Solving Steel Alloying Using Differential Evolution and SOMA

Michal Holiš, Lenka Skanderová, Martin Placěk, Jiří Dvorský, Ivan Zelinka

Solving Steel Alloying Using Differential Evolution and SOMA

Michal Holis, Lenka Skanderova, Martin Placek, Jiri Dvorsky, and Ivan Zelinka

Department of Computer Science, VSB – Technical university of Ostrava,
17. listopadu 15, 708 33 Ostrava – Poruba, Czech republic,
{michal.holis, lenka.skanderova, martin.placek,
jiri.dvorsky, ivan.zelinka}@vsb.cz

Abstract. This paper proposes method for solving steel alloying problem using evolution algorithms SOMA and differential evolution. Both algorithms belong to the family of the evolution algorithms but the main ideas of these algorithms are different. In differential evolution new offspring is created during the evolution, the individuals are crossed and mutated, while in SOMA the individuals move in the space of the possible solutions and the mutation is replaced by perturbation. The main goal of this paper is to discover how much these algorithms are usable and suitable to solve the problem of steel alloying.

Keywords: steel alloying, evolutionary algorithms, Differential evolution, SOMA

1 Introduction

1.1 Steel Alloying

It takes many steps to manufacture steel and every factory's step configuration may vary from the others. Commonly the process follows path similar to this: iron ore is smelted in iron smelters and distributed as a hot metal with temperature about 1400°C to converter or electric arc furnace to improve its quality for following treatments. Hot metal is mixed with scrap and slag formers, the heat is heated up approximately to 1650°C and first ferro-alloys are charged into the heat during tapping. This point is the first appliance of alloying process described below. Ladle with the heat is going to secondary metallurgy process, typically ladle furnace or vacuum treatment. This treatment part is exceptionally skipped, otherwise the steel quality is improved with accurate amount of alloys to achieve requested steel composition defined by steel grade and the temperature is modified for the last step. The final step of this process is casting on continuous casting machines to obtain final steel product like slabs, blooms or billets. More on this process can be found for example in book by Ahindra Ghosh and Amit Chatterjee [12].

Target of the whole process is primarily to manufacture steel of requested quality (this is commonly referred to as target or final steel grade), secondly to reduce cost of the materials and energy used in the process.

To manufacture steel of certain grade, with specific chemical, electrical and mechanical properties, it is crucial to be sure that the steel is composed of correct elements in correct ratio's. Composition of the steel is modified in each step throughout the whole process with technique known as alloying.

Alloying is process of continually charging the steel with certain amount's of alloying materials (compounds of chemical elements) to gradually obtain the requested steel quality. Each of the materials has certain given ratio of all elements it is composed of. Also price of the material is known. Question is how to optimally choose amount of charged materials in each step when there are many charging materials composed of many chemical elements, with one element being commonly present in many materials. We also have to consider price of the final solution, since some materials are very cheap and some can be very expensive.

To solve the problem we are always presented with these input data:

- weight of the input steel,
- input steel's composition vector (vector of ratios representing each element's presence in steel),
- list of alloying materials, their chemical solution and their unit prices,
- target steel grade represented with three vectors – one being optimal ratio of all chemical elements in final steel, second represents minimal ratio of any given element in final steel and last one represents maximal ratio of any given element in final steel.

Commonly in practice this problem is solved with Dantzig's simplex algorithm, which is a linear programming solution that with given input parameters finds cheapest solution in given final steel's grade range. This however does not allow us to fine-tune optimal balance between precision of final solution and it's price. This can be solved by iteratively running the algorithm with different steel grade ranges and trying to found solution that we consider to be optimal both in precision and cost. However since the ranges are vectors there are so many combinations of minimal and maximal values that it is very inefficient and unpractical to proceed in this way. More on this matter can be found in book by Dantzig, George B. [13].

In this paper, we present how to overcome problems of the common solution with usage of evolution algorithms and fitness function suitable for solution of this problem is presented.

