IDENTIFYING INITIAL CONDITION IN HEAT CONDUCTION TRANSFER BY A GENETIC ALGORITHM: A PARALLEL APPROACH

Leonardo D. Chiwiacowsky
Haroldo F. de Campos Velho
Airam J. Preto
Stephan Stephany
Laboratório Associado de Computação e Matemática Aplicada
Instituto Nacional de Pesquisas Espaciais
Caixa Postal 515, CEP 12245-970 São José dos Campos, SP - Brasil
12201-970, São José dos Campos - SP - Brazil
[leodc, haroldo, airam, stephan]@lac.inpe.br

Abstract. *A parallel implementation of a genetic algorithm is proposed to solve the one-dimensional inverse heat conduction problem of estimating the initial temperature distribution from the transient temperature noisy profile at a given time, considering insulated boundary conditions. This inverse problem is formulated as an optimization problem and the objective function is given by the square difference between experimental and simulated temperatures added to a regularization term. Genetic algorithms are efficient search methods based on natural and genetic selection of population being inherently parallel. In the proposed genetic algorithm, the genotype of each individual is composed by a set of real numbers and the code was parallelized by means of calls to the MPI library. A sub-population (demes) is assigned to each processor and individuals are free to migrate to any other processor, according to the island model migration policy. The inversions were accomplished for two population sizes (336 and 1008), being the individuals equally distributed among the processors. A distributed memory parallel machine was used and tests were performed with the number of processors ranging from 2 to 16. Processing times show efficiencies between 0.9 and 1.14 for the 1008-individual population and between 0.83 and 0.89 for the 336-individual population.*

Keywords: *Parallel Genetic Algorithms, Inverse Problems, Heat Conduction.*

1. INTRODUCTION

Genetic Algorithms (GA) are efficient search methods based on principles of natural selection and population genetics. They are being successfully applied to problems in business, engineering and science (Cantú-Paz, 1995). However there exist some problems in their utilization which can all be addressed with some form of Parallel GA (PGA):

- For some kind of problems, the population needs to be very large and the memory required to store each individual may be considerable. In some cases this makes it impossible to run an application efficiently using a single machine, so some parallel form of GA is necessary.
- Fitness evaluation is usually very time-consuming. Thus, a practical way to optimize this operation is to the use of parallel processing.
- Sequential GAs may get trapped in a sub-optimal region of the search space thus becoming unable to find better quality solutions. PGAs can search in parallel different subspaces of the search space, thus making it less likely to become trapped by low-quality subspaces.

However, the most important advantage of PGAs is that in many cases they provide better performance than single population-based algorithms, even when the parallelism is simulated on conventional machines. The reason is that multiple populations evolve in different directions, i.e. toward different optimal (Nowostawski, 1999a, Nowostawski, 1999b).

2. GENETIC ALGORITHMS

Simple GAs operate on a fixed-sized population of fixed-length individuals. In general, the individuals are a binary string that encodes the variables of the problem that the algorithm is trying to optimize. However, in this work, the individuals are encoded by a real-valued string. Simple GAs use basically three operators: selection, crossover and mutation.

The **selection** operator identifies the fittest individuals of the current population to serve as parents of the next generation. The fitness value of each individual is given by a problemdependent function. The selection mechanism can take many forms, but it always ensures that the best individuals have a higher probability to be selected to reproduce to form a new generation.

The primary exploration mechanism for GAs is **crossover**. This operator randomly chooses a pair of individuals among those previously selected to breed and exchanges substrings between them. The exchange occurs around randomly selected crossing points.

The **mutation** operator is usually considered a secondary operator. Its main function is to restore diversity that may be lost from the repeated application of selection and crossover. This operator simply takes one string from the population and randomly alters some value within it. Following nature's example, the probability of applying the mutation operator is very low compared to the probability of applying the crossover operator.

Genetic algorithms are not guaranteed to find an optimal solution and their effectiviness is determined largely by the population size (Goldberg, 1991, Cantú-Paz, 1995). As the population size increases, the GA has a better chance of finding the global solution, but the computation cost also increases as a function of the population size (Goldberg, 1991).

With serial GAs, we have to choose between getting a good result with a high confidence and pay a high computational cost or loosen the confidence requirement and get (possibly poor) results fast. In contrast, parallel GAs can keep the quality of the results high and find them fast because, using parallel machines, larger populations can be processed in less time. This

keeps the confidence factor high and the response time low, opening opportunities to apply genetic algorithms in timeconstrained applications. Additionally, parallel GAs may evolve several different independent solutions that may be recombined at later stages to form better solutions.

3. THE INVERSE HEAT CONDUCTION PROBLEM

Forward heat conduction problems are focused on determination of the temperature field of the medium, when the boundary and initial conditions, heat source/sink terms (if any), and the physical properties of the material are known. On the other hand, an inverse heat conduction problem is concerned with the estimation of such quantities (boundary or initial conditions, source/sink terms, physical properties) from the temperature and/or heat flux measurements.

