Regular paper

PARALLEL GENETIC ALGORITHM OPTIMIZATION OF DIE PRESS

Pavlina MIHAYLOVA[1], Kostadin BRANDISKY[2]

Abstract: This paper presents an optimization based on genetic algorithm computed in a cluster of computers with application to problems in electrical engineering. Some basic steps and technologies used for implementing this approach, as well as its advantages and disadvantages are discussed. The main goal is to show the merits of distributed optimization and to determine its suitability in the area of numerical field analysis.

Keywords: Cluster computing, Genetic algorithm, Optimization, Finite element method

INTRODUCTION

Everyday design of electromagnetic devices has to be accurate, fast and simple to use, as well as meet certain requirements such as low price, better performances, reduced size etc. However, the creation of adequate design models and their numerical solution can sometimes consume considerable time and hardware resources in the case of complex problems. On a single computer the solution of such models can take hours or days.

As a result, it becomes much more sensible and cheaper to use the available computer resources for example by organizing them in computer clusters, if the solution/optimization algorithm permits this. The present work is based on such an approach. It aims to figure out the suitability of parallel GA optimization for electromagnetic devices, modelled with the Finite Element Method (FEM).

The computation time for serial GA execution becomes high for time consuming fitness functions such as those including finite element analysis (FEA) at each objective function call. A better alternative is to take advantage of the intrinsically parallel nature of GAs and to perform the generation of new populations in parallel, on different processors.

Some of the existing possibilities for programming parallel applications include using MPI libraries for C++ and Fortran 90 compilers, using the PVM libraries, using the MATLAB Distributed Computing toolbox for cluster of computers, or using OpenMP for multi-core processors.

MPI functions are in fact standard nowadays, but they are complicated and require a lot of coding and fine tuning. Instead, MATLAB's Distributed Computing Toolbox (DCT) is chosen in this work. DCT includes functions based on MPI to implement communication among tasks. Its main responsibility is to develop distributed MATLAB applications and execute them in a cluster of computers, thus reducing execution time compared to running the same algorithm on a single computer.

For the purpose of the parallel GA optimization in this paper, MATLAB's Genetic Algorithm and Direct Search toolbox is used. It can be easily connected to the DCT in such a way that the fitness function evaluations at each generation are computed in parallel.

FEA is performed using FEMM 4.0 [1] to create a 2D model, in combination with Lua-scripting for easier control of the design variables. The modelled device is taken from the TEAM Workshop Problem 25 - Optimization of die press [2].

PROBLEM FORMULATION

Figure 1 shows the model of a die press with electromagnet that can set up a strong magnetic field [2], [3]. It is used to orient the magnetic powder in a component and produce anisotropic permanent magnet. The magnetic powder is inserted in the cavity. The die molds are set to form the radial flux distribution. The orientation and strength of the magnetic field should be controlled in order to obtain the required magnetization in the component that is being magnetized. The objective is to find the optimal shape of nonlinear die molds in order to obtain the desired magnetic field in the cavity.

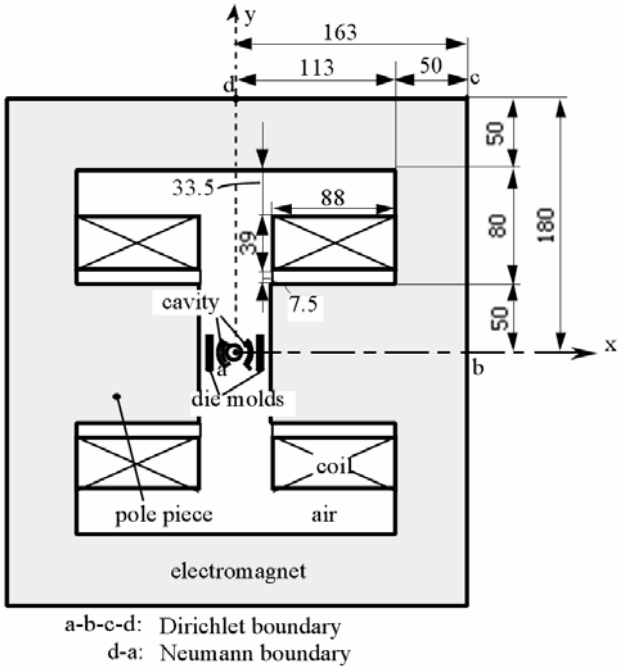

Fig.1 – *Die press with electromagnet – whole view*

[1] Technical University of Sofia, Kliment Ohridski 8, Sofia 1000, Bulgaria, E-mail: pulli@abv.bg

[2] Technical University of Sofia, Kliment Ohridski 8, Sofia 1000, Bulgaria, E-mail: kbran@tu-sofia.bg

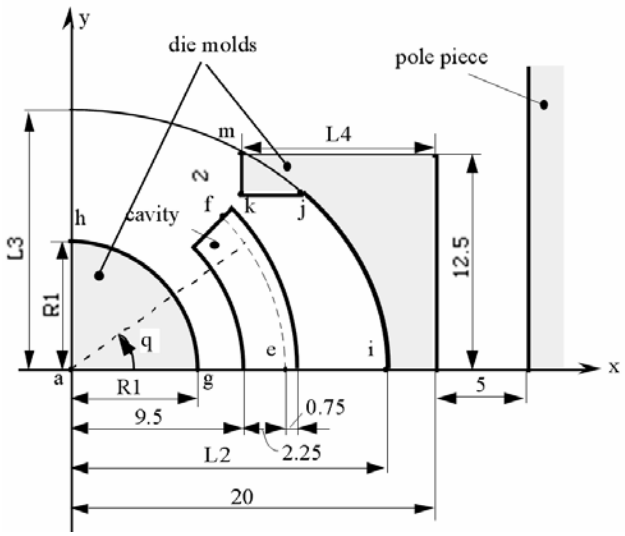

Fig.2 - *Die press with electromagnet – enlarged view*

The geometry of the whole device is shown on figure 1. A detailed view of the die molds is presented on figure 2. The die press and electromagnet are made of steel with B-H curve shown in figure 3. Each coil has 4253 ampere-turns.

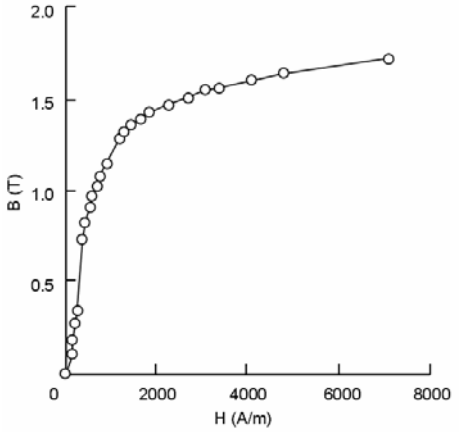

Fig.3 – *BH curve of the steel*

The x- and y- flux density components B_x and B_y at the points along the line *e-f* in the cavity must satisfy:

$$B_x = 0.35\cos\theta\ ,T \qquad (1)$$

$$B_y = 0.35\sin\theta\ ,T \qquad (2)$$

where $0^o \le \theta \le 45^o$ is the angle measured from the x-axis and $r_0 = 11.75mm$ is the radius along which the points are measured. These equations logically lead to the objective function (OF), which has to be minimized:

$$OF = \sum_{i=1}^{n} \left\{ \left(B_{xicalc} - B_{xireq}\right)^2 + \left(B_{yicalc} - B_{yireq}\right)^2 \right\} \qquad (3)$$

where *n*=10 is the number of points at which the magnetic flux density is examined.

It has been proved that the above specified flux distribution can be obtained if the shape of the die press is parametrized by a circle for the inner and an ellipse for the outer die. This justifies the choice of the following design variables:

R_1 - radius of the inner mold, [mm]

L_2 and L_3 - axes of the ellipse, [mm]

L_4 - length of the top piece of the outer mold, [mm]

The only constraints in this optimization are the ranges specified for the values of the design parameters, which are shown in table 1.

Table I
Ranges of the design variables

Design variable	Lower limit	Upper limit
R_1	5	9.4
L_2	12.6	18
L_3	14	45
L_4	4	19

PARALLEL OPTIMIZATION USING GA

GA is a stochastic optimization method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The GA repeatedly modifies a population of individual solutions in order to create the next generation, using three main types of rules – selection, crossover and mutation. At each step, the GA selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. The GA can be applied to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear.

Some optimization problems involve computationally expensive objective functions. At each generation of the GA, the objective, or fitness function must be evaluated at all the points in the current population. This can be very time consuming for expensive (computationally intensive) fitness functions as FEM analysis. When the fitness function is expensive enough (the computation time of the fitness function is sufficiently higher than the communication time between the master processor and slave processors), it is possible to distribute these objective evaluations across processors, to compute them in parallel, and see speedup in the overall execution of the optimization algorithm.

An attempt for such distributed computation is made in the current work. In MATLAB this can be done without changing the fitness function or the GA code. Only the distributed computing environment needs to be set up and then a special function that will distribute the fitness function across the worker machines has to be used. In the terms of MATLAB, the job in this case is to optimize the die press model using GA. The population size divided by the number of available workers gives the number of tasks (fitness function evaluations, i.e. FEAs) that

each worker should accomplish at the different generation stages.

CLUSTER COMPUTING USING MATLAB

The Distributed Computing Toolbox (DCT) and MATLAB's Distributed Computing Engine (MDCE) [4] facilitate the coordination and execution of independent operations simultaneously on a cluster of computers, speeding up execution of time-consuming jobs. Each job is a complete large-scale operation to perform, which is broken down into segments called tasks. The DCT defines the job and its tasks in a client session usually on the machine where the user programs MATLAB. The MDCE on its turn performs the job execution by evaluating each of its tasks and returning the result to the client. The job manager is the part of the engine that coordinates the execution of jobs and the evaluation of their tasks. It distributes the tasks for evaluation to the engine's individual sessions called workers (figure 4).

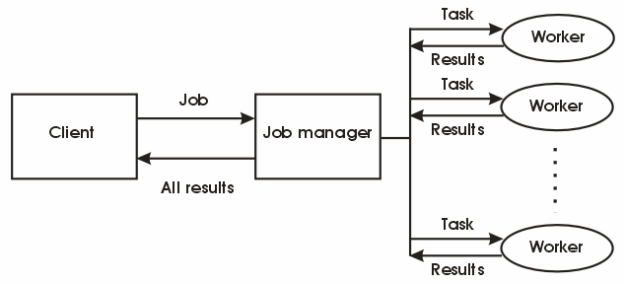

Fig.4 – *Basic cluster computing configuration*

The job manager can be run on any machine on the network. It runs jobs in the order in which they are submitted, unless any jobs in its queue are promoted, demoted, canceled, or destroyed. A MDCE setup usually includes many workers that can execute all tasks simultaneously. The MDCE daemon makes it possible for these processes to communicate with each other on different machines. Each worker is given a task from the running job by the job manager, executes the task, returns the result to the job manager, and then is given another task. The job manager then returns the results of all the tasks in the job to the client session. It is generally not important which worker executes a specific task.

RESULTS

The simulations of the die press model were performed with a FE mesh of nearly 15000 nodes. The population size was chosen to be 50. The optimal solution was obtained on the 20th generation.

Figure 5 shows enlarged view of the flux lines plot for the optimized device in the die molds and the air around them. Dirichlet boundary conditions have been used in order to solve only one quarter of the problem.

In the initial design (at random point in the design variable range) the field along the contour *e-f* was far from satisfactory. Figures 6 and 7 show the values of B_x , B_y and |B| along this contour for the final optimized design.

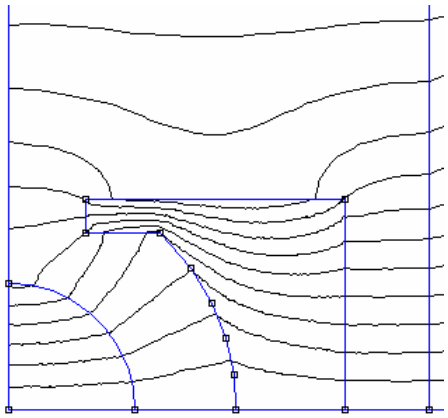

Fig.5 – *Final design – flux lines plot*

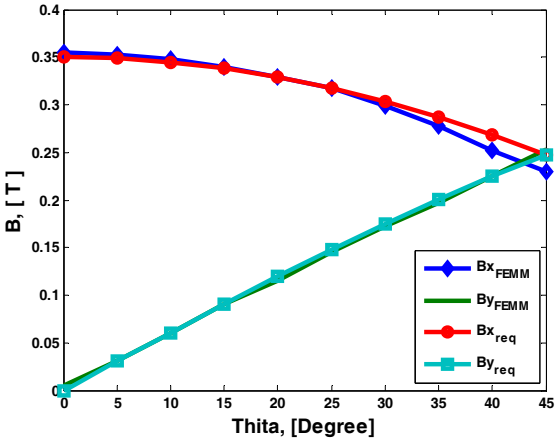

Fig.6 – *B_x and B_y plotted along the line e-f*

As seen from figure 7, the objective has been obtained and the magnitude of the flux density along the contour is very close to 0.35 T. The GA succeeded in finding a solution which fulfills the specified requirements.

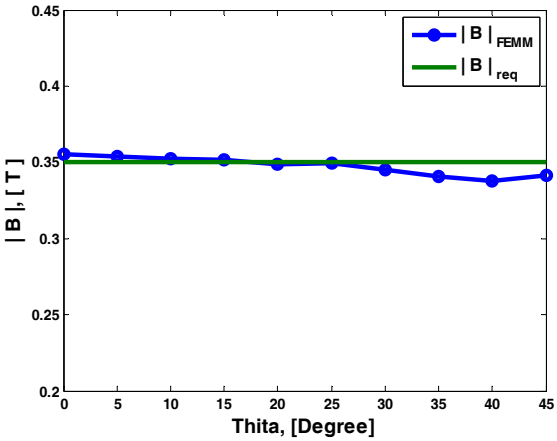

Fig.7 – *| B | plotted along the line e-f*

Table 2 gives the final solution, obtained after 1000 OF calls:

Table II
Final results

Quantity	Value
R_1 , [mm]	7.497

L_2 , [mm]	13.537
L_3 , [mm]	14
L_4 , [mm]	15.406
OF	1.33e-3

To evaluate DCT performance for the die press model a number of sample experiments was made on a cluster of 16 identical computers, each of which having CPU AMD Athlon XP 2500+ and RAM 512 MB. All computers were connected with 100 Mbit/s Ethernet network.

If T_s is the time taken to run the serial algorithm on one processor and if T_{par} is the time taken by its parallel version on N processors, then speedup and efficiency of the parallel algorithm are defined as:

$$\text{Speedup} = S_N = T_s / T_{par} \quad (4)$$

$$\text{Efficiency} = E_N = S_N / N \quad (5)$$

Execution time and efficiency were measured for various numbers of workers at two different population sizes, over 20 generations (figure 8 and 9). The best performances were obtained for 2 and 4 workers.

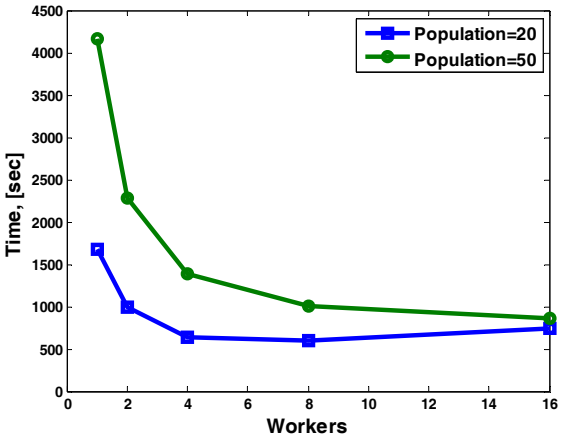

Fig.8 – *Execution time versus number of workers for two population sizes*

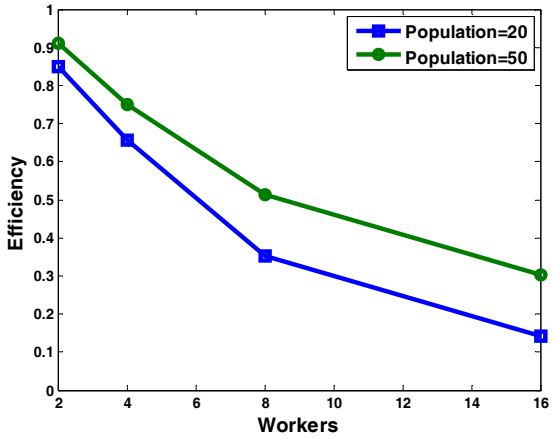

Fig.9 – *Efficiency versus number of workers for two population sizes*

The following discussion of the results can be made:

1. DCT assigns different number of tasks to each worker if the population size is not divisible by the number of workers. However, such noneven task distribution may in some cases lead to increase of communication time, as one part of the workers will always wait for the others, which have additional tasks. An important consideration is to ensure that each node performs the same amount of work, i.e. the system is load balanced.

2. The smaller the population and the bigger the number of workers is, the lower the efficiency is – because of the smaller number of tasks assigned to each worker for sequential execution in the framework of one generation. The goal of the parallel algorithm designer should be to make the grain size (relative amount of work done between synchronizations – communications) as large as possible, while keeping all the processors busy.

3. The increase in the number of generations influences almost linearly the execution time.

4. The unconscionable increase of workers can diminish efficiency if the fitness function is not expensive enough and communication time becomes commensurable with computation time. As mentioned above the effect of communication on speedup is reduced, in relative terms, as the grain size increases. As it can be seen from figure 10, efficiency increased considerably in the case of 8 and 16 workers for a model with denser mesh (nearly 30000 nodes), i.e. for a fitness function that requires higher computation time.

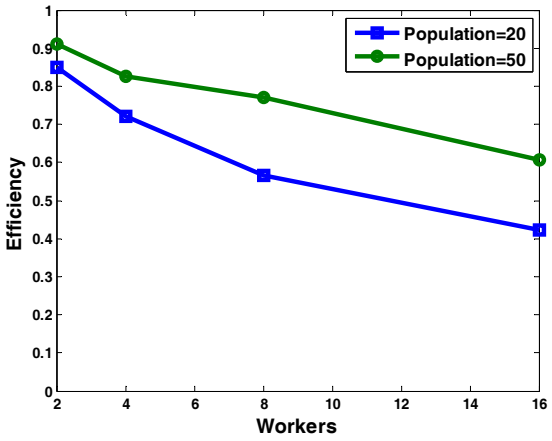

Fig.10 – *Efficiency versus number of workers for two population sizes – model with denser mesh*

CONCLUSION

Parallel GA optimization successfully determines the global optimum (as serial GA also does), but with at lower time cost depending on the number of workers and the complexity of the problem. An important issue is to balance the tasks between the workers in a way that the efficiency does not become undesirably low. The discussed approach for parallel GA optimization can also be implemented and will be advantageous for more time-consuming problems such as 3D models, or easily adapted to other inherently parallel optimization methods – for example, preparation of data sets for supervised

neural networks training or for design of experiment method.

The DCT is flexible and easy to use. Moreover, as it is directly connected to MATLAB, one can use built-in or custom functions in a distributed environment. In combination with GAs it gives the electrical engineer a powerful optimization tool that implements the advantages of GAs for solving complex problems in Electromagnetics.

REFERENCES

[1] D. Meeker: "*FEMM user's manual*", 2005.

[2] N. Takanashi: "Optimization of die press model", TEAM Workshop problem 25.

[3] P. Alotto, C. Eranda, B. Brandstaetter and al.: "Stochastic algorithms in electromagnetic optimization", IEEE Transactions on Magnetics, Vol. 34, No.5, September 1998.

[4] MathWorks: "*Distributed computing Toolbox User's Guide*", 2006.

Pavlina Mihaylova was born on 18.10.1982 in Sofia, Bulgaria. In 2001 she finished the National School of Finance and Economy with excellent marks. In July, 2005 she received B.Sc. degree in Computer Systems and Technologies at the Technical University of Sofia, Bulgaria with the thesis "CAD system for analysis of induction motors". Currently she continues her M.Sc study at the Technical University of Sofia.

During her studies she won three times the first prize at the competitions in Theoretical Electrical Engineering – twice at local competitions at TU-Sofia, and once at the national competition in 2004. She knows excellent German and English and also speaks French and Russian. She is author and co-author of 3 papers. Some of her interests are: Programming, Finite Element Method, MATLAB, Numerical methods.