1.2 Evolution algorithms

The evolution algorithms have been successfully used to solve many practical and theoretical problems, see [9, 15, 16]. In [4] the differential evolution is used as classifier for the features in the data set, in [1] the algorithm SOMA is used to machining allocation of clutch assembly. Evolutionary techniques are applied in connection with, e.g. [2, 5, 6], too. These algorithms are still improved, see [7, 8]. They are based on two essential principles – crossing and mutation. These principles proceed from Darwin's evolution theory and Mendel's principle of

crossing. They work with the population of individuals, which are developing (improving) during the evolution. Better individuals survive while worse ones die. In this paper differential evolution and SOMA have been chosen.

Differential evolution Differential evolution (DE) is a population – based stochastic algorithm for global optimization. It was introduced by Ken Price and Rainer Storn and in this paper the original version is used for experiment design [11]. Although this algorithm is very simple and efficient, in [17] authors proved that it can not ensure the global convergence and they proposed two hybrids algorithms named QAISDE and GDISE to improve DE.

At the beginning of the algorithm the first population is generated. Each individual in population has its own set of parameters. The number of parameters is called dimension, we denote it D. In addition each individual has its fitness value. This value is very important because it says how good is this individual in current population. Fitness is the value of the cost function. Each parameter of the individual has low bound and high bound. It is not able to cross over these bounds [18].In this article DE/best/1/exp has been chosen.

When the first population is generated, for each individual in population following steps are done:

1. Three other neighbors are chosen randomly (these individuals may not be the same).
2. The noise vector is created, see Eq. (1).
3. New individual is created. The random number from the interval $[0, 1]$ is generated. If the number is smaller than crossing threshold CR, the parameter from the noise vector is chosen as a parameter of the new individual. Otherwise the parameter from the actual parent is taken.
4. If the new individual has better fitness than its parent, it will replace the parent, otherwise the new individual is forgotten and the parent stays in the population.

The noise vector equation of DE/best/1/exp:

$$v_j = x^G_{r3,j} + F\left(x^G_{r1,j} - x^G_{r2,j}\right). \tag{1}$$

where v_j is the noise vector, $x^G_{r3,j}$, $x^G_{r1,j}$ and $x^G_{r2,j}$ are three randomly chosen individuals and F denotes the mutation constant.

Self organizing migration algorithm (SOMA) Unlike differential evolution, SOMA works on the principle of one population, which is changing in time. Application of this algorithm can be found for example in [19, 20]. The beginning is the same like in DE. The first population is generated. But in SOMA no offspring is created. The principle of SOMA is the individuals are moving in the space of possible solutions and in the each generation (we say migration) they have a new positions. In [21] the novel multiobjective SOMA has been introduced.In this paper the strategy All to one has been chosen. At the begin the

new population is generated. As well as in DE, each individual has its parameters and fitness, low and high bound. When we have the population the best individual is chosen (individual with the best fitness) – we call it leader. The following steps are done next for each individual in population:

1. The perturbation vector $\boldsymbol{\alpha}$ is generated according to the parameter PRT, which is usually set to 0.11. The $\boldsymbol{\alpha}$ is composed of 0 and 1 and it is important for the direction of the individual moving. (Individual can move to the leader in many directions.)
2. Individual will move to the leader. It moves by steps. For each step the next position and the new fitness is computed. Moving is finished when the parameter $Step$ spreads the value of the parameter $PathLength$.
3. After migration of the individual its best position, which has been reached during the migration, is chosen.
4. If the best reached position of the individual is better than the original one, the original one is replaced by the new position.

The crossing is replaced by philosophy that each individual moves in the space and each individual remembers the coordinates of the position, where the resolution has been the best in scope of its way to the leader. Moving of the individual is directed by the equation below [10].

$$x_{i,j}^{ML+1} = x_{i,j,start}^{ML} + \left(x_{L,j}^{ML} - x_{i,j,start}^{ML}\right) t\boldsymbol{\alpha}_j \tag{2}$$

where $x_{i,j}^{ML+1}$ is a new position of the individual, $x_{i,j,start}^{ML}$ denotes the individual's start position, $x_{i,j,start}^{ML}$ is the actual leader's position, $Step$ is marked as t and $\boldsymbol{\alpha}$ denotes the value in j^{th} position of $\boldsymbol{\alpha}$.

