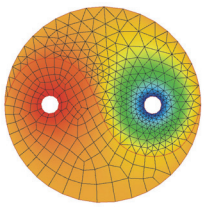
Publishing House
AKAPIT

COMPUTER METHODS IN MATERIALS SCIENCE

Informatyka w Technologii Materiałów

Vol. 9, 2009, No. 3



BIO-INSPIRED OPTIMIZATION STRATEGIES IN CONTROL OF COPPER FLASH SMELTING PROCESS

ŁUKASZ SZTANGRET, ANDRZEJ STANISŁAWCZYK, JAN KUSIAK

Akademia Górniczo-Hutnicza, al. Mickiewicza 30, 30-059 Kraków
Corresponding Author: szt@agh.edu.pl (Ł. Sztangret)

Abstract

The paper presents optimization methods that are inspired by the mechanisms occurring in nature and their application in the determination of the values of signals controlling a copper flash smelting process. The furnace model that is used in the optimization process, was created on the base of artificial neural network. The control was aimed at the obtaining of the required values of SO_2, CO_2 and NO_x concentrations in exhaust gases at a specific concentrate's composition. The optimization process was based on a static furnace model and, therefore, the control consisted in the problem of static optimization. Block limitations were imposed on all the decision variables. The paper gives the theoretical background of each method, the way of implementation and the results obtained with its respective application. In addition, the results were compared with the results obtained by using one of deterministic methods.

Key words: copper flash smelting process, bio-inspired optimization strategies, control of metallurgical processes

1. INTRODUCTION

Humans have always watched nature and have tried to imitate its perfect mechanisms. Optimization algorithms as described in this chapter are also modelled on those mechanisms existing in nature: evolution, behaviour of population of individuals, etc. Optimization algorithms inspired by nature belong to the group of heuristic algorithms. Heuristic algorithms have the feature of not ensuring the finding of the optimum solution. There is no evidence at all that a heuristic algorithm will find the optimum solution in a limited number of iterations. Their wide application is justified by the fact that the optimised objective function may have any nature (non-linear, discontinuous or multimodal). Optimization of such functions by using the classical methods is more difficult and sometimes even impossible. The considered in the work algorithms inspired by nature are:

- genetic algorithms,
- evolutionary algorithms,
- particle swarm optimization,
- simulated annealing method.

These algorithms, that are described below, have been used to determine the optimum values of the control parameters in the one-stage process of copper smelting in a flash smelting furnace. The optimization was carried out based on the static process model that was built on the basis of the artificial neural networks. Process and model descriptions are presented in works (Kusiak, 2009) and (Stanisławczyk & Kusiak, 2009). As the model does not take into account the process dynamics, the optimum control determination is a problem of static optimization.

2. GENETIC ALGORITHMS

Genetic algorithms (GA) (Cytowski, 1996; Goldberg, 1995; Gwiazda, 2007) are optimization

 ISSN 1641-8581

methods that are inspired by natural selection, evolution and inheritance existing in nature. They originated from the work that was carried out at the University of Michigan under John Holland's direction. These methods apply a few simple mechanisms: natural selection, genetic recombination and mutation. Genetic algorithms differ from the classical ones primarily by the following key issues:

1) encoded form of the solution,
2) search starts from a certain population of initial solutions,
3) use of probabilistic selection rules.

The greatest advantage of genetic algorithms is a lack of restriction on the form of the objective function. It can be non-linear, multimodal and even discontinuous. In addition, GAs do not require knowledge of the gradient or higher derivatives of the objective function. They use the probabilistic selection rules which makes that genetic algorithms do not always find the optimal solution. Therefore, if analytical optimization methods can be applied for the problem considered, they should be used, because they usually turn out to be more effective. Genetic algorithms should be used to resolve problems, for which other methods turn out to be too difficult for implementation or excessively time-consuming.

The terminology used in the genetic algorithms description is rather unique because and has been taken directly from a biology. The basic terms, together with their explanation, are shown below.

