Global transformer design optimization using deterministic and non-deterministic algorithms

E. I. Amoiralis, M. A. Tsili, A. G. Kladas

Abstract — The present paper compares the application of one deterministic and three non-deterministic optimization algorithms to global transformer design optimization. One deterministic optimization algorithm (Mixed Integer Nonlinear Programming), is compared to three non-deterministic approaches (Harmony Search, Differential Evolution and Genetic Algorithm). The comparison yields significant conclusions on the efficiency of the algorithms and the selection of the most suitable for the transformer design optimization problem.

Index Terms—transformers, design optimization, deterministic algorithms, non-deterministic algorithms, mixed integer nonlinear programming, harmony search, differential evolution, genetic algorithm.

I. INTRODUCTION

In optimum design of transformers, the main target is to minimize the manufacturing cost. Therefore, the objective function is a cost function with many terms, including material costs, labor costs, and overhead costs. These component costs, as well as the constraint functions, must be expressed in terms of a basic set of design variables [1].

Deterministic methods provide robust solutions to the transformer design optimization problem. In this context, the deterministic method of geometric programming has been proposed in [2] in order to deal with the design optimization problem of both low frequency and high frequency transformers. Furthermore, the complex optimum overall transformer design problem, which is formulated as a mixed-integer nonlinear programming problem, by introducing an integrated design optimization methodology based on evolutionary algorithms and numerical electromagnetic and thermal field computations, is addressed in [3]. However, the overall manufacturing cost minimization is scarcely addressed in the technical literature, and the main approaches deal with the cost minimization of specific components such as the magnetic material [4], the no-load loss minimization [5] or the load loss minimization [6]. Techniques that include mathematical models employing analytical formulas, based on design constants and approximations for the calculation of the transformer parameters are often the base of the design process adopted by transformer manufacturers [7].

Apart from deterministic methods, Artificial Intelligence techniques have been extensively used in order to cope with the complex problem of transformer design optimization, such as genetic algorithms (GAs) that have been used for transformer construction cost minimization [8] and construction and operating cost minimization [9][10], performance optimization of cast-resin distribution transformers with stack core technology [11], toroidal core transformers [12], furnace transformers [13], small low-loss low frequency transformers [14] and high frequency transformers [15]. GA is also employed for the optimization of distribution transformers cooling system design in [16]. Neural network techniques are also employed as a means of design optimization as in [17] and [18], where they are used for winding material selection and prediction of transformer losses and reactance, respectively. The comparison of deterministic and non-deterministic optimization algorithms is scarcely encountered in the relevant literature, as in [19] where GA and Simulated Annealing are compared to Geometric Programming for high-frequency power transformer optimization.

It is therefore clear that global transformer optimization remains an active research area, since several approaches for its implementation have not yet been investigated. The present paper compares the application of one deterministic and three non-deterministic optimization algorithms to global transformer design optimization. The applied deterministic optimization algorithm is the Mixed Integer Nonlinear Programming (MINLP), while the three non-deterministic algorithms are: Harmony Search (HS), Differential Evolution (DE) (the use of both HS and DE for transformer design optimization is introduced in this paper) and Genetic Algorithm.

The paper is organized as follows: Section II describes the mathematical formulation of the transformer design optimization problem. Sections III, IV, V and VI provide the necessary theoretical background for MINLP, HS, DE and GA, respectively. Section VII presents the results of the application of the four algorithms to three different distribution transformer ratings. Finally, Section VIII concludes the paper.

II. MATHEMATICAL FORMULATION OF TRANSFORMER DESIGN OPTIMIZATION PROBLEM

The objective of transformer design optimization is based on the minimization of the overall transformer cost function:

$$\min Z(\chi) = \min \sum_{j=1}^{k} c_j f_j(\chi)$$  \hspace{1cm} (1)

where $c_j$ and $f_j$ are the unit cost (€/kg) and the weight (kg) of each component $j$ (active and mechanical part), and $\chi$ is the vector of the four design variables, i.e. the number of low voltage turns, the magnetic induction magnitude $(\Phi)$, the width of core leg $(D)$ and the core window height $(G)$ (Fig. 1).