2 Experiment design

Our method is based on SOMA and DE using in steel alloying process. There were two main problems in experiment design - how to represent an individual in evolutionary algorithms and how to define its fitness function. From the view of steel alloying process, there were 14 compounds, which are added to the original steel composition. These compouds are consist of the elements. In result we needed the elements amount. In evolution algorithms the individuals consist of parameters. In this paper the parameters of individuals are represented by the compounds, which were added to the original steel composition. There were 14 compounds, so the individual's dimension (number of parameters) D has been 14. Each compound has been represented by one parameter in individual. In the end of the evolution the amount of the elements have been recomputed from the amount of compounds.

Fitness function presented in this paper is designed in the following manner:

– parameters of each member in population represent weight of each alloying material that is to be charged to the steel (every member has as many parameters as there are alloying materials),

- first the function computes weight of each element in input steel using it's weight and it's composition vector,
- using the value of each parameter of current member and composition vector of material it represents we can compute the weight of each chemical element that would be added to the steel, if materials the member represents would be charged to the steel,
- using weights of elements in original steel and computed weights of elements in charged materials the final presence of each material in final steel is computed,
- value of the fitness function is then equal to absolute distance of the optimal solution vector and this solution's vector, if this vector is out of range represented by minimal and maximal vector of solution, then penalization is applied to the fitness value,
- additionally price of all used materials is computed and added to the fitness value, to do this constant defining importance of price of final solution over it's accuracy is applied, with this constant we can control and fine-tune the solution to our specific needs.

2.1 Fitness Function Equation

If $Cmin_j < \left| \frac{M_i \cdot Mat_{ij} + S_j \cdot W}{\sum_{k=0}^{n_m} M_k + W} - C_j \right| < Cmax_j$ then fitness function is defined as follows (this is the case when given member represents valid solution that is in required composition range).

$$\sum_{i=0}^{n_m} \left[\left(\sum_{j=0}^{n_e} \left| \frac{M_i \cdot Mat_{ij} + S_j \cdot W}{\sum_{k=0}^{n_m} M_k + W} - C_j \right| \right) + P_i \cdot M_i \cdot PW \right] \tag{3}$$

Otherwise we apply additional penalization constant to equation to discard the member from solution.

$$\left\{ \sum_{i=0}^{n_m} \left[\left(\sum_{j=0}^{n_e} \left| \frac{M_i \cdot Mat_{ij} + S_j \cdot W}{\sum_{k=0}^{n_m} M_k + W} - C_j \right| \right) + P_i \cdot M_i \cdot PW \right] \right\} \cdot T \tag{4}$$

Notation used in these equation is summarized in Table 1.

For both evolution algorithms, DE and SOMA, the same dimension has been set, $D = 14$. In DE's experiments we were changing the parameters: the number of individuals in the population NP, the number of generations G, mutation constant F, and crossing threshold CR in order to find better evaluation of the cost function. In SOMA experiments these parameters were also modified: number of the migration cycles $Migrations$, the stopping position of the migrating individual $PathLength$, and the step size of the migrating individual $Step$. All tested combinations of input parameters are given in Table 2.