Term	Meaning
chromosome	series of genes; encoded form of a potential solution
gene	single component of a chromosome
allele	gene's value
locus	gene's position in a chromosome
individual	potential solution (point in the search space)
genotype	set of a specific individual's chromosomes
phenotype	individual in non-coded form
population	set of individuals of a specific size
fitness	numerical value defining the quality of a solution represented by a selected individual
parent	individual selected for a crossing operatotion
descendant	individual obtained in the crossing operation
crossing	process of genes recombination, leading to the generation of new descendants as a result of the exchange of fragments of parents' chromosomes
mutation	process of a single gene change in a chromosome
selection	selection of individuals, usually best adapted to the parents' population

Genetic algorithms process the population of individuals. Each individual is encoded by means of a chain of defined length by using a finite alphabet. *Zero – one* alphabet is most often used and individuals are coded with a natural binary code (NBC). Using NBC code, it is easy to code the natural numbers from the interval $[0, 2^n-1]$, where n is the length of the chain of genes. The encoding of the real numbers from the interval $[u, v]$ with the assumed accuracy r can be achieved by the linear mapping of the interval $[u, v]$ onto the interval $[0, 2^n-1]$. If an individual's genotype is marked g and their phenotype f, then there is the following relationship:

$$f = u + gr \tag{1}$$

where: r is the accuracy and is given by the equation:

$$r = \frac{v-u}{2^n - 1} \tag{2}$$

Pseudocode of a general genetic algorithm is as follows:

```
t ← 0;
initiation of P^t population;
for each i ∈ P^t do compute f(i);
repeat
        T^t ← (P^t) selection;
        O^t ← (T^t) crossing and mutation;
        for each i ∈ O^t do compute f(i);
        P^(t+1) ← O^t;
        t ← t + 1;
until not stop-condition (t, P^t));
```

The first step consists of creation of a baseline population. This step is performed only once and consists of sampling the appropriate number of individuals. A new, continually improved baseline population, is created in each algorithm's iteration, by the application of the following three genetic operators:

1) selection (reproduction),
2) crossing,
3) mutation.

The selection operator is responsible for the selection of individuals creating the population of parents. The selection is random, but depends on the objective function values of separate individuals. The higher the objective function value, the higher the individual's chance of being selected. There are many ways of execution of the selection operation. The method of the roulette wheel is the simplest one. The probability of specific individual selection is equal to the individual's objective function value

and the entire population fitness index. The population fitness index is equal to the sum of the objective function values for all individuals comprised by the population. The method name refers to a wheel divided into sectors. Each individual has its own sector, which is proportional to the objective function value. A single wheel's turn selects one individual copied to the parents' population. Then, crossing and mutation operators are executed on the parents' population. The crossing operator proceeds in two stages. In the first, two individuals are randomly selected from the parents' population. In the second, the pair selected is subjected to the crossing operation with some probability (crossing probability p_k is an algorithm parameter). The crossing point k ($k<n$) is drawn and next, the genes in positions from $k+1$ to n inclusive are exchanged between parent individuals, creating two descendant individuals. The crossing operation is illustrated by the following example.

In the first stage, let the following two individuals (parents) have been selected for crossing:

10101010

01010101

In the second stage, the crossing point $k = 4$ has been drawn and the genes have been changed in positions from 5 to 8, creating the following two descendant individuals.

$$\begin{array}{l} 1010|1010 \\ 0101|0101 \end{array} \rightarrow \begin{array}{l} 1010|0101 \\ 0101|1010 \end{array}$$

In the next step the descendant individuals are mutated (also with some probability according the mutation probability p_m). A number from the interval $[0, 1]$ is drawn for each gene in the chromosome. If is lower or equal p_m, then the gene's value is changed into the opposite. The operation of a mutation operator has been shown below for one of the two previously obtained descendant individuals. Assuming that the mutation probability is $p_m = 0.1$ and the following sequence of numbers has been drawn for the first descendant: 0.82; 0.35; **0.05**; 0.24; 0.74; 0.16; 0.63; 0.47, an individual before and after mutation will have the following form:

10100101

10000101.

After mutation, completion individuals are recorded in the descendant population, which becomes the baseline population for the next algorithm iteration. The number of algorithm iterations should be limited to a certain maximum value, in which after exceeding, the algorithm will end the search, irrespectively to the results obtained so far. The end of the searching should also occur at the moment of obtaining a solution considered sufficiently close to the optimal one.

3. EVOLUTIONARY ALGORITHMS

Evolutionary algorithms (EA) (Arabas, 2001) are based on the same assumptions as the genetic algorithms. However, contrary to GA, individuals comprised by the population are not encoded. Evolutionary algorithms comprise three basic strategies:

1) strategy (1+1),
2) strategy (μ+λ),
3) strategy (μ,λ).