The minimization of the objective function is subject to:

$$DNLL + DLL - 1.10 \cdot (GNLL + GLL) < 0$$ \hspace{1cm} (2)

$$DNLL - 1.15 \cdot GNLL < 0$$ \hspace{1cm} (3)

$$DLL - 1.15 \cdot GLL < 0$$ \hspace{1cm} (4)
where $D_{NLL}$ denotes the designed no-load loss (W), $D_{LL}$ the designed short-circuit impedance (%), $G_{NLL}$ the guaranteed no-load loss (W), $G_{LL}$ the guaranteed short-circuit impedance (%), $H_{c}$ is the heat dissipated (by convection) through the transformer cooling system (W), while $D$, $G$, $E_{u}$ are the geometric characteristics of the active part (Fig. 1), and $l_{b}$ and $u_{b}$ are nx1 matrices of lower and upper bounds on $x$. The coefficients appearing in (2)-(5) are based on the tolerances specified by IEC 60076-1 [20], while the respective coefficients in (6)-(9) are based on the transformer manufacturer specifications.

![Active part configuration of the three-phase wound core power transformer considered.](image)

**Fig. 1.** Active part configuration of the three-phase wound core power transformer considered.

### III. MIXED INTEGER NONLINEAR PROGRAMMING

Recently, the area of MINLP [21] has experienced tremendous growth and a flourish of research activity. In the transformer design optimization area, MINLP techniques are very suitable and effective due to the fact that the design variables can assume not only continuous values but also integer values (e.g., number of winding turns). In this context, this paper proposes a BB optimization algorithm tailored to a MINLP formulation, completing previous research [22]. MINLP refers to mathematical programming with continuous and discrete variables and nonlinearities in the objective function and constraints. A general MINLP can be written as:

$$
\min f(x, y)
$$

subject to

$$
\begin{align*}
H(x, y) &= 0 \\
G(x, y) &\leq 0
\end{align*}
$$

where $x$ is a vector of $n$ continuous variables and $y$ is a vector of $m$ integer variables ($R$ denotes the real numbers and $Z$ denotes the integers). The function $f$ is a scalar valued objective function, while the vector functions $H$ and $G$ express linear or nonlinear constraints.

BB algorithms for MINLP [23] constitute a well-known approach for solving combinatorial optimization problems to optimality. Essentially, BB techniques use an implicit enumeration scheme for exploring the search space in an “intelligent” way. This is done by partitioning the search space and producing upper and lower bounds of the solutions attainable in each partition. Thus, the search performed by the algorithm can be represented as a tree that is traversed in a certain way. The most efficient (in terms of the number of iterations required to find the optimum and prove its optimality) is to use a depth-first traversal.

The proposed recursive BB algorithm solves continuous optimization problems, while constraining some variables into sets of standard values, which may consist of discrete or integer values. The associated discrete programming problem is recursively divided into two sub-problems, by fixing the discrete variables to the closest above and below standard values. The search starts by solving a nonlinear programming (NLP) relaxation, and using the solution as the lower bound of the problem. If the solutions of the discrete variables are all equal to the values defined at the standard discrete set, then the optimum solution is reached and the search is stopped. Otherwise, the search branches on the first discrete variable that has non-standard solution. The closest discrete values above and below the current solution are identified. If both above and below values exist, the NLP with the fixed above values becomes the first sub-problem. The first discrete variable with non-standard solution is identified. Subsequently, a new equality constraint to fix this variable to the above value is added to the original constraints, and the NLP sub-problem subject to the updated constraints is solved. If the NLP sub-problem converges, and yields the superior solution over the existing lower bound, then this solution becomes the new lower bound. The branching continues recursively to the next discrete value with non-standard solution. Otherwise, the node is fathomed. If this happens, the algorithm backtracks to the ascendant node, and then resumes branching at the sub-problem associated with below values.

