Multi-criteria optimization strategies for production chains

Jan Kusiak[1], Paweł Morkisz[2], Piotr Oprocha[2,a], Wojciech Pietrucha[3] and Łukasz Sztangret[1]

[1]*Faculty of Metals Engineering and Industrial Computer Science, AGH University of Science and Technology, al. A. Mickiewicza 30, 30-059 Kraków, Poland*
[2]*Faculty of Applied Mathematics, AGH University of Science and Technology, al. A. Mickiewicza 30, 30-059 Kraków, Poland*
[3]*Luxoft Poland Sp. z o.o., Krakowska 280, Z1, 32-080 Zabierzów, Poland*

Abstract. The aim of this paper is presentation of some optimization strategies applicable in the optimization of multi-stage and multi-threads chain structures (linear or tree-structured, acyclic graphs) with multiple intermediate goal functions. The inspirations for this type of analysis are production chains often seen in industrial plants. Production in these plants consists of sequences of multiple units connected linearly or in tree-structured graphs in which, at various production stages, intermediate quality criteria, related to the semi-products, may occur.

1 Introduction

Typical metallurgical industrial process can usually be divided into several smaller elements, corresponding to intermediate stages of production. They can arise in a natural way in correspondence to physical stand (e.g. furnace, hot rolling mill, laminar cooling, etc.) or representing smaller processes taking place in such stand (e.g. smelting and casting). In general, metallurgical production chains have a few elements in common:

- They do not have cycles. Delivered materials have to pass some production path without turning back. Branching is allowed, however, because some semi-products can be manufactured differently to reach the final form.
- Each stage requires some input parameters (materials, semi-products) with admissible range, demanded by technological restrictions for proper execution of reactions/processing of that stage.
- Each stage allows some controlling of performance by adjusting selected input parameters that can be controlled by an operator of the process at that stage.
- A number of quality output parameters are generated at the end of processing of each stage to judge performance or quality of the production.
- Output semi-products are described by some parameters characterizing their important technological features.

To illustrate the term of production chain better, an example of input-output parameters in two-stage model of production of blister copper (Cu_b) in Kennecott-Outokumpu process, considered in [1], is presented in Figure 1. In this particular case we have one global input parameter x (raw material – concentrate of Cu, Fe and S) that cannot be modified, two intermediate output-input parameters between two stages (Cu_2S, FeS) and global output parameter - blister copper Cu_b. Each stage has 3 control parameters (F – mass of fuel, O_2 – mass of oxygen in blast, N_2 – mass of nitrogen in blast) and two quality parameters (C_{Cu}, C_{FeO}; and C_{Fe3O4}, C_{Cu2O}) which are the concentrations of respective parameters (semi-products). Such a situation is typical in some sense. In general situation, each stage can be represented in the form of equations:

$$x_{i+1} = F_i(x_i, p_i), \tag{1}$$

$$q_i = Q_i(x_i, p_i), \tag{2}$$

where x_i, p_i, q_i are vectors representing input materials, control parameters and quality evaluation, respectively, and index i represents the i-th stage. While it is not written explicitly, each stage has attached a sequence of limiting equations on admissible values of input vector x_i as well as admissible range of control parameters p_i.

Clearly, when there are more than 2 stages then the production chain can have branching points, creating a tree-structured arrangement of production stages, as presented on Figure 2. Chosen strategies that can take advantage of the additional information about the structure of production chain will be presented in the paper.

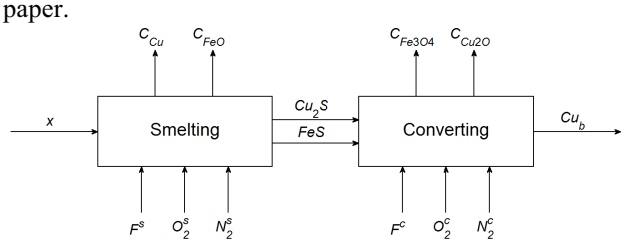


Figure 1. Example of two stage linear chain of industrial process (blister copper production [1,2]).