These strategies differ between themselves in population size, the operators used and in the way individuals are selected for the baseline population for the next iteration. A brief description of each of them is shown below.

Strategy (1+1)

In strategy (1+1), the population consists of one individual only. The crossing operator does not exist therein. In each of the next algorithm's iteration, one new individual is generated by means of a mutation operator. The selection is limited to the selection of an individual of the highest objective function value. The algorithm's pseudocode is shown below.

```
t ← 0;
X^t initialisation;
repeat
        Y^t ← (X^t) mutation;
        if (f(Y^t) > f(X^t))
                X^{t+1} ← Y^t;
        else
                X^{t+1} ← X^t;
        t ← t + 1;
until (not stop-condition (t, X^t));
```

The mutation operator creates a new individual by adding a random number to each gene. The number added is a product of a mutation range and a random number of normal distribution N(0,1):

$$Y_i^t = X_i^t + \sigma\xi_{N(0,1),i} \tag{3}$$

where: σ is the mutation range, ξ is a random variable of normal distribution N(0,1).

The mutation range is an algorithm's parameter, which changes during its operation. The change algorithm is called as the *1/5 success rule*. It operates as follows:

1) if within consecutive k iterations the number of successful mutations (i.e. when the descendant individual turns out better than its parent) is higher than 1/5 of all mutations, then the mutation range is thereby increased: $\sigma' = c_i \sigma$,
2) if the number of successful mutations is lower than 1/5 of all mutations, then the mutation range is thereby decreased: $\sigma' = c_d \sigma$,
3) if the number of successful mutations is equal to 1/5 of all mutations, then the mutation range does not change.

Constants c_i and c_d have the following values:

- $c_d = 0.82$,
- $c_i = 1/0.82$.

The advantage of the strategy (1+1) is the speed of the operation, resulting from the fact that the population consists of only one individual. Its disadvantage is low resistance to local minima.

Strategy (μ+λ)

In strategy (μ+λ), a population consisting of μ individuals is processed. Each individual consists of two chromosomes. One contains a vector of independent variables **X** (representing a point in the search space), the second a vector of standard deviation values **σ** that is used by the mutation operator. A crossing operator exists apart from the mutation operator. The selection operator, by sampling with a replacement, from the baseline population of size μ creates a parents' population of individuals of the size λ (hence the strategy name). Exactly as in the case of genetic algorithms, the probability of drawing a specific individual is proportional to its objective function value. The strategy (μ+λ) pseudocode is shown below.

```
t ← 0;
initialisation of P^t population;
for each i ∈ P^t do compute f(i);
repeat
    T^t ← (P^t) selection;
    O^t ← (T^t) crossing and mutation;
    for each i ∈ O^t do compute f(i);
    P^(t+1) ← μ of the best individuals from P^t ∪ O^t;
    t ← t + 1;
until (not stop-condition (t, P^t));
```

The baseline population, as in the case of genetic algorithms, is randomly created. The process of mutation may be divided into three stages. In the first stage, a number of the normal distribution N(0,1) is drawn. In the second, each individual's standard deviations change according to the relationship:

$$\sigma_i' = \sigma_i e^{\left(\tau' \xi_{N(0,1)} + \tau \xi_{N(0,1),i}\right)} \tag{4}$$

where: τ and τ' are the algorithm's parameters, which values are expressed as follows:

$$\tau = \frac{K}{\sqrt{2n}} \tag{5a}$$

$$\tau' = \frac{K}{\sqrt{2\sqrt{n}}} \tag{5b}$$

The value of K is usually taken as equal *1*, while n is equal to the dimension of a decision space.

Having updated the values of vector **σ,** the values of the independent variables' vector **X** are modified:

$$X'_i = X_i + \sigma'_i \xi_{N(0,1),i} \tag{6}$$

The advantage of a mutation operator in strategy (μ+λ) compared with strategy (1+1) is the lack of an arbitrary value characterising the mutation (success number). The effect of a mutation range adaptation in strategy (μ+λ) is a consequence of the selection mechanism (favouring better individuals). The crossing operator may be executed in different ways. The method commonly used is that, which evaluates the mean value of the chromosomes of parents. Random number of the uniform distribution of an interval [*0, 1*] is used, and is denoted as $\xi_{U(0,1)}$. Descendant individuals are created according to the following relationships:

