Use of genetic algorithms for optimal design of electrical resistive furnaces insulation

Hristo Nenov[1] and Borislav Dimitrov[2]

Abstract – **The using of genetic algorithms approach for design of chamber resistive furnaces (CRF) insulation is presented in this paper. The solution of the specified optimization problem and finding the optimal parameters are the basic precondition for increasing the efficiency of resistance furnaces.**

Keywords – **genetic algorithms, optimization, electrical resistive furnaces.**

I. INTRODUCTION

Electric resistance furnace chambers (EFC) are designed for heat treatment of steel parts - annealing, normalization and more. They are powerful consumers of electricity, which is why the problems related to improving the energy effectiveness are particularly relevant. The analysis of the processes run in the chamber resistance furnaces using model and numerical methods gives significant opportunities for optimization when setting different target functions. Proper results are achieved by working with detailed models, whose parameters correspond to maximum facility.

II. ANALYSIS

A. Genetic algorithm

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. You can apply the genetic algorithm to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear. The genetic algorithm can address problems of mixed integer programming, where some components are restricted to be integer-valued.

[1]eng. Hristo Nenov, Ph.D– Technical University of Varna, Bulgaria, assistant professor. E-mail – ico762001@gmail.com

[2]eng. Borislav Dimitrov, Ph.D– Technical University of Varna, Bulgaria, associated professor.

E-mail – bdimitrov@processmodeling.org.

The genetic algorithm uses three main types of rules at each step to create the next generation from the current population:

- Selection rules select the individuals, called parents that contribute to the population at the next generation.
- Crossover rules combine two parents to form children for the next generation.
- Mutation rules apply random changes to individual parents to form children.

The genetic algorithm differs from a classical, derivative-based, optimization algorithm in two main ways, as summarized in the following table.(Table 1).

TABLE I
ALGORITHM COMPARE

Classical Algorithm	Genetic Algorithm
Generates a single point at each iteration. The sequence of points approaches an optimal solution.	Generates a population of points at each iteration. The best point in the population approaches an optimal solution.
Selects the next point in the sequence by a deterministic computation.	Selects the next population by computation which uses random number generators.

B. Genetic Algorithm Terminology

Fitness Functions

The fitness function is the function you want to optimize. For standard optimization algorithms, this is known as the objective function. The toolbox software tries to find the minimum of the fitness function.

Write the fitness function as a file or anonymous function, and pass it as a function handle input argument to the main genetic algorithm function.

Individuals

An individual is any point to which you can apply the fitness function. The value of the fitness function for an individual is its score.

Populations and Generations

A population is an array of individuals. For example, if the size of the population is 100 and the number of variables in

the fitness function is 3, the population is represented by a 100-by-3 matrix. The same individual can appear more than once in the population. For example, the individual (2, -3, 1) can appear in more than one row of the array.

At each iteration, the genetic algorithm performs a series of computations on the current population to produce a new population. Each successive population is called a new generation.

Diversity

Diversity refers to the average distance between individuals in a population. A population has high diversity if the average distance is large; otherwise it has low diversity. In the following figure, the population on the left has high diversity, while the population on the right has low diversity.

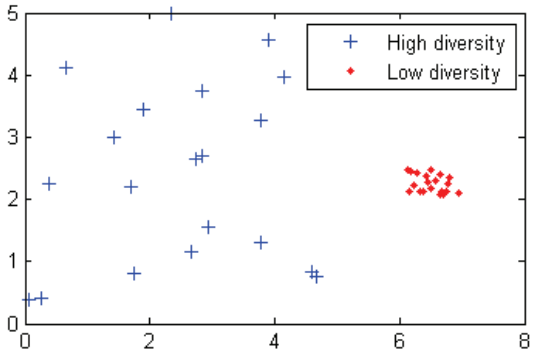

Fig.1. Diversity example

Diversity is essential to the genetic algorithm because it enables the algorithm to search a larger region of the space.

Fitness Values and Best Fitness Values

The fitness value of an individual is the value of the fitness function for that individual. Because the toolbox software finds the minimum of the fitness function, the best fitness value for a population is the smallest fitness value for any individual in the population.

Parents and Children

To create the next generation, the genetic algorithm selects certain individuals in the current population, called parents, and uses them to create individuals in the next generation, called children. Typically, the algorithm is more likely to select parents that have better fitness values.

C. How it works

1. The algorithm begins by creating a random initial population.
2. The algorithm then creates a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population. To create the new population, the algorithm performs the following steps:

   a. Scores each member of the current population by computing its fitness value.

   b. Scales the raw fitness scores to convert them into a more usable range of values.

   c. Selects members, called parents, based on their fitness.

   d. Some of the individuals in the current population that have lower fitness are chosen as elite. These elite individuals are passed to the next population.

   e. Produces children from the parents. Children are produced either by making random changes to a single parent—mutation—or by combining the vector entries of a pair of parents—crossover.

