Regular paper

SEGMENTAL PM DC MOTOR OPTIMIZATION USING PARALLEL GENETIC ALGORITHM

Petar IVANOV[1], Kostadin BRANDISKY[2]

Abstract: This paper presents a new client-server implementation of a parallel genetic algorithm and its application for optimization of a segmental direct current motor. The optimization objective function is minimal volume of the permanent magnet (i.e. minimal cost), subject to minimum torque constraint. The obtained results and system's efficiency are discussed.

Keywords: Genetic algorithm, Parallel computation, CAD system, Finite element method

INTRODUCTION

Unceasing devices miniaturization trend, that can be seen during the last years, requires increasingly smaller constructive components. On the other hand, market economics requires ever decreasing overall cost of the products. Engineers are challenged to invent new (or to develop existing) devices, which have minimal size and end-price, considering necessary characteristics.

DC motors are widespread (portable devices, automobiles, tools, toys etc.). Under the increasing rivalry circumstances, reducing their overall cost is very important. Main components of the end price are the permanent magnet price and the copper conductor price. This paper deals with the permanent magnet volume minimization and thus overall device cost minimization.

In the optimization of electrical machines two approaches are used for obtaining the magnetic field distribution and calculating force characteristics (machine torque, for example)[1]:

a) circuit approach – based on working out and solving equations of an equivalent magnetic circuit

б) numerical field computation approach – based on finite element or finite difference method

The first approach is less time-consuming, but the accuracy of the results is not high. The accuracy is high enough in cases when the magnetic circuit represents well the main flux path and stray field is minimal.

The second approach is more accurate, but requires more computation time – from minutes (for solving 2D models), till several hours (for 3D field analysis). In addition, the number of the design parameters has to be reduced (3-8), in order to reduce the computation time.

In the optimization of the PM DC motor, presented in this paper, the second approach is used – finite element method. Calculating of the machine's torque and the objective function (permanent magnet volume) is accomplished with CAD system FEMM v3.4 [6]. It allows automated building and solving of 2D models, through LUA-script.

Genetic strategy is used, for the purpose of the optimization. The genetic algorithm incorporated in the presented software is based on Genocop III program [4]. Series of improvements (aiming to reduce the number of objective function computations) and innovations of the original algorithm are introduced. Furthermore, Genocop III is transformed from sequential, to parallel form. Synchronous Master-Slave parallelization is used. This allows preserving of the original algorithm's accuracy and effectiveness.

The developed client-server system allows the great number of computations of the objective function (OF), typical for stochastic algorithms, to be performed in parallel on theoretically unlimited number of computers.

New high level communication protocol, based on TCP, is developed especially for the presented system. The protocol is very fast and reliable – communication between clients and server takes a few milliseconds, which is insignificant delay, compared to the amount of time, necessary to obtain the OF (which is of order of few minutes). This provides almost N-times increase of the performance, when N computers are involved in parallel computation.

Parallel processing provides finding of the optimal design in reasonable amount of time (a few hours). Genetic algorithm finds global optimum of the OF and the Finite Element Method ensures high accuracy of OF and constraints computation. All these elements are incorporated in the software system presented in this paper.

GENETIC ALGORITHMS

Main features of genetic algorithms

Genetic algorithms (GA) are powerful, highly efficient search methods. They have high probability of finding the global optimum. GAs are based on evolutionary mechanism and genetic laws of nature (crossover, mutation, selection, reproduction etc.)

One of the most important advantages of GAs is, that the OF is not required to be continuous, convex or differentiable (because derivatives are not used). The quality of

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

the solution does not depend on initial values of input parameters. Usually, in GAs, the initial values are randomly generated.

GAs are capable of optimizing a function with linear constraints for the input parameters, linear and nonlinear equations and inequalities, used as an additional input parameters constraints.