$$a = \xi_{U(0,1)} \tag{7}$$

$$X'^1 = aX^1 + (1-a)X^2 \tag{8a}$$

$$X'^2 = aX^2 + (1-a)X^1 \tag{8b}$$

$$\sigma'^1 = a\sigma^1 + (1-a)\sigma^2 \tag{8c}$$

$$\sigma'^2 = a\sigma^2 + (1-a)\sigma^1 \tag{8d}$$

Contrary to genetic algorithms, all individuals are subjected the mutation and crossing procedures. The baseline population for the next algorithm's iteration is created by selecting μ individuals from the total of the baseline population and descendant populations.

Strategy (μ,λ)

That strategy is similar to the strategy (μ+λ). The only difference is that the baseline population in the next algorithm's iteration is created by selecting μ individuals from the descendant population only.

4. PARTICLE SWARM OPTIMIZATION METHOD

The particle swarm optimization (PSO) method is based on the mechanisms observed in the nature. However, contrary to GA and EA algorithms, is not based on the evolution theory but on the behaviour of the individuals' population. Particles (identified with the solutions of the problem considered) traverse the decision space (the area inhabited by the population) following the particle representing the best hitherto behaviour, at the same time remembering the best position, in which they have been so far. Each particle is described by two vectors: the position vector and velocity vector. In each algorithm's iteration, a new velocity vector is determined and the change of the particle position occurs based on it. The algorithm's pseudocode is shown below.

```
k ← 0;
initialisation of R^k swarm;
repeat
        for each i ∈ R^k do compute f(i);
        update p^g;
        for each i ∈ R^k do
                update p^i;
                determine velocity vector v_k^i;
                determine position vector x_k^i;
        k ← k + 1;
until (not stop-condition (t, R^k));
```

The swarm's initialisation consists in giving the particles a random position and velocity. The position should be sampled from the permissible area. The size of this area should be considered when sampling the velocity. If the velocity is too low, the swarm will not be able to search the entire permissible area; while excessively high velocity makes, the particles 'bumps' against the limits. The velocity vector changes according to the relationship:

$$v_{k+1}^i = wv_k^i + c_1 r_1 \left(p^g - x_k^i\right) + c_2 r_2 \left(p^i - x_k^i\right) \quad (9)$$

where: x_k^i and v_k^i are the position and velocity of the *i-th* particle in the *k-th* iteration, respectively; p^g defines the best position found so far by the whole swarm; p^i is the best solution found so far by the *i-th* particle; w is defined as the inertia coefficient; c_1 and c_2 are acceleration coefficients (called also training coefficients); r_1 and r_2 are random numbers from the [*0, 1*] interval of the uniform distribution.

The new particle's position is defined as follows:

$$x_{k+1}^i = x_k^i + v_{k+1}^i \quad (10)$$

After displacement of all particles to their new position, they are subjected to an assessment and the swarm leader is chosen.

The determination of the coefficients values affects the swarm behaviour. The value of the inertia coefficient is usually selected from the [*0, 1*] interval. A higher value is favourable for the global searching of the solution space, and a lower value for the local searching. Usually, its value is constant throughout the entire optimization process. However, it also may change. Then, at the beginning, it assumes a high value, enabling global searching, and while approaching to the maximum that is sought, it gradually decreases. Acceleration coefficients are usually equal and selected from the [*0, 2*] interval. When selecting their values, the maximum velocities, which the particles should not exceed, should be considered. The exceeding of the maximum number of iterations or obtaining a satisfactory solution is taken as the criteria of the computation completion (stop criteria).

5. SIMULATED ANNEALING ALGORITHM

Contrary to the optimization methods described above, the simulated annealing (SA) algorithm (Cytowski, 1996) is not inspired by live creatures' behaviour, but rather by the process of metal or metal alloy annealing. Annealing consists of heating a material up to a defined temperature, maintaining it at this temperature for some time and then slow cooling. The simulated annealing algorithm searches a minimal solution of the objective function (GA and EA search the maximal solution). In the simulated annealing algorithm (as in strategy (1+1)), there is only one current solution. Based on it, new solution is created in each iteration, which – depending on the objective function value – may become the current solution for the next iteration. If the newly created solution is better than the previous one, it always becomes the current solution, and when it is worse, it becomes the current solution with some probability. This probability depends on the difference in the objective function values and on the algorithm's parameter that is identified with

the temperature of the annealing process and is expressed by the equation:

$$p = e^{\frac{\Delta f}{T}} \tag{11}$$

where: Δf is the difference between the objective function values of the current solution and of the newly created solution; T is the current temperature.

