Parallel Simulated Annealing Using CILK Language: Applications in Estimating Transport Parameters for Groundwater Contaminants

L. Li[1], D. A. Barry[2] and J. Morris[3]

[1]School of Engineering and Technology
Deakin University, Geelong, VIC 3217

[2]Department of Environmental Engineering,
University of Western Australia, Nedlands, WA 6907

[3]Department of Electrical and Electronic Engineering,
University of Western Australia, Nedlands, WA 6907

Abstract This paper presents an application of parallel simulated annealing (PSA) using CILK language for parameter estimation in groundwater contaminant transport. An asynchronous PSA algorithm with multiple Markov chains is applied and programmed in CILK. The algorithm was used to determine the transport parameters of groundwater contaminant by curve-fitting breakthrough curves. The tests were conducted on a PC network with three processors running Linux. The results show that the speedup by three processors is close to 3 while two processors achieve a speedup of only 1.29. The results also show that communication load did not increase with the number of processors. On the contrary, the communication load in the case of two processors is greater than that with three processors. The cause of such a behaviour is yet to be determined. Although further tests with a larger number of processors are needed in order to ascertain the performance of the algorithm, the results so far indicate that the PSA is very efficient.

1. INTRODUCTION

Optimization problems are often encountered in applied groundwater hydrology. Inverse problems, such as the determination of transport parameters (e.g., flow velocities and dispersion coefficients) through curve-fitting experimental data to a predictive model, essentially involve optimization. For example, the two-region model (TRM) is often used to determine the parameters for contaminant transport in natural soils (Parker and van Genuchten, 1984). In non-dimensional form, the governing equations are:

$$\beta R\frac{\partial c}{\partial t}+(1-\beta)R\frac{\partial c_s}{\partial t}=\frac{1}{P}\frac{\partial^2 c}{\partial x^2}-\frac{\partial c}{\partial x} \quad (1a)$$

$$(1-\beta)R\frac{\partial c_s}{\partial t}=\omega(c-c_s) \quad (1b)$$

where c and c_s are the normalised concentration of contaminant in liquid and solid phase, respectively; t is the non-dimensional time (t^*V/L where t^* is time, V is the flow velocity and L is the travel distance of contaminant); β is the ratio of the mobile region to the total pore volume; R is the retardation factor; P is the column Péclet number (VL/D where D is the dispersion coefficient); and ω is the non-dimensional reaction rate, $\alpha(1-\beta)RL/V$ (α is the reaction rate coefficient). Variable definitions are collected in the Notation list.

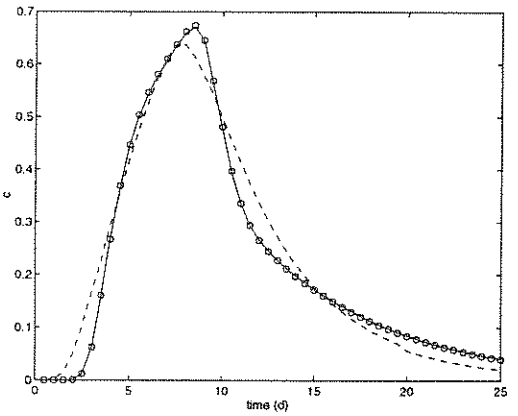

Figure1: Fitted BTCs by CXTFIT. Circles are test data generated by CXTFIT with preset parameters. Solid and dashed curves are the fitted BTCs with different initial parameter choices.

Most previously developed techniques for solving optimization problems in parameter estimation are gradient-type approaches which do not guarantee a global optimal solution (Murtagh and Saunders,

1980). If local optimal solutions are accepted in practice, non-unique solutions can result from different initial estimates of the transport parameters. Parker and van Genuchten (1984) used non-linear least squares fitting in their program, CXTFIT. Depending on the initial parameter choices, the results can be non-unique as they are only locally optimal. Figure 1 shows two different fitted breakthrough curves (BTC) which result from different initial parameter sets.

