Structural switched reluctance motor optimization using the genetic algorithm

Natan Tzvi Shaked and Raul Rabinovici

Department of Electrical and Computer Engineering
Ben Gurion University of the Negev, Beer-Sheva, Israel
natis@ee.bgu.ac.il and rr@ee.bgu.ac.il

Abstract — **This paper uses the Genetic Algorithm (GA) in order to make a structural optimization of the switched reluctance motor (SRM). The objective function in this optimization is chosen to minimize the outer volume of the motor, or in other words, to maximize its output power. This paper explains the optimization procedure step by step and shows the results and the conclusions of a simulation which was done in order to check this procedure.**

Index Terms **— Switched Reluctance Motor, Structural Optimization, Genetic Algorithm.**

I. Introduction

Although the basic principles of the switched reluctance motor (SRM) are well documented in the literature [1-5], only few articles have been written about the optimization of this motor, and even less use evolutionary methods for this optimization process. The Genetic Algorithm (GA) is a powerful optimization technique and is one of the most widely known methods for evolutionary computation today [6-12]. The Genetic Algorithm has many advantages: parallel ability of working on a population of points, finding the global optimum with higher probability (rather than the local one), ability of handling discrete parameters, and reasonable computation time. This paper presents the adaptation of the Genetic Algorithm for structural optimization of the SRM. Later paper [13] is going to use the Genetic Algorithm in order to minimize the torque ripple of this motor.

II. Motor Design Optimization Principles

We can look at the optimization of electrical machinery as a typical problem of nonlinear programming [14]. We have to find n variables $X=(x_1,x_2,\ldots,x_n)$ which minimize or maximize a scalar function $F(X)$ under m scalar conditions: $G(X)=\left(g_1(X),g_2(X),\ldots,g_m(X)\right)$. The above described X vector contains n independent variables which could be electromagnetic or geometric quantities. Common examples for these independent variables may be: the air gap diameter, height and width of stator and rotor teeth, stator and rotor radiuses, etc. The scalar function $F(X)$ is called the objective function (since the minimization or the maximization of this function is the objective of the optimization). Common examples for the objective function may be the cost of the machine, the output power of the machine, etc. The $G(X)$ vector contains m scalar conditions $g_i(X)\in G(X)$, $i=1...m$ which are frequently called constraints. Common examples for constraints may be starting torque, geometrical or electrical limitations, etc.

III. Genetic Algorithm Principles

The Genetic Algorithm was introduced by John Holland form the University of Michigan. Holland's goal was formally study of the adaptive processes in nature and the development of similar methods for computer systems. In his book [6] (which was originally published in 1975), Holland presented the Genetic Algorithm framework. With similarities to biological evolution, the GA describes a population of chromosomes which are strings. These chromosomes are composed of genes which are (traditionally) bits (0s or 1s). In each iteration, the population moves from its current form into a new form by a kind of natural selection, which is realized by the GA operators (crossovers and mutations), and by the selection function. Fig. 1 shows a basic version of the Genetic Algorithm. As it can be seen form this figure, after the generation number t and the population $Pop(t)$ are initialized, an evaluation of this population is done by using the fitness (or evaluation) function. Next, there is a loop which continues until the termination condition is satisfied. Inside of this loop, a recombination of the population $Pop(t)$ is done in order to create the offspring $C(t)$ of the t'th generation. This is done using the GA operators which are crossover and mutation. An evaluation of the offspring is done, and then the selection function selects the individuals for the next generation, $t+1$, which yields the new population, $Pop(t+1)$. This process continues until the algorithm converges on the 'best' solution, which is hopefully the optimal solution of the problem. The termination condition is the indication of when we should stop the optimization and declare the 'best' solution. As we use the Genetic Algorithm for an optimization problem, we will refer to it from now on in the context of an optimization problem.

A. The GA Operators

There are mainly two kinds of GA operators: crossover operators and mutation operators. Crossover takes two individuals from the population and produces two new individuals. Mutation takes a single individual

```
Genetic Algorithm:
begin
 t = 0;
 initialize Pop(t);
 evaluate Pop(t);
 while (not terminate condition) do
 begin
  recombine Pop(t) to yield C(t);
  evaluate C(t);
  select Pop(t+1) from Pop(t) and C(t);
  t = t + 1;
 end
end
```

Fig. 1. General structure of the Genetic Algorithm

and alters it. The application of these two types of operators depends of the encoding method we use for the chromosomes. The encoding methods can be classified as follow: binary encoding, real-number encoding, integer of literal permutation encoding, and general data structure encoding. Real-number encoding is best used for function optimization problem [11]. Three examples for each kind of these operators are shown next.