If the newly created solution has a lower value of the objective function (is better), then the probability that it will become the current solution in the next iteration is higher than 1. The algorithm's pseudocode is as follows:

```
k ← 0;
X^k initialisation;
T^k ← T^max;
repeat
        Y^k ← determine (X^k,T^k);
        if (random([0,1]) < exp((f(X^k) – f(Y^k)) / T^k))
                X^{k+1}←Y^k;
        else
                X^{k+1} ← X^k;
        T^{k+1} ← reduce (T^k);
        k ← k + 1;
until (not stop-condition (t, X^k));
```

The new solution is drawn from a certain neighbourhood X^k, and its value is proportional to the temperature T^k. A normal distribution of the expected value X^k and standard deviation T^k, or uniform distribution, may be used.

The temperature is decreased in each algorithm's iteration. There is a number of temperature decreasing methods. However, usually the temperature is decreased linearly in the following way:

$$\begin{aligned} T^{k+1} &= \alpha T^k \\ \alpha &\in (0,1) \end{aligned} \tag{12}$$

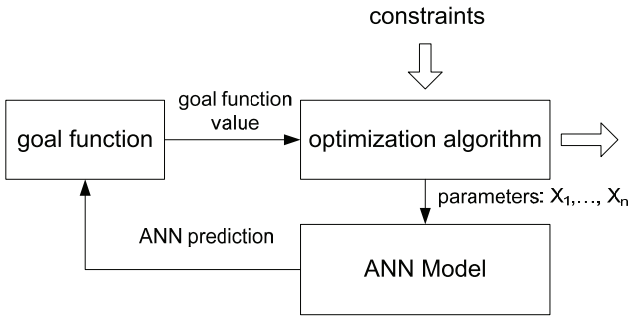


Fig. 1. *Optimization flow.*

6. OPTIMIZATION RESULTS

The optimization of the flash smelting furnace was performed on the basis of its model created using the artificial neural networks. The optimization flow chart is shown in figure 1.

Because the model used in the considered flash smelting furnace is a static one, the optimization procedure can be considered as the static optimization problem. The main goal of the optimization was the determination of the values of eighteen control signals ensuring the achieving of the preset values of the SO_2, CO_2 and NO_x concentrations in the exhaust gases for a specific concentrate's (input materials) composition. The optimization was carried out for the following concentrate's composition:

- C_{org} analysis: 7.19
- Cu analysis: 28.17
- S analysis: 12.54
- Pb analysis: 1.91
- SiO_2 analysis: 17.87
- CaO analysis: 5.8
- H_2O content: 0.32
- minus mesh fraction: 49.8
- plus mesh fraction: 0.2

The following preset values of concentrations in the exhaust gases were assumed:

- SO_2: 16.5 [%]
- CO_2: 45 [%]
- NO_x: 900 [ppm]

The objective function has the following form:

$$\varepsilon = \sqrt{\frac{1}{3}\sum_{i=1}^{3}\left(\frac{y_i - y_i^*}{y_i^*}\right)^2}\,100[\%] \tag{13}$$

where: y_i is the current value of *i-th* output of the model; y_i^* is the preset value of *i-th* output of the model.

All of the optimization methods described above have been implemented and used to determine the optimum values of the signals controlling a copper flash smelting process. The obtained results of heuristic optimization methods were compared with results obtained using the trust region method (TR) (Moor & Sorensen, 1983). In that last method, in each iteration, the objective function is approximated in a certain neighbourhood of the current point (called trust region) by another, simpler function. Then, the approximated function is minimised in the trust region. The minimum found becomes the current point for the next iteration. Because of simplicity, an approximation of the following type is normally used:

$$f(x)=\frac{1}{2}x^T Hx+g^T x+c \qquad (14)$$

where: H – Hessian of the approximating function, g – gradient of the approximating function, c – constant.

The block limitations, according to the technological requirements, were imposed on all decision variables. The stop criteria were as follows:

- the optimization accuracy $\varepsilon < 1\%$,
- the maximum iterations number N_{max}.