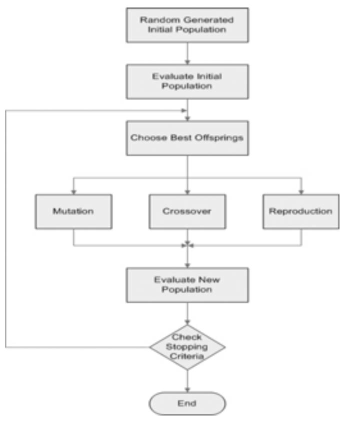

Fig. 1 - *Typical Genetic Algorithm*

Genetic algorithms (Fig. 1) start with an initialization procedure, which creates the initial population. It is generated randomly, obeying input problem specific constraints. After that, the initial population is evaluated and the best solutions are selected to reproduce and form the next generation.

In each generation, relatively "good" individuals reproduce to give offspring that replace the relatively "bad" individuals. The basic mechanism in GAs is Darwinian evolution: bad treats are eliminated from the population, because they appear in individuals, which do not survive the process of selection. The OF plays the role of the environment, which determines which individuals are "good" (give good solutions) and have to reproduce, and which are "bad" and have to "die". Population size is a constant through the evolution process.

Evolution of the individuals is performed by applying genetic operators, modelled on the genetic processes occurring in nature. Genetic operators are two main types: *mutation* and *crossover*.

Mutation is unary operator, defined as selecting a random component $k \in (1,...,n)$ of the vector $\overline{X} = (x_1,..,x_k,..,x_n)$ and changing it with a step in promising direction.

Arithmetical crossover is binary operator, defined as a linear combination of two vectors. If vectors $\overline{x_1}$ and $\overline{x_2}$ are to be crossed, the resulting offsprings [2] are:

$$\overline{x_1'} = \alpha\overline{x_1} + (1-\alpha)\overline{x_2} \qquad (1)$$

$$\overline{x_2'} = \alpha\overline{x_2} + (1-\alpha)\overline{x_1} \qquad (2)$$

On each iteration step, each individual $\overline{x_i}$ from the current population is evaluated by computing the OF $f(\overline{x_i})$, which is a measure of its fitness. Then, a selection process, based on individuals' relative fitness, is applied. Survivals are recombined, using genetic operators, to form the new population.

Main strategy parameters of GAs [3] are:

- population size
- probability of mutation
- probability of crossover
- probability of reproduction
- stopping criteria – maximum generation number or ε - condition.

Genocop III

The structure of Genocop is based on genetic algorithm. It borrows different ideas from other stochastic algorithms. The system uses floating point representations for chromosomes (genes, input parameters) as evolutionary strategies do. One of the operators (arithmetical crossover) has its origin in scatter search algorithms. Ideas of simulated annealing algorithms (smaller changes with lower temperature) are incorporated in other operator – non-uniform mutation [2].

The genetic operators, used in Genocop are:

- uniform mutation
- boundary mutation
- non-uniform mutation
- arithmetical crossover
- simple crossover
- heuristic crossover

The operators are discussed in details in [2].

The Genocop system maintains two separate populations, where a development in one population influences evaluations of the individuals in the other population [4]. The first population consists of so-called search-points from S, which satisfy linear constraints of the problem. The second population consists of so-called reference points from F. These points are fully feasible – they satisfy all constraints (linear and nonlinear). Reference points $\overline{R}$ are evaluated directly by the OF (i.e. $eval(\overline{R}) = f(\overline{R})$). Evaluation of the unfeasible search – points $\overline{S}$, is done as follows: system selects one of the reference points, say $\overline{R}$ (better individuals have better chances to be selected); new random point $\overline{Z}$ from a segment between $\overline{S}$ and $\overline{R}$ is created [3]:

$$\overline{Z} = \alpha\overline{S} + (1-\alpha)\overline{R} \qquad (3)$$

where α is a random number in the range (0, 1). Generation process continues until a feasible $\overline{Z}$ is found. Once $\overline{Z}$ is found, individual $\overline{S}$ is evaluated:

$$eval(\overline{S}) = eval(\overline{Z}) = f(\overline{Z}) \qquad (4)$$

If $f(\overline{Z})$ is better than $f(\overline{R})$, then the point $\overline{Z}$ replaces $\overline{R}$ as a new reference point. Additionally, $\overline{Z}$ replaces $\overline{S}$ with some probability of replacement p_r.