From chemical point of view fourteen, $D = 14$, chemical compounds have been used: FeMn (with high carbon level), FeMn (with medium carbon level),

Table 1. The variables description from the Eqs. (3) and (4)

Variable	Description
n_m	Total number of available alloying materials (compounds) used.
n_E	Total number of chemical elements that materials consists of.
M	Individual being tested for it's fitness value. Bottom index specifies index of parameter whose value we want to obtain. Each parameter of member represents weight in kilograms (kg) of one material that would be charged to steel. Number of parameters of one member equals to number of all available alloying materials.
Mat	Material composition. First index specifies material, which's composition we are interested in. Second denotes index of element. Result is number in range $[0, 1]$ (1 means 100% of material is composed with that element, 0 that element is not contained in compound) that represents element's representation in given material (compound).
S	Original steel composition vector. Index denotes element whose representation we are interested in. Resulting number is again in range $[0, 1]$.
W	Original steel's weight in kg.
C	Desired composition vector. Index denotes element whose representation in steel we want to obtain. Resulting number is again in range $[0, 1]$.
$Cmin$	Desired minimal composition vector. Index denotes element whose representation in steel we want to obtain. Resulting number is again in range $[0, 1]$.
$Cmax$	Desired maximal composition vector. Index denotes element whose representation in steel we want to obtain. Resulting number is again in range $[0, 1]$.
P	Unit price of material for each kg.
PW	Constant defining importance of final solution's cost over it's precision.
T	Penalization constant. Very high number that is used to multiply the final fitness function's value in case the solution is outside the $[Cmin, Cmax]$ range.

FeMn (with low carbon level), FeSiMn, FeCr (with high carbon level), FeCr (with low carbon level), FeNi, FeMo, FeW, FeNb, FeV, NV, FeTi and Cu. Composition of these compounds can be found in Table 4. For each element the minimum value, maximum value and optimal value in final steel alloy have been set, see Table 3.

Table 2. Experiments setting for algorithms DE and SOMA.

(a) Experiments setting for DE.

D	NP	G	F	CR
14	1000	1500	0.8	0.6
14	1000	1500	0.8	0.5
14	1000	1500	0.7	0.6
14	1000	1500	0.7	0.5
14	1000	1500	0.5	0.7
14	1000	1500	0.5	0.6
14	1000	1500	0.5	0.5
14	500	1000	0.8	0.6
14	500	1000	0.8	0.5
14	500	1000	0.7	0.6
14	500	1000	0.7	0.5
14	500	1000	0.5	0.7
14	500	1000	0.5	0.6
14	500	1000	0.5	0.5

(b) Experiments setting for DE.

D	NP	G	F	CR
14	50	2000	0.8	0.6
14	50	2000	0.8	0.5
14	50	2000	0.7	0.6
14	50	2000	0.7	0.5
14	50	2000	0.5	0.7
14	50	2000	0.5	0.6
14	50	2000	0.5	0.5
14	50	1500	0.8	0.6
14	50	1500	0.8	0.5
14	50	1500	0.7	0.6
14	50	1500	0.7	0.5
14	50	1500	0.5	0.7
14	50	1500	0.5	0.6
14	50	1500	0.5	0.5

(c) Experiments settings for SOMA ALLToOne.

D	NP	$Migrations$	$PathLength$	$Step$	PRT
14	1000	200	3	0.11	0.1
14	1000	200	3	0.3	0.1
14	1000	200	3	0.1	0.1

3 Experimental Results

Our experimental results are summarized in Table 5. Highlighted rows represent runs that were unable to reach satisfying solution. First column of the table contains description of given settings along with it's parameters. Second column contains best achieved fitness value of all runs and next columns contain information about final computed solution of the steel. Progress of the best fitness values is show in Figures 1 and 2. One hundred test runs using given setting were performed and each test run is displayed as one line in these charts. Only selection of plots of runs that were able to reach satisfying solution are presented in this paper.

Table 3. The table of the chemical elements whose amounts we need to know. The amounts are stated in % and are set for 1,000 kilograms of the steel.

	Values [%]		
Element	Minimal	Optimal	Maximal
C	0.003	0.035	0.040
Si	0.065	0.075	0.085
Mn	0.952	0.962	0.972
P	0.000	0.001	0.006
S	0.000	0.001	0.002
Cr	1.268	1.288	1.308
Ni	0.940	0.960	0.980
Mo	0.000	0.000	0.010
Cu	0.000	0.000	0.005
Nb	0.000	0.000	0.002
V	0.000	0.000	0.002
Al	0.000	0.003	0.013
Ti	0.000	0.000	0.005
N	0.000	0.000	0.005
W	0.000	0.000	0.005
Fe	96.671	96.676	96.681

Table 4. Materials and their solution used in our experiments. Column *Material Code* contains code of material and in other columns presence of each element in percents is shown.