For individual optimization algorithms, the parameters were as follows:

Algorithm	Parameters	
GA	Maximum iterations' number	100
	Genes number	10
	Individuals' number in the population	20
	Mutation probability	0.05
	Crossing probability	0.8
	Crossing method	One-point crossing
PSO	Maximum iterations' number	100
	Particles' number in the swarm	20
	Acceleration coefficient c_1	1
	Acceleration coefficient c_2	1
	Inertia coefficient	0.8
(1+1)	Maximum iterations' number	1000
	Initial mutation range	1
	Modification of the mutation range with respect to the number of iterations	50
(μ+λ), (μ,λ)	Maximum iterations' number	100
	Individuals' number in the baseline population	20
	Individuals' number in the parents' population	40
	Crossing method	Averaging crossing
SA	Maximum iterations' number	1000
	Initial temperature	1000
	Temperature decrease coefficient α	0.9
	Temperature decrease, each iteration	10
	New solution generation method	$Y^k = N(X^k, T^k)$
TR	Maximum iterations' number	1000
	Starting point	randomly selected from the permissible area

For each algorithm, the optimization was performed ten times. Examples of results obtained by different optimization procedures are given in table 1, while graphs of the optimization error in the consecutive iterations of considered optimization procedures are shown in figure 2.

The optimal values of the control signals, found by using described optimization methods, substantially differ from each other. Based on that fact, it is possible to conclude that the objective function has local minima. The values marked in bold are permissible from the technological point of view, however these values may occur only temporarily. In case of the implementation of the control system, respective limits for these parameters should be narrowed.

Minimum, average and maximum values of the optimization error and the number of objective function calls for each method are given in table 2 and are graphically presented in figures 3 and 4.

It is seen that the smallest errors of the optimal values of the control signals were obtained by using the simulated annealing algorithm and additionally at the smallest average number of objective function calls. A low number of objective function calls was also observed for the strategy (1+1), but the optimization errors were higher. A low number of the objective function calls resulted from the fact that both algorithms did not process the solutions' population but only one current solution. The results obtained by using the particle swarm optimization and the genetic algorithm were similar, both in terms of the number of the objective function calls and of the obtained errors. The worst results were obtained by using evolutionary algorithms. In strategy (1+1), the number of objective function calls was low, indeed, but the optimization errors were high. Strategy (μ+λ) produced low optimization errors, but the number of objective function calls was high. Strategy (μ,λ) turned out to be the worst. The optimization error seldom fell below 1%, and hence the average number of the objective function calls was that high. The trust region method gave results slightly better than the simulated annealing method. However, the number of the objective function calls was higher.

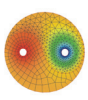


Table 1. *Optimization results.*

		GA	PSO	(1+1)	(μ+λ)	(μ,λ)	SA	TR
Decision variables value	Concentrate feed in burner 1	21.65	24.88	28.91	28.09	26.47	28.43	27.45
	Concentrate feed in burner 2	29.03	26.73	29.53	24.09	20.41	20.87	20.18
	Concentrate feed in burner 3	21.74	25.32	28.92	29.24	24.53	27.14	22.18
	Concentrate feed in burner 4	24.13	23.27	27.23	24.12	24.48	22.71	23.89
	Dusts feed	10.23	**18.60**	**14.89**	11.66	8.89	**19.15**	**4.00**
	IOS feed in burner 1	0.63	1.09	0.47	0.09	0.19	0.43	**0.26**
	IOS feed in burner 2	0.14	0.75	0.32	0.21	0.00	0.00	**0.02**
	IOS feed in burner 3	0.60	0.66	1.34	0.64	**1.50**	0.00	**0.62**
	IOS feed in burner 4	1.06	0.27	0.02	1.32	1.37	1.09	**0.00**
	IOS feed in dust burner	0.10	0.00	0.00	0.29	**1.98**	**2.02**	**0.25**
	Oil into the reaction shaft	82.92	68.92	80.36	80.58	76.92	96.36	66.61
	O_2 concentration in the blast	79.77	82.54	78.44	82.15	68.18	82.11	82.46
	Overoxidation	235.82	252.08	246.97	261.71	297.06	**324.21**	225.66
	Aeration in burner 1	260.04	251.90	254.68	244.80	**168.93**	269.99	**166.43**
	Aeration in burner 2	**166.34**	193.42	185.88	214.97	219.22	224.81	**162.57**
	Aeration in burner 3	185.68	226.95	193.22	**163.04**	258.74	**165.52**	**160.30**
	Aeration in burner 4	172.27	198.97	193.64	180.63	225.42	247.18	**163.31**
	Aeration in dust burner	247.06	289.52	298.69	188.93	241.39	299.27	**159.18**
Outputs	SO_2	16.5588	16.5288	16.5005	16.3828	16.7344	16.5679	16.5297
	CO_2	44.5917	44.7069	44.1335	44.8468	44.7061	44.8148	45.0351
	NO_x	903.8781	896.2734	897.9237	910.4171	908.8353	910.4653	902.9310
Optimization error		0.6153	0.457	1.1198	0.8083	1.066	0.7507	0.2196
Number of objective function calls		1240	600	1001	2180	4020	702	793