Simulated annealing (SA) is a technique for finding the global solution to optimization problems (van Laarhoven and Arts, 1987). The concepts underlying simulated annealing come from statistical mechanics and the algorithm is analogous to the process of slowly cooling a heated solid until it reaches its low energy ground state (van Laarhoven and Arts, 1987). A typical structure of a SA algorithm (SAA) is as follows (Lee and Lee, 1996):

```
s ← s0;
Tem ← Tem0;
while the outer loop break condition is not met do {
        while the inner loop break condition is not met do {
            perturb current configuration (i.e., parameter set)
                        s to s*;
            evaluate cost function and find,
                        ΔC = C(s*)-C(s);
            accept s* with the probability min(1, e^-ΔC/Tem);
            update sbest;
        }
        lower(Tem);
        s ← sbest;
}
```

We applied this algorithm to the above curve-fitting process and found that the solution converges to the global optimum in all the tests as shown in Figure 2 where the averaged relative error of the estimated parameters is plotted for each test. Four parameters, V, D, β, and α were fitted, while R and T (the pulse time of contaminant release) were fixed (equal to 1 and 6 days, respectively). Also plotted in Figure 2 are the results obtained using CXTFIT.

This technique, however, requires very long computation time (Nabhan and Zomaya, 1995). It was found that the SAA requires at least 10 times more computation time than CXTFIT.

Here, we present an application of CILK to simulated annealing in order to achieve optimization in a parallel computing environment. CILK is a C-based runtime system for multiple-threaded parallel programming developed by the Laboratory for Computer Sciences, Massachusetts Institute of Technology (CILK-2.0, 1995). Tests are conducted on the above example problem, i.e., using the TRM to determine the transport parameters by curve-fitting a breakthrough curve. The approach, however, can be easily applied to other, more complicated problems.

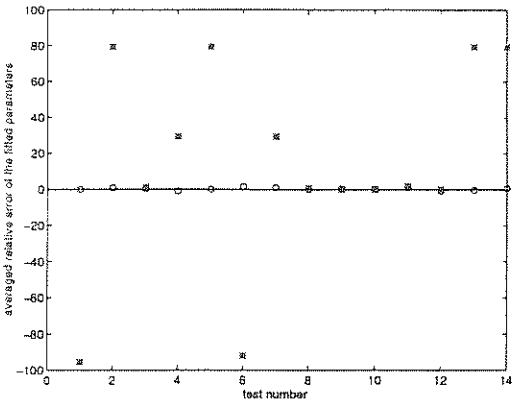

Figure 2: Averaged relative error of estimated parameters. Circles are results from SAA while stars from CXTFIT. The solid line indicates the global optimum.

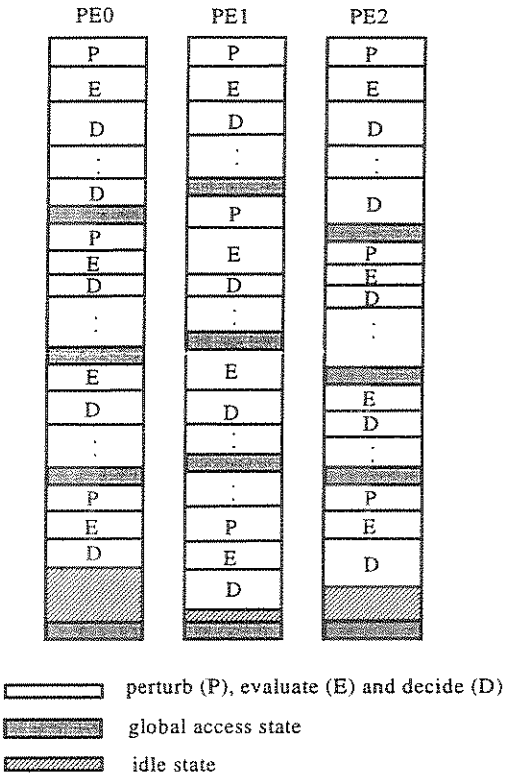

Figure 3: Schematic diagram of a AMMC PSA based on three processors. PE0, PE1 and PE2 are the three processors.

2. ASYNCHRONOUS PSA ALGORITHM WITH MULTIPLE MARKOV CHAINS

Parallel simulated annealing algorithms can be classified into two types (Lee and Lee, 1996): SMC PSA (single Markov chain parallel simulated annealing) and MMC PSA (multiple Markov chain parallel simulated annealing). SMC PSA allows only a single search path (Markov chain) for all the processors. While SMC PSA preserves the sequential nature of SA, following a single search

path might be an unnecessary restriction, possibly causeing large communication loads (Banerjee et al., 1990). On the other hand, MMC PSA allows each processor to follow its own search path (Arts and Korst, 1989). These processors may communicate with each other with a fixed period (synchronously) or dynamically (asynchronously).

In the case of asynchronous MMC PSA (AMMC PSA), each processor broadcasts its best state to other processors and updates its state using global best state from other processors independently (Lee and Lee, 1996). A processor need not wait for others during its search and thus no idle state exists in the processors except at the end when the solutions are collected from the processors to determine the final solution (Figure 3).

3. IMPLEMENTATION OF AMMC PSA USING CILK

The CILK language is an extension to C that provides an abstraction of threads in explicit continuation-passing style. A CILK program is preprocessed to C and then linked with a runtime library to run on a computer network (CILK-2.0, 1995).
An outline of the AMMC PSA program in CILK is shown below (not all threads are displayed):

```
cilk int main{
        variable declaration;
        for i=1 to N_processor {
                spawn thread_setBC @(i);
                /* broadcasting initial global data to
all processors */
        }
        sync; /* synchronous point */
        for i=1 to N_processor {
                spawn anneal(...) @(i);
                /* starting the MMC PSA on all
processors */
        }
        sync;
        collect_solutions;
/* collect solution from all processors */
printout_result;
}

thread anneal(...) {
        ...
        for i=1 to size_of_the_MC {
                perturb parameter_set; /* generate
random moves in parameter space */
evaluate cost and calculate ΔC;
accept new_move with the probability
min(1, e^-ΔC/Tem);
if update local_best, broadcast it to
other processors;
        }
        spawn anneal(...);
}
```

Each new move of the contents of parameter set is generated randomly. Two pseudo-random numbers are generated for each parameter, one for the magnitude of the move and the other for the direction. If the new move produces a local best, the result will be broadcast to all other processors. However, these processors only process the received information at the end of the current thread. This involves comparing the received result with each processor's local best (LB) with the possibility of updating its LB.

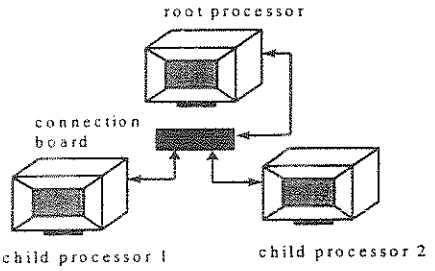

Figure 4: Schematic diagram of the PC network at the Department of Electrical and Electronic Engineering, University of Western Australia.

We conducted tests with the TRM curve-fitting problem on a PC network, which possesses three processors running Linux (Figure 4).

4. RESULTS AND DISCUSSION

To examine the speedup of the algorithm, tests were run on one, two and three processors. The program was terminated when the averaged relative error of the parameters was less that 1%. The real computation time for each test was recorded. During all the tests, the program was run with 99% CPU resources available, i.e., no other jobs were present.

