

# Restart Operator for Optimization Heuristics in Solving Linear Dynamical System Parameter Identification Problem

Ivan Ryzhikov<sup>1,2</sup><sup>a</sup> and Christina Brester<sup>1,2</sup><sup>b</sup>

<sup>1</sup>*Department of Environmental and Biological Sciences, University of Eastern Finland, Kuopio, Finland*

<sup>2</sup>*Institute of Computer Science and Telecommunications, Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia*

**Keywords:** Dynamical System, Restart, Heuristics, Meta-heuristics, Parameter Identification, Evolutionary Algorithm, Bio-inspired Algorithms.

**Abstract:** In this study, the parameter identification problem for linear dynamical systems is considered. The system is assumed to be represented as a linear differential equation in general form, so the right-hand side equation contains input function and its derivatives. This problem statement extends the order reduction problem, where we need to find the equation of the lower order to approximate the real system output observations. Considered problem is reduced to an optimization one. The reduced problem is complex, and we propose the combination of stochastic optimization algorithm and restart operator. This operator aim is to prevent the algorithm stagnation by starting the search over again if no remarkable solution improvement is detected or if algorithm searches in the area where stagnation had been detected.


## 1 INTRODUCTION


In this paper, we consider parameter identification problem for dynamical system and its approach using optimization heuristic with specific operator that controls the search. Dynamical system parameter identification problems (Ramsay and Hooker, 2017) are complex and appears in different application fields (Gennemark and Wedeling, 2009). The main idea is to identify the parameters of the differential equation so its solution would fit the observation data the most. We assume that we know the degrees of the left-hand side and right-hand side equations and initial point of the dynamical system. The problem of parameter identification for the differential equation of the second order finds plenty of applications and is considered in different studies. In most of them, the evolution-based algorithms are applied to solve the reduced identification problem: genetic algorithm (Parmar and Prasad, 2007), big bang big crunch (Desai and Prasad, 2011) and cuckoo search (Narwal and Prasad, 2016). In this case, the considered approach generalizes the order reduction problem so that any of possible degree, both state and input

variables. There are also studies on identification of the single output dynamical system parameters, when the right-hand side equation is just the control function. That means, that considered in this study approach extends the class of dynamical systems by adding the input derivatives to the right-hand side equation.

Many of optimization algorithms utilized to solve real world problem are stochastic. There are different implementations of the general idea on how the natural systems evolve. However, what all these algorithms have in common is exploration of the searching space and seeking for the better alternative. There are plenty of adaptation schemes and algorithms interaction schemes, which allow increasing searching performance. Also, there are plenty of problem-oriented modifications, which improve performance for optimization problems.

This study focuses on pairing algorithm with restart operator for solving identification problem. In that sense, we develop a heuristic that is applicable for different algorithms despite of their basic idea and its implementation. Proposed restart heuristic identifies and prevents algorithm stagnation by

<sup>a</sup> <https://orcid.org/0000-0001-9231-8777>

<sup>b</sup> <https://orcid.org/0000-0001-8196-2954>

starting another search if the algorithm is trapped in a local optimum area or in the area that was found in previous runs. Stagnation happens when algorithm cannot get out of the local optimum area and at the same time cannot sufficiently improve the best alternative in it. Here we assume that another algorithm launch would at least explore the search space more.

The so-called restart operator checks if any statistic exceeds predefined limitation. If it happens than the search starts over again, so, probably another optimum will be found. This operator improves the optimization algorithm performance for some optimization problems. It can be also applicable for managing the computational resources in multiple runs of stochastic algorithms. That approach has some benefits in solving time-consuming problems, where the exploration is critical, and the global optimum is hard to be found.

To investigate the algorithm performances with and without the restart operator, we examine optimization algorithms on a set of data samples, produced by different dynamical systems. All examples are considered noiseless ones.

## 2 IDENTIFICATION PROBLEM

In this chapter, we consider the dynamical system identification problem. First, we discuss the mathematical model of the dynamical system and then suggest the way to calculate the objective function. Objective function appears when we reduce the problem to the unconstrained global optimization problem on the real vector space.