Mathematically, the inverse problems are classified as ill-posed problems. A problem is considered well-posed if the following requirements are satisfied: the solution of the problem exists, it is unique and stable with respect to the input data (Hadamard, 1923). The existence of a solution for an inverse heat transfer problem can be assured based on physical reasoning, but the requirement of uniqueness can only be formally proved for some special cases. Also, the inverse problem solution is generally unstable. Therefore, small perturbations in the input data, like random errors inherent to the measurements used in the analysis, can cause large oscillations on the solution.

3.1 The direct problem

The direct (forward) problem consists of a transient heat conduction problem in a slab with adiabatic boundary condition and initially at a temperature denoted by $f(x)$. The mathematical formulation of this problem is given by the following heat equation:

$$\frac{\partial^2 T(x,t)}{\partial x^2} = \frac{\partial T(x,t)}{\partial t}, \quad x \in (0,1), \quad t > 0, \tag{1}$$

$$\frac{\partial T(x,t)}{\partial x} = 0, \qquad x = 0; \qquad x = 1, \quad t > 0, \tag{2}$$

$$T(x,t) = f(x), \qquad x \in [0,1], \quad t = 0, \tag{3}$$

where $T(x,t)$ (temperature), $f(x)$ (initial condition), x (spatial variable), and t (time variable), are nondimensional quantities. The solution of the direct problem for $x \in (0,1)$ and $t > 0$ for a given initial condition $f(x)$ is given by

$$T(x,t) = \sum_{m=0}^{\infty} e^{-\beta_m^2 t} \, \frac{1}{N(\beta_m)} \, X(\beta_m, x) \int_0^1 X(\beta_m, x') f(x') \, dx' \,, \tag{4}$$

where $X(\beta_m, x)$ are the eigenfunctions associated to the problem, and β_m and $N(\beta_m)$ are, respectively, the corresponding eigenvalues and *norms* (Özisik, 1980). The function f is assumed to be bounded satisfying Dirichlet's conditions in the interval $[0,1]$ (Carslaw 1959). The eigenfunctions, eigenvalues, and the norm for the problem defined by equations (1)-(3) can be precisely expressed as

$$X(\beta_m, x) = \cos(\beta_m x), \qquad \beta_m = m\pi, \quad (m = 0, 1, 2, ...); \quad \text{and}$$

$$N(\beta_m) = \begin{cases} 1, & \text{at} \quad m = 0; \\ 1/2, & \text{at} \quad m = 1, 2, ... \end{cases}$$

Assume τ is the final time, this solution could be expressed in a discrete form

$$T(x_j,\tau) = \sum_{m=0}^{M_{\max}} e^{-m^2\pi^2\tau}\ \frac{\cos(m\pi x_j)}{N_m} \sum_{i=1}^{I} \frac{1}{2}\ \triangle x\left[\cos(m\pi x_i)\, f_i + \cos(m\pi x_{i+1})\, f_{i+1}\right]\ , \tag{5}$$

where I is the total subdivision number of the interval of interest (integration domain), and the series (4) was truncated considering $M_{\max}$ terms. This solution could be expressed also in a compact form

$$T_j = \sum_{i=1}^{I} \left[f_i\, C_{ij} + f_{i+1}\, C_{i+1j}\right]\ , \tag{6}$$

where

$$C_{ij} = \sum_{m=0}^{M_{\max}} e^{-m^2\pi^2\tau}\ \frac{\cos(m\pi x_j)}{N_m} \frac{1}{2}\ \triangle x\ \cos(m\pi x_i)\ . \tag{7}$$

Hence, the temperature profile $T(x,\tau)$ can be determined by solving the matrix system

$$\mathbf{T}^{\mathrm{Mod}} = \mathbf{T}^{\tau} = \mathbf{C}^{\tau}\mathbf{f}, \tag{8}$$

where τ is the final time.

3.2 The inverse problem

As mentioned before, a transient heat conduction problem in a slab is considered, with both boundaries kept insulated. The goal is to estimate the unknown initial temperature distribution f, from the knowledge of the measured temperatures T at the time $t = \tau > 0$, for a finite number of different locations in the domain.

In order to solve this ill-posed problem the use of a regularization technique was required, and the 0th-order Tikhonov regularization was chosen (Tikhonov, 1977). The regularized solution is obtained by choosing the function $f_\xi(x)$ that minimizes the following functional form:

$$J(f_\xi) = \left\| T^{\mathrm{Mod}} - T^{\mathrm{Exp}} \right\|_2^2 + \xi \left\| f^{(k)} \right\|_2^2, \tag{9}$$

where $T^{\mathrm{Exp}} = T^{\mathrm{Exp}}(x,\tau)$ is the experimental data ($t = \tau > 0$), T^{Mod} is the temperature obtained using the initial condition recovered (f_ξ), ξ is the regularization parameter, and $\|\cdot\|_2$ is the norm 2.

The inverse problem is formulated as an optimization problem, which was solved by the parallel implementation of a specifically developed genetic algorithm method, which fitness function is represented by the functional form, Eq. (9).

4. THE PARALLEL GENETIC ALGORITHM - PGA

An important feature of GA's is their suitability for parallelization. Nowadays GA's have been widely employed , but parallel implementations are recent. The most common GA parallelization techniques (Cantú-Paz, 1995) are:

- Global Parallelization: in this approach all genetic operators and the evaluation of all individuals are explicitly parallelised.
- Coarse Grained: such approach requires a division of population into some number of demes (subpopulations) that are separated one from another ('geographic isolation'). The individuals compete only within a deme. This approach introduces an additional operator called *migration* that is used to send some individuals from one subpopulation to another.
- Fine Grained: such approach requires a large number of processors because the population is divided into a large number of small demes, and each one evolves separately, but subject to migration.

Many GA researchers believe that a PGA, with its multiple distributed subpopulations and local rules and interactions, is a more realistic model for the evolution of species in nature than a single large population. Indeed, by analogy with natural selection, a population is typically composed by many independent subpopulations that occasionally interact. Parallel genetic algorithms also naturally fit the model of how evolution is viewed: a large degree of independence exists in the "global" population (Levine, 1994).

Two migration models are used in coarse grained GA implementations: island Model and the Stepping Stone Model. In the first model, the population is partitioned into small subpopulations that are geographically isolated and individuals can migrate to any other subpopulation. In the last model, the population is partitioned in the same way, but migration is restricted to neighboring subpopulations (Fig.1). In this work, the island Model was adopted.

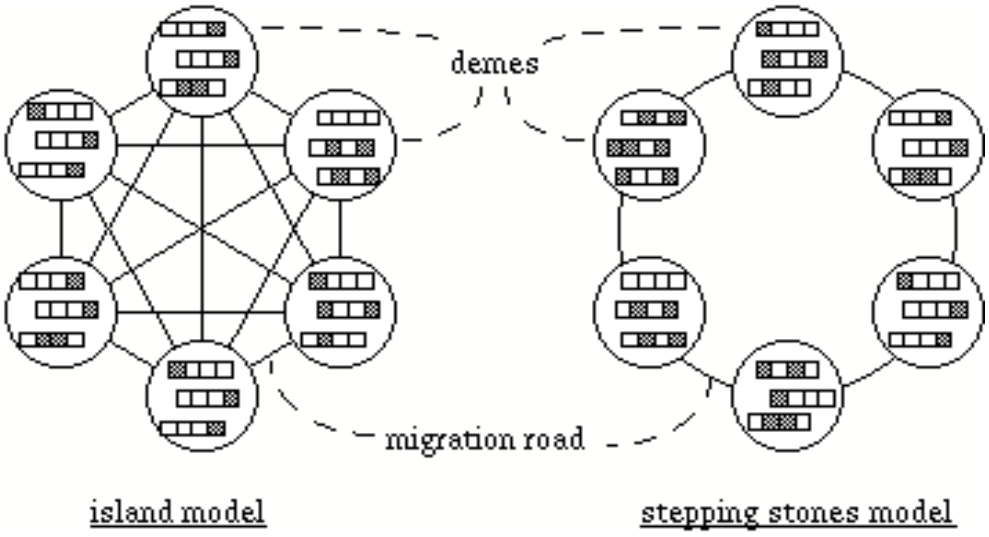

Figure 1: population structures used in the coarse grained approach

4.1 The island model genetic algorithm

The island model genetic algorithm (IMGA) is analogous to the island model of population genetics. In this model, a GA population is divided into several subpopulations, each of which being randomly initialized, and being evolved by an independent sequential GA. Occasionally, fit strings migrate between subpopulations.

The migration of strings between subpopulations is a key feature of the IMGA. This serves to increase the overall selective pressure since additional reproductive trials are allocated to

those strings that are fit enough to migrate (Levine, 1994). At the same time, the introduction of migrant strings into the local population helps to maintain genetic diversity, since the migrant string arrives from a different subpopulation which has evolved independently.

The IMGA is itself a logical model that is very suitable for parallel architectures. In this case each island is mapped to a processor that runs the sequential GA on its own subpopulation. This allows to perform more reproductive trials in a fixed time period, provided that the parallelization overhead due to interprocessor communication (migration of strings) does not impose a significant penalty on processing time. Synchronization between processors is loose as selection and other GA operators are applied locally in each subpopulation.