The test data, i.e., the BTC, was obtained using a CXTFIT simulation with the following preset parameters: V = 0.5 m/d, D = 0.05 m^2/d, R = 1, T = 6 d, β = 0.5 and ω = 0.9. Fitted parameters are V, D, T, β and ω. The initial parameter estimations for the tests are: V = 0.1 m/d, D = 0.1 m^2/d, T = 4 d, β = 0.1 and ω = 0.1. Since the search path is determined by pseudo-random numbers, it is necessary to run the tests a number of times with the same preset initial values. Thus, ten tests were conducted for each case with a particular number of processors (30 tests in total). Each test generated a unique sequence of random numbers since the seed for random number generation changed with the time when the program started.

The results are shown in Figure 5 where the real computation time is plotted against the number of processors. While the data are scattered for each set of tests with a particular number of processors, the trend of computation time decreasing with the number of processors is evident. The solid line shows the averaged computation time (CT).

We use the averaged CT to estimate the speedup (sp), i.e.,

$$sp(n) = \frac{CT(n)}{CT(1)} \qquad (2)$$

where n is the number of processors. The result show that the speedup with three processors is close to 3, the ideal value (Lee and Lee, 1996). However, the speedup in the case of two processors is only 1.29 (Figure 6). This suggests that the communication load may be higher in the case of two processors than that with three processors.

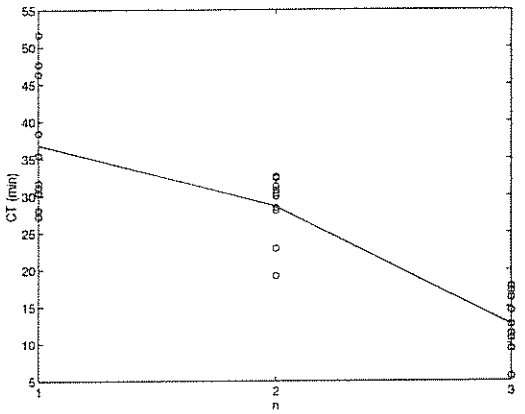

Figure 5: Real computation time against the number of processors. Circles are results from the tests and solid line indicates the averaged performance.

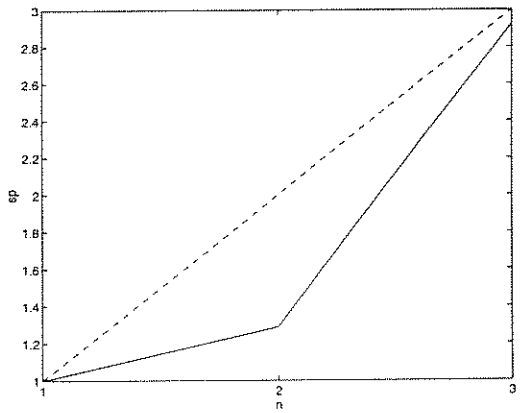

Figure 6: Speedup against the number of processors. The dashed line indicates the ideal speedup (i.e., $sp(n) = n$).

To examine the communication load, we recorded the time at each evaluation step during the test. Figure 7 shows the typical results from tests with one, two and three processors, respectively. The curves are almost linear, indicating that the CPU was solely used by the testing program. The slope of the curve increases as a result of multiple processors. Such an increase is due to the communication load introduced by the PSA. The results also indicate that the communication load in the case of two processors is larger than that with three processors. Although this is consistent with the results of speedup shown previously, it is a rather unusual phenomenon, the cause of which is yet to be determined. In particular, further tests should be conducted with a larger number of processors to examine the relationship between the communication load and number of processors.

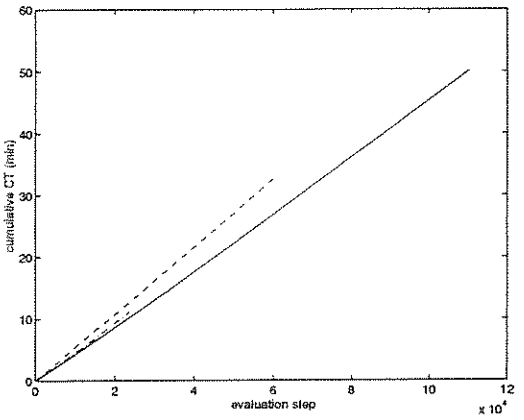

Figure 7: Cumulative real computation time against evaluation step. Solid line is the case with one processor, dashed line with two and dot-dashed line with three processors.