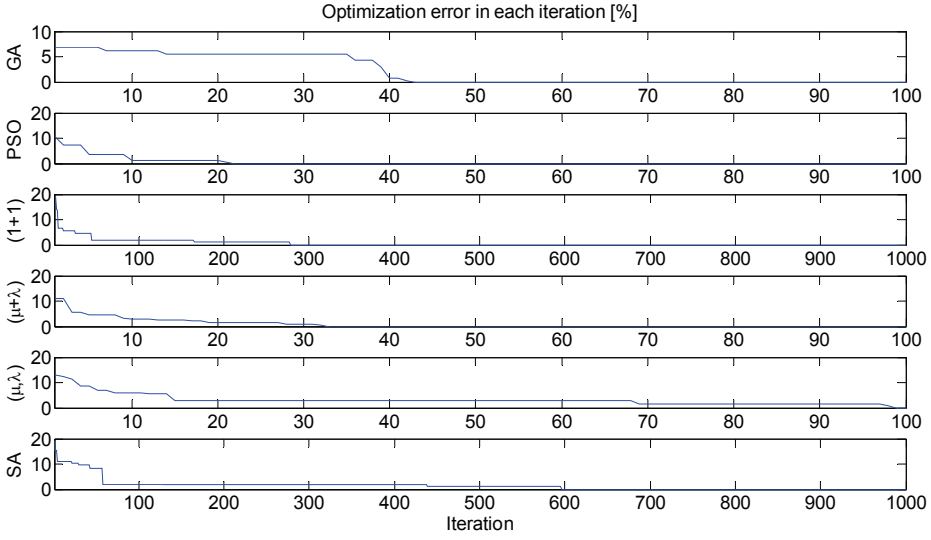


Fig. 2. *Values of the optimization error in consecutive iterations.*

7. SUMMARY

The main goal of the work was an attempt of application of chosen bio-inspired methods in the optimization of industrial processes. The objective was optimization of the contents of chosen parameters of the exhaust gases of the copper flash smelting process. The results obtained by the considered bio-inspired algorithms were validated with the results of one of deterministic methods, which was the trust region method. From the technological point of view, the best results of the control signals were obtained from the genetic algorithm, strategy (1+1) and (μ+λ). Nearly all obtained optimal values of the control signals are permissible. Obtained results confirm that the bio-inspired optimization methods are effective in a case of solving complex optimization problems.

Table 2. *Minimum, average and maximum values of the optimization error and numbers of the objective function calls of different methods.*

Algorithm		ε [%]	Number of objective function calls
GA	min	0.2542	280
	ave	1.08442	1468
	max	1.8164	2020
PSO	min	0.3527	340
	ave	0.73108	908
	max	1.8169	2020
(1+1)	min	0.645	223
	ave	1.37915	705.8
	max	2.314	1001
(μ+λ)	min	0.5463	260
	ave	0.78505	1604
	max	1.2525	4020
(μ,λ)	min	0.9081	3900
	ave	1.40822	4008
	max	2.0228	4020
SA	min	0.4561	561
	ave	0.71009	718.2
	max	0.9696	896
TR	min	0.07	459
	ave	0.52	960
	max	0.87	1717