	Element Values [%]															
Material Code	C	Si	Mn	P	S	Cr	Ni	Mo	Cu	Nb	V	Al	Ti	N	W	Fe
FeMnHC	7	6	78	0.05	0.02	0	0	0	0	0	0	0	0	0	0	8.93
FeMnMC	2	3	88	0.1	0.02	0	0	0	0	0	0	0	0	0	0	6.88
FeMnLC	0.5	1.8	90	0.05	0.02	0	0	0	0	0	0	0	0	0	0	7.63
FeSiMn	0.5	25	65	0.05	0.02	0	0	0	0	0	0	0	0	0	0	9.43
FeCrHC	7	1.5	0	0.03	0.06	65	0	0	0	0	0	0	0	0	0	26.41
FeCrLC	0.05	1.5	0	0.03	0.02	67	0	0	0	0	0	0	0	0	0	31.4
FeNi	0.02	0	0	0	0	0	99.8	0	0	0	0	0	0	0	0	0.18
FeMo	0.05	0.8	0	0.05	0.1	0	0	60	0	0	0	0	0	0	0	39
FeW	0.1	0.8	0.2	0.03	0.02	0	0	6	0.1	0	0	4	0	0	80	8.75
FeNb	0.1	2	0	0.08	0.08	0	0	0	0	65	1.5	0.8	0	0	0	30.44
FeV	0.2	1.5	5	0.05	0.05	0	0	0	0	0	60	0	0	0	0	33.2
NV	5	0.2	0.5	0.05	0.5	0.2	0	0	0.15	0	75	0.3	0	16	0	2.1
FeTi	0.2	0.5	0.1	0.05	0.05	0	0	0	0	0	0	5	70	0	0	24.1
Cu	0	0	0	0	0	0	0	0	99	0	0	0	0	0	0	1
Al pigs	0	0.15	0	0	0	0	0	0	0.5	0	0	99.2	0.04	0	0	0.11
Al pigs	0	1	0	0	0	0	0	0	0	0	0	97	0	0	0	2
C	98	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2

From the results measured in our experiments we can see, that by using higher values then 0.5 for F in DE the evolution requires higher number of generations to successfully reach valid solution. With $F = 0.5$ we were able to reach satisfying solution even with only 50 members in whole population. Although some of the runs were unsuccessful and were unable to present valid solution, the runs that completed successfully yielded correct result reliably and we have not recorded single case when they would fail.

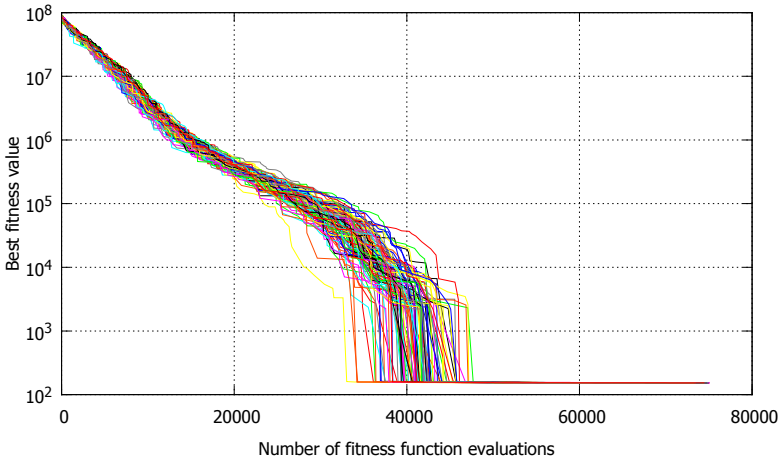

Fig. 1. Differential Evolution (NP: 50, G: 1500, F: 0.5, CR: 0.7)

4 Conclusion

We have proposed alternative way to compute steel alloying recipe. Method using evolution algorithms has several advantages when compared to most commonly used Simplex method. It allows to incorporate final cost of used alloying materials into equation. Result of Simplex method is always cheapest solution in given range. Using our fitness function described in Section 2 we can control the importance of precise solution of final steel over it's price.