The slope of the evaluation time curve is calculated for each test and is plotted in Figure 8. Again, although the data are scattered, the general trend is evident, i.e., slope(n = 2) > slope (n = 3) > slope(n = 1).

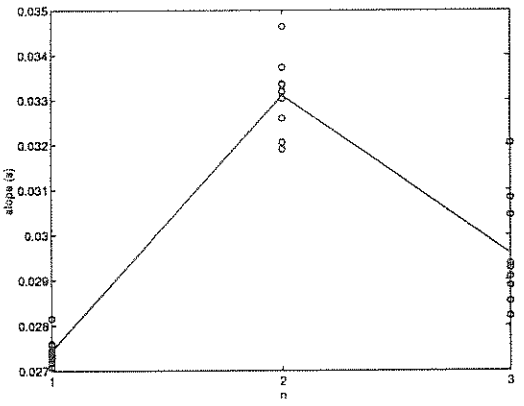

Figure 8: Slope of the evaluation time curve against the number of processors. Solid line indicates the averaged values.

5. CONCLUSIONS

Simulated annealing has been demonstrated to be an effective optimisation method in estimating parameters through curve-fitting the model prediction to the data. Although the technique is

computationally intensive, parallel annealing can be explored.

In this paper, we have shown that asynchronous MMC PSA can be implemented using the CILK language and be used to reduce the computation time without sacrificing the accuracy of the solution. The results show that the speedup is high, especially in the case of three processors. Due to the unexpected high communication load, the speedup is relatively low in the case of two processors. Further tests are necessary to ascertain both the speedup and communication load of the algorithm.

NOTATION LIST

c	normalised concentration in liquid phase	
c_s	normalised concentration in solid phase	
CT	real computation time	[T]
D	dispersion coefficient	$[L^2T^{-1}]$
L	travel distance	[L]
n	number of processors	
R	retardation factor	
s	parameter set	
sp	speedup	
t	non-dimensional time	
t^*	time	[T]
T	pulse time of contaminant release	[T]
Tem	temperature	[k]
V	flow velocity	$[LT^{-1}]$
α	reaction rate	$[T^{-1}]$
β	mobile region ratio	
ω	non-dimensional reaction rate	

REFERENCES

Aarts, E., and J. Korst, *Simulated annealing and Boltzman machines*, John Wiley & Sons, New York, 1989.

Banerjee, P., M. H. Jones, and J. S. Sargent, Parallel simulated annealing for cell placement on hypercube multiprocessors, *IEEE Trans. On Parallel and Distributed Sys.*, 1, 91-106, 1990.

Cilk-2.0 Reference Manual, the Massachusetts Inst. of Tech., Cambridge, USA, 1995.

Lee, S.-Y., and K. G. Lee, Synchronous and asynchronous parallel simulated annealing with multiple Markov chains, *IEEE Trans. On Parallel and Distributed Sys.*, 7, 993-1007, 1996.

Murtagh, B. A., and M. A. Saunders, MINOS/AUGMENTED user's manual, *Tech. Rep. 80-4*, Sys. Optimisation Lab., Dept. Of Oper. Res., Stanford Univ., Stanford, California, 1980.

Nabhan, T. M., and A. Y. Zomaya, A parallel simulated annealing with low communication overhead, *IEEE Trans. On Parallel and Distributed Sys.*, 6, 12261233, 1995.

Parker, J. C., and M. Th. Van Genuchten, Determining transport parameters from laboratory and field tracer experiments, *Vir. Agric. Ex. Stat. Bull. 84-3*, Vir. Poly. Inst. And State Univ., Blacksburg, 1984.

Van Laarhoven, P. J. M., and E. H. L. Aarts, *Simulated annealing: Theory and applications*, D. Reidel Publ. Com., Boston, 1987.