### 2.1 Model

Let the system model be described with the linear differential equation:

$$a_n x^{(n)} + \dots + a_0 x = b_m u^{(m)} + \dots + b_0 u, \quad (1)$$

where  $a_i \in R, i = 0 \dots n$  and  $b_i \in R, i = 0 \dots m$  are the model coefficients,  $x$  is the system state and  $u$  is the system input,  $n$  and  $m$  are the maximum degrees of the system output and the system input, respectively. We consider systems where the output  $x$  is observable and the input  $u$  is known.

Since  $n$  and  $m$  are the highest degrees, then parameter  $a_n \neq 0$ , so we can simplify (1):

$$x^{(n)} + \dots + \tilde{a}_0 x = \tilde{b}_m u^{(m)} + \dots + \tilde{b}_0 u. \quad (2)$$

Now the model can be described with equation (2) and the model output on input  $u$  can be found by solving the Cauchy problem:

$$\begin{aligned} x^{(n)} + \dots + \tilde{a}_0 x &= \tilde{b}_m u^{(m)} + \dots + \tilde{b}_0 u, \\ x(0) &= v, \end{aligned} \quad (3)$$

where  $v$  is the vector of initial values. The equation (2) is time invariant; by that reason, we assign initial time point as 0 in (3). In this research we also assume that  $v = 0$ .

We solve considered Cauchy problem (3) for equation (2) numerically with Runge-Kutta integration scheme.

### 2.2 Objective Function

Let us denote the parameters we need to identify as  $\tilde{a} = (\tilde{a}_0, \dots, \tilde{a}_{n-1})^T$  and  $\tilde{b} = (\tilde{b}_0, \dots, \tilde{b}_m)^T$ . The system observation data is  $Y = \{y_i\}, i = 1 \dots N$  and observation times is  $T = \{t_i\}, i = 1 \dots N$ , where  $N$  is the sample size. In this study, without loss of generality, we assume that the observation times are the same as integration time points. The last means that the solution of (3) at time  $t_i$  is  $y_i$ . We also assume that we know the initial point  $v$  and the input function  $u$ .

Now we can reduce the identification problem to the optimization problem, in which we need to minimize the objective function:

$$F(\tilde{a}, \tilde{b}) = \frac{\sum_{i=1}^N \|x(t_i) - y_i\|}{\max(Y) - \min(Y)}, \quad (4)$$

where  $x(t)$  is the solution of the Cauchy problem (3) for given  $\tilde{a}$  and  $\tilde{b}$  parameters and the difference between  $\max(Y)$  and  $\min(Y)$  is used here for scaling the error. The scaling of the error statistics provides better interpretation of the real error between the solution and observation data.

In general, objective function (4) is complex, multimodal and requires the global optimization approach to find the parameters that reach the extremum of function.

We use specific fitness function for the evolution-based optimization algorithms:

$$f(\tilde{a}, \tilde{b}) = \left(1 + F(\tilde{a}, \tilde{b})\right)^{-1}, \quad (5)$$

where function  $f \in [0, 1]$ , and the greater this function is, the better solution is. This fitness function will be used in further examination of the results.

### 3 SOLVING REDUCED PROBLEM

We propose applying stochastic search heuristic combined with restart operator for solving the reduced problem formulated in (4). Standard and common optimization heuristics were applied. Algorithms and details of their implementation are considered below. The restart operator was designed on the basis of (Ryzhikov et al., 2016).

#### 3.1 Optimization Algorithms

As the main optimization algorithms, the real-value genetic algorithm, differential evolution algorithm and particle swarm optimization were used.

The settings of the algorithms were the following: 200 or 500 iterations and 100 alternatives in iteration. The initial population is uniformly distributed on  $[-5, 5]^{n \times m}$ . The optimization problem is considered as unconstrained, so there are no boundaries for the variables during the search.

Real value genetic algorithms has tournament selection with tournament size equal to 10. We used 2-parent uniform crossover, where each variable of the offspring is taken randomly from one of its parents, according with the scheme of uniform crossover in evolutionary strategies (Beyer and Schwefel, 2001). The mutation is implemented by the following scheme:

$$var = \begin{cases} r_g, r_m > m_p, \\ var \end{cases} \quad (6)$$

where  $var$  is a variable chosen in alternative that is processed by the mutation operator,  $m_p$  is mutation probability,  $r_m$  is uniformly generated value on  $[0, 1]$  and  $r_g$  is randomly generated value on  $[-5, 5]$ .