3. Replaces the current population with the children to form the next generation.

Creating the Next Generation

At each step, the genetic algorithm uses the current population to create the children that make up the next generation. The algorithm selects a group of individuals in the current population, called parents, who contribute their genes—the entries of their vectors—to their children. The algorithm usually selects individuals that have better fitness values as parents. You can specify the function that the algorithm uses to select the parents in the Selection function field in the Selection options.

The genetic algorithm creates three types of children for the next generation:

Elite children are the individuals in the current generation with the best fitness values. These individuals automatically survive to the next generation.

Crossover children are created by combining the vectors of a pair of parents.

Mutation children are created by introducing random changes, or mutations, to a single parent.

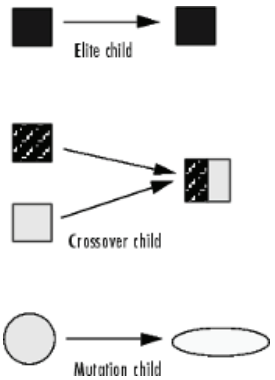

Fig.2. Children types

The algorithm stops when one of the stopping criteria is met.

Crossover Children

The algorithm creates crossover children by combining pairs of parents in the current population. At each coordinate of the child vector, the default crossover function randomly selects an entry, or gene, at the same coordinate from one of the two parents and assigns it to the child. For problems with linear constraints, the default crossover function creates the child as a random weighted average of the parents.

Mutation Children

The algorithm creates mutation children by randomly changing the genes of individual parents. By default, for unconstrained problems the algorithm adds a random vector from a Gaussian distribution to the parent. For bounded or linearly constrained problems, the child remains feasible.

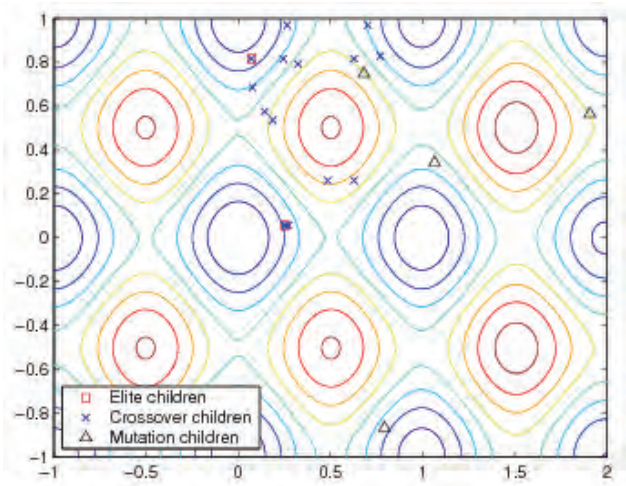

Fig.3. Children types

Stopping Conditions for the Algorithm

The genetic algorithm uses the following conditions to determine when to stop:

- Generations — the algorithm stops when the number of generations reaches the value of Generations.
- Time limit — the algorithm stops after running for an amount of time in seconds equal to Time limit.
- Fitness limit — the algorithm stops when the value of the fitness function for the best point in the current population is less than or equal to Fitness limit.
- Stall generations — the algorithm stops when the weighted average change in the fitness function value over Stall generations is less than Function tolerance.
- Stall time limit — the algorithm stops if there is no improvement in the objective function during an interval of time in seconds equal to Stall time limit.
- Function Tolerance — the algorithm runs until the weighted average relative change in the fitness function value over Stall generations is less than Function tolerance. The weighting function is 1/2n, where n is the number of generations prior to the current.
- Nonlinear constraint tolerance — The Nonlinear constraint tolerance is not used as stopping criterion. It is used to determine the feasibility with respect to nonlinear constraints. Also, a point is feasible with respect to linear constraints when the constraint violation is below the square root of Nonlinear constraint tolerance.

Reproduction Options

Reproduction options control how the genetic algorithm creates the next generation. The options are

Elite count — the number of individuals with the best fitness values in the current generation that are guaranteed to survive to the next generation. These individuals are called elite children. The default value of Elite count is 2.

When Elite count is at least 1, the best fitness value can only decrease from one generation to the next. This is what you want to happen, since the genetic algorithm minimizes the fitness function. Setting Elite count to a high value causes the fittest individuals to dominate the population, which can make the search less effective.

Crossover fraction — the fraction of individuals in the next generation, other than elite children, that are created by crossover. Setting the Crossover Fraction describes how the value of Crossover fraction affects the performance of the genetic algorithm.

Mutation and Crossover

The genetic algorithm uses the individuals in the current generation to create the children that make up the next generation. Besides elite children, which correspond to the individuals in the current generation with the best fitness values, the algorithm creates

Crossover children by selecting vector entries, or genes, from a pair of individuals in the current generation and combine them to form a child

Mutation children by applying random changes to a single individual in the current generation to create a child

Both processes are essential to the genetic algorithm. Crossover enables the algorithm to extract the best genes from different individuals and recombine them into potentially superior children. Mutation adds to the diversity of a population and thereby increases the likelihood that the algorithm will generate individuals with better fitness values.

Some of the advantages of a GA include that it

• Optimizes with continuous or discrete variables;

• Doesn't require derivative information;

• Simultaneously searches from a wide sampling of the cost surface;

• Deals with a large number of variables;

• Is well suited for parallel computers;

• Optimizes variables with extremely complex cost surfaces (they can jump out of a local minimum);

• Provides a list of optimum variables, not just a single solution;

• May encode the variables so that the optimization is done with the encoded variables, and

• Works with numerically generated data, experimental data, or analytical functions.

These advantages are intriguing and produce stunning results when traditional optimization approaches fail miserably.

D. Optimisation of chamber resistive furnaces (CRF) insulation

The main factor determining the energy performance of CRF is the loss. The realization of the assigned tasks is reduced to: define minimum losses - for stationary furnaces operating in continuous mode (industrial, large-sized furnace, etc.); define minimum size and weight - use the CRF if necessary with incidentals – furnace that are non-essential facilities and are required to take minimum space, mobile furnace mounted on a platform, laboratory ones, etc.

The object of research is a CRF. As a target function to solve the optimization problem is set minimization of heat losses.

The parameters involved in the mathematical model that describes the heat losses of the furnace are:

- number of layers
  - ✓ fire resistant layer size/weight
  - ✓ insulation layer size/weight
  - ✓ thickness of the layers
- rated temperature
- productivity
- voltage
- specific heat capacity
- thermal conductivity
- density
- degree of blackness
- size of the walls.

As restrictive conditions are determined the range of variation of the material specific heat, size of the walls and thickness of the insulating layers.

On the fig.4 is shown the situation before the process of optimization

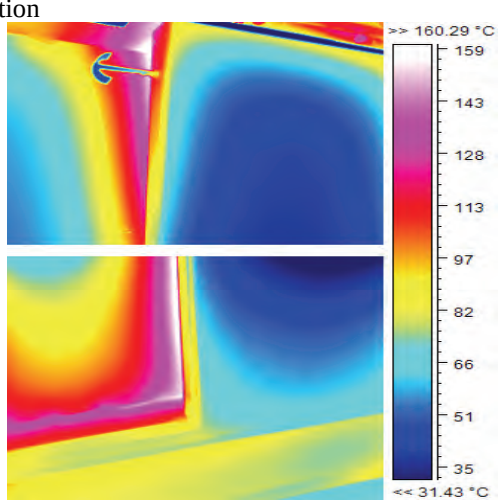

Fig.4. infra-red thermo picture of CRF before optimization (manhole chamber)

The figure clearly shows that the resulting temperature of the outer shell (≅ 160 C°) of the furnace is unacceptable to the conditions of work of this type of facility.

The optimization process, correct the values of the different parameters of the CRF, and after changes to the technical characteristics of the furnace, the following results were achieved (Fig.5).

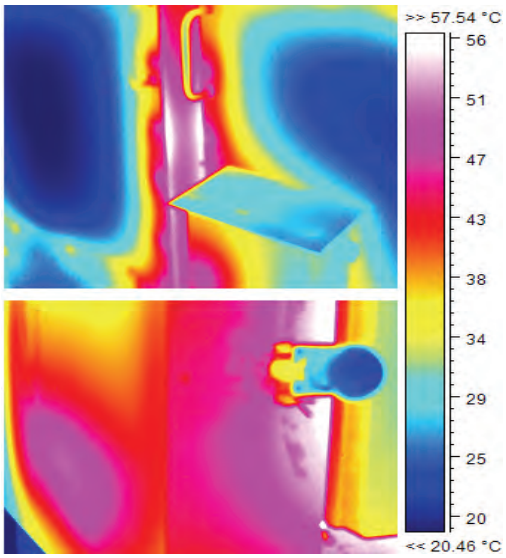

Fig.4. infra-red thermo picture of CRF after optimization (manhole chamber)

It is clear from the figure that after optimization, the maximum temperature which is obtained in the furnace housing is now only 57 C°.

III. Conclusion

Genetic algorithms are a powerful tool for solving optimization problems with many variables and complex calculations. In some cases, they have an advantage over traditional (calculus-based) optimization methods. Their uses help the improvement of mathematical models describing the various furnaces and solve some specific problems in the operation of this type of systems.

Acknowledgement

This paper is developed in the frames of project “Improving energy efficiency and optimization of electro technological processes and devices”, № МУ03/163 financed by the National Science Fund.

References

[1] P. Watson, K. C. Gupta, “EM-ANN Models for Microstrip Vias and Interconnects”, IEEE Trans., Microwave Theory Tech., vol. 44, no. 12, pp. 2395-2503, 1996.

[2] B. Milovanovic, Z. Stankovic, S. Ivkovic and V. Stankovic, "Loaded Cylindrical Metallic Cavities Modeling using Neural Networks", TELSIKS'99, Conference Proceedings, pp.214-217, Nis, Yugoslavia, 1999.

[3] S. Haykin, *Neural Networks*, New York, IEEE Press, 1994.