In this paper, the Sequential Quadratic Programming (SQP) method is proposed for solving transformer cost minimization NLP sub-problems with BB [23], which enforces early detection and termination of infeasible or inferior NLP solutions. The SQP implementation consists of three main stages: 1) at each major iteration a positive definite quasi-Newton approximation of the Hessian of the Lagrangian function is calculated using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, 2) at each major iteration of the SQP method, a quadratic programming problem is solved, and 3) the line search algorithm is a safeguarded quadratic polynomial method, which requires fewer function evaluations but more gradient evaluations.

### IV. HARMONY SEARCH ALGORITHM

The Harmony Search algorithm (HSA) is a new metaheuristic population search algorithm proposed by Geem et al. [24]. HSA was derived from the natural phenomena of musicians’ behavior when they collectively play their musical instruments (population members) to come up with a pleasing harmony (global optimal solution). This state is determined by an aesthetic standard (fitness function).
The HSA is simple in concept, less in parameters, and easy in implementation. It has been successfully applied to various benchmarking, and real-world problems like traveling salesman problem [25]. The main steps of HS are as follows [24][26].

1. Initialize the algorithm parameters.
2. Initialize the harmony memory.
3. Improvise a new harmony.
4. Update the harmony memory.
5. Check the termination criterion.

These steps are described in the next subsections.

A. Initialization of algorithm parameters

The algorithm parameters are: the harmony memory size (HMS), the number of solution vectors in the harmony memory; harmony memory considering rate (HMCR); pitch adjusting rate (PAR); and the number of improvisations (NI), or stopping criterion. The harmony memory is a memory location where all the solution vectors (sets of decision variables) are stored. Here HMCR and PAR are parameters that are used to improve the solution vector, which are defined in Step 3.

B. Initialization of Harmony Memory

In this step, the HM matrix with as many randomly generated solution vectors as the HMS:

\[
HM = \begin{bmatrix}
x^1_1 & x^1_2 & \ldots & x^1_N & x^1_{N+1} \\
x^2_1 & x^2_2 & \ldots & x^2_N & x^2_{N+1} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
x^{HMS-1}_1 & x^{HMS-1}_2 & \ldots & x^{HMS-1}_N & x^{HMS-1}_{N+1} \\
x^{HMS}_1 & x^{HMS}_2 & \ldots & x^{HMS}_N & x^{HMS}_{N+1} 
\end{bmatrix}
\] (12)

There is a possibility of infeasible solutions which violate the constraints. However, the algorithm forces the search towards feasible solution area. Static penalty functions are used to calculate the penalty cost for an infeasible solution. The total cost for each solution vector is evaluated using

\[
\text{fitness}(\vec{x}) = f(\vec{x}) + \sum_{i=1}^{M} \alpha_i \left( \min[0, g_i(\vec{x})]^2 \right) + \sum_{j=1}^{P} \beta_j \left( \min[0, h_j(\vec{x})]^2 \right)
\] (13)

where \(\alpha_i\) and \(\beta_j\) are the penalty coefficients for the inequality and equality constraints \(g_i(\vec{x})\) and \(h_j(\vec{x})\), respectively. Generally, it is difficult to find a specific rule to determine the values of the penalty coefficients, and hence, these parameters remain problem dependent.

C. Improvisation of a new harmony

A new harmony vector \(\vec{x}' = (x'_1, x'_2, \ldots, x'_N)\) is generated, based on three criteria: 1) memory consideration, 2) pitch adjustment, and 3) random selection. Generating a new harmony is called improvisation. According to memory consideration, \(i\)-th variable \(x'_i = (x_i - x^{HMS}_i)\) The HMCR, which varies between 0 and 1, is the rate of choosing one value from the historical values stored in the HM, while (1-HMCR) is the rate of randomly selecting one value from the possible range of values, as shown in (14):

\[
\text{if } (\text{rand}() < \text{HMCR}) \quad x'_i \leftarrow x'_i \in \{x^1_i, x^2_i, \ldots, x^{HMS}_i\} \\
\text{else} \quad x'_i \leftarrow x'_i \in X_i 
\] (14)