[a] Corresponding author: oprocha@agh.edu.pl

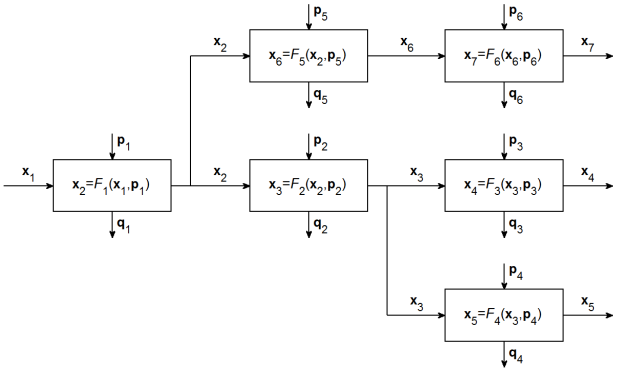


Figure 2. Example of hypothetical industrial process with a tree-structure.

It is also worth noting, that to calculate values q_i and x_i on deeper levels of the tree, it is necessary to set up an arrangement of stages. For example, to evaluate a value of q_4 (Figure 2) the following relationship has to be calculated:

$$q_4 = F_4(x_3, p_4) = F_4(F_2(F_1(x_1, p_1), p_2), p_4). \quad (3)$$

The indexing of nodes (production stages) of the tree will be performed using a standard Breadth-first Search (BFS) algorithm (see Figure 3).

2 Optimization Strategies

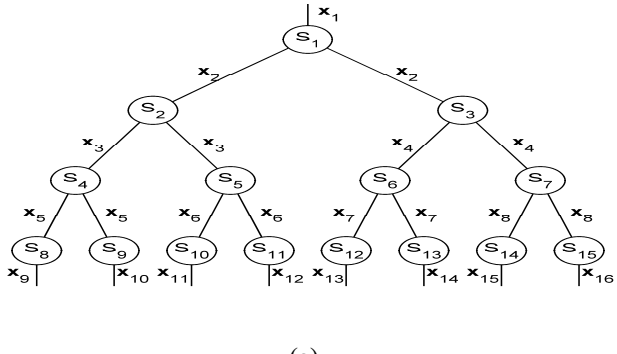
Since the knowledge about the structure of the production stages is known, such information can be used when choosing optimization strategy. When the structure is linear it is possible to look for optimal control parameters stage by stage. If there are branching points, the order in which the consecutive stages are dealt with, may be important for the final quality assessment. If the structure of relations between stages is not known (or is not at the point of interest), the **global approach** (GLO) can be applied. In this last approach, the global objective function composed by these stages is considered (see Section 2.2).

2.1. Standard searching in graphs

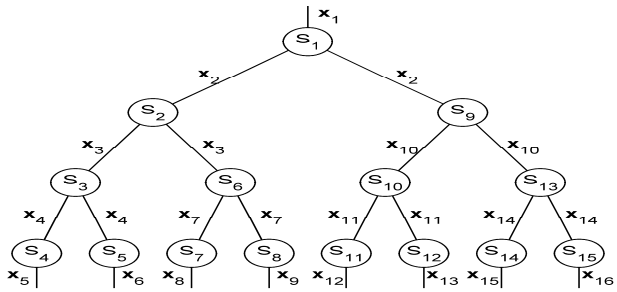
There are two standard graph algorithms (e.g. see [3]) allowing passing through all stages along the edges connecting them.

- Breadth-first search (BFS): in this approach all neighbouring (children) nodes are explored first, before going to next level of neighbours (Figure 3a).
- Depth-first search (DFS): in this approach tree is explored to the deepest possible level before backtracking (Figure 3b).

Note that while in DFS not all neighbouring nodes are checked simultaneously, their input-parameter bounds have to be checked. It can be simply explained. Let the output vector x_{i+1} is an input vector of all children nodes of i-th stage. If this output vector is not an admissible input for all children nodes, then it might happen that after backtracking the x_{i+1} cannot be used.



(a)



(b)

Figure 3. Indexing of nodes (stages) according to BFS (a) and DFS (b) algorithm.

It is worth noting, that we have to use BFS or DFS algorithm to order nodes, since we need some ordering when calculating values of functions F_i and Q_i. As we said, in our calculations we order nodes (order of propagation of x_i) by BFS algorithm, see Figure 3a.

2.2. Pareto-optimal solutions and order of children nodes

It is clear that in practice there is not a single (one) global process optimization criterion, since e.g. it is impossible to obtain best quality of final product, minimizing consumption of energy or fuel consumed in reactions, maximizing the speed of production and minimizing wastes, etc. Therefore, only the Pareto-optimal solution is the only aim of the research (the reader not familiar with this topic is referred to [4,5]).

A standard technique in this case is to transform multi-objective task into single objective task, assigning some weights w_j to each coordinate of each quality variable in vectors q_j. It results in the following objective function:

$$F = w_1 q_{1,1} + \cdots + w_k q_{1,k} + \cdots + w_s q_{n,r}, \quad (4)$$

where s is the total number of quality assessments in considered industrial process. Then it is reasonable to visit children nodes of i-th stage following their importance for the final value of function F. For this reason, for each child node its accumulated weight is calculated, summing up its weight w_j with weights of all of his children nodes on lower levels. These weights are used next in the DFS and BFS searches to order children nodes of stage i, that is children of node i are visited

following the order induced by accumulated weights. Let the *i*-th stage has *k* children nodes and accumulative weights *v* of sub-trees induced by these children are:

$$v_1 \leq v_2 \leq \cdots \leq v_k. \quad (5)$$

Then the child with weight v_1 is visited first, next with v_2, and so on. Note that if weight w_i of each quality criterion q_i is the same (say $1/s$) then condition $v_i<v_j$ means that *i*-th subtree has less nodes than *j*-th. It means in our strategies, that sub-trees with the simplest structure are visited first, leaving more complex sub-trees for later.
While in standard DFS or BFS the order of children nodes is not that important, its importance will be much higher when transferring credits as described in the next section. Function *F* will be an objective function in GLO algorithm, hence GLO does not use additional knowledge about the structure of the industrial process and relations between stages (this knowledge is only used indirectly when calculating the value of *F*).

2.3. Credits

In our search for optimal solution, an upper limit *N* on the number of allowed evaluations of functions F_i and Q_i for *i*-th stage has to be set up. It may happen, however, that the optimal solution for this stage is found using *n* evaluations with $n<<N$. Saved credit $c_0=N-n$ of remaining evaluations can then be forwarded for the evaluation of children nodes.

2.3.1 DFS with transferable credits (DFS CRE)

This approach is almost the same as standard DFS with one simple modification. The limit of evaluations for the first child is set to $N+c_0$ and then is propagated onto consecutive levels, following recursion in DFS. In some situations, credit *c* may stack to quite large number compared to *N*, which in turn may help in finding solution when *N* evaluations are not sufficient. Now suppose that after visiting all descendants of the first child the search is finished with c_1 remaining evaluations. Then, this credit is forwarded to the next child, so its limit of executions is now $N+c_1$. Finally, suppose that there were *k* children and at the end there is c_k remaining evaluations. In other words, let it be assumed that the optimum was found for all *l* stages in sub-tree induced by *i*-th node in $r<lN$ iterations, that is below total *lN* limit. In that case the remaining number $c_k=lN-r+c_0$ is returned to the parent node and propagated to subsequent children nodes, following node of *i*-th stage in DFS recursion.

2.3.2 BFS with transferable credits (BFS CRE)

The case of BFS is a little more complex. Since all children nodes have to be visited simultaneously, it is necessary to split credit c_0 between these nodes. Let *k* children of *i*-th stage have accumulated weights given by (5). Then *j*-th child receives credit $c_j=c_0v_j$ and this credit is then propagated recursively into deeper levels using the same order of visit as in standard BFS algorithm.