Real value genetic algorithm is combined with local optimization, where for randomly chosen 10 alternatives we chose the variable also randomly and make a step of 0.1 to random direction. If after this step we discover a better solution, we substitute the current one with the new one.

Differential evolution algorithm is applied in its standard form (Storn and Price, 1997). The crossover probability is set as 0.2 and the differential weight is 1.4. The parameter values were chosen according to brief investigation of the algorithm performance for the current problem.

Particle swarm optimization algorithm is also applied in its standard form (Kennedy and Eberhart,

1995), with  $\varphi_1 = 1.2$  and  $\varphi_2 = 1.2$ . The reason for that choice of parameters is the same as for the differential evolution algorithm.

#### 3.2 Restart Operator

The restart operator checks the stagnation via the following inequality:

$$best_i - best_{i-w_r} < t_r, \quad (7)$$

where  $best_i$  is the best fitness value ever found by the algorithm in the current run by the  $i$ -th iteration,  $best_{i-w_r}$  is the best fitness value by the  $i - w_r$ -th iteration,  $w_r$  is a restart window size and  $t_r$  is a threshold. So, if the difference between best fitness values does not change sufficiently (does not exceed  $t_r$ ) for the last  $w_r$  iterations, it is decided to start the algorithm again.

If we do the restart, we also keep the best alternative found by the algorithm. So if the best solution in another run is close to the one found earlier, we also do the restart. According to that, there is another condition to check:

$$\bigcup_{a \in H_r} \|a - a_i\| < d_r, \quad (8)$$

where  $H_r$  is the set of alternatives, which was found by the algorithm by the time when the restart initiated and  $a_i$  is the best alternative found in the current algorithm run by the  $i$ -th iteration,  $d_r$  is another restart operator parameter.

### 4 EXAMINATION

For examination of the algorithm performance we considered differential equations given in the Table 1. These equations were had been used to generate the sample on the time interval  $[0, 10]$  and then we took 200 points out of each solution. As a control function we used  $\sin(t)$ .

Table 1: Parameters for identification problems.

Number	Parameters
1	$\tilde{a} = (-1, -2, -1)^T$ $\tilde{b} = (1, 2)^T$
2	$\tilde{a} = (-1, -2, -4, -1)^T$ $\tilde{b} = (1, 2)^T$
3	$\tilde{a} = (-1, -3, -4)^T$ $\tilde{b} = (-1, 1)^T$

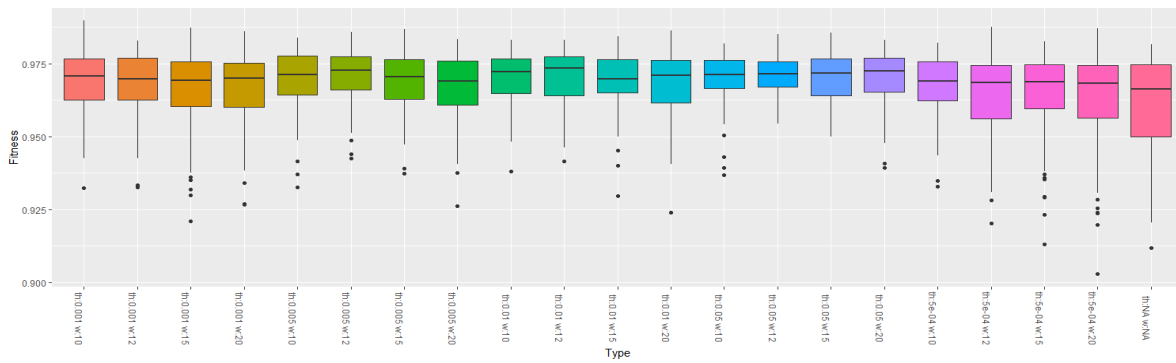


Figure 1: Best fitness value boxplot for different settings: window size + threshold. Statistic for the algorithm without restart operator is on the right. Problem 1 from the Table 1 solved by real-valued genetic algorithm with 200 iterations.