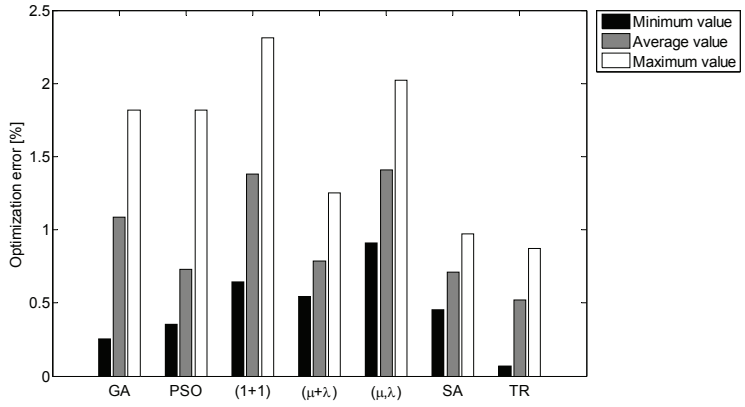


Fig. 3. *Minimum, average and maximum values of the optimization error of each method.*

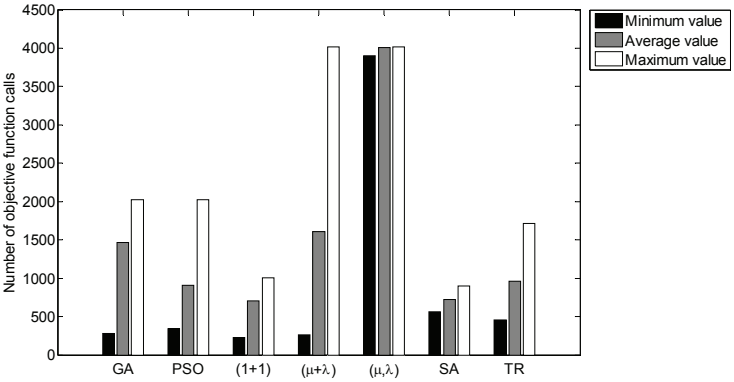


Fig. 4. *Minimum, average and maximum values of the number of objective function calls for each method.*

ACKNOWLEDGEMENTS

The financial support of the MNiSzW, project No 3 T08B 034 30 is acknowledged.

References

1. Arabas, J., 2001, *Wykłady z algorytmów ewolucyjnych*, Wydawnictwa Naukowo-Techniczne, Warszawa.
2. Cytowski, J., 1996, *Algorytmy genetyczne. Podstawy i zastosowania*, Akademicka Oficyna Wydawnicza PLJ, Warszawa.
3. Goldberg, D.E., 1995, *Algorytmy genetyczne i ich zastosowania*, Wydawnictwa Naukowo-Techniczne, Warszawa.
4. Gwiazda, T.D., 2007, *Algorytmy genetyczne. Kompendium*, t.1,2 Wydawnictwa Naukowe PWN SA, Warszawa.
5. Kusiak, J., 2009, Copper flash smelting process. Modelling and control, *Computer Methods in Material Science*, 9, 3, 362-369.
6. Moor, J.J., Sorensen, D. C., 1983, Computing a Trust Region Step, *SIAM Journal on Scientific and Statistical Computing*, 3, 553-572.
7. Stanisławczyk, A., Kusiak, J., 2009, Neural network modelling of a copper flash smelting process gas phase, *Computer Methods in Material Science*, 9, 3, 376-381.

OPTYMALIZACJA PROCESU ZAWIESINOWEGO WYTOPU MIEDZI OPARTA O METODY INSPIROWANE PRZEZ NATURĘ

Streszczenie

Artykuł przedstawia metody optymalizacji inspirowane występującymi w przyrodzie mechanizmami oraz ich zastosowanie w wyznaczaniu wartości sygnałów sterujących piecem zawiesinowym do wytopu miedzi. Stosowany w procesie optymalizacji model pieca, stworzony został w oparciu o sztuczne sieci neuronowe. Celem sterowania było uzyskanie pożądanych wartości stężeń SO_2, CO_2 oraz NO_x w gazach wylotowych przy danym składzie koncentratu. Proces optymalizacji opierał się na statycznym modelu pieca, dlatego zagadnienie sterowania sprowadziło się do problemu optymalizacji statycznej. Na wszystkie zmienne decyzyjne zostały nałożone ograniczenia kostkowe. W pracy przedstawiono podstawy teoretyczne każdej metody, sposób implementacji oraz uzyskane przy jej zastosowaniu wyniki. Dodatkowo wyniki te porównano z wynikami otrzymanymi jedną z metod deterministycznych.

Submitted: May 06, 2009
Submitted in a revised form June 06, 2009
Accepted: June 17, 2009