Let P_1 and P_2 be two vectors denoting two individuals (patents) taken form the population. Let C_1 and C_2 be the new individuals (children) which are “born” using P_1 and P_2. **Simple crossover** generates a random number α from a uniform distribution from 1 to the number of genes in the chromosome and produces (using the parents) two new individuals according to the following equations:

$$c_{1i} = \begin{cases} p_{1i} & \text{: if } i < \alpha \\ p_{2i} & \text{: otherwise} \end{cases} ; c_{2i} = \begin{cases} p_{2i} & \text{: if } i < \alpha \\ p_{1i} & \text{: otherwise} \end{cases} \qquad (1)$$

where p_{1i}, p_{2i}, c_{1i}, and c_{2i} are the i 'th gene of the first parent, the second parent, the first child, and the second child respectively. **Arithmetic crossover** creates two complimentary linear combinations according to the following equations:

$$\begin{aligned} C_1 &= \alpha \cdot P_1 + (1-\alpha) \cdot P_2 \\ C_2 &= (1-\alpha) \cdot P_1 + \alpha \cdot P_2 \end{aligned} \qquad (2)$$

where $\alpha \sim U(0,1)$. Fig. 2 shows an example for the arithmetic crossover. **Heuristic crossover** utilizes some fitness information: it uses the fitness values of the two parents in order to determine the direction of the search. The offspring are created from of the parents according to the following equations:

$$C_1 = B + \alpha(B - W); C_2 = B \qquad (3)$$

where: $B = best(P_1, P_2)$, $W = worst(P_1, P_2)$, $\alpha \sim U(0,1)$. If α is chosen such that one or more of its genes fall outside of the allowable upper or lower bounds, it is possible that C_1 will not be feasible. That is why heuristic crossover has a parameter that may be set by the user. This parameter is the number of times of trying to find α that results in a feasible chromosome. If a feasible chromosome is not produced this number of tries, the worst of the two parents, W, is set to be C_1. **Boundary mutation** replaces the value of the randomly chosen gene with either the upper or lower boundary of the gene (chosen randomly). The idea behind this operator is that the global solution for many optimization problems usually lies on the boundaries of the feasible region, so it may be beneficial to search there. **Uniform mutation** replaces the value of the randomly chosen gene with a uniform random value selected between the upper and the lower boundaries of this gene. Fig. 3 shows an example for the uniform mutation. **Non-uniform mutation** increases the probability that the amount of the mutations will be close to zero as the generation number increases. This kind of mutation is designed for fine-tuning capabilities and for achieving high precision. Remembering that in mutation there is only one parent and one child, let p_k be the k ’th gene of the parent and the selected gene for the mutation (so $P = (p_1, p_2, \ldots p_k, \ldots, p_h)$), and let c_k be the k ’th gene of the child of this parent (so $C = (p_1, p_2, \ldots c_k, \ldots, p_h)$). This gene is randomly selected from the following two choices:

$$\begin{aligned} c_k &= p_k + f(t, HighBoundary[p_k] - p_k) \\ c_k &= p_k - f(t, p_k - LowBoundary[p_k]) \end{aligned} \qquad (4)$$

where the function f is defined as follows:

$$f(t, \gamma) = \gamma \cdot a(1 - t / MaxGen)^b \qquad (5)$$

where a is a random number from [0,1], b is a parameter determining the degree of the non-uniformity, and $MaxGen$ is the generation maximum number. The function $f(t,\gamma)$ returns a value in the range $[0,\gamma]$ such that this value approaches zero as the generation number t increases. If the offspring is not feasible, we reduce the value of a until it is.

B. Selection Function

The selection function plays an important role in the Genetic Algorithm: it chooses which individuals are

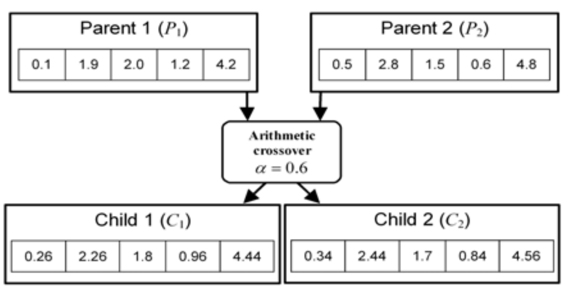


Fig. 2. Crossover Example: arithmetic crossover

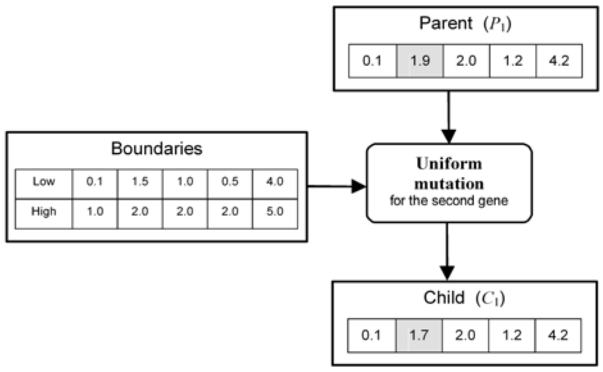


Fig. 3. Mutation Example: uniform mutation

going to 'live' (be in the next generation) and which individuals are going to 'die' (not be in the next generations). Like the Darwinian selection, better individuals have better chances to survive (being selected). Usually, the selection function assigns the k'th individual a selection probability of $Prob_k$ based on its fitness value. Then, a series of random numbers is generated and compared against the cumulative probability of the population which is defined by:

$$CProb_i = \sum_{k=1}^{i} Prob_k \tag{6}$$

If $CProb_i < U(0,1) < CProb_{i+1}$ then the i'th individual is selected for being in the next generation. Most of the selection methods are different form each other due to the different assignment of selection probabilities. Three examples of typical selection functions are shown next.

Roulette wheel selection determines the selection probability for each chromosome as proportional to the fitness value. This method is based of spinning an imaginary "roulette wheel" the number of times equal to the size of the population. In this case, the probability $Prob_k$ for each individual is defined by:

$$Prob_k = \frac{FitnessValue_k}{\sum_{j=1}^{N_{Pop}} FitnessValue_j} \tag{7}$$

where N_{pop} is the population size. **Normalized geometric selection** is a type of a ranking method. It assigns the probability $Prob_k$ based on the rank of the k'th solution after sorting all solutions according to:

$$Prob_k = \frac{q}{1-(1-q)^{N_{pop}}}(1-q)^{\rho-1} \tag{8}$$

where q is the probability of selecting the best individual, ρ is the rank of the individual (1 is the best). **Tournament selection** is different from the above described selection methods since according to this method we do not have to assign selection probabilities, all we have to do is selecting k individuals randomly (with replacements), and to choose the best of them (using the fitness function) to be in the new population. We repeat this process until we have the same number of individuals in the new population as we had in old one.

C. Termination Function

The termination function determines when the Genetic Algorithm has to stop running and the optimal solution has to be declared. The common reasons for that may be: when there is no improvement in the best solution in a specified number of generations, when the sum of deviations among individuals becomes smaller than a known threshold, or when a specified maximum number of generations is reached.

D. Fitness Function and Constraints

It has already been explained that in the evaluation stage each individual gets a fitness value which determines its chances of being in the next generation. In an unconstrained optimization problem this is usually done by the objective function $F(X)$ which its minimization or maximization is the objective of the optimization problem. In a constrained optimization problem, the constraint vector $G(X)$ has to be taken into consideration. This is going to affect the original objective function in such a way than if any of constraints is not satisfied for one of the individuals, its fitness value is going to be so bad, that this individual will probably not going to be chosen for the next generation. This can be done by penalizing the objective function for these cases, which yields the final fitness function $E(X)$. This technique transforms the constrained optimization problem into unconstrained problem. One way to do this is by adding a penalty term to the objective function:

$$E(X) = F(X) + PenaltyTerm(X) \tag{9}$$

The structure of $PenaltyTerm$ has to be such that if the individual that is checked, is feasible - we have to get $PenaltyTerm = 0$, which means that $E(X) = F(X)$. This can be done by choosing the $PenaltyTerm$ to be:

$$PenaltyTerm(X) = r \cdot \sum_{k=1}^{m} W_k \max\left(g_k(X),0\right)^{\beta} \tag{10}$$

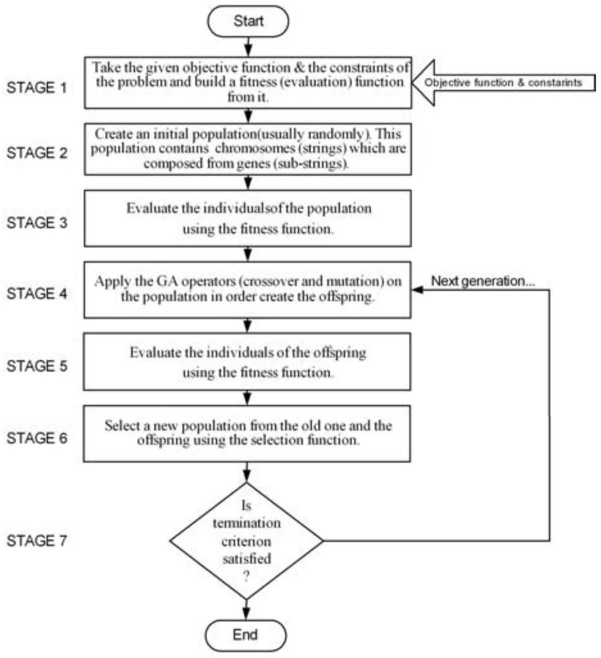
where m is the number of constrains in the $G(X)$ vector [$g_i(X) \in G(X)$], W_k's are the weighting factors, and r and β are the penalty factors. The above equation assumes that we want to solve a minimization task and that the constraints are given in the following forms (one of them or both of them):

$$g_i = LowBoundaryValue - ActualValue \tag{11}$$

$$g_i = ActualValue - HighBoundaryValue \tag{12}$$

So if the actual value of the variable is bigger than the low boundary value for (11) and the actual value of the variable is smaller than the high boundary value for

(12), we are inside of this variable boundaries. In this case, we get $g_i < 0$, so penalty term is equal to zero and the fitness function is equal to the objective function. The other case is when the actual value is smaller than the low boundary value for (11) or higher than the high boundary value for (12). In this case, we get $g_i > 0$, so the penalty term is unequal to zero and the fitness function are unequal to the objective function. The penalty factors r and β should be adjusted to make a sensible ratio between the objective function and the penalty term. These factors can be obtained by a simple trail and error procedure. Too high penalty factors mean a fast but wrong convergence, whereas too small penalty factors mean a very slow convergence. The weighting factor of the k'th constraint should be higher compared to the rest of the weighting factors if we want to give the condition of not satisfying this constraint a higher effect on the penalty term. Fig. 4 shows the overall 7-stages GA including the fitness function building stage.