2.4. Particle Swarm Optimization

In our tests the Particle Swarm Optimization (PSO), which is a global search method, was applied. PSO is a method representing probabilistic approach, performing global search for optimal solution. PSO belongs to a group of algorithms motivated by nature, and is widely used and well described in the literature, e.g. see [6,7,8,9]. Let us recall basic information about PSO algorithm. In PSO algorithm it is necessary to simultaneously execute optimization of *k* different instances, so-called particles, and then evolution of the algorithm is in some sense determined by interactions of these particles and their performance measured by quality expressed in terms of a objective function *F* whose minimum is searched for. The search begins with randomly chosen position vector z_0^k of each particle. Position vector z_i^k of *k*-th particle in *i*-th iteration is modified by the relation:

$$z_{i+1}^k = z_i^k + v_i^k, \quad (6)$$

where v^k_i is velocity vector updated in each iteration by the formula:

$$v_{i+1}^k = wv_i^k + c_1r_{i,1}^k\left(p^g - z_i^k\right) + c_2r_{i,2}^k\left(p^k - z_i^k\right), \quad (7)$$

where: p^g denotes the best position found so far by the whole swarm; vector p^k represents the best solution found so far by the *k*-th particle; *w* is defined as the inertia coefficient; c_1 and c_2 are acceleration coefficients (called also training coefficients); $r^k_{i,1}$and $r^k_{i,2}$ are numbers in [0;1] picked at random with the uniform distribution.
This approach is in some sense independent of the choice of functions F_i, Q_i and in practice works effectively for complex problems as well as for simple problems.

3 Hypothetical structures of process

3.1. Balanced and unbalanced binary trees

Our first testing ground will be balanced binary tree with four levels, that is: root stage has two children nodes, each of them has two children, etc., forming finally a tree with the number of nodes on each level 1-2-4-8 as presented on Figure 4.

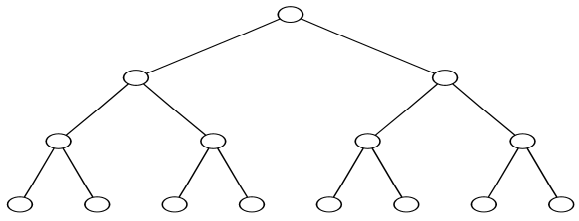


Figure 4. Process of balanced tree structure 1-2-4-8.

As a second example of hypothetical process tree with unbalanced structure will be examined. Such a situation seems more likely to happen in practice, because usually concrete industrial processes have different lengths measured by the number of intermediate stages. A tree with structure 1-2-4-4-4 as presented on Figure 5 will be

considered. Note, that the number of nodes is exactly the same as in 1-2-4-8 tree.

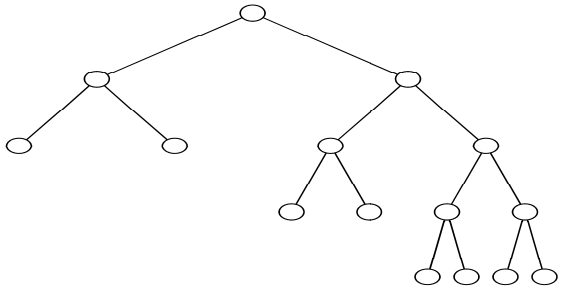


Figure 5. Process of unbalanced tree structure 1-2-4-4-4.

3.2. Functions Q_i and F_i

As it was said earlier, the aim of the paper is to investigate performance of the optimization strategies (and influence of credits in optimization). For this reason the simple, multi dimensional function was considered. This will minimize influence of randomness in our studies (there is zero probability of stuck of algorithm in local minimum) however there are still various complexity levels of the problem. Functions used in each node are the same, given by the formula:

$$q_i = Q_i(x_i, p_i) = \sum_{j=1}^{m} \left(x_{i,j} - p_{i,j}\right)^2 \quad (8)$$

$$x_{i+1} = F_i(x_i, p_i) = p_i. \quad (9)$$

Notice, that the control parameter p_i has a direct influence on the position of global minimum of function F_s for each children node s of i-th node, hence optimization in consecutive stages is not completely independent. The complexity of the process can be augmented by increasing number of dimensions m. When m is large, search space is also large, which makes that finding minimum by probabilistic algorithms is much more difficult.

4 Comparison of performance

Performance of the following five strategies was compared:

i. Global optimization (GLO - optimal control parameters of all stages set simultaneously),

ii. BFS without credits (BFS),

iii. DFS without credits (DFS),

iv. BFS with credits (BFS-CRE),

v. DFS with credits (DFS-CRE).

Let the number of evaluations of quality function Q_i for each stage be limited to N. In methods without credits this limit is strict. In methods with credits, remaining (not "consumed") evaluations can increase limits for consecutive nodes. In global approach we have global limit of $N{\cdot}s$ evaluations for the whole process (per particle), where s is the number of stages. The following 'global' limit of evaluations for the whole procedure is also introduced:

$$L_g = N \cdot s \cdot p \cdot t, \quad (10)$$

where s is a number of stages, p is a number of particles, t is a limit for the 'time' of the optimization runs. When the algorithm calculates the value of any of the functions $Q_1,...,Q_s$, the index L_g becomes $L_g = L_g - 1$. In particular, to calculate value of F in (4) we need exactly s evaluations. It means that one step of GLO consumes s *'credits'* from L_g, and one step of each other method consumes 1 '*credit'*, however finally it is necessary to check all s stages (nodes of the tree). Then the limit in (10) allows at least t optimization attempts. However, if one run of the method is finished faster (or it is quickly decided that the whole optimization leads to a failure) the next optimization can be restarted faster. Thank that more runs than t can be performed. It is clear that the number of evaluations in one optimization run will be higher in methods with credits than in these without credits, and hence the average number of optimization attempts will be smaller. Since in every step of methods (i)-(v) it is necessary to preform evaluation of each particle of PSO method, the limit (10) may be simplified to the form:

$$L = N \cdot s \cdot t, \quad (11)$$

having in mind that in fact 1 credit in L means p evaluations (one for each particle in PSO method), so p credits in terms of limit L_g.

5 Results

In all numerical experiments optimal solution using PSO algorithm with p=40 particles is searched for. Tests for different values of dimension parameter m in (8) were performed (m=1,5,7,10). Higher value of m is correlated with smaller chance of finding minimum by PSO.

5.1. Process of balanced tree structure

Every of the methods (i)-(v) was tested on the tree 1-2-4-8 (Figure 4) within the total limit of evaluations on functions F_i provided by (11) which in our case is L=1500000 (for N=1000 and t=100). As it was said before, each method can be executed at least 100 times within the limit L, however, this number can be changed if the method is terminated faster (due to success or failure). The number of full optimization runs of every method for different value of m is presented in Figure 6 and the number of attempts which were successful (method finished finding minimum) are presented on Figure 7.

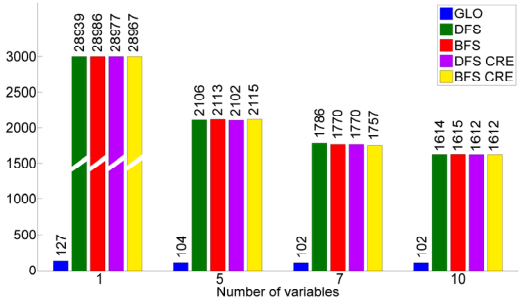


Figure 6. Average number of full optimization runs within limit L (tree 1-2-4-8).

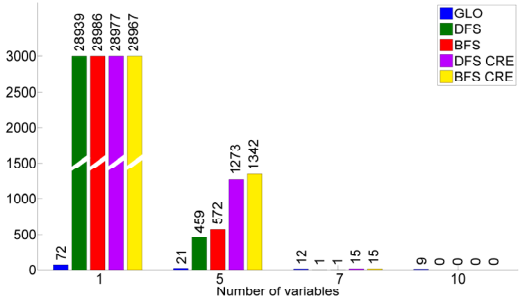


Figure 7. Average number of full optimization runs finished with success within limit L (tree 1-2-4-8).