All runs presented in our experiments that were able to achieve less then approximately 155 fitness value provide satisfactory results and are fully prepared to be used in real life environment on real life cases.

We would like to extend our work and try out more evolution algorithms to compare their results. Mainly we would like to design larger experiment to compare performance and precision of larger set of evolution algorithms on this problem.

Table 5. Experimental results.

Settings	Fitness	Alloying elements [%]															
		C	Si	Mn	P	S	Cr	Ni	Mo	Cu	Nb	V	Al	Ti	N	W	Fe
DE (1,000, 1,500, 0.5, 0.5)	153.518	0.040	0.083	0.969	0.001	0.001	1.270	0.943	0.000	0.001	0.000	0.001	0.011	0.000	0.000	0.000	96.680
DE (1,000, 1,500, 0.5, 0.6)	153.578	0.039	0.083	0.972	0.001	0.001	1.269	0.942	0.000	0.003	0.000	0.001	0.008	0.002	0.000	0.000	96.680
DE (1,000, 1,500, 0.5, 0.7)	153.689	0.040	0.085	0.957	0.001	0.001	1.273	0.949	0.000	0.003	0.000	0.002	0.010	0.000	0.000	0.000	96.680
DE (1,000, 1,500, 0.7, 0.5)	15,789.470	0.042	0.084	0.958	0.001	0.001	1.273	0.940	0.000	0.000	0.007	0.003	0.012	0.002	0.000	0.004	96.674
DE (1,000, 1,500, 0.7, 0.6)	100,347.300	0.055	0.083	0.918	0.001	0.001	1.274	0.948	0.003	0.014	0.002	0.003	0.014	0.007	0.000	0.004	96.674
DE (1,000, 1,500, 0.8, 0.5)	146,710.100	0.050	0.091	0.953	0.001	0.001	1.287	0.933	0.009	0.007	0.002	0.008	0.007	0.004	0.001	0.022	96.624
DE (1,000, 1,500, 0.8, 0.6)	334,186.400	0.075	0.105	0.947	0.001	0.001	1.195	0.916	0.007	0.015	0.003	0.026	0.028	0.013	0.005	0.009	96.656
DE (500, 1,000, 0.7, 0.5)	139,756.300	0.060	0.084	0.895	0.001	0.001	1.305	0.955	0.009	0.001	0.013	0.007	0.008	0.006	0.001	0.002	96.654
DE (500, 1,000, 0.7, 0.6)	325,398.300	0.065	0.078	0.803	0.001	0.001	1.278	0.960	0.028	0.012	0.016	0.032	0.021	0.016	0.000	0.015	96.675
DE (500, 1,000, 0.8, 0.5)	352,591.900	0.085	0.121	0.883	0.001	0.001	1.205	0.969	0.006	0.014	0.003	0.015	0.033	0.002	0.000	0.010	96.652
DE (500, 1,000, 0.8, 0.6)	926,096.500	0.209	0.169	0.895	0.001	0.001	1.144	0.706	0.035	0.042	0.009	0.037	0.022	0.024	0.006	0.021	96.678
DE (50, 1,500, 0.5, 0.5)	153.164	0.040	0.083	0.972	0.001	0.001	1.271	0.941	0.000	0.000	0.000	0.001	0.010	0.001	0.000	0.000	96.680
DE (50, 1,500, 0.5, 0.6)	153.021	0.040	0.085	0.972	0.001	0.001	1.269	0.941	0.000	0.002	0.000	0.001	0.009	0.000	0.000	0.000	96.679
DE (50, 1,500, 0.5, 0.7)	152.896	0.040	0.085	0.970	0.001	0.001	1.270	0.940	0.000	0.000	0.000	0.001	0.010	0.000	0.000	0.000	96.681
DE (50, 2,000, 0.5, 0.5)	152.956	0.040	0.085	0.971	0.001	0.001	1.268	0.942	0.000	0.001	0.000	0.001	0.009	0.001	0.000	0.000	96.681
DE (50, 2,000, 0.5, 0.6)	152.858	0.040	0.085	0.971	0.001	0.001	1.270	0.943	0.000	0.000	0.000	0.000	0.009	0.000	0.000	0.000	96.681
DE (50, 2,000, 0.5, 0.7)	152.666	0.040	0.085	0.971	0.001	0.001	1.269	0.940	0.000	0.001	0.000	0.000	0.011	0.000	0.000	0.000	96.681
SOMA (1,000, 200, 0.1, 0.1)	152.582	0.040	0.085	0.972	0.001	0.001	1.268	0.940	0.000	0.000	0.000	0.000	0.012	0.000	0.000	0.000	96.681
SOMA (1,000, 200, 0.1, 0.11)	152.717	0.040	0.085	0.972	0.001	0.001	1.268	0.942	0.000	0.000	0.000	0.000	0.011	0.000	0.000	0.000	96.680
SOMA (1,000, 200, 0.1, 0.3)	152.935	0.040	0.085	0.971	0.001	0.001	1.268	0.940	0.000	0.004	0.000	0.000	0.008	0.000	0.000	0.000	96.681
		Desired composition vectors															
Minimal composition vector		0.030	0.065	0.952	0	0	1.268	0.940	0	0	0	0	0	0	0	0	96.671
Ideal composition vector		0.035	0.075	0.962	0.001	0.001	1.288	0.960	0	0	0	0	0.003	0	0	0	96.676
Maximal composition vector		0.040	0.085	0.972	0.006	0.002	1.308	0.980	0.010	0.005	0.002	0.002	0.013	0.005	0.005	0.005	96.681