Only search-points are subject to genetic operators. Best search-points are moved to reference population.

Some reference points are moved into the population of search-points, where they undergo transformation.

PARALLEL GENETIC ALGORITHMS

Classification of parallel GAs

The basic idea behind most parallel programs is to divide a big task in many small subtasks, which are to be solved simultaneously using multiple processors. This approach (known as "divide and conquer") is applied in GAs in many different ways. There are three main types of parallel GAs:

1) single-population master-slave GAs – they use a single population (like the serial GAs), but the evaluation of each individual's fitness (OF) is distributed among several processors. Because the fitness of an individual is independent from the rest of the population, during this phase there is no need of communication between processors. Communication occurs only when slave processors receive subset of individuals to evaluate, and when the slaves return the fitness values.

Master-slave algorithms can be two main types: synchronous and asynchronous. In synchronous algorithm, master processor stops and waits to receive solutions of all the population. After that, genetic operators are being applied to all the population. Synchronous master-slave GA has exactly the same properties as serial GA. The only difference is speed. In asynchronous implementation of GA, algorithm does not stop to wait any slow processors, but it does not work exactly as a classical GA.

2) single-population fine-grained GA – they consists of one spatially-structured population. Selection and recombination of the individuals is restricted to small neighbourhood, but neighbourhoods overlap, permitting interaction among all individuals. Usually individuals are placed in the nodes of a 2-D grid and can interact with surrounding 4 neighbours. Analysis showed that performance of these algorithms degraded as the size of the neighbourhood is increased [4].

3) multiple-population coarse-grained GA – they are more sophisticated, as they consists of several subpopulations (deme), which evolve independently and exchange individuals occasionally. This exchange of individuals is called *migration*. As the size of the demes is small, they converges faster. On the other hand, the quality of the solution might be poorer, because different areas of search space are examined. Very important factor for the quality of the solution is the migration rate. With a low migration rate, the demes evolved independently. Observations indicate, that there is a critical migration rate, below which the found solution is not satisfactory, and above which the solution is similar to serial GA's one [4].

There is another class parallel GAs – hierarchical parallel GAs – which combines previous ones. At a higher level, they are multiple-population algorithms with single population parallel GAs (either master-slave or fine-grained) at the lower level. Hierarchical parallel GAs combine the benefits of its components and it promises better performance than any of them alone.

Parallel Genocop

The presented client-server optimization system uses specially developed parallel version of the genetic algorithm Genocop III.

Main features and improvements in Parallel Genocop:

- The number of computation of the OF is reduced several times. The evaluation of the OF in arithmetic test cases takes milliseconds. Several hundreds computations more would have taken a few seconds. But in real practical problems, as is discussed in this paper DC motor, evaluation of the OF takes minutes and the slow down would have been several hours. Therefore it is very important to minimize the number of OF computations. The original Genocop algorithm is improved to avoid second (or multiple) evaluation of the problem with same input parameters.
- ε-condition stopping criteria is added. This allows algorithm to be stopped before achieving maximum generation number if the difference between best solutions of two consecutive generations is small enough.
- Ability of setting output parameters constraints is added. Practical optimization tasks demand it. In the test case presented below, minimum permanent magnet volume is obtained, subject to minimal torque constraint.
- All solutions (from all generations) are kept in a database and can be used later for studying each input parameter's influence over the obtained result.
- Evolution process can be inspected – each population's best solution is saved.
- Evaluation of individuals fitness (OF) is done simultaneously on several computers (clients). Number of clients is not fixed and can be changed without affecting optimization process.
- Intelligent restart - computational process can be interrupted at any time. Parallel Genocop is developed, so that all algorithm parameters are saved on each iteration. In this way algorithm can continue its normal processing from the same step, at which it was stopped. This is very important feature, because optimization could take many hours, even days, and restarting of the algorithm (for example, by interrupted power supply) would have caused a significant computation time loss.
- Ability to change the stopping criteria during the algorithm's run, is provided. This allows user to continue evolution process, without restarting the algorithm.
- Using CAD system for evaluation of the OF - CAD systems allow modelling (2D and 3D) of real devices and their evaluation with high accuracy. This avoids OF approximation (with Response Surface Modelling, Radial Basis Functions, Kriging models etc.) and the inevitable approximation errors.
- First generation can be initialized with randomly generated values (as in a classical algorithm) or with user-defined initial values.