The proposed IMGA follows a single-program multipledata (SPMD) model: each processor executes the same program on different data (their respective subpopulations) according to its rank. The communication and synchronization between processors occurs when there is migration of strings.

4.2 Parameters of the island model genetic algorithm

An IMGA requires the definition of some parameters:

- the type of sequential GA to be executed in the nodes
- the number of strings that will migrate
- how often the migration will take place
- which strings will migrate
- which strings will be replaced due to the migration

Although the definition of the migration parameters has been intensively studied, intuition is still more used than analysis, with quite good results (Hüe, 1997).

Choosing the right time for migration and which individuals should migrate appears to be more difficult. Species may evolve quickly in small isolated populations. Nevertheless, migrations should occur after a time long enough for allowing the development of goods characteristics in each subpopulation. It also appears that migration is a trigger for evolutionary changes. If migration occurs after each new generation, the algorithm is more or less equivalent to a sequential GA with a larger population. In practice, migration occurs either after a fixed number of iterations in each deme or at uniform periods of time. Migrants are usually selected randomly from the best individuals in the population and they replace the worst ones in the receiving deme. In fact, intuition is still strongly recommended to fix migration rate and migration interval: there are no fixed rules and a personal “cooking recipe” may give good results (Hüe, 1997).

There are two reasons to send a string to another subpopulation. One is to increase the fitness of the other subpopulation. The other is to help the other subpopulation to maintain diversity. As in the sequential GA, the competing themes of selective pressure and diversity arise. If a subpopulation consistently and frequently receives similar, highly fit strings, these strings become predominant in the population, and the GA will focus its search on them at the expense of a lose of diversity. On the other hand, if random strings are received, diversity may be maintained, but the subpopulation’s fitness will likely not improve.

Concerning the migration policy, the best individual was chosen as the migrant and it replaces the worst individual in the receiving subpopulations, while for the migration frequency, it was adopted an empirical value which is determined as function of the maximum number of generations.

4.3 The parallel genetic algorithm - island model

This Parallel Genetic Algorithm was written in Fortran90 and the parallelization was implemented using calls to the Message Passing Interface (MPI) communication library (Pacheco, 1995). Each processor corresponds to an island of the Island Model, and its initial subpopulation is randomly generated and evolve independently from the others subpopulations until the migration operators are activated.

In the current Island Model implementation, each processor evolves a population composed by fixed-length and real-valued strings individuals (Michalewicz, 1996) that encode the variable to be optimized (f_ξ). Besides the standard evolutionary operators (selection, crossover, mutation,...), it was introduced a new operator called **epidemical** (Medeiros, 2002) (see also (Chiwiacowsky, 2003)). The pseudocode of the proposed parallel genetic algorithm follows, with the new evolutionary operator:

Parallel_Genetic_Algorithm
Define and initialize the evolutionary parameters;
$t \leftarrow 0$;
Generate a random population $Pop(t)$;
Calculate the fitness $J(f_\xi)$ *of each individual in* $Pop(t)$;
While (condition)
  $t \leftarrow t + 1$;
  Select a pair of parents (X_1, X_2) *from* $Pop(t-1)$;
  Cross over the pair (X_1, X_2);
  Mutate the new string X *generated;*
  Insert the individual X *in the worst location of* $Pop(t-1)$;
  Calculate the fitness of the individual X;
  $X^*(t) \leftarrow Best[Pop(t)]$;
  If (migration) then
    $X_{migrante} \leftarrow X^*(t)$
    Send individual $X_{migrante}$ *to the other processors;*
    $X_{recv} \leftarrow$ *recv_idivduo(*$X_{migrante}$*);*
    $X_{delete} \leftarrow Worst[Pop(t)]$;
    Replace X_{delete} *by* X_{recv};
  End_If
  If $(X^*(t) = X^*(t-1))$ *then*
    $cont\_epidemic \leftarrow cont\_epidemic + 1$;
    If $(cont\_epidemic > limit)$ *then*
      Apply Epidemical on $Pop(t)$;
    End_If
  End_If
End_While
End

Following, the evolutionary operators employed in this work are presented:

- Tournament Selection (Mitchell, 1996). This operator uses a random number *rand* from the interval [0,1) with uniform distribution. $bigger$ is the best fitness individual and $smaller$ is the worst fitness individual:

bigger:=rand; smaller:=rand; val:=0.75;
if (rand $<$ *val) then*
    position:=bigger;
else
    position:=smaller;
endif

- Geometrical Crossover (Michalewicz, 1996). This crossover operator breeds only one offspring from two parents. From the parents x_i and y_i the offspring is represented by:

$$z_i = x_i^{\mu}\, y_i^{1-\mu}, \tag{10}$$

where μ is a number between [0, 1]. A typical value is $\mu = 1/2$, where the same weight is given to both parents.

- Non-uniform Mutation (Michalewicz, 1996). This mutation operator is defined as:

$$x_i' = \begin{cases} x_i + \triangle(t, l_{sup} - x_i) & \text{if a random binary digit is } 0, \\ x_i - \triangle(t, x_i - l_{inf}) & \text{if a random binary digit is } 1, \end{cases} \tag{11}$$

and

$$\triangle(t, y) = y\left[1 - rand^{\left(1-\frac{t}{T}\right)^b}\right],$$

where $rand$ is a random number from the interval [0,1) with uniform distribution, T is the maximal generation number, t is the generation number, and b is a system parameter determining the degree of non-uniformity.

- Epidemical Strategy (Medeiros, 2002) (see also (Chiwiacowsky, 2003)). This innovative operator is activated when a specific number of generations is reached without improvement of the best individual. Then, all the individuals are *affected by a plague*, and only those that have the best fit (e.g., first 2% with the best fit in the population) *survive*. The remaining individuals *die* and are substituted by new individuals with new genetic variability, such as immigrants arriving, in order to evolve the population. Two parameters need to be chosen: one determines when the strategy will be activated, i.e. the number of generations without improvement of the best individual fit, while the other parameter determines the amount of individuals that will survive the *plague*.

5. NUMERICAL RESULTS

In this article we have described a parallel genetic algorithm to solve the backward heat conduction problem involving the estimation of the unknown initial condition for a one-dimensional slab. The chosen test case assumed the following initial temperature profile, given by the triangular test function

$$f(x) = \begin{cases} 2x, & 0 \le x < 0.5, \\ 2(1-x), & 0.5 \le x \le 1; \end{cases} \tag{12}$$

The experimental data (*measured temperatures at a time* $\tau > 0$) are obtained from the exact solution of the direct problem by adding a random perturbation error to the exact solution of the direct problem in order to generate noisy data

$$T^{exp}(t) = T(t) + \sigma\,\mathcal{R}, \tag{13}$$

where σ is the standard deviation of the errors and $\mathcal{R}$ is a random variable taken from a normal distribution such that $\mathcal{R} \sim \mathcal{N}$ormal(0;1). For numerical purposes, it has been adopted $\tau = 0.01$ s and $\sigma = 0.05$ and a spatial grid consisting of 101 points ($M_{\max} = 101$). In order to accomplish the inversions, some parameter of the parallel genetic algorithm are required

- Fixed population size of: 1008 or 336 individuals;
- Geometrical crossover operator : $\mu = 1/2$;
- Non-uniform mutation operator: $b = 5$;
- Mutation probability: 5%;
- Epidemical Operator: the best 5 individuals have been kept;
- Fixed maximal generation number: 50000.

5.1 Parallel performance

Usually, the performance of a parallel implementation can be roughly evaluated by the **speedup** and **efficiency** values. Let T_1 be the processing time using the sequential code and T_p, the time spend using the parallel code executed by p processors. The speedup S_p is defined by

$$S_p = \frac{T_1}{T_p} \tag{14}$$

while the efficiency E_p, by

$$E_p = \frac{S_p}{p} \leq 1 \ . \tag{15}$$

A speed-up equal to p is called linear and yields $E_p = 1$. A possible side effect in parallelization is cache memory access optimization due to the partitioning of the domain. This may cause speed-ups greater than p when using p processors and it is called super-linear speed-up (Pit, 1995). However, the overhead due to interprocessor communication and synchronization may cause poor speed-ups.

5.2 Performance analysis

The performance values in this work were computed for the optimization routine (the AGP). Timing information was provided by successive calls to the MPI function "MPI_WTIME" that reads the system wall time. Numerical tests were performed in a 17-node distributed memory parallel machine with IA32 architecture. Two different population sizes, both multiple of the used number of processors p, were selected: 1008 and 336 individuals. Elapsed times, speed-ups and efficiencies are given in Tables 1 and 2.

The execution times for different number of processors considering the two population sizes are presented in Fig. 2 and Table 3. As expected, the times obtained for the 1008-population are higher than those for the 336-population. It can be noted that the ratio of the execution times for these two populations decreases as the number of processors is increased.

Figure 3 shows the speed-up values for the two population sizes in function of the number of processors. A superlinear speed-up can be observed for the 1008-population, while the 336-population presents a speed-up close to linear. This can also be observed in the efficiency values, presented in Fig. 4. The efficiency is greater than 1 for the 1008-population when more than 4 processors are used.

Table 1: Elapsed Time, Speed-up and Efficiency for the 1008-individual population

Nproc	Time (s)	S_p	E_p
1	8341	1.0	1.0
2	4630	1.8015	0.9007
3	2794	2.9853	0.9951
4	2098	3.9757	0.9939
6	1287	6.4810	1.0802
7	1117	7.4673	1.0668
8	947	8.8078	1.1010
9	836	9.9773	1.1086
12	630	13.2396	1.1033
14	527	15.8273	1.1305
16	457	18.2516	1.1407

Table 2: Elapsed Time, Speed-up and Efficiency for the 336-individual population

Nproc	Time (s)	S_p	E_p
1	2137	1.0	1.0
2	1283	1.6656	0.8328
3	834	2.5624	0.8541
4	629	3.3975	0.8494
6	428	4.9930	0.8322
7	344	6.2122	0.8875
8	300	7.1233	0.8904
12	201	10.6318	0.8860
14	174	12.2816	0.8773
16	161	13.2733	0.8296

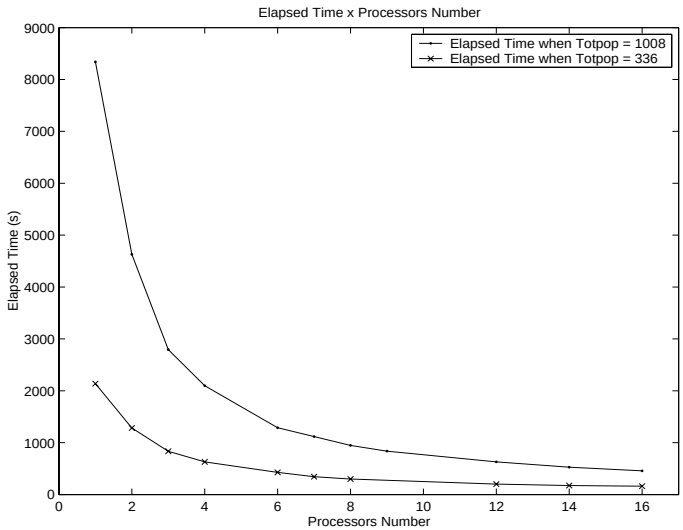

Figure 2: Execution times versus the number of processors considering the two population sizes

Table 3: Execution times versus the number of processors considering the two population sizes

Nproc	Totpop = 1008	Totpop = 336	Time Ratio
1	8341	2137	3.9031
2	4630	1283	3.6087
3	2794	834	3.3501
4	2098	629	3.3355
6	1287	428	3.0070
7	1117	344	3.2471
8	947	300	3.1567
12	630	201	3.1343
14	527	174	3.0287
16	457	161	2.8385

The total execution time is divided into processing and communication times. A bigger population requires higher processing times, but the amount of data that need to be communicated during the migration is the same for both populations. This implies that the 1008-population would have better speed-up and efficiency values as the number of processors increases, as confirmed in the preceding figures and tables. However, the total execution times are always greater for the bigger population and its corresponding speed-up and efficiencies were calculated using only the sequential time of the same population as a reference.

Therefore, the superlinear speed-up for the bigger population is due to bad memory access in the sequential version. The use of more processors in the parallel version improves cache memory access, causing the superlinear effect for more than 4 processors.

On the other hand, the opposite behaviour can be observed for the 336-population. As the size of data to be computed by each processor is smaller the performance of the sequential version is better for the 336-population: the ration between the sequential times of the two populations is 3.90, while their size ratio is 3, denoting that memory access is better for the smaller population. Thus, the 1008-population can take better advantage of cache memory access than the 336-population with the parallelization.

However, despite the smaller total execution time, the 336-population presents lower values of speed-up and efficiency, since the amount of processing decreases as more processors are used, while the amount of communication remains constant.

The good accuracy of the initial condition reconstruction is shown in Figs. 5 and 6, for both the 336-population and the 1008-population. It should be noted that the quality of the estimation depends on the specific experimental data, the fitness of the PGA-initial population, the population size and also on the parameters of the PGA.

6. FINAL COMMENTS

The inverse problem of estimating the unknown initial condition of heat conduction transfer in a one-dimensional slab was solved using a specific parallel genetic algorithm, that uses a new evolutionary operator called *epidemical* (Medeiros, 2002). The application of PGA's to optimization problems is an open problem, since there are several parameters to be adjusted in order to achieve the best results. The parameter "tuning" of the PGA is an empirical task requiring a great amount of interaction. The results obtained using the PGA method described in this work presented good accuracy and show to be competitive with classical deterministic optimization methods.

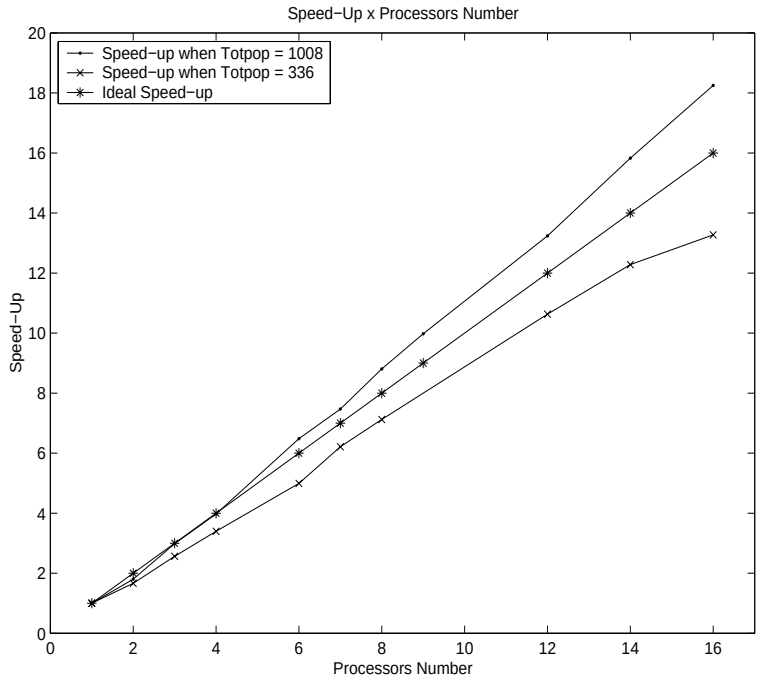

Figure 3: Speed-up versus the number of processors considering the two population sizes

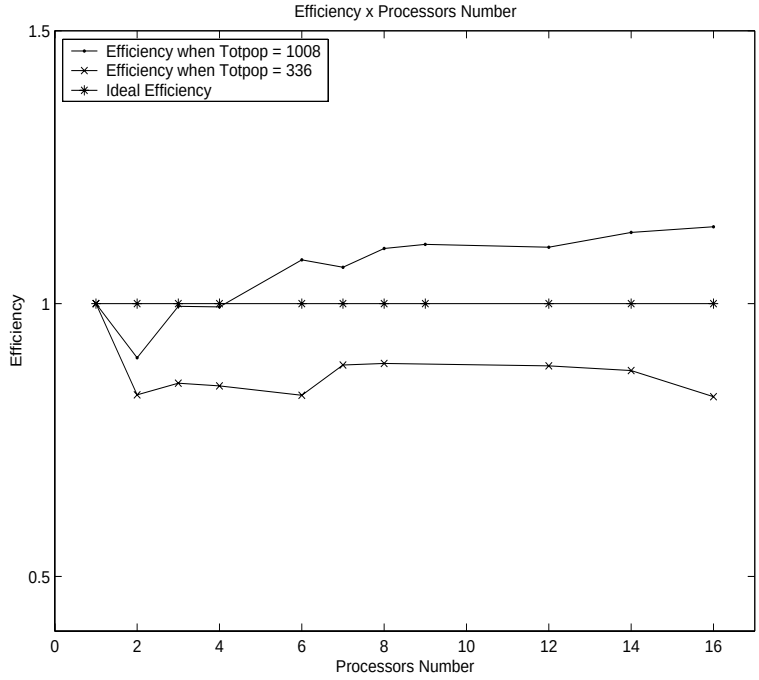

Figure 4: Efficiency versus the number of processors considering the two population sizes

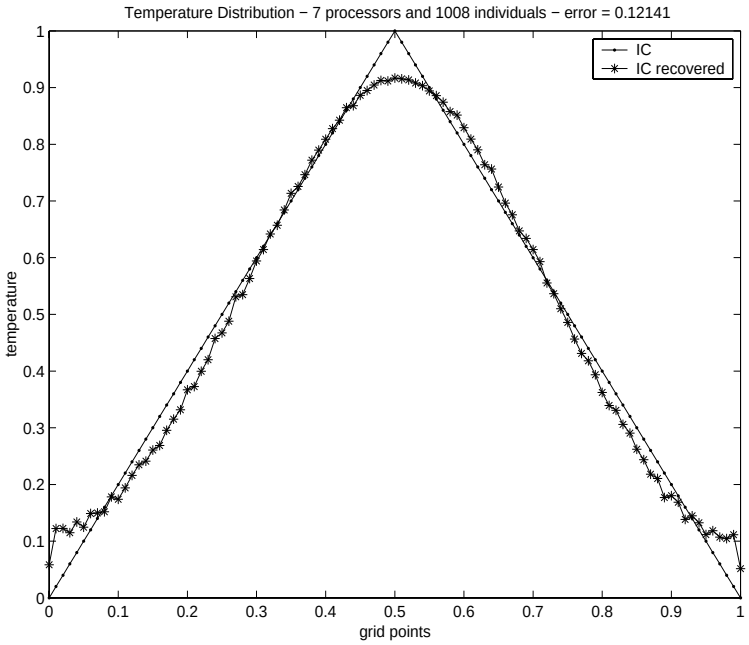

Figure 5: PGA solution for the 1008-population

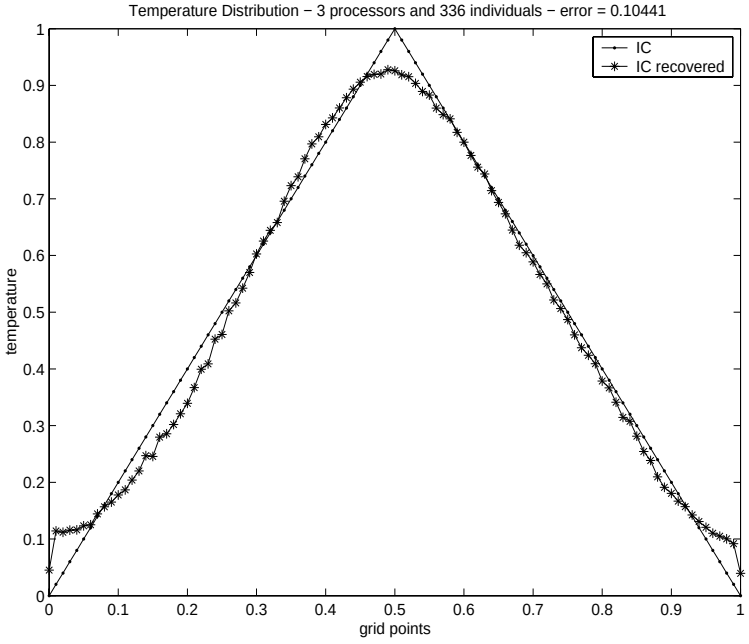

Figure 6: PGA solution for the 336-population

Acknowledgements

Authors L.D. Chiwiacowsky and H. F. de Campos Velho acknowledge the finantial support by FAPESP, The State of São Paulo Research Foundation (process 00/09488-6), CAPES, the Brazilian Post-graudation Support Agency, and CNPq, the Brazilian Council for Scientific and Technological Development (process 300466/95-1). Authors S. Stephany and A.J. Preto thanks FAPESP for the support given to this study through a Research Project grant (process 01/03100-9).

REFERENCES

Cantú-Paz, E., 1995, "A summary of research on parallel genetic algorithms", IlliGAL Report N^o 95007, University of Illinois.

Carslaw, H. S. & Jaeger, J. C., 1959,"Conduction of heat in solids", Oxford University Press, London.

Chiwiacowksy, L. D. & Campos Velho, H. F., 2003, "Different Approaches for the Solution of a Backward Heat Conduction Problem", Inverse Problems in Engineering.

Goldberg, D. E., 1991, "A comparative analysis of selection schemes used in genetic algorithms, eds Gregor Rawlins, Foundations of Genetic Algorithm", Morgan Kaufmann Publishers, USA.

Hadamard, J., 1923, "Lectures on the Cauchy problem in linear partial differential equations", Yale Universtity Press, New Haven.

Hüe, X., 1997, "Genetic algorithms for optimisation - background and applications", Edinburgh Parallel Computer Centre Rept., The University of Edinburgh.

Levine, D., 1994, "A parallel genetic algorithm for the set partitioning problem", Mathematics and Computer Science Division, Argone National Laboratory Rept., University of Illinois.

Medeiros, F. L. L., 2002 Hybrid Genetic Algorithm as a Search Scheme of Steady State of Dynamical Systems, M.Sc. Thesis, Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos (SP), Brazil – in portuguese.

Michalewicz, Z., 1996, "Genetic algorithms + data structures = evolution programs", Springer-Verlag, USA.

Mitchell, M., 1996, "An introduction to genetic algorithms", MIT Press, USA.

Nowostawski, M & Poli, R., 1999, "Dynamic demes parallel genetic algorithm", Proceedings of Third International Conference on Knowledge-based Intelligent Information Engineering Systems KES'99, May 13, Adelaide, South Australia.

Nowostawski, M & Poli, R., 1999, "Parallel genetic algorithm taxonomy", Proceedings of Third International Conference on Knowledge-based Intelligent Information Engineering Systems KES'99, May 13, Adelaide, South Australia.

Özisik, M. N., 1980, "Heat conduction, Wiley Interscience", USA, 1980.

Pacheco, P. & Ming, W. C., 1995, “Introduction to message passing programming - MPI user guides in FORTRAN and C”, Department of Mathematics, University of San Francisco, USA.

Pit, L.J., 1995, “Parallel genetic algorithms”, Department of Computer Science, Master Thesis, Leiden University, Holanda.

Tikhonov, A. N.& Arsenin, V. Y., 1977,“Solutions of Ill-Posed Problems”, Winston & Sons, Washington, D.C.