Figure 8 presents relative number of successes of given method expressed by the ratio:

$$Sr = 100 \cdot {}^{completed\_runs}/_{starts}\,[\%]. \quad (12)$$

In other words, Figure 8 shows relative percentage of chance that single instance of the method will finish with success.

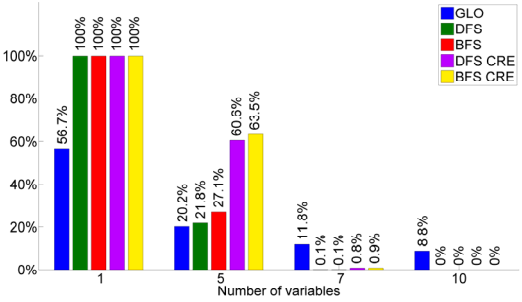


Figure 8. Relative success rate of every method (tree 1-2-4-8).

To decrease influence of randomness in our numerical experiment, every complete cycle of computations (sequence of optimizations within limit set L) was repeated 100-times for each method. Therefore, Figures 6-8 present average values calculated from 100 repetitions of every method.

5.2. Process of unbalanced tree structure

The same tests as in Section 4.1 were repeated for the process of the unbalanced tree structure 1-2-4-4-4 (Figure 5). Results of our numerical experiment are presented on Figures 9,10 and 11. Data presented in these figures follows exactly the same methodology as in section 4.1 in particular value of L, tested values of m, etc.

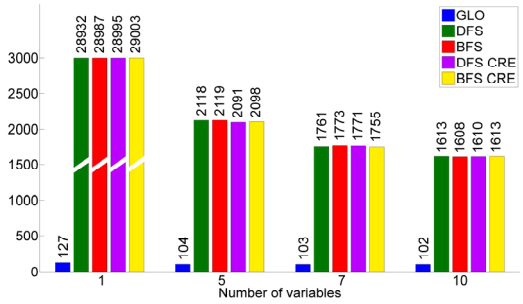


Figure 9. Average number of full optimization runs within limit L (tree 1-2-4-4-4).

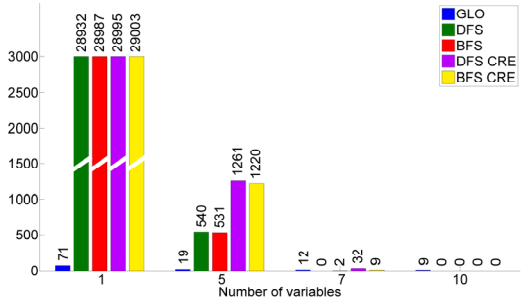


Figure 10. Average number of full optimization runs finished with success within limit L (tree 1-2-4-4-4).

The main difference now is that right side of the tree is much deeper, so in particular weights w_i defining division of credits between children nodes in BFS are now different. On the other hand, there is no visible change of the problem for GLO because the number of control parameters is the same (the same number of stages s as in tree 1-2-4-8). As before, each complete cycle of L evaluations was repeated 100 times.

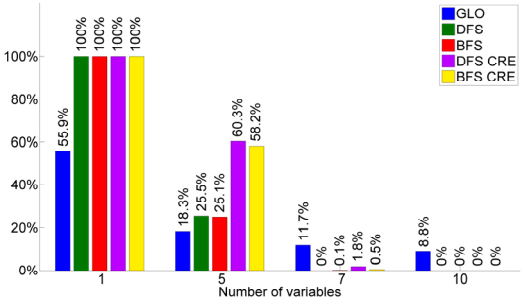


Figure 11. Relative success rate of every method (tree 1-2-4-4-4).

6 Conclusions

Five different optimization strategies of multi-stage industrial processes with two different tree structures were examined. According to the performed test, the following conclusions can be withdrawn.