CLIENT-SERVER ENVIRONMENT

Main features

The presented client-server system is a multi-thread MDI Windows application, developed with the modern object oriented language C#. Application structure is given on Figure 2:

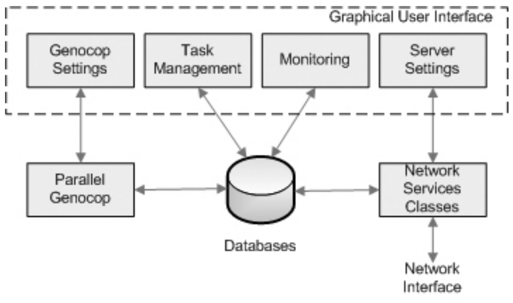

Fig. 2 - *Application structure*

The application has a convenient graphical user interface, which allows new tasks to be defined and existing ones to be edited. All input data is stored in MS Access databases. Integrated parallel Genocop also uses databases for saving its current state.

The application allows multiple tasks to be processed simultaneously. Each task has its individual settings for the GA. Three-level priority system is implemented (low, normal and high priority), which is based on circular discipline (to avoid infinite delay of low-level tasks).

Two types of problems can be solved: optimization problems (finding an~~d~~ optimum using GA) and problems in which a walk through all nodes of n-dimensional mesh in n-dimensional search space is performed. The second type allows input parameters to vary in given intervals with predefined step. All possible input parameter combinations are generated and solved.

The application displays each task's progress. There is an indication of current server state (number of connected clients, running tasks etc.). The system can also display some client information (computer name, username, OS, IP Address etc.). There is a statistics for the number of solved problems by each client.

User interface allows setting of classes, which provide network communication (IP Address, Port etc.).

Communication protocol

Especially for the application, a high-level protocol, based on TCP and Windows Sockets (in blocking mode), is developed. It provides number of confirmed services:

- file exchange
- starting an application on client's computer (for starting the CAD system, which evaluate the OF)
- system functions (establish and terminate a connection, authentification etc..)

Availability of own protocol makes unnecessary using other products and libraries (like MPI, PVM etc..) for communication purposes. This facilitate the application's installation and configuration (no other products are installed or tuned).

Client-server interactions follow the algorithm, given on Figure 3.

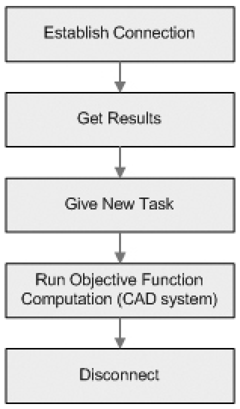

Fig. 3 – *Protocol algorithm*

1. Connection is established on client's demand, after client had finished his task (or if he still does not have one).
2. The server gets the results from the OF evaluation as a file (usually text file). This step is skipped, if the client has not received task yet.
3. The server gives a task to the client. If the task is new for the client, all necessary files are sent (script files, input files etc.). If the client evaluates just another individual from the population, only input file is sent.
4. Server sends a command to start computation of the OF (start the CAD system with the appropriate input parameters and scripts).
5. Connection is terminated.

This implementation of the communication protocol has the following advantages:

- Only sever needs to have real static IP-address. Clients could have dynamic and masked IP-addresses. The advantage of this implementation is, that the server could handle every client's computer, that has internet connection.
- Very good performance – gathering of the results and setting a new task is accomplished with the exchange of very short files (less than 1 KB) and commands. The whole process consists of transmitting several bytes, which is done in milliseconds.
- During the process of evaluating the OF, clients and server are not connected to each other. This leads to very low network usage and saves system resources.

PROBLEM FORMULATION

The presented optimization problem is taken from [1], where an optimization with RSM and evolutionary algorithm is applied.