where \(\text{rand}()\) is a uniformly distributed random number between 0 and 1 and \(X_i\) is the set of the possible range of values for each decision variable. For example, an HMCR of 0.85 indicates that HSA will choose decision variable value from historically stored values in HM with 85% probability or from the entire possible range with 15% probability. Every component obtained with memory consideration is examined to determine if pitch is to be adjusted. This operation uses the rate of pitch adjustment as a parameter as shown in the following:

\[
\text{if } (\text{rand}() < \text{PAR}) \quad x'_i = x'_i \pm \text{rand}() \times bw \\
\text{else} \quad x'_i = x'_i
\] (15)

where \(bw\) is an arbitrary distance bandwidth for the continuous design variable and \(\text{rand}()\) is uniform distribution between 0 and 1.

D. Update of harmony vector

If the new harmony vector \(\vec{x}' = (x'_1, x'_2, \ldots, x'_N)\) has better fitness function than the worst harmony in the HM, the new harmony is included in the HM and the existing worst harmony is excluded from the HM.

E. Check of the termination criterion

The HSA is terminated when the termination criterion (e.g., maximum number of improvisations) has been met. Otherwise, steps 3 and 4 are repeated.

In our case, various values of parameters for the HSA were tested and the following ones were chosen for the transformer design optimization process:

- \(\text{HMS} = 6\)
- \(\text{HMCR} = 0.9\)
- \(0.4 \leq \text{PAR} \leq 0.9\)
- \(0.0001 \leq \text{bw} \leq 1\)
- \(\alpha_i = 1000\)
- \(\beta_j = 1000\)

V. DIFFERENTIAL EVOLUTION ALGORITHM

Differential Evolution (DE) is a parallel direct search method which utilizes NP D-dimensional parameter vectors

\[
x_{i,G}, i = 1,2,3,\ldots, NP
\] (16)

as a population for each generation G. NP does not change during the minimization process. The initial vector population is chosen randomly and should cover the entire parameter space. As a rule, we will assume a uniform probability distribution for all random decisions unless
otherwise stated. In case a preliminary solution is available, the initial population might be generated by adding normally distributed random deviations to the nominal solution \( x_{nom,0} \). DE generates new parameter vectors by adding the weighted difference between two population vectors to a third vector. Let this operation be called mutation. The mutated vector’s parameters are then mixed with the parameters of another predetermined vector, the target vector, to yield the so-called trial vector. Parameter mixing is often referred to as “crossover” in the ES-community and will be explained later in more detail. If the trial vector yields a lower cost function value than the target vector, the trial vector replaces the target vector in the following generation. This last operation is called selection. Each population vector has to serve once as the target vector so that NP competitions take place in one generation. More specifically DE’s basic strategy can be described as follows [27][28]:

A. Mutation

For each target vector \( x_{i,G}, i \), a mutant vector is generated according to

\[
v_{i,G+1} = x_{i,G} + F \cdot (x_{r1,G} - x_{r2,G})
\]

with random indexes \( r1, r2, r1 \in \{1, 2, ..., NP \} \), integer, mutually different and \( F \geq 0 \). The randomly chosen integers \( r1, r2, and \( r3 \) are also chosen to be different from the running index \( i \), so that \( NP \) must be greater or equal to four to allow for this condition. \( F \) is a real and constant factor \( \in [0, 2] \) which controls the amplification of the differential variation \( (x_{r1,G} - x_{r2,G}) \).

B. Crossover

In order to increase the diversity of the perturbed parameter vectors, crossover is introduced. To this end, the trial vector:

\[
u_{i,G+1} = (u_{i,G+1}, u_{i,G+1}, ..., u_{i,G+1})
\]

is formed where,

\[
u_{k,G+1} = \begin{cases} u_{k,G+1} & \text{if } \text{randb}(j) \leq CR \text{ or } j = \text{rnbr}(i) \\ x_{k,G} & \text{if } \text{randb}(j) > CR \text{ and } j = \text{rnbr}(i) \end{cases}
\]

\( j = 1, 2, ..., D \).

In (19) \( \text{randb}(j) \) is the \( j \)th evaluation of a uniform random number generator with outcome \( \in [0, 1] \). \( CR \) is the crossover constant \( \in [0, 1] \) which has to be determined by the user. \( \text{rnbr}(i) \) is a randomly chosen index \( \in 1, 2, ..., D \) which ensures that \( u_{i,G+1} \) gets at least one parameter from \( v_{i,G+1} \).

C. Selection

To decide whether or not it should become a member of generation \( G+1 \), the trial vector \( u_{i,G+1} \) is compared to the target vector \( x_{i,G} \) using the greedy criterion. If vector \( u_{i,G+1} \) yields a smaller cost function value than \( x_{i,G} \), then \( x_{i,G+1} \) is set to \( u_{i,G+1} \); otherwise, the old value \( x_{i,G} \) is retained.

VI. GENETIC ALGORITHM

GA is a method for solving optimization problems based on natural selection, the process that drives biological evolution. The GA repeatedly modifies a population of initial solutions. At each step, it selects individuals at random from the current population to be the parents. It then uses the parents to produce children for the next generation. Crossover and mutation operators are applied to the parents to generate new children. Over successive generations, the population evolves toward an optimal solution. In contrast to more traditional numerical techniques, the parallel nature of the stochastic search done by GA often makes it very effective in finding the global optimum. GA is less susceptible to getting stuck at local optima than gradient search methods. Also, GA is much less sensitive to initial conditions and is widely used in various optimization problems [29].

Genes are the basic building blocks of genetic algorithms. A gene is a binary encoding of a parameter. A chromosome in a computer algorithm is an array of genes. Each chromosome has an associated cost function, assigning a relative merit to that chromosome. The algorithm begins with a large list of random chromosomes. Cost functions are evaluated for each chromosome. The chromosomes are ranked from the most-fit to the least-fit according to their respective cost functions. Unacceptable chromosomes are discarded, leaving a superior species-subset of the original list. Genes that survive become parents, by swapping some of their genetic material to produce two new offspring. The parents reproduce enough to offset the discarded chromosomes. Thus, the total number of chromosomes remains constant &er each iteration. Mutations cause small random changes in a chromosome. Cost functions are evaluated for the offspring and the mutated chromosome, and the process is repeated. The algorithm stops after a set number of iterations, or when an acceptable solution is obtained [30].

The algorithm begins by defining a chromosome as an array of parameter values to be optimized. If the chromosome has \( n_{par} \) parameters (an \( N \)-dimensional optimization problem), given by \( p_1, p_2, ..., p_{n_{par}} \), then the chromosome is written as

\[
\text{chromosome} = [p_1, p_2, ..., p_{n_{par}}]
\]

Each chromosome has a cost function, found by evaluating the objective function \( f \) at \( p_1, p_2, ..., p_{n_{par}} \). The cost function is represented by

\[
\text{cost} = f (p_1, p_2, ..., p_{n_{par}})
\]

The parameters, \( p_n \), can be discrete or continuous. If the parameters are continuous, either some limits need to be placed on the parameters, or they should be restricted to a handful of possible values. One way to limit the parameters is to encode them in a binary sequence, such as

\[
q_n = \sum_{m=1}^{L_n} a_{n,m} \cdot 2^{1-m} Q
\]

where \( q_n \) is the quantized version of \( p_n \), \( L_n \) is the number of quantization levels for \( q_n \), \( a_{n,m} \) is the array containing the binary sequence representing \( q_n \) and \( Q \) is the largest quantization level half the largest-possible value of \( q_n \). The binary-encoded parameter, \( q_n \), does not have to
mathematically relate to $p_s$, as in (22). Instead, $q_s$ may just represent some value of $p_s$. The implementation of genetic algorithms described here only works with the binary encoding of the parameters, and not the parameters themselves. Whenever the cost function is evaluated, the chromosome must first be decoded.

After devising a scheme to encode and decode the parameters, a list of random chromosomes is generated. Each chromosome has an associated cost, calculated from the cost function in (21). At this point, the unacceptable chromosomes are discarded. Unacceptable is user defined. Typically, the top $t$ are kept (where $t$ is even), and the bottom $N_{chrom} - t$ are discarded (where $N_{chrom}$ is the number of chromosomes).

The next step, after ranking and discarding the chromosomes, is to pair the remaining $N_{chrom}/2$ chromosomes for mating. Any two chromosomes can mate. Some possible approaches are to pair the chromosomes from top to bottom of the list, pair them randomly, or pair them 1 with $N_{chrom}/2$, 2 with $N_{chrom}/2-1$, etc. Once paired, new offspring are formed from the pair-swapping genetic material.

After the surviving $N_{chrom}/2$ chromosomes pair and mate, the list of $N_{chrom}/2$ parents and $N_{chrom}/2$ offspring results in a total of $N_{chrom}$ chromosomes (the same number of chromosomes as at the start).

At this point, random mutations alter a small percentage of the bits in the list of chromosomes, by changing a “1” to a “0” or visa versa. A bit is randomly selected for mutation of the bits in the list of chromosomes, by changing a “1” to a “0” or the opposite. A bit is randomly selected for mutation of the bits in the list of chromosomes, by changing a “1” to a “0” or versa.

Mer the mutations take place, the costs associated with the offspring and mutated chromosomes are calculated, and the process is repeated. The number of generations that evolve depends on whether an acceptable solution is reached, or a set number of iterations is exceeded. After a while, all of the chromosomes and associated costs become the same, except for those that are mutated. At this point, the algorithm should be stopped.

In our case, 30 runs of the GA algorithm are performed and the best solution is chosen as the optimum one. The population type is bit string of size equal to 20. A random initial population is created, that satisfies the boundary and linear constraints of the optimization problem. Rank fitness scaling is employed, scaling the raw scores based on the rank of each individual, rather than its score. Stochastic uniform selection function is used, which lays out a line in which each parent corresponds to a section of the line of length proportional to its expectation. The algorithm moves along the line in steps of equal size, one step for each parent. At each step, the algorithm allocates a parent from the section it lands on. The first step is a uniform random number less than the step size. As far as mutation and crossover functions are concerned, the first one is adaptive feasible (it randomly generates directions that are adaptive with respect to the last successful or unsuccessful generation - a step length is chosen along each direction so that linear constraints and bounds are satisfied) and the second one is scattered (it creates a random binary vector, selects the genes where the vector is a 1 from the first parent, and the genes where the vector is a 0 from the second parent, and combines the genes to form the child.). The limit for the fitness function is set to 0.5, while the maximum number of generations (iterations) is equal to 500.

VII. RESULTS AND DISCUSSION

The MINLP and HS, DE, GA optimization algorithms are applied for the design optimization of three 20-0.4kV distribution transformers, of 160kVA, 400kVA and 630 kVA rating. Tables I, II and III compare the respective optimization results. Figure 2 compares the loss values of the optimum transformer designs yielded by the four optimization methods in the cases of the 160, 400 and 630 kVA transformers. Figure 3 compares the construction cost of the optimum transformer designs yielded by the four optimization methods in the cases of the 160, 400 and 630 kVA transformers. Figure 4 compares the cost difference of the optimum transformer designs yielded by the non-deterministic methods, i.e. HS, DE and GA to the optimum transformer designs yielded by MINLP in the cases of the 160, 400 and 630 kVA transformers.

The comparison of the results presented in Tables I-III and Figs. 2-4 results to the following conclusions:

- MINLP yields the best solution in the case of 160 and 400kVA transformers, while in the case of the 630kVA transformer the cost difference between the optimum transformer yielded by MINLP is 1.08% compared to the best solution, yielded by HS.

- Among the non-deterministic methods, HS produces the best solution for all the examined transformer cases, while GA produces the worst one.

- A small difference in the technical characteristics of the optimum solutions produced by the four optimization methods (no load and load losses as depicted in Fig.2). According to these characteristics, GA produces the best solution in terms of no-load loss, while MINLP produces the best solution in terms of load loss.

<table>
<thead>
<tr>
<th>Characteristics of the optimum transformer design</th>
<th>MINLP</th>
<th>HS</th>
<th>DE</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low voltage turns</td>
<td>27</td>
<td>29</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>$D$ (mm)</td>
<td>186</td>
<td>181</td>
<td>189</td>
<td>202</td>
</tr>
<tr>
<td>$G$ (mm)</td>
<td>186</td>
<td>209</td>
<td>211</td>
<td>240</td>
</tr>
<tr>
<td>$B$ (Gauss)</td>
<td>17399</td>
<td>17355</td>
<td>17285</td>
<td>16661</td>
</tr>
<tr>
<td>No load loss (W)</td>
<td>431</td>
<td>414</td>
<td>408</td>
<td>378</td>
</tr>
<tr>
<td>Load loss (W)</td>
<td>2133</td>
<td>2199</td>
<td>2203</td>
<td>2252</td>
</tr>
<tr>
<td>Short-Circuit Impedance (%)</td>
<td>4.30</td>
<td>4.26</td>
<td>4.24</td>
<td>3.92</td>
</tr>
<tr>
<td>Cast (€)</td>
<td>2195</td>
<td>2215</td>
<td>2225</td>
<td>2312</td>
</tr>
<tr>
<td>Cost Difference compared to the cheapest solution (%)</td>
<td>0.00</td>
<td>0.92</td>
<td>1.40</td>
<td>5.33</td>
</tr>
</tbody>
</table>
TABLE II
Comparison of the Optimization Algorithms for the 400 KVA Transformer

<table>
<thead>
<tr>
<th>Characteristics of the optimum transformer design</th>
<th>MINLP</th>
<th>HS</th>
<th>DE</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low voltage turns</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>D (mm)</td>
<td>236</td>
<td>254</td>
<td>230</td>
<td>248</td>
</tr>
<tr>
<td>G (mm)</td>
<td>247</td>
<td>260</td>
<td>247</td>
<td>275</td>
</tr>
<tr>
<td>B (Gauss)</td>
<td>18000</td>
<td>17998</td>
<td>17922</td>
<td>16846</td>
</tr>
<tr>
<td>No load loss (W)</td>
<td>860</td>
<td>862</td>
<td>856</td>
<td>709</td>
</tr>
<tr>
<td>Load loss (W)</td>
<td>4289</td>
<td>4319</td>
<td>4279</td>
<td>4551</td>
</tr>
<tr>
<td>Short-Circuit Impedance (%)</td>
<td>4.40</td>
<td>4.10</td>
<td>4.40</td>
<td>4.35</td>
</tr>
<tr>
<td>Cost (€)</td>
<td>4021</td>
<td>4021</td>
<td>4034</td>
<td>4207</td>
</tr>
<tr>
<td>Cost Difference compared to the cheapest solution (%)</td>
<td>0.00</td>
<td>0.01</td>
<td>0.31</td>
<td>4.61</td>
</tr>
</tbody>
</table>

TABLE III
Comparison of the Optimization Algorithms for the 630 KVA Transformer

<table>
<thead>
<tr>
<th>Characteristics of the optimum transformer design</th>
<th>MINLP</th>
<th>HS</th>
<th>DE</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low voltage turns</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>D (mm)</td>
<td>251</td>
<td>284</td>
<td>283</td>
<td>284</td>
</tr>
<tr>
<td>G (mm)</td>
<td>259</td>
<td>286</td>
<td>286</td>
<td>286</td>
</tr>
<tr>
<td>B (Gauss)</td>
<td>18000</td>
<td>17999</td>
<td>17981</td>
<td>17996</td>
</tr>
<tr>
<td>No load loss (W)</td>
<td>1327</td>
<td>1321</td>
<td>1320</td>
<td>1321</td>
</tr>
<tr>
<td>Load loss (W)</td>
<td>4463</td>
<td>4463</td>
<td>4460</td>
<td>4463</td>
</tr>
<tr>
<td>Short-Circuit Impedance (%)</td>
<td>6.34</td>
<td>5.51</td>
<td>5.46</td>
<td>5.51</td>
</tr>
<tr>
<td>Cost (€)</td>
<td>7390</td>
<td>7311</td>
<td>7316</td>
<td>7312</td>
</tr>
<tr>
<td>Cost Difference compared to the cheapest solution (%)</td>
<td>1.08</td>
<td>0.00</td>
<td>0.07</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Fig. 2. Comparison of (a) no load and (b) load loss values of the optimum transformer designs yielded by the four optimization methods in the case of the 160, 400 and 630 kVA transformers.

Fig. 3. Cost comparison of the optimum transformer designs yielded by the four optimization methods in the case of the 160, 400 and 630 kVA transformers.

Fig. 4. Cost difference of the optimum transformer designs yielded by the non-deterministic methods, i.e. HS, DE and GA to the optimum transformer design yielded by MINLP in the case of the 160, 400 and 630 kVA transformers.

VIII. CONCLUSIONS

The present paper compared the application of one deterministic and three non-deterministic optimization algorithms to global transformer design optimization. The applied deterministic optimization algorithm is the Mixed Integer Nonlinear Programming (MINLP), while the three non-deterministic algorithms are: Harmony Search (HS), Differential Evolution (DE) (the use of both HS and DE for transformer design optimization is introduced in this paper) and Genetic Algorithm.

The comparison of the results in three different distribution transformer ratings concluded that the non-deterministic method (MINLP) yields the best solution in the case of 160 and 400kVA transformers, while in the case
of the 630kVA transformer the cost difference between the optimum transformer yielded by MINLP and the best solution, yielded by HS, is approximately equal to 1%. No significant deviations between the technical characteristics of the optimum solutions of the four methods are observed. The deterministic method is therefore the most appropriate one in the considered transformer design optimization problem.

IX. REFERENCES


X. BIOGRAPHIES

Eleftherios I. Amoiralis was born in Greece, in 1980. He received the Diploma in Production and Management Engineering, the M.Sc. in Industrial Engineering, and the Ph.D. degree in the field of Electric Power Systems from the Technical University of Crete (TUC), Greece, in 2004, 2005, and 2008, respectively. Since 2004, he has been occupied as researcher in many research projects. From 2005 to 2008, he was with Schneider Electric AE as a freelancer. He is currently an Assistant Professor in the Technological Educational Institute ofchalkida. His current research interests include transformer cost evaluation, energy-efficient transformers, optimal transformer sizing, transformer design optimization as well as artificial intelligence. Dr. Amoiralis is member of IEEE and the Technical Chamber of Greece.

Marina A. Tsili was born in Greece, in 1976. She received the Diploma in Electrical and Computer Engineering in 2001 and the Ph.D. degree in 2005 from the National Technical University of Athens, Greece. From 2005 to 2006 she worked for the Distribution Division of the Public Power Corporation of Greece, in high and medium voltage substation studies. In 2007, she joined the Hellenic Transmission System Operator as a power systems engineer. Since 2005, she is collaborating with the National Technical University of Athens as a research associate. Her research interests include transformer and electric machine modeling as well as analysis of generating units by renewable energy sources. She is a member of IEEE and the Technical Chamber of Greece.

Antonios G. Kladas was born in Greece, in 1959. He received the Diploma in Electrical Engineering from the Aristotle University of Thessaloniki, Greece in 1982 and the DEA and Ph.D. degrees in 1983 and 1987, respectively from the University of Pierre and Marie Curie (Paris 6), France. He served as Associate Assistant in the University of Pierre and Marie Curie from 1984-1989. During the period 1991-1996 he joined the Public Power Corporation of Greece, where he was engaged in the System Studies Department. Since 1996 he joined the Department of Electrical and Computer Engineering of the National Technical University of Athens (NTUA), where he is now Professor. His research interests include transformer and electric machine modeling and design as well as analysis of generating units by renewable energy sources and industrial drives.