- As it was mentioned earlier, GLO strategy does not take advantage of the knowledge about the structure of the process (relations between nodes/stages), therefore it should behave similarly on balanced or unbalanced

tree structure, assuming that the number of stages and their dimensions are the same. It is confirmed by computational results presented in Figures 6-11. Average number of executed full optimization runs and the successful optimizations (and as a consequence, the success rate) were nearly the same in both cases. Comparison of the number of optimizations runs in sequential methods (ii)-(v) shows a huge difference between them and GLO. It is clear that these methods can terminate much faster, because in optimistic scenario they can decide that whole optimization is going to be a failure during first N evaluations for the first stage. If they do not succeed, they terminate the process, saving $N\cdot(s-1)$ evaluations. In the case of GLO it is necessary to wait until the real end to realize the failure, hence in such a case the whole $N\cdot s$ credits are spent. This situation is clearly visible in Figures 6 and 9, where number of started optimizations is close to minimum of t attempts (GLO can earn additional attempts only by finding optimal solution before spending the limit of $N\cdot s$ credits; then some credits within L are *``saved''*). Notice, that all the other (sequential) strategies are executed much more times than t.

- Comparison of strategies (ii), (iii) and (iv), (v). It was intuitively expected that (ii) and (iii) will be executed more often on average than their counterparts with credits, because they are able to continue computations longer. This statement is not supported by results of the experiment presented in Figure 6 and 9. On the other hand, success rate of DFS-CRE and BFS-CRE was much higher than for DFS and BFS. This suggests that while transferring *``credits''* between stages does not change expected execution time of the optimization much, it can highly decrease chances of failure. If the DFS and BFS (with of without *``credits''*) are compared, then it can be seen that visiting stages in optimization following order provided by BFS behaves slightly better when tree is balanced and DFS may be a better option in a case of unbalanced tree. The differences are not high however, so probably the above claim needs confirmation in a more extensive study with various different tree structures.
- Finally it can be seen that when complexity of the problem increases (the size of parameters space) GLO approach becomes more appropriate. It is most visible in the case of m=10 where it was the only strategy that was successful. This is situation in a sense similar to our previous observations from [10], where the influence of aggregation of stages on reliability of sequential optimization was considered.

Acknowledgements

The research leading to results presented in this paper was supported by National Science Centre in Poland (NCN), Grant no. DEC-2013/11/B/ST8/00352.

References

1. P. Jarosz, J. Kusiak, S. Małecki, P. Oprocha, Ł. Sztangret, M. Wilkus: Mathematical Problems in Engineering (2015) doi: 10.1155/2015/182679
2. P. Jarosz, J. Kusiak, S. Małecki, P. Morkisz, P. Oprocha, W. Pietrucha, Ł. Sztangret, *Metamodelling and optimization of copper blister two-stage production process*, JOMJ (2016), (*in print*)
3. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to algorithms,* 3rd ed. (MIT Press, Cambridge, 2009).
4. K. Deb, *Multi-objective optimization using evolutionary algorithms*. (John Wiley & Sons, Chichester, 2001).
5. S. Rao, *Engineering Optimization: Theory and Practice*. (John Wiley & Sons, New York,1996).
6. R.C. Eberhart, J. Kennedy, *Proceedings of the Sixth International Symposium on Micro Machine and Human Science* (IEEE Press,1995).
7. R.C. Eberhart, J. Kennedy, J., *Proceedings of IEEE International Conference on Neural Networks* (IEEE Press, 1995).
8. S. Helwig, R. Wanka, *Proceedings of the 2007 IEEE Swarm Intelligence Symposium* (IEEE Press, 2007).
9. I.C. Trelea, *Information Processing Letters*, **85** (2003).
10. J. Kusiak, P. Morkisz, P. Oprocha, W. Pietrucha, Ł. Sztangret, *On aggregation of stages in multi-criteria optimization of chain structured processes*, submitted to ICAISC 2016.