The optimized machine is a small two-pole permanent magnet DC machine with a rated power output of 120 W at 2400 rpm, supply voltage 12 V and rated current 17.11 A. The ferrite permanent magnets have a segmental shape. The rotor has 12 slots. The winding consists of two parallel paths [1].

The purpose of the optimization is to find minimal permanent magnet volume, subject to following constraints (Table 1):

Table 1

Output constraints

Torque	$T \geq 0.44$ N.m
Stator Radius	$R_{out} < 40$ mm
k_{fill}	$k_{fill} < 0.35$

$$k_{fill} = \frac{S_c.nc}{S_{sl}} \tag{4}$$

S_c - conductor area; S_{sl} - total slot area

The following input parameters (shown on fig. 4) are subject to optimization (Table 2):

Table 2

Design variables and limits

Var.	Unit	Lower	Upper	Description
dy	mm	4.5	8	yoke thickness
hm	mm	5	9	PM height
Rr	mm	22	27	rotor outer radius
tw	mm	2.5	4.5	tooth width
L	mm	50	60	machine length
nc	-	13	19	conductors in a slot

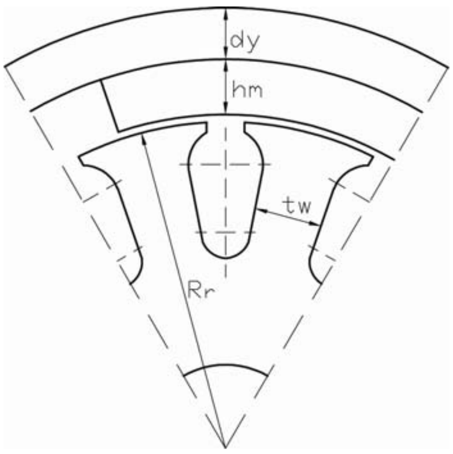

Fig. 4 – *Input parameters*

Experiment conditions

Experimental work is accomplished in a computer laboratory on 16 identical computers, PC type IA32, connected to 100 Mbit/s Ethernet network. Configuration of client computers is given is Table 3:

Table 3

Client computer configuration

Motherboard	K7 Triton GB-7N400E
CPU	AMD Athlon XP 2500+
RAM	512 MB DDR 400 MHz
Video card	ATI Radeon 9600 – 128 MB
HDD	80 GB
OS	Windows XP Pro SP2

As a server is used a laptop ASUS A3E-5003 with following configuration (Table 4):

Table 4

Server Configuration

CPU	Pentium-M 730 1.6 GHz
RAM	512 MB DDR
HDD	60 GB
OS	Windows XP Pro SP2

Population size is 35. Maximum number of generations is 100. Probability of applying a genetic operator is 5.71 % (same for all operators). Random generated initial population is used.

Optimization results

Optimal solution is found at the 38-th generation. GA was not stopped and went through all 100 generations for 3h 15 min 11 sec. Average time for evaluation of one individual is 24.66 sec. It is important to say, that time for evaluation is not constant and strongly depends on input parameters (time vary from 13 to 67 sec).

The same computations took approximately 48h when performed on a single computer. Parallel processing of 16 PCs reduced computation time about 14.8 times.

Optimization results for the objective function (permanent magnet volume) and torque are given in Table 5.

Table 5

Optimization results

Var.	Unit	Initial	Optim. [1]	Optim.
Torque	N.m	0.393	0.363	0.487
Volume	m^3	5.62E-5	3.55E-5	2.89E-5
k_{fill}	-	0.39	0.36	0.27

Resulting permanent magnet volume is reduced by 48.58 %. Motor torque is increased by 23.92 %. The obtained results are better than the results in [1]. This is explained with the not very accurate quadratic polynomial approximation of the objective function used in [1]. Besides, in the presented case here, there are no constraints for $H_{c\max}$, output power (P) and efficiency (η), used before in [1].

The initial values of the original device's input parameters, optimal values, obtained in [1] and optimal values, obtained by the presented client-server implementation of parallel GA are summarized in Table 6.