First, let us apply the real-valued genetic algorithms with restart operator and 200 iteration search to all of these problem and for the following restart operator parameters:  $d_r \in \{0.01, 0.05, 0.1\}$ ,  $t_r \in \{0.1, 0.05, 0.01, 0.005, 0.001, 0.0005\}$  and  $w_r \in \{10, 12, 15, 20\}$ . We considered all the combinations of these parameters and made 40 independent launches of the algorithm. For the first experiment we take the problem number 1 from the Table 1.

In Figure 1 one can see the boxplot of the fitness values of the best solutions found by algorithms for 40 runs. In that figure, one can see the performance of different combinations of the restart parameters: window size and threshold. The box on the right represents the statistic for the algorithm without the restart operator. All the cases with restart operator outperform the standard algorithm, since the majority of the fitness values and their median value is closer to 1.

All the Figures have the statistic for standard algorithm represented on the right. The legend or axis labels named NA means that the parameter is missing and so the statistic represents the algorithm without the restart operator. Boxplots are represented in their common way.

In Figures 2, 3 and 4 one can see the fitness boxplots for various values of the restart parameters: window size  $w_r$ , threshold  $t_r$ , and distance  $d_r$ , respectively.

All the figures prove that different settings of the restart outperform the standard algorithm. We also can say that the threshold is the parameter is the most influencing one.

Now let us examine the same algorithm for the problems 2 and 3 from the Table 1 and make the same tests for that problem. Boxplots that show how different combinations of threshold and window size affect the algorithm performance is given in Figure 5 and in Figure 6.

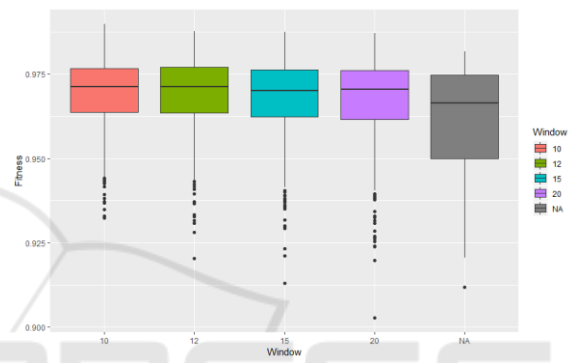


Figure 2: Fitness values boxplot for different restart window sizes.

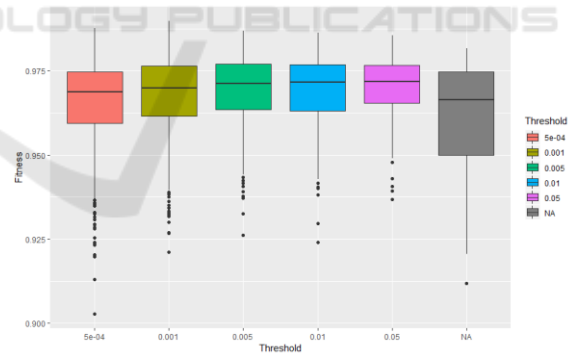


Figure 3: Fitness values boxplot for different restart threshold parameter values.

Figure 5 we can see some settings, which make algorithm not as efficient as its standard implementation. The situation shown in Figure 6 is worse. For problem 3 the standard algorithm outperforms most of the algorithms with the restart operator. But here we can see the trend: algorithms with restart with small window size are not worse than the original algorithm.

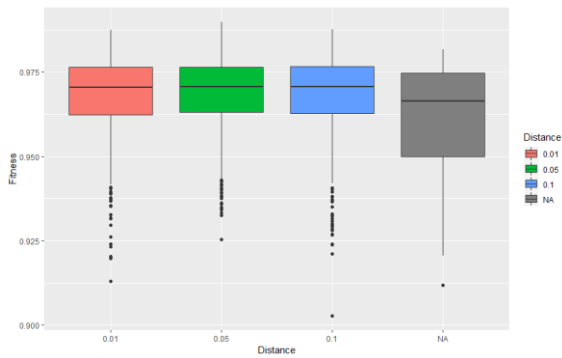


Figure 4: Fitness values boxplot for different distances.

The further investigation would be based on solving problem 1 from the Table 1. First of all, let us check what if we increase the number of objective function calculations to 500. To make the results clearer here we use the window size equal 15, distance to 0.01 and vary only the thresholds. The similar boxplots are presented on the Figure 7. Starting from here we will consider the following

values for the restart threshold:  $t_r \in \{0.005, 0.001, 0.0005, 0.0001, 0.00005\}$ .

