation strategies under various physical, environmental and budgetary constraints. As PGO has been designed to be an easy-to-use general framework for parallel optimization modeling, integrating PGO with MGO is simple and straightforward. Basically, the MGO code is modified to play the role of the computational module residing and running on individual nodes. Each run of MGO returns to the server a single objective function value associated with

each chromosome. The optimization algorithms in the original MGO code are no longer used; instead the parallel GA kernel residing and running on the server carries out the optimization tasks.

We then apply the parallelized MGO code to the optimization of a groundwater remediation system at the CS-10 site located within the Massachusetts Military Reservation (MMR) in Cape Cod, Massachusetts (Zheng and Wang, 2002). TCE is the primary contaminant at the CS-10 site. Extensive field sampling has led to the delineation of a TCE plume approximately 5 km long, 2 km wide, and up to approximately 43 m thick. The groundwater remediation system for the CS-10 site involved nine pumping wells intended to contain and eventually remove the TCE plume. The nine pumping wells must not exceed a total extraction rate approximately $10 \text{ m}^3/\text{min}$, the design capacity for the on-site treatment plant. After the extracted water was treated at the on-site treatment plant, it would be re-injected into the infiltration trenches on the downstream edges of the plume. More detailed information on the case study can be found in Zheng and Wang (2002).

Using the MMR example as a demonstration for the new parallel MGO for comparison with the original serial MGO, we optimize the pumping rates of the nine wells to achieve the maximum amount of contaminant mass by the end of the project horizon while satisfying all the constraints discussed in Zheng and Wang (2002). In this application, the number of generation (G) is 50, the population size (N) is 200 and the run time (T) is about 25 minutes for running both flow and transport simulations on a single CPU. We carry out the optimization in 36 heterogeneous computational nodes, with 72 CPUs in total. The optimization takes 52 hours to converge. Figure 6 illustrates the evolution of objective function (contaminant mass removed) with generations. Without

integrating with PGO, the non-parallelized optimization achieved a similar optimal objective function value with the identical GA solution parameters, but it would take about half a year to accomplish it.

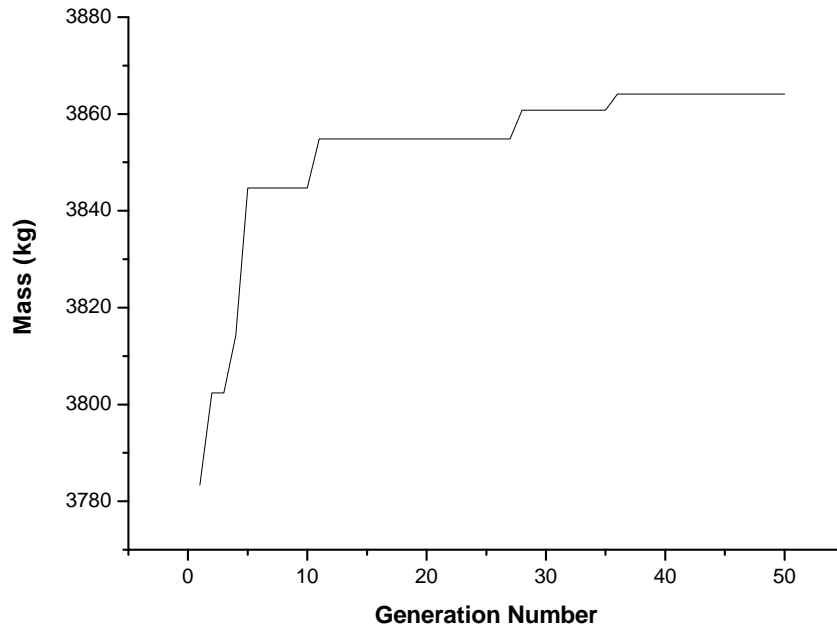


Fig. 6. Relationship between mass removal and generation number

5 Conclusions and Future Work

This paper has presented a parallel computing framework for general GA-based optimization problems. Coupling with the Genetic Algorithm and parallel technology, PGO is very suitable for highly nonlinear model optimization. We demonstrate the effectiveness and versatility of PGO in previous two applications.

Compared with the traditional optimization software based on local gradient-based search technology, PGO does better in searching and finding the optimal

or near-optimal parameters for highly-nonlinear model and the search results are insensitive to the choice of initial parameter values. More importantly, PGO provides an easy-to-use parallel computing framework for integrating with any existing software and modeling codes. Further, with the use of DBMS data structure, PGO is very suitable for integrating with a wider Grid environment to enable the the service-oriented Grid computing. The Grid-enabled version of PGO is currently under development.