Table 6

Values of the design variables

Var.	Unit	Initial	Optim. [1]	Optim.
dy	mm	4.95	7.45	7.01
hm	mm	8.0	5.22	5.06
Rr	mm	26.35	22.88	22.07
tw	mm	4.0	2.5	4.03
L	mm	50.0	57.44	50.02
nc	-	18	14	19

The magnetic field distributions in the initial and the optimized motor are shown on fig. 5 and 6.

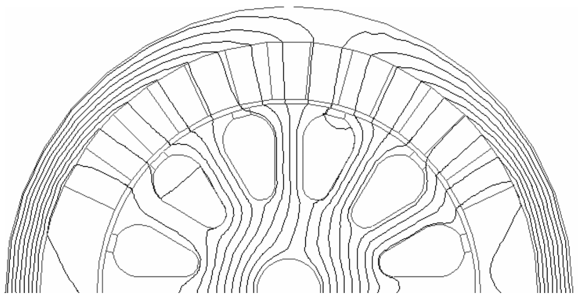

Fig. 5 – *The initial motor*

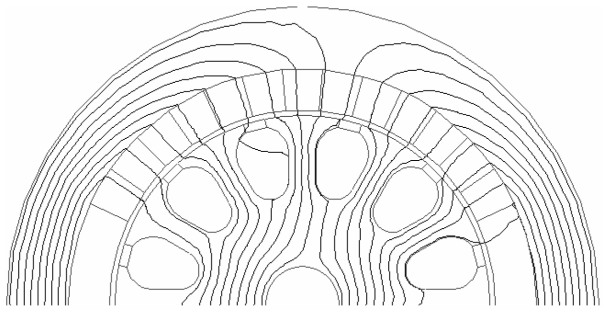

Fig. 6 – *The optimized motor*

CONCLUSION

The presented client-server environment with integrated parallel genetic algorithm gives possibility practical optimization problems to be solved by CAD system simulations. The discussed software system can be applied also in artificial neural network training, in design of the experiment method, etc.

In the problem presented here the CAD system FEMM v3.4 is used, but the application allows using other CAD systems also (MagNet, ANSYS, etc.) for objective function evaluation. CAD systems, due to integrated finite element method, provide very high accuracy of the solution.

A disadvantage of using CAD systems is that all possible combinations of input parameters inside the predefined limits must lead to physically possible models. Otherwise, the CAD system is not able to create and simulate the model.

The parallel processing significantly decreases computation time, necessary for obtaining the optimal solution. Its significance is even greater when CAD systems, performing 3D modelling and simulation are used (for example MagNet), because in these cases the computation of the objective function might take hours.

The main idea on which the presented software is based, is to use idle computational resources. In educational institutions many available computers stand idle during the night and holidays. The developped client-server system allows these computational resources to be combined in a computational cluster and used for scientific purposes.

REFERENCES

[1] K. Brandisky, R. Belmans, U. Pahner: "Optimization of a segmental PM DC motor using FEA, statistical experiment design method and evolutionary strategy", SPEEDAM Symposium, 1994, Taormina, Italy.

[2] Z. Michalewicz: "Evolutionary computation techniques for nonlinear programming problems", Evolutionary Computation, 4: 1—32, 1996.

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

[4] Z. Michalewicz, G. Nazhiyath: "Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints", Proceedings of the Second IEEE International Conference on Evolutionary Computation, pp. 647-651, IEEE Press, 1995.

[5] E. Cantu-Paz: "A survey of parallel genetic algorithms", Tech. Rep., The University of Illinois, 1997, IlliGAL Report No. 97003.

[6] D. C. Meeker, Finite Element Method Magnetics, Version 3.4 (http://femm.foster-miller.net/).

Petar Ivanov was born on 04.10.1983 in Varna, Bulgaria. In 2002 he graduated Natural Science and Mathematics High School with excellent marks. In 2006 he is going to receive Bachelor degree in Computer Systems and Technologies at Technical University of Sofia. He is Scholarship student of DAAD.

He won second place in National Competition in Theoretical Electrical Engineering, Varna 2004. Some of his interests are: Programming, Electronics, CAD systems.