Start
STAGE 1
Take the given objective function & the constraints of the problem and build a fitness (evaluation) function from it.
Objective function & constarints
STAGE 2
Create an initial population(usually randomly). This population contains chromosomes (strings) which are composed from genes (sub-strings).
STAGE 3
Evaluate the individualsof the population using the fitness function.
STAGE 4
Apply the GA operators (crossover and mutation) on the population in order create the offspring.
Next generation...
STAGE 5
Evaluate the individuals of the offspring using the fitness function.
STAGE 6
Select a new population from the old one and the offspring using the selection function.
STAGE 7
Is termination criterion satisfied ?
End
Fig. 4. 7-stages constrained GA optimization

IV. Structural Motor Optimization via Genetic Algorithm

Until now we have interpreted the Genetic Algorithm in the general meaning of optimization problems, from now on we are going to interpret it in the meaning of structural motor optimization problems [14-17]. In this case, the population contains individuals which each one of them is a motor design. Each motor design (chromosome) is characterized by a set of motor design variables $x_i \in X$ (genes). The constraints $g_i \in G(X)$ are usually geometric or electromagnetic limitations. The objective function $F(X)$ is some desirable goal for the resultant motor.

V. Optimization of SRM via Genetic Algorithm

The switched reluctance motor of [17] has been chosen in order to check the above described structural motor optimization. This is a 60KW, 231V, 3 phases, 4700 rpm, air cooled SRM. The fixed parameters of this motor are: maximum current density 10007×10^3 [A/m^2], shaft diameter $D_{sh} = 0.039$ [m], maximum flux density $B_{\max} = 2.094$ [T], and switching frequency $f_1 = 940$ [Hz].

A. SRM Objective Function

The objective function was chosen to minimize the outer volume of the motor [17-18]:

$$MotorOuterVolume = \pi \cdot (D_o/2)^2 \cdot L_{env} \tag{13}$$

where D_o is the machine outer diameter, and L_{env} is the envelope length of the machine. Let sr be the split ratio of the rotor outer diameter and the machine outer diameter:

$$sr = D_r / D_o \tag{14}$$

where D_r is the rotor outer diameter. Using (14) in (13) gives:

$$MotorOuterVolume = 1/4 \cdot \pi \cdot (D_r/sr)^2 \cdot L_{env} \tag{15}$$

The envelope length, L_{env}, may be estimated as the stack length, L_{stk}, plus two turn overhangs [18]. The overhang length can be estimated as 1/4 of the stator pole pitch, so the envelope length can be expressed as:

$$L_{env} \cong L_{stk} + 1/2 \cdot \pi \cdot D_r / N_s \tag{16}$$

So we can rewrite (15) as follows:

$$\begin{aligned} &MotorOuterVolume \cong \\ &1/4 \cdot \pi \cdot (D_r/sr)^2 \cdot (L_{stk} + 1/2 \cdot \pi \cdot D_r / N_s) \end{aligned} \tag{17}$$

The goal of the optimization process is to find the motor which has the minimal volume depending on some specific parameters. In our case, these parameters are going to be D_r, sr, and L_{stk}. In other words:

$$x_1 = D_r \quad ; \quad x_2 = L_{stk} \quad ; \quad x_3 = sr \tag{18}$$

We want to find $X = (x_1, x_2, x_3)$ which brings the motor to its minimal volume. This means that in our case, the objective function is:

$$F(X) = MachineOuterVolume \tag{19}$$

Using (18) and (19) in (17) gives:

$$F(X) = \frac{1}{4} \cdot \pi \cdot \left(\frac{x_1}{x_3}\right)^2 \cdot \left(x_2 + \frac{\pi}{2N_s} \cdot x_1\right) \tag{20}$$

B. SRM Optimization Constraints

The above described general optimization problem was: “Find n variables $X = (x_1, x_2, \ldots, x_n)$ which optimize a scalar function $F(X)$ under m scalar conditions $G(X) = (g_1(X), g_2(X), \ldots, g_m(X))$." In our case $n = 3$ and the constraints ("scalar conditions") are:

- **Motor aspect ratio:**
  $AR = x_1 / x_2$, $0.3 \le AR \le 3$
- **Slot-fill factor** ff **[17]:**
  $ff \le ff_{\max}$, where $ff_{\max}$ is the maximum slot-fill factor which depends on the type of the coil and the amount of insulation.
- **Steady state average torque** T_{avg} **:**
  $T_{avg} \ge \mathrm{K} \cdot T_o$, where T_o is the full load output torque and K is the overload factor.
- **Efficiency** η **:**
  $\eta_{\min} \le \eta \le \eta_{\max}$, where $\eta_{\min}$ and $\eta_{\max}$ are the maximum and minimum allowable efficiencies respectively. ($\eta = P_{shaft} / (P_{shaft} + P_{losses})$ [18], where P_{shaft} is the machine shaft power, and P_{losses} are the losses in the machine and in the converter).
- **Maximum generalized power factor** PF_m [5]**:**
  $0.55 \le PF_m \le 0.9$

and three domain constraints (which restrict the search into the practical values and preserve a reasonable computation time):