Acknowledgements

This research was jointly funded by the Innovation Knowledge Project of Chinese Academy of Sciences (project No. KZCX3-SW-428), L. Zheng's Young Talents Award from the Chinese Academy of Sciences, and the ChinaGrid Project (project No. CG2003-GA002 and CG2003-GA005). Many thanks to Joop Kroes (Alterra, Wageningen University and Research Centre) for supporting the source code of TTUTIL library (van Kraalingen and Rappoldt, 2000).

References

- Bayer, P., Finkel, M., 2004. Evolutionary algorithms for the optimization of advective control of contaminated aquifer zones. *Water Resources Research* 40 (6), w06506, doi:10.1029/2003WR002675.
- De Jong, K. A., 1975. An analysis of the behavior of a class of genetic adaptive systems. Ph.D. thesis, University of Michigan.

- De Jong, K. A., 1985. Genetic algorithms: A 10 year perspective. In: Grefenstette, J. J. (Ed.), ICGA. Lawrence Erlbaum Associates, pp. 169–177.
- Doherty, J., 2004. PEST: Model-Independent Parameter Estimation. Watermark Numerical Computing, 5th Edition.
- Erickson, M., Mayer, A., Horn, J., 2002. Multi-objective optimal design of groundwater remediation systems: application of the niched Pareto genetic algorithm (NPGA). *Advances in Water Resources* 25 (1), 51–65.
- Foster, I., Kesselman, C. (Eds.), 1999. The grid: blueprint for a new computing infrastructure. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Goldberg, D. E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, Mass.
- Holland, J., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI.
- Karonis, N. T., Toonen, B., Foster, I., 2003. MPICH-G2: A grid-enabled implementation of the message passing interface. *Journal of Parallel and Distributed Computing* 63 (5), 551–563.
- Karpouzou, D. K., Delay, F., Katsifarakis, K. L., de Marsily, G., 2001. A multipopulation genetic algorithm to solve the inverse problem in hydrogeology. *Water Resources Research* 37 (9), 2291–2302.
- Kroes, J., van Dam, J., 2003. Reference Manual SWAP version 3.0.3. Alterra, Green World Research, Wageningen, The Netherlands.
- Liu, C., Zhang, X., Zhang, Y., 2002. Determination of daily evaporation and evapotranspiration of winter wheat and maize by large-scale weighing lysimeter and micro-lysimeter. *Agricultural and Forest Meteorology* 111 (2), 109–120.
- McDonald, M. G., Harbaugh, W. W., 1988. A modular three-dimensional finite-difference ground water flow model. *Techniques of Water-Resources*

- Investigations, Book 6. US Geological Survey, Reston, VA.
- Michalewicz, Z., 1996. Genetic Algorithms + Data Structures = Evolution Programs, 3rd Edition. Springer Verlag.
- Poeter, E. P., Hill, M. C., 1998. Documentation of UCODE: A computer code for universal inverse modeling. Water-Resources Investigations Reports 98-4080, U.S. Geological Survey.
- Prasad, K. L., Rastogi, A. K., 2001. Estimating net aquifer recharge and zonal hydraulic conductivity values for Mahi Right Bank Canal project area, India by genetic algorithm. *Journal of Hydrology* 243, 149–161.
- Ritzel, B. J., Eheart, J. W., Ranjithan, S., 1994. Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. *Water Resources Research* 30 (5), 1589–1603.
- van Kraalingen, D., Rappoldt, C., 2000. Reference manual of the FORTRAN utility library TTUTIL v. 4. Plant Research International, Wageningen, The Netherlands.
- Wang, Q. J., 1991. The genetic algorithm and its application to calibrating conceptual rainfall-runoff models. *Water Resources Research* 27 (9), 2467–2471.
- Zheng, C., Wang, P. P., 1999. MT3DMS: Documentation and user's guide. Contract report SERDP-99-1, U.S. Army Eng, R&D Center, Vicksburg, MS.
- Zheng, C., Wang, P. P., 2002. A field demonstration of the simulation-optimization approach for remediation system design. *Ground Water* 40 (3), 258–265.
- Zheng, C., Wang, P. P., May 2003. MGO-A Modular Groundwater Optimizer Incorporating MODFLOW/MT3DMS. University of Alabama and Groundwater Systems Research Ltd.