Increasing the computational resources for the original algorithm caused an increase of its performance, so it became much closer to the algorithms with restart and even outperformed some of its settings. But if we look at the Figure 1, we will see that algorithms with smaller resources have the same efficiency.

What if we use the half of the computational resources for hybridization of the real-valued genetic algorithm with a local search? The boxplots are given in Figure 8, where it is shown that combination of the local search and restart increased the overall efficiency.

The further investigation will be provided for differential evolution algorithm and particle swarm optimization for solving problem 1 with 500 iterations. For these algorithms we consider windows equals 15 and 75. The similar boxplots for differential evolution are given in Figure 9 and Figure 10.

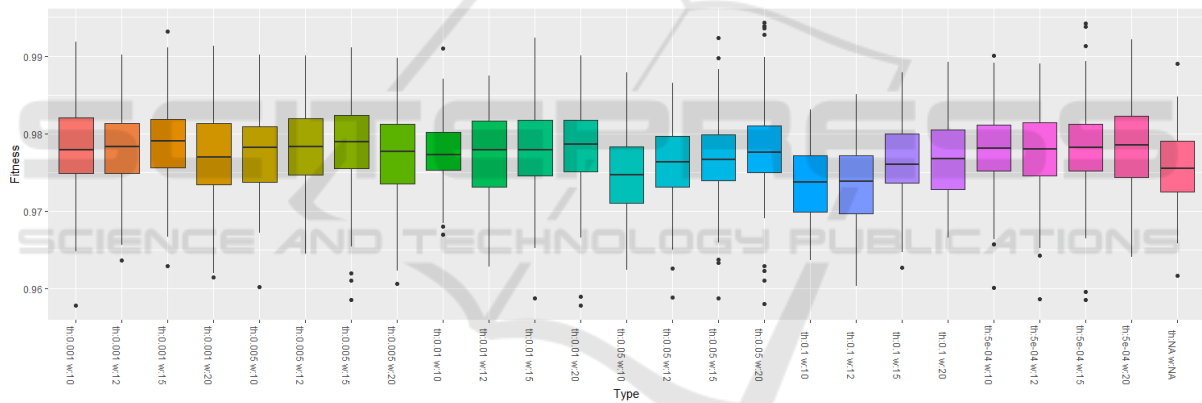


Figure 5: Best fitness value boxplot for different settings: window size + threshold. Statistic for the algorithm without restart operator is on the right. Problem 2 from the Table 1 solved by real-valued genetic algorithm with 200 iterations.

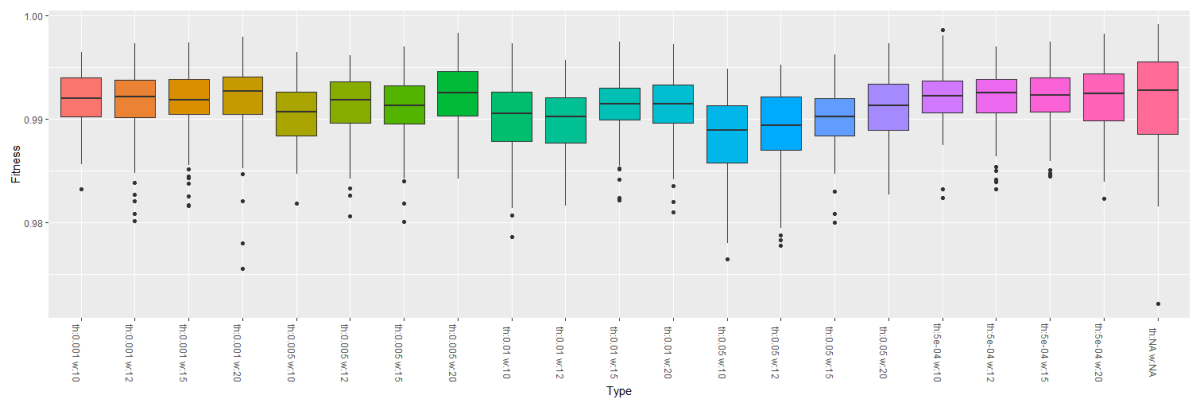


Figure 6: Best fitness value boxplot for different settings: window size + threshold. Statistic for the algorithm without restart operator is on the right. Problem 3 from the Table 1 solved by real-valued genetic algorithm with 200 iterations.