- **Rotor outer diameter:**
  $0.05 \le D_r \le 0.5$
- **Rotor axial length:**
  $0.05 \le L_{stk} \le 0.5$
- **Split ratio of the rotor outer diameter and the motor outer diameter:**
  $0.5 \le sr \le 0.63$

As it can be seen from Fig. 4, the objective function and the constraints are input into "STAGE 1" box and the fitness function is built using them. Before we explain how it is done, we have to understand that in order to build the fitness function, the constraints, which were defined above, have to be expressed in (11) and (12) forms. Applying this rules in our optimization problem, we should write the above described constraints as follows:

$$\begin{aligned}
&g_1 = AR - 3,\ g_2 = 0.3 - AR;\ g_3 = ff - ff_{\max};\\
&g_4 = \mathrm{K} \cdot T_o - T_{avg},\ g_5 = \eta - \eta_{\max};\ g_6 = \eta_{\min} - \eta;\\
&g_7 = PF_m - 0.9,\ g_8 = 0.55 - PF_m;\\
&g_9 = D_r - 0.5,\ g_{10} = 0.05 - D_r;\\
&g_{11} = L_{stk} - 0.5,\ g_{12} = 0.05 - L_{stk};\\
&g_{13} = sr - 0.63\ ,\ g_{14} = 0.5 - sr
\end{aligned} \tag{21}$$

C. 7-stage Genetic Algorithm Optimization

Using (20) and (21), the fitness function in our case can be written as:

$$\begin{aligned}
E(X) = \frac{1}{4} \cdot \pi \cdot \left(\frac{x_1}{x_3}\right)^2 \cdot \left(x_2 + \frac{\pi}{2N_s} \cdot x_1\right)\\
+ r \cdot \sum_{k=1}^{14} W_k \cdot [\max(g_k, 0)]^{\beta}
\end{aligned} \tag{22}$$

This is actually "STAGE 1" of Fig. 4. In order to perform "STAGE 2" of this figure we may randomize a set of SRM designs. This set is the initial population and it contains N_{pop} members. Since in our case, the X vector contains three variables (x_1, x_2 and x_3), all we have to do is to randomize N_{pop} values for each one of these variables, and in that we actually "create" N_{pop} SRM designs. The evaluation of the initial population is done in "STAGE 3" of Fig. 4. If one or more constraints are not satisfied, the fitness value of the current SRM design will be penalized by adding something to its objective function, and get a higher "effective volume", which means that there are less chances for this design to be in the next generation of the population. After applying the Genetic Algorithm operators in order to create the offspring ("STAGE 4") and after evaluating the offspring ("STAGE 5"), a selection is performed so the better individuals are copied to the population of next generation ("STAGE 6"). Then, a termination criterion is checked ("STAGE 7"). If it is not satisfied then we go back to "STAGE 4", otherwise we end the optimization procedure and pronounce the best SRM design which has been found. The termination criterion we choose for our optimization is a maximum number of generations.

D. The Model of Simulation

The analytical model that is being used here is the one which was presented by Miller & McGlip [19]. The paper shows how to build the magnetization curves at aligned, unaligned, and intermediate positions based on the unaligned- and the aligned unsaturated-inductance. These inductances can be achieved from the structural quantities of the motor [1,25]. After using this analytical model, we can draw the magnetization curves (the flux-linkage as a function of the phase-current and the rotor position, $\psi(i,\theta)$) of the optimized SRM design which was chosen by the optimization procedure.

The voltage equation of the motor for one phase can be written as:

$$v = R \cdot i - \frac{\partial \psi}{\partial t} \tag{23}$$

where R is the phase resistance and v is the applied voltage which may be chosen as:

$$v = \begin{cases} V & \text{if } \theta_0 < \theta < \theta_c \\ -V & \text{if } \theta_c < \theta < \theta_q \\ 0 & \text{if } \quad \theta > \theta_q \end{cases} \tag{24}$$

where V is the dc supplied voltage, and θ_0, θ_c, θ_q are the turn-on, turn off and quenching angles respectively. The way of choosing these angels is explained in [2,3]. Under the assumption of constant velocity, (24) can be written as:

$$\frac{\partial \psi}{\partial \theta} = \frac{1}{\omega_m} \cdot (v - R \cdot i) \quad (25)$$

where ω_m is angular velocity of the rotor. Taking an integral from both sides of (25), we can express the flux linkage as:

$$\psi = \frac{1}{\omega_m} \int (v - R \cdot i) d\theta \quad (26)$$

In other words, we can say that:

$$\psi_{new} \approx \psi_{prev} + (v - R \cdot i) \cdot \frac{\Delta\theta}{\omega_m} \quad (27)$$

where ψ_{new} is the flux linkage in the current integration step, ψ_{prev} is the flux linkage in the previous integration step, and $\Delta\theta = \omega_m dt$ is the integration step length. The phase current i must be updated from the new value of the flux linkage ψ and the rotor position θ in each integration step. Following the procedure in [20], the phase-current i as a function of rotor position θ can be calculated. Usually, the other phase-currents have the same waveform, but phase-shifted by a step angle. After achieving the magnetization curves, $\psi(i,\theta)$, and the phase-current, $i(\theta)$, the coenergy and stored-energy can be calculated as follows:

$$W_f = \int i \, d\psi \quad (28)$$

$$W_c = \int \psi \, di \quad (29)$$

Then, the instantaneous electromagnetic torque can be calculated in the following ways:

$$T_e = \frac{\partial W_c(i,\theta)}{\partial \theta} = -\frac{\partial W_f(\psi,\theta)}{\partial \theta} \quad (30)$$

Using this equation, the torque of each one of the phases can be calculated and the sum of all torques from all phases gives the total torque produced by the motor. Then, the average torque can be calculated as well.

VI. RESULTS

The GA was applied for 50 generations which, as we shall see, was enough for converging on the global optimum. The objective function (which defines the target of the optimization) was chosen to minimize the outer volume of the motor (as stated in (13)). Table I shows the results of the optimization for various population sizes, number of crossovers, and number of mutations which were the inputs of the optimization procedure (together with the above mentioned fixed parameters). The outputs of the optimization procedure were the optimized variables (D_r, L_{stk}, and sr). From these optimized variables, the optimized outer volume of the motor can be calculated. Other motor parameters that can also be calculated are shown in Table II. Fig. 5 shows a trace of the best fitness function (for design 1) through the 50 generations of the optimization. As it can be seen from this figure, the rough converging of the Genetic Algorithm to the global optimum is very fast, but its "fine-tuning" may take much longer. From this figure, it can also be seen that 50 generations are enough for getting close enough to the global optimum. This fact is right almost for all designs in the range we checked, so 50 generations limit was chosen to be the termination condition of the GA. Another termination condition that may be chosen is the change in the fitness function of the current generation compared to the previous generation, but it was not necessary in our case. We have learned that if someone wants to use a maximum number of generations limit as the termination condition for the GA, he must try it first for a large number of generations and see where the change of the fitness function is small enough. Only then he can determine the maximum generations limit. If he will not do it, he might use too low maximum number of generations and stop the optimization process before reaching the global optimum, then he may "find" a wrong optimum. On the other hand, we should remember that too high maximum number of generations means unreasonable computation time. The maximum number of generations is only one of the parameters that may affect the truthfulness of the optimization process, the Genetic Algorithm may be affected from other parameters as well. Two of these parameters are the penalty factors r and β. As it was predicted: too small r and β cause a very slow converging, whereas too big r and β cause a fast, but wrong converging. A trail and error procedure has shown that $r = 1 \times 10^{-5}$ and $\beta = 2$ are good penalty factors in our case. The population size is another parameter that must be chosen correctly. Trying to do the optimization with less than 100 designs in the population, gave in some cases wrong results. This is the reason that in Table I we use with sizes of population which are bigger than 100. The probabilities of crossovers and mutations are also important: too high probability of crossovers may lead us to converging to a local optimum, and too high probability of mutations may lead us to a primitive random search. On the other hand, too low probability of crossovers or mutations may not be enough for the changing of population through the generations. We have found that an adequate mutations and crossovers percentage in our case may be 1-4% and 10-40% (of the population size) respectively. Fig. 6 shows the magnetization curves for one cycle of flux-linkage $\psi(i,\theta)$, which were found by the above mention model. From this curves and form the voltage equation of one phase, we can reach to the phase-current waveform (Fig. 7). Then, the coenergy and the stored-energy can be calculated. From the coenergy (or the stored-energy), the instantaneous torque can be found. Fig. 8 shows the dynamic torque curves of each one of the phases, whereas Fig. 9 shows the total torque and the average torque of the motor. Fig. 5-9 refer to design 1 of Tables I-II.

VII. Conclusion

This paper has explained the principles of the Genetic Algorithm and shown how we can adapt this algorithm for a constrained motor optimization. Then, it has shown how this can be done for the switched reluctance motor. An accurate step-by-step procedure of doing a structural optimization for the SRM was discussed and a simulation of this procedure was done in order to check it. We have found that the Genetic Algorithm is a fast and reliable algorithm for a structural SRM optimization, as long as its parameters (size of population, number of mutations and crossovers, etc) are properly configured. An important conclusion of the simulation we have done is that the first thing one should do, before making his final experiments with the Genetic Algorithm, is learning the problem in order he can configure these parameters in a proper way. This learning may take time (because for example we may have to try the Genetic Algorithm with a large number of generations in order to find a good termination condition), but we must remember this "learning stage" is very important to the truthfulness of the final result, which is, in our case, the optimal SRM design. Later paper [14] is going to deal with torque ripple minimization using the GA and is also going to compare the GA with a deterministic optimization method called the Simplex Method.

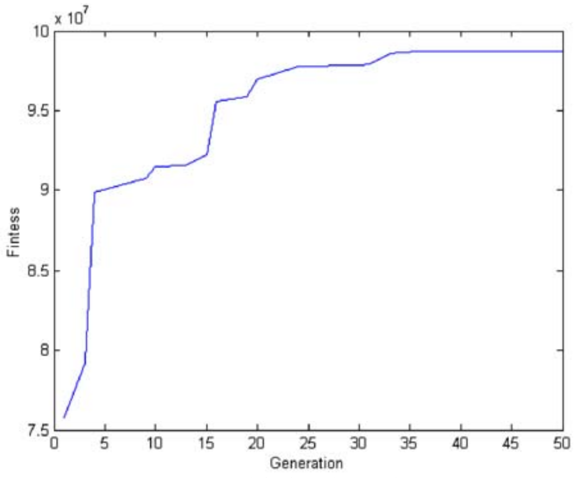


Fig. 5. Trace of the best fitness function

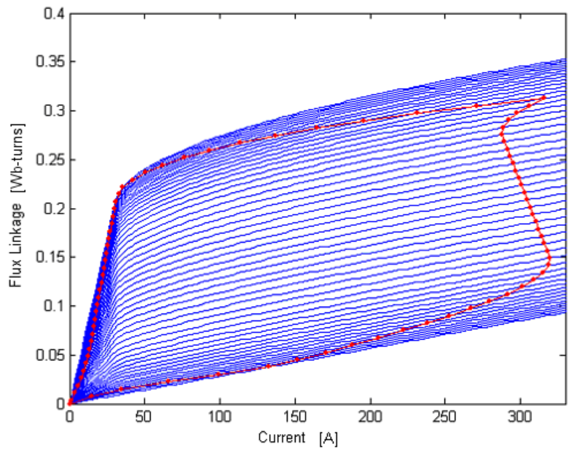


Fig. 6. Magnetization curves of the described motor

TABLE I
GA Optimization – Input Parameters, Out Parameter And Objective Function

	Parameter	Design 1	Design 2	Design 3	Design 4
Inputs	Population size N_{pop}	100	150	200	300
	Percentage of crossovers	20%	10%	20%	20%
	Percentage of mutations	2%	2%	2%	1%
Outputs	Rotor diameter D_r [m]	0.1398	0.1442	0.1372	0.1397
	Motor stack length L_{stk} [m]	0.2156	0.1942	0.2343	0.1922
	Split ratio $sr = D_r / D_0$	0.6287	0.6286	0.6291	0.6292
***F*(*X*)**	Motor output volume V_{out} [m^3]	0.0098	0.0096	0.0101	0.0089

TABLE II
GA Optimization – Calculated Parameters

	Parameter	Design 1	Design 2	Design 3	Design 4
Calculated parameters	Machine outer diameter D_o [m]	0.2224	0.2294	0.2181	0.2220
	Internal rotor diameter $D_{r\ \text{int}}$ [m]	0.0911	0.0927	0.0901	0.0910
	Internal stator diameter $D_{s\ \text{int}}$ [m]	0.1734	0.1789	0.1700	0.1731
	Stator diameter D_s [m]	0.1403	0.1447	0.1377	0.1402
	Average torque [Nm]	125.4586	125.221	124.954	125.961