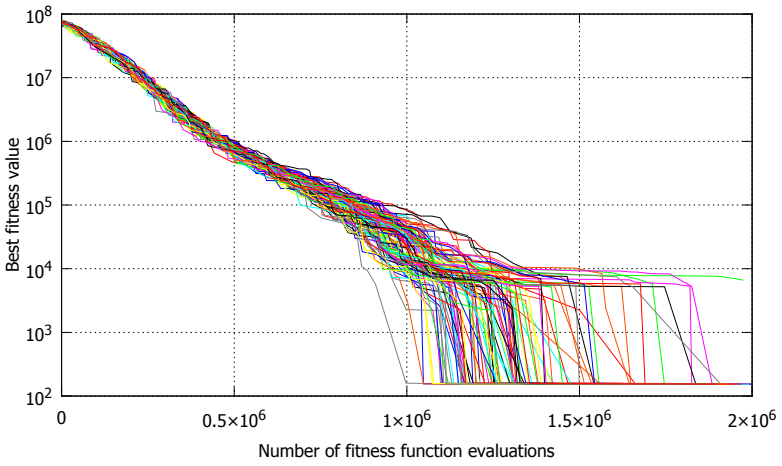

Fig. 2. SOMA (NP: 1000, $Migrations$: 200, PRT: 0.1, $Step$: 0.3)

Acknowledgement This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and by the Development of human resources in research and development of latest soft computing methods and their application in practice project, reg. no. CZ.1.07/2.3.00/20.0072 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic.