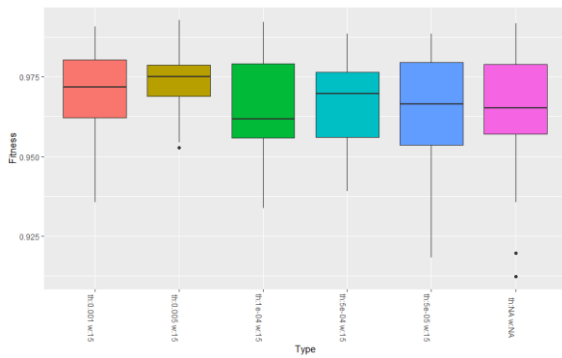


Figure 7: Fitness values statistics for different restart parameters. Problem 1 from the Table 1 solved by real-valued genetic algorithm with 500 iterations.

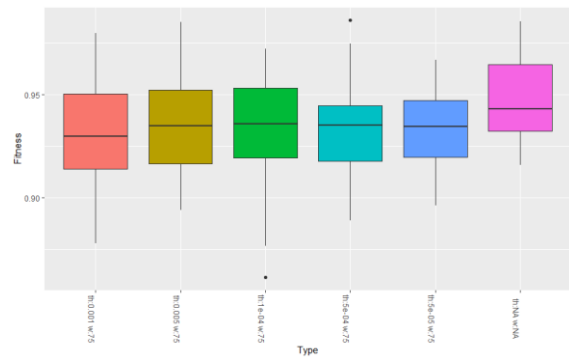


Figure 10: Fitness values statistics for different restart parameters. Problem 1 from the Table 1 solved by differential evolution, window size is 75.

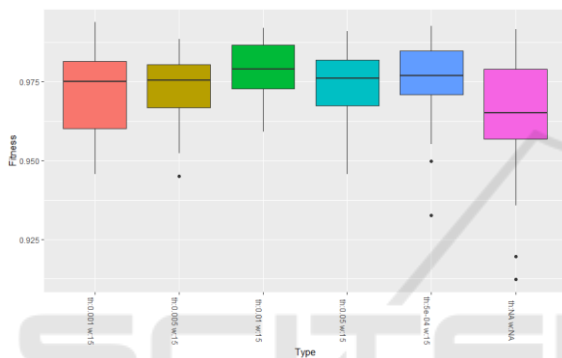


Figure 8: Fitness values statistics for different restart parameters. Problem 1 from the Table 1 solved by real-valued genetic algorithm with local search.

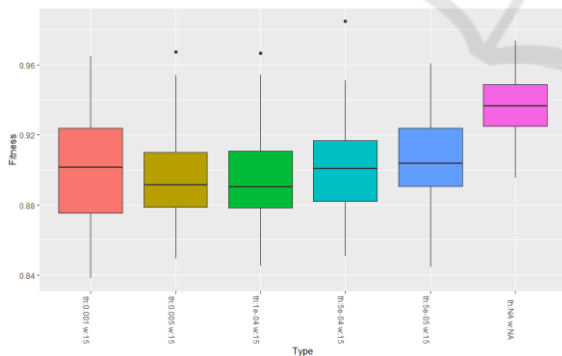


Figure 9: Fitness values statistics for different restart parameters. Problem 1 from the Table 1 solved by differential evolution, window size is 15.

Figure 9 shows that restart does not improve the performance of the differential evolution algorithm for solving the considered problem. One can also see that the differential evolution performance is much lower than the real-value genetic algorithm's one.

Figure 10 shows that increasing the window size make the algorithm behaviour closer to the original algorithm, as it was expected. According to Figure 10 and Figure 3 there are problems for which restart operator does not improving the performance of the differential equation parameters search.

Let us do the same experiments for the particle swarm optimization algorithm. The boxplot diagrams for window size equal 15 and 75 are given in Figure 11 and Figure 12, respectively.

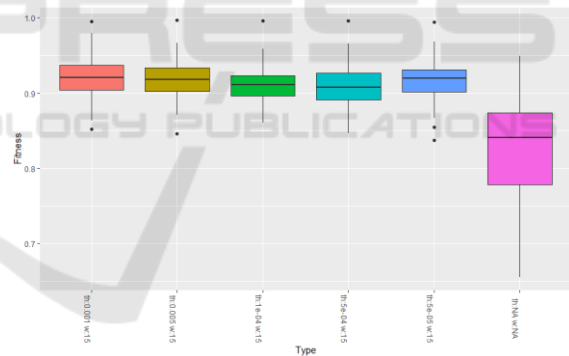


Figure 11: Fitness values statistics for different restart parameters. Problem 1 from the Table 1 solved by particle swarm optimization, window size is 15.

Figure 11 shows that particle swarm optimization algorithm is outperformed by the real-valued genetic algorithm and differential evolution. But at the same time, we can see that restart sufficiently improved its performance and the maximum fitness value is even close to 1.

Increasing the window size has the same effect as for the differential evolution. Restart performance is getting closer to the original algorithm's one.



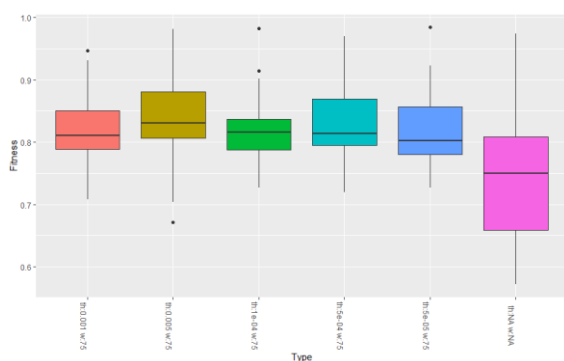


Figure 12: Fitness values statistics for different restart parameters. Problem 1 from the Table 1 solved by particle swarm optimization, window size is 75.

## 5 CONCLUSIONS

In this study we considered parameter identification problem for the linear dynamical models. This problem differs from the other parameter identification problems because it is assumed that the right-hand side equation contains input function and its derivatives.

The considered problem is complex and multimodal and requires a specific approach. This is proven by the low efficiency of particle swarm optimization and differential evolution algorithms applied to this problem.

In this work, an operator is proposed that can improve the algorithms' performance preventing its stagnation. But this operator has different effect on different optimization algorithms: it improved the genetic algorithm and particle swarm optimization algorithm but decayed the differential evolution performance. We can also conclude that the restart operator is parameter sensitive.

The goal of the further research is to explore other heuristics to improve the algorithms efficiency in solving linear dynamical system parameter identification problem.

## ACKNOWLEDGEMENTS

The reported research was funded by Russian Foundation for Basic Research and the government of the region of the Russian Federation, grant № 18-41-243007.

## REFERENCES

- Beyer, H.-G., Schwefel, H.-P., 2002. Evolution Strategies: A Comprehensive Introduction. *Journal Natural Computing*, 1(1):3–52.
- Desai, S., Prasad, R., 2013. A novel order diminution of LTI systems using Big Bang Big Crunch optimization and Routh approximation. *Appl. Math. Model.* 37, 8016–8028
- Gennemark, P., Wedelin, D., 2009. Benchmarks for identification of ordinary differential equations from time series data. *Bioinformatics (Oxford, England)*, 25(6), 780–786.
- Narwal, A., Prasad, B.R., 2016. A novel order reduction approach for LTI Systems using Cuckoo Search optimization and stability equation. *IETE J. Res.* 62(2), 154–163
- Parmar, G., Prasad, R., Mukherjee, S. 2007. Order reduction of linear dynamic systems using stability equation method and GA. *Int. J. comput. Inf. Eng.* 1(1), 26–32.
- Ramsay J., Hooker, G., 2017. *Dynamic Data Analysis. Modeling Data with Differential Equations.* Springer Series in Statistics. Springer.
- Ryzhikov, I., Semenkin, E., Sopov, E. 2016. A Meta-heuristic for Improving the Performance of an Evolutionary Optimization Algorithm Applied to the Dynamic System Identification Problem. *IJCCI (ECTA) 2016*: 178-185
- Storn, R., Price, K. 1997. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization.* 11 (4): 341–359.
- Kennedy, J., Eberhart, R. 1995. "Particle Swarm Optimization". *Proceedings of IEEE International Conference on Neural Networks. IV.* pp. 1942–1948.