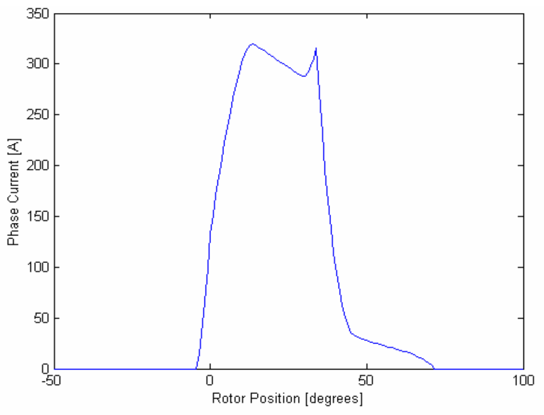


Fig. 7. Phase-current

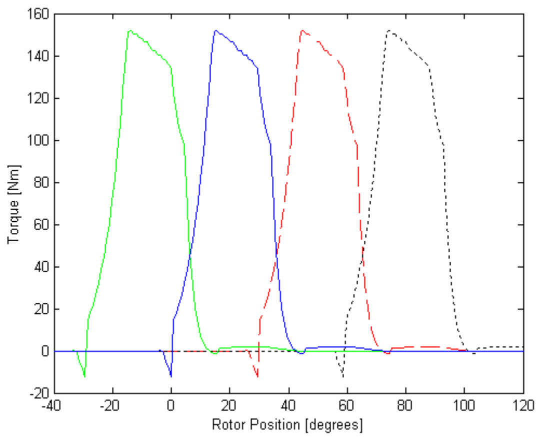


Fig. 8. Phase torques

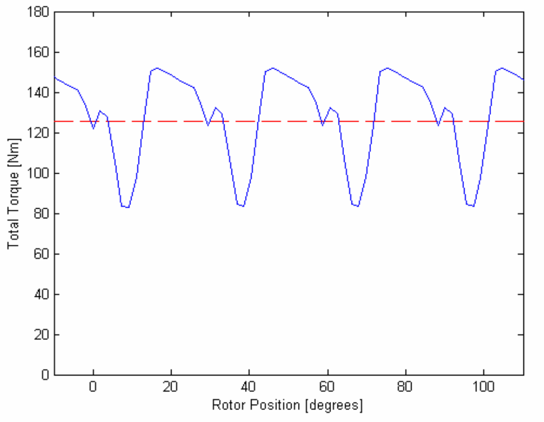


Fig. 9. Total torque and average torque

References

[1] R. Krishnan, *Switched reluctance motor drives: modeling, simulation, analysis, design, and applications*, CRC Press, 2001.

[2] T. J. E. Miller, *Switched reluctance motors and their control*, Magna Physics Publishing, 1993.

[3] T. J. E. Miller, *Electronic control of switched reluctance machines*, Newnes Power Engineering Series, 2001.

[4] T. J. E. Miller, *Brushless permanent-magnet and reluctance motor drives*, Ch. 7, Oxford University Press, 1989.

[5] I. Boldea, and S.A. Nasar, *Electric drives*, CRC Press, 1999.

[6] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT Press, 1992.

[7] P. Vas, *Artificial-intelligence-based electrical machines and drives: application of fuzzy, neural, fuzzy-neural, and genetic-algorithm-based techniques*, Oxford University Press, 1999.

[8] M. D. Vose, *The simple genetic algorithm: foundations and theory*, MIT Press, 1999.

[9] M. Mitchell, *An introduction to genetic algorithms*, MIT Press, 1996

[10] Y. Rahmat-Samii, and E. Michielssen, *Electromagnetic optimization by genetic algorithms*, Wiley, 1999.

[11] M. Gen, and R. cheng, *Genetic algorithms and engineering optimization*, Wiley, 2000.

[12] R. S. Zebulum, M. A. Pacheco, and M. M. B. R. Vellasco, *Evolutionary electronics: automatic design of electronic circuits and systems by genetic algorithms*, CRC Press international series on computational intelligence, 2002.

[13] N. T. Shaked, and R. Rabinovici, "Minimization of the torque ripple in switched reluctance motors by optimizing the phase-current profile," *Ninth International Conference on Optimization of Electrical and Electronic Equipment*, OPTIM'04, Brasov, Romania, May 20-22, 2004.

[14] M. Poloujadoff , M. Nurdin, and A. Faure, "Synthesis of squirrel cage motors, a key to optimization," *IEEE Trans. on Energy Conversion*, Vol. 6, No. 2, 1991.

[15] S. Palko, "Structural optimization of an induction motor using a genetic algorithm and finite element method," Acta Polytechnica Scandinavia, Electrical Engineering Series, No. 84, 1996.

[16] N. Bianchi, and S. Bolognani, "Design optimization of electric motors by genetic algorithms," *IEE Proc., Electr. Appl.*, Vol. 145, No. 5, pp 475-483, 1998.

[17] A. El-Wakeel, and A. C. Smith, "Optimal design of switched reluctance motors using the genetic algorithms," *International Conference on Electrical Machines*, 2002.

[18] A. El-Wakeel, S. A. Gawish, and M. A. L. Badr, "Systematic design procedure of switched reluctance motors," *Proceedings of the International Conference on Electrical Engineering*, 1999.

[19] T. J. E. Miller, and M. McGlip, "Nonlinear theory of the switched reluctance motor for rapid computer-aided design," *IEE Proc – Pt. B.*, Vol. 137, No. 6, pp 337-347, 1990.

[20] T. J. E. Miller, "Optimal design of switched reluctance motors, IEEE Trans. on Industrial electronics," Vol. 49, No. 1, pp 15-27, 2002.

[21] C. Roux, and M. M. Morcos, "On the use of simplified model for switched reluctance motor," *IEEE Trans. on Energy Conversion*, Vol. 17, pp 400-405, No. 3, 2002.

[22] R. Rabinovici, "Torque estimation for switched reluctance motors by on-line procedure," *Nonlinear Electromagnetic Systems*, pp 19-22, IOS Press, 1996.

[23] J. Faiz, and J. W. Finch, "Aspects of design optimization for switched reluctance motors," *IEEE Trans. on Energy Conversion*, Vol. 8, No. 4, 1993.

[24] P. J. Lawreson, J. M. Stephenson, J. Corda, and N. N. Fulton, "Variable speed switched reluctance motors," *IEE Proc. - Pt. B*, Vol. 127, No. 4, pp 253-265, 1980.

[25] A. V. Radun, "Design considerations for the switched reluctance motor,“ *IEEE Trans. on Industry Applications*, Vol. 31, No. 5, pp 1079-1087, 1995.

[26] P. N. Materu, and R. Krishnan, "Estimation of switched reluctance motor losses," *IEEE Trans. on Industry Applications*, Vol. 28, No. 3, pp 668-679, 1992.