References

1. Coelho, L. dos Santos: Self-organizing migration algorithm applied to machining allocation of clutch assembly, Mathematics and Computers in Simulation, Volume 80, Issue 2, October 2009, Pages 427-435
2. Senkerik, R., Zelinka, I., Oplatkova, Z.: Evolutionary Techniques for Deterministic Chaos Control, Technological Developments in education and automation, 391-396, 10.1007/978-90-481-3656-8_71, 2010
3. De Falco, I.: Differential Evolution for automatic rule extraction from medical databases, Applied soft computing, Pages 1265 - 1283, 10.1016/j.asoc.2012.10.022, 2013
4. Koloseni, D., Lampinen, J., Lukka, P.: Differential Evolution Classifier with Optimized Distance Measures for the Features in the Data Sets, Book Advances in Intelligent Systems and Computing, Volume: 188, Pages: 103-111,2013
5. Zelinka, I., Chadli et al: An investigation on evolutionary reconstruction of continuous chaotic systems, Mathematical and Computer Modelling, Volume: 57, Pages: 2-1, 2013
6. Senkerik, R. et al: Performance comparison of differential evolution and SOMA on chaos control optimization problems, International journal of bifurcation and chaos, Volume 22, DOI: 10.1142/S021812741230025X, 2012

7. Sun, Y. et al: A hybrid co-evolutionary cultural algorithm based on particle swarm optimization for solving global optimization problems, Elsevier, Neurocomputing, Volume 98, Pages 76-89, 2012
8. Zhou, Y., Li, X., Gao, L.:A differential evolution algorithm with intersect mutation operator, Applied soft computing, Volume 13, Pages: 390 - 401, DOI: 10.1016/j.asoc.2012.08.014, 2013
9. Alguliev, R., Aligulizev R., Isazade, N.:DESAMC+DocSum: Differential evolution with self-adaptive mutation and crossover parameters for multi-document summarization, Knowledge - based systems, Pages: 21 - 38, DOI: 10.1016/j.knosys.2012.05.017, 2012
10. Onwubolu, GC., Babu, BV.: New Optimization Techniques in Engineering, Springer, ISBN 3-540-20167-X Springer - Verlag Berlin Heidelberg New York, 2004
11. Chakraborty, U., K.: Advances in Differential Evolution: Springer - Verlag Berlin Heidelberg, 2008, ISBN 978 - 3 - 540 - 68827 - 3
12. Ghosh, A., Chatterjee, A. Ironmaking and steelmaking: theory and practice. New Delhi: Prentice-Hall of India, 2008. ISBN 978-812-0332-898.
13. Dantzig, G. B., Thapa, M. N.: Linear programming. New York: Springer, c1997-2003, 2 v. ISBN 03879861382-.
14. Zhang, Y., Gong, DW., Zhang, JH: Robot path planning in uncertain environment using multi-objective particle swarm optimization, Neurocomputing, Pages 172 - 185, DOI: 10.1016/j.neucom.2012.09.019, 2013
15. Chakaravarthy, GV, Marimuthu, S., Sait, AN.:Performance evaluation of proposed Differential Evolution and Particle Swarm Optimization algorithms for scheduling m-machine flow shops with lot streaming, Journal of intelligent manufacturing, Volume 24, Pages: 175 - 191, 10.1007/s10845-011-0552-2 , 2013
16. Subramanian, P.et al: PRISM: PRIority based SiMulated annealing for a closed loop supply chain network design problem, Applied soft computing, Volume 13, Pages: 1121 - 1135, DOI: 10.1016/j.asoc.2012.10.004, 2013
17. Chengfo, S.: Improved differential evolution algorithms, Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference, 25 – 27 May 2012, Pages 142 - 145
18. Price, KV., Storn RM., Lampinen JA: Differential Evolution A Practical Approach to Global Optimization, Springer, 1997
19. Zelinka, I., Skanderova, L.: Investigation on Evolutionary Control and Optimization of Chemical Reactor, Soft computing models in industrial and environmental applications, Advances in Intelligent Systems and Computing, Volume 188, Pages: 469 –474, 2013
20. Pavlech, M.:Self-organizing Migration Algorithm on GPU with CUDA, Soft computing models in industrial and environmental applications, Advances in Intelligent Systems and Computing, Volume 188, Pages 173 – 182, 2013
21. Kadlec, P., Raida, Z.:A Novel Multi-Objective Self-Organizing Migrating Algorithm, Radioengineering, Volume 20, Pages 804 – 816, 2011