Maximizing Efficiency: A Comparative Study of SOMA Variants and Constraint Handling Methods for Time Delay System Optimization

Roman Senkerik
Tomas Bata University in Zlin
Zlin, Czech Republic
senkerik@utb.cz

Tomas Kadavy
Tomas Bata University in Zlin
Zlin, Czech Republic
kadavy@utb.cz

Peter Janku
Tomas Bata University in Zlin
Zlin, Czech Republic
janku@utb.cz

Michal Pluhacek
Tomas Bata University in Zlin
Zlin, Czech Republic
pluhacek@utb.cz

Hubert Guzowski
AGH University of Science and Technology
Krakow, Poland
guzowski@agh.edu.pl

Libor Pekar
Tomas Bata University in Zlin
Zlin, Czech Republic
pekar@utb.cz

Radek Matusu
Tomas Bata University in Zlin
Zlin, Czech Republic
rmatusu@utb.cz

Adam Viktorin
Tomas Bata University in Zlin
Zlin, Czech Republic
aviktorin@utb.cz

Maciej Smołka
AGH University of Science and Technology
Krakow, Poland
smolka@agh.edu.pl

Aleksander Byrski
AGH University of Science and Technology
Krakow, Poland
olekb@agh.edu.pl

Zuzana Komínková Oplatková
Tomas Bata University in Zlin
Zlin, Czech Republic
oplatkova@utb.cz

ABSTRACT

This paper presents an experimental study that compares four adaptive variants of the self-organizing migrating algorithm (SOMA). Each variant uses three different constraint handling methods for the optimization of a time delay system model. The paper emphasizes the importance of metaheuristic algorithms in control engineering for time-delayed systems to develop more effective and efficient control strategies and precise model identifications.

The study includes a detailed description of the selected variants of the SOMA and the adaptive mechanisms used. A complex workflow of experiments is described, and the results and discussion are presented. The experimental results highlight the effectiveness of the SOMA variants with specific constraint handling methods for time delay system optimization.

Overall, this study contributes to the understanding of the challenges and advantages of using metaheuristic algorithms in control engineering for time delay systems. The results provide valuable insights into the performance of the SOMA variants and can help guide the selection of appropriate constraint handling methods and the adaptive mechanisms of metaheuristics.

CCS CONCEPTS

• **Mathematics of computing** → **Evolutionary algorithms**.

KEYWORDS

SOMA, swarm algorithms, parametric optimization, time delay system

ACM Reference Format:
Roman Senkerik, Tomas Kadavy, Peter Janku, Michal Pluhacek, Hubert Guzowski, Libor Pekar, Radek Matusu, Adam Viktorin, Maciej Smołka, Aleksander Byrski, and Zuzana Komínková Oplatková. 2023. Maximizing Efficiency: A Comparative Study of SOMA Variants and Constraint Handling Methods for Time Delay System Optimization. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3583133.3596417

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0120-7/23/07…$15.00
https://doi.org/10.1145/3583133.3596417

1 INTRODUCTION

Control engineering is a critical field vital in ensuring the efficient operation of complex systems. However, various observed quantities in the system and control loop do not act simultaneously. The latency between some action and its impact can appear [5]. Optimizing either the control or identification of a complex system with

time delay is challenging due to numerous interdependent variables and constraints. A time delay in the system response further complicates the problem since the system is infinite-dimensional because of an infinite number of its modes[17].

Metaheuristic algorithms [4] have emerged as a promising tool for optimizing the control of complex systems and identification of models with time delay due to their ability to handle high-dimensional optimization problems and their robustness to noise and uncertainty. In recent years, various metaheuristic algorithms, such as particle swarm optimization (PSO), artificial bee colony (ABC), genetic algorithms (GA), differential evolution (DE), and hybrid versions with other metaheuristic algorithms have been proposed for optimizing the control of a complex system with time delay [6, 10, 15]. Using metaheuristic algorithms in control engineering for time-delayed systems offers a promising approach for developing more effective and efficient control strategies and precise model identifications. By leveraging the strengths of these algorithms, researchers and practitioners can develop more robust, scalable, and effective control systems.

However, in addition to these advantages, several challenges and disadvantages are associated with using these algorithms in a specific domain of time delay systems. One of the challenges is a black-box nature, when metaheuristic algorithms can be challenging to understand, as they operate as a black box and do not provide insight into the underlying mechanisms that drive their optimization process. This lack of understanding makes it challenging to develop effective optimization strategies that leverage the full potential of these algorithms. Other common challenges are scalability, and the need for hyperparameter tuning of a metaheuristic algorithm, as there may be many parameters that interact with one another in complex ways.

The paper's organization is the following: After the brief state-of-the-art section and research motivation, the solved problem of optimization of time delay system model identification is presented, followed by selected variants of a metaheuristic algorithm. These variants are introduced in detail, including explanations of all adaptive mechanisms. Subsequently, a complex workflow of experiments is described, followed by a summary of the results and discussion.

1.1 Related Works and Motivation

As stated before, there exists a gap in knowledge about the effectiveness of metaheuristic algorithms in optimizing the parameters of a model with time delay and the impact of different algorithm parameter settings and proper choice of core/adaptive functionality on their performance. To gain a better understanding, this research paper aims to investigate and identify the most effective internal mechanisms influencing population behavior and reveal possible weaknesses of these adaptation mechanisms for different settings of the search space range and methods for dealing with the constraints defined by the optimization problem itself. This clearly defines the motivation for research, which is not the extensive benchmarking study of various metaheuristic algorithms.

We are aware that recent research [2, 18, 24] has highlighted the need to move beyond creating new algorithms and focus on understanding the function and taxonomy of existing metaheuristic algorithms. Therefore, previous research with the same optimization problem modified and supplemented the chosen algorithms with mechanisms supporting exploratory behavior and knowledge sharing [14, 19]. The results demonstrated these approaches' effectiveness compared to classical evolutionary algorithms, such as genetic algorithms (GAs). Another study [7] then attempted to identify more closely the problem of choosing the boundaries of the search space and the fact that some of the search parameters may lie very close to the lower bound. At the same time, the upper bound is essentially unknown due to the black-box nature of the problem. Another observation was that a specific implementation of the CMA-ES algorithm [9] (from the jMetalPy framework) supporting bi-population and restarting [8] could handle the optimization problem better than the classical evolutionary algorithm, genetic algorithms (GA), and the Nelder-Mead [20] optimization method, used as a baseline technique. However, the above referred study focused on the aforementioned fitness landscape issue and the discrepancy between the seemingly low (suitable) fitness value and the resulting inferior system identification and stability in the frequency domain. Thus, it did not focus on benchmarking and comparing several optimization methods.

All these facts were behind the self-organizing migrating algorithm (SOMA) choice, the experiments' workflow, and the nature of the obtained data processing (not benchmarking). The SOMA has gained renewed interest from the research community due to its adaptive nature, exploration capabilities, and ability to solve complex problems. Recently, many powerful modern versions have been introduced [23]. We have chosen the original version as the baseline method and three state-of-the-art variants when each of these variants includes a different adaptive technique focusing on either population organization, automatic movement control on the search space, data-driven population analysis (clustering), and an ensemble method for strategy and hyperparameter selection.

2 TIME-DELAYED SYSTEM

This section describes a model for optimizing the identification of a time-delayed system. It also discusses the challenges that need to be dealt with in this area, especially when the objective function and sampling points or frequency are not properly chosen, and the findings from a recent study [7] point to the optimization problem near the search space's boundaries.

2.1 General Description

The considered time-delay identification problem [21] is defined by a model transfer function $G_m : \mathbb{C} \rightarrow \mathbb{C}$ (1),

$$G_{m,\mathbf{p}}(s) = \frac{b_0 + b_{0,\tau} e^{-\tau_0 s}}{s^3 + a_2 s^2 + a_1 s + a_0 + a_{0,\theta} e^{-\theta s}} e^{-\tau s}. \tag{1}$$

Parameters of such a model form a 9-dimensional real vector

$$\mathbf{p} = \left[b_0, b_{0,\tau}, \tau_0, \tau, a_2, a_1, a_0, a_{0,\theta}, \theta \right]. \tag{2}$$

As usual, we assume that some of the parameters are related due to the static gain, i.e.

$$k = \frac{b_0 + b_{0,\tau}}{a_0 + a_{0,\theta}}, \tag{3}$$

where the value of k is well known (or estimated). In our case we used the value $k = 0.0322$.

2.2 Constrains

To achieve appropriate properties of solutions (such as stability, feasibility, and minimum-phase conditions) we use the following constraints:

$$\begin{aligned}
&\tau_0 > 0, \tau > 0, \theta > 0,\\
&a_2 > 0, a_1 > 0, a_0 + a_{0,\theta} > 0,\\
&a_2 a_1 > a_0,\\
&a_2 a_1 > a_0 + a_{0,\theta},\\
&\frac{a_{0,\theta}}{\sqrt{(a_0 - a_2\omega^2)^2 + \omega^2(a_1 - \omega^2)^2}} < 1, \quad \forall \omega > 0,\\
&|b_0| > |b_{0,\tau}|,\\
&a_0 \neq 0, a_{0,\theta} \neq 0, b_{0,\tau} \neq 0.
\end{aligned} \tag{4}$$

Our main task is to find such parameter values that

$$G_{m,\mathbf{p}}(\mathbf{j}\,\omega_i) = A_i + \mathbf{j}\,B_i \tag{5}$$

for some fitting points $\omega_1, \ldots, \omega_n$ and some measured values of $A_1, \ldots, A_n$ and $B_1, \ldots, B_n$, where $\mathbf{j}$ is the imaginary unit ($\mathbf{j}^2 = -1$), see Table 1.

To solve (5) using optimization methods we reformulate it using the classical least-square approach. It consists in the construction of a cost (or fitness) function,

$$C(\mathbf{p}) = \sum_{i=1}^{n} \left[\left(\Re G_{m,\mathbf{p}}(\mathbf{j}\omega_i) - A_i\right)^2 + \left(\Im G_{m,\mathbf{p}}(\mathbf{j}\omega_i) - B_i\right)^2 \right] \tag{6}$$

This way we obtain the final version of our optimization problem, which is to find such parameter values $\mathbf{p}^*$ that

$$C(\mathbf{p}^*) = \min_{\mathbf{p}\in\mathcal{D}} C(\mathbf{p}), \tag{7}$$

where $\mathcal{D}$ is the set of all $\mathbf{p} \in \mathbb{R}^9$ satisfying (3) and (4). The experiments were conducted in $\mathbb{R}^8$ since parameter b_0 was calculated using (3).

2.3 Challenges for Optimization

Solving parametric optimization problems usually brings several challenges. Solving an inverse problem consists of minimizing a misfit function on a set of fitting points. However, this set has to be designated arbitrally. Distribution of frequencies are crucial for the process stability, However, wrongly selected fitting points (frequencies) can cause very low sensitivity to the changes in model parameters and obtained results did not satisfy quality validation.

Another major issue affecting the outcome is the nature of the delay system itself. If feedback loops inside the process include delays (i.e., the so-called state or internal delays), the system is infinite-dimensional because of an infinite number of its modes. Only a limited set of model parameters determines its properties driven by an infinite set of model-free response components. For most time-delay systems, only a subset of the so-called dominant modes has a decisive impact on system features in the time domain as well as in the frequency domain [17]. This poses a challenge for metaheuristics, as there is a need to search a multimodal constrained space, and moreover, certain search parameters may be very close to the boundaries of the search space. All this imposes demands on the configuration of the metaheuristic algorithm.

Table 1: Observation data

i	ω_i	A_i	B_i
1	0.0002	0.03238	-0.00284
2	0.0003	0.03213	-0.00424
3	0.0005	0.03137	-0.00694
4	0.0008	0.02962	-0.01063
5	0.001	0.02813	-0.01278
6	0.0012	0.02645	-0.01465
7	0.0015	0.02371	-0.01692
8	0.0018	0.02087	-0.01857
9	0.002	0.01899	-0.01936
10	0.003	0.01063	-0.02054
11	0.005	0.00057	-0.01713
12	0.008	-0.00540	-0.01110
13	0.01	-0.00704	-0.00795
14	0.011	-0.00757	-0.00658
15	0.012	-0.00795	-0.00531
16	0.014	-0.00843	-0.00296
17	0.016	-0.00860	-0.00074
18	0.018	-0.00846	0.00147
19	0.02	-0.00795	0.00377
20	0.025	-0.00346	0.00982

3 METAHEURISTIC ALGORITHMS

The selected variants of the SOMA are described in the following subsections together with hyperparameter settings describing the algorithm configurations and the problem instances to be solved. Why this algorithm was chosen is explained in the "motivation" section 1.1. In following pseudocodes, *D* represents the dimension of optimized problem, *MAXFES* is the maximum number of objective function evaluations (optimization budget), and *NP* is population size, and lastly, the *FEs* parameter represents the number of evaluations of the objective function currently spent at the observed moment during the run of the algorithm.

3.1 Baseline SOMA

SOMA is a population-based metaheuristic algorithm that modifies traditional crossover and mutation operations to simulate a social group of individuals. The algorithm follows a simple process in the basic variant of SOMA, called All-to-One. At the start of each iteration, known as the migration loop, the fittest individual is selected as the leader. The remaining individuals then move towards the leader in the search space, taking jumps determined by the *step* parameter until they reach the final position given by the *pathLength* parameter. Each step is evaluated using the fitness function, and the best position, including the initial position of the individual, is selected as the new position of the individual in the next migration loop. The All-to-One variant of SOMA has been described in a recent book and survey [1, 23]. The exact position of each step is calculated according to (8).

$$x_{i,j}^{k+1} = x_{i,j}^{k} + (x_{L,j}^{k} - x_{i,j}^{k}) \cdot t \cdot PRTVector_j \tag{8}$$

Where $x_{i,j}^{k+1}$ is the new position of *i*-th solution (for iteration *k+1*) for dimension *j*, and $x_{i,j}$ is the current position of the *i*-th solution. The $x_{L,j}$ represents a position of a leader (the leader selection depends on the used SOMA strategy). Parameter *t* represents steps from *i*-th solution to the leader. Solution *i* is migrating, by discrete steps, and the best-found solution on *t*-th position is propagated into a new iteration of the algorithm. The *t* parameter is generated in a range starting from 0 to *pathLength* with step size *step*.

The $PRTVector_j$ represents an important mechanism in SOMA. It is generated for each new *t* step. This vector determines which dimensions will be changed in a particular step *t*. In other words, in which dimensions the solution will "head" or "be perturbed" towards the *leader* position or not. Since the SOMA was developed in the context of the control optimization challenge, this vector is called "perturbation". In contrast to other metaheuristics, its control must be understood as a certain threshold value not a mutation probability. The $PRTVector_j$ consists only of values 0 or 1. These values are generated based on the value of *PRT* parameter; the process is detailed in equation (9), where a *rand* is a pseudo-random number from a uniform distribution within the range of 0 to 1.

$$PRTVector_j = \begin{cases} if\ rand_j < PRT, & 1 \\ otherwise, & 0 \end{cases} j = 1, .., D \tag{9}$$

In addition to the All-To-One strategy, there are other basic strategies, namely All-To-Random, and All-To-All.

All-To-Random. This is a strategy that can be considered more exploratory. This strategy contains a *leader* individual as in All-To-One strategy. However, the *leader* is selected randomly from all individuals in the population at the beginning of each iteration.

All-To-All. In this modified strategy, the concept of a *leader* is absent, and all individuals migrate toward each other in the same manner as in the All-To-One strategy. Once an individual completes its migration, it returns to its original position. All individuals' position updates occur after they have completed their migrations. Compared to the All-To-One approach, this strategy explores a larger search space, enabling faster convergence to local or global optima of the optimized problem.

Algorithm 1 SOMA ATO

```
Set D, NP, and MAXFES
Set step, pathLength and PRT
while Stopping criterion not met do
    select the best solution - leader x_L from population
    for i = 1 to NP do
        for t = 0 to pathLength with t+ = step do
            generate PRTVector_i by eq. (9)
            migrate x_i to x_L by eq. (8)
        end for
        save the best x_i to new population
    end for
    record the best solution
end while
```

3.2 ESP SOMA

The ESP SOMA was first presented in 2019 [11] as a response to the problem of choosing optimal setting of the control parameter and strategy of the original SOMA. Such a proper choice strongly influences performance. The performance varies over different optimization tasks and could change over different stages of SOMA execution. Therefore, the adaptation mechanism of control parameters and used strategy for each individual in a population has been introduced in ESP SOMA. The adaptation modification is inspired by Ensemble of Mutation and Crossover Strategies and Parameters in Differential Evolution (EPSDE) [25].

The algorithm's functionality can be described as follows: each individual has its *PRT* value and strategy. In the initialization step, each particle has randomly obtained a *PRT* value from a predefined set and one of the possible strategies: All-To-One, All-To-All, All-To-Random. The migration step is followed by an adaptation step, where the *PRT* and strategies assigned to individuals are adjusted. If the particular individual did not improve its objective function value for a threshold number of subsequent iterations, then the individual is forced to pick a new strategy and *PRT* value based on roulette selection.

3.3 SOMA-CLP

SOMA-CLP [13], is the updated version of its predecessor SOMA-CL [12], which utilizes the idea of data-driven control of two different migration strategies. The first strategy All-To-Random serves mainly as an explorer, which maps the search space. The second strategy is named All-To-Cluster-Leaders and is used to exploit promising areas obtained thanks to the clustering technique. SOMA-CLP uses a linear adaptation of the *PRT* control parameter (10) to generate a perturbation vector, promoting the global transition from the tendency of exploration to exploitation.

$$PRT = 0,08 + 0,90 \cdot \frac{FEs}{MaxFEs} \tag{10}$$

All-To-Random Migration Strategy and Identification of Cluster Leaders. A critical component of this strategy involves storing each evaluated solution in a memory, denoted as $\boldsymbol{M}$. This memory of all previously-visited solutions is utilized in the subsequent subsection. The objective is to select a few promising solutions within this memory and use them as candidate leaders for the next phase, which focuses on exploitation. The selection of leaders is carried out in two stages. In the first stage, a clustering method is used to group all solutions based on their parameter values (i.e., positions within the search space) into several clusters. Specifically, the k-means clustering method [16] is employed. In the second stage, only the solutions with the best fitness within their respective clusters are referred to as cluster leaders. These cluster leaders are sorted based on their fitness function values in ascending order and subsequently employed in the All-To-Cluster-Leaders strategy.

All-To-Cluster-Leaders Migration Strategy. The leader is selected for each migrating individual from the cluster leaders using the Rank Selection technique [26]. This migration strategy has its own parameters $pathLength_L$ with step size $step_L$, to support local search capabilities. After the end of a single iteration of this strategy, the

Algorithm 2 ESP SOMA

```
Set D, NP, and MAXFES
Set gap, step, pathLength, adaptivePRT
for i = 1 to NP do
    if adaptivePRT == 0 then
        PRT_i = 0.3
    else
        PRT_i = random from {0.1, 0.3, 0.5, 0.7, 0.9}
    end if
    strategy_i = random from {ATO, ATA, ATR}
    counter_i = 0
end for
while Stopping criterion not met do
    for i = 1 to NP do
        if strategy_i == ATO then
            x_L = best solution x
        else if strategy_i == ATR then
            x_L = random solution x
        else
            P = whole population x
            x_L = {P} - x_i
        end if
        for t = 0 to pathLength with t+ = step do
            generate PRTVector_i
            migrate x_i to x_L
        end for
        save best x_i to new population
        if x_i not improved then
            counter_i += 1
            if counter_i > gap then
                counter_i = 0
                PRT_i = roulette PRT
                strategy_i = roulette strategy
            end if
        else
            counter_i = 0
        end if
    end for
    record the best solution
end while
```

Algorithm 3 SOMA-CLP

```
Set D, NP, NP_L, and MAXFES
Set step, pathLength
Set step_L, pathLength_L
while Stopping criterion not met do
    M = Ø
    update PRT by eq. (10)
    for i = 1 to NP do
        x_L = pick random solution x
        for t = 0 to pathLength with t+ = step do
            generate PRTVector by eq. (10)
            migrate x_i to x_L by eq. (8)
            save each evaluated solution into M
        end for
    end for
    k-means clustering method for solutions stored in M
    keep only best-solution from each cluster
    sort the remaining solutions
    for i = 1 to NP do
        x_L = Rank Selection from cluster leaders
        for t = 0 to pathLength_L with t+ = step_L do
            generate PRTVector by eq. (10)
            migrate x_i to x_L by eq. (8)
        end for
    end for
    record the best solution
end while
```

whole process (algorithm) is again started from strategy All-To-Random and the memory $\boldsymbol{M}$ is cleared.

3.4 SOMA T3A

The SOMA T3A was introduced in 2019 [3] and was tested in the CEC 2019 100 digits competition, where it achieved 4th place. This algorithm features several modifications compared to the original SOMA design, with a particular emphasis on the population organization process. This process consists of three repeated activities: organization, migration, and update.

The organization process involves two primary activities: the selection of individuals who will migrate (termed migrants) and the selection of a leader. Initially, m individuals are randomly selected from the population, and the n best individuals are chosen from these, where n≤m. These individuals become migrants. For leader selection, k individuals are randomly selected from the population, and the best individual from this set becomes the leader. The selected migrants then move towards the chosen leader, except when the leader is also one of the migrants (in which case, that individual skips the migration process).

In the migration process, the *PRT* parameter, to which SOMA exhibits significant sensitivity in the SOMA [22], has an adaptive function. The control of this parameter is based on the philosophy of transitioning from exploration to local search, gradually increasing from low values to 1, according to the following equation (11):

$$PRT = 0,05 + 0,90 \cdot \frac{FEs}{MAXFES} \tag{11}$$

Another change is the gradual reduction of the step size (12), and the number of jumps is fixed and thus not bound to the *pathLength* parameter, as it is in the classic version of SOMA. This means that each individual moves toward the *leader* by a certain number of jumps (hence parameter *Njumps*). At these stepping points, the individual is evaluated by a fitness function.

$$step = 0,15 - 0,08 \cdot \frac{FEs}{MAXFES} \tag{12}$$

Although this version of the algorithm removed the need to set the *pathLength* and *step* parameters, it added three new parameters for managing the organization process (m, n, and k).

Algorithm 4 SOMA T3A

```
Set D, NP and MAXFES
Set m, n, k and Njumps
while Stopping criterion not met do
    update PRT by eq. (11)
    update step by eq. (12)
    choose randomly m individuals from population
    choose the best n Migrants out of m individuals
    for i = 1 to n do
        choose randomly k individuals from population
        choose the leader x_L from k individuals
        if the x_i is not the x_L then
            for t = 1 to Njumps do
                generate PRTVector_i by eq. (9)
                migrate x_i to x_L by eq. (8)
                checking boundary
                re-evaluate fitness function
                updated the better position of the x_i
            end for
        end if
    end for
    record the best solution
end while
```

Table 2: Parameter search ranges

Name	L limit 1	L limit 2,3	H limit 1	H limit 2	H limit 3
b_{0D}	0	0	10	100	500
τ_0	0	0	100	500	500
τ	0	0	250	1000	1000
a_2	0	0	1000	2500	2500
a_1	0	0	1000	2500	2500
a_0	-100	-250	100	250	250
a_{0D}	-100	-250	100	250	250
θ	0	0	250	1000	1000

4 EXPERIMENT DESIGN

This research aimed to investigate in detail the behavior of different adaptive strategies of the chosen algorithm on a black box optimization problem with a specific location of the solution near the boundaries of the search space. For this reason, the experiment was divided into three case studies (problem instances) for a total of three different settings of the search space boundaries. These are defined in Table 2.

Each problem instance was then solved using 12 configurations of the SOMA. These 12 configurations contained four different variants of the SOMA, each for three different constraint handling methods.

The aim was also to observe how the search capabilities of the algorithm and the final results would be affected under different constraints handling methods and under different adaptive strategies of the SOMA. The nature of the optimization problem allows evaluating the fitness function even in the case of infeasible solutions (as long as they lie within bounds). Therefore, complex methods to avoid infeasible solutions on an algorithmic level were not implemented, but a method of random re-initialization, penalization, and finally ignoring constraints was chosen, followed by a post hoc analysis of the best results from repeated runs.

Table 3: Algorithm instances

No.	Description
1	SOMA All-To-One Random re-initialization
2	SOMA All-To-One Penalization
3	SOMA All-To-One No constraints
4	SOMA CLP Random re-initialization
5	SOMA CLP Penalization
6	SOMA CLP No constraints
7	SOMA ESP Random re-initialization
8	SOMA ESP Penalization
9	SOMA ESP No constraints
10	SOMA T3A Random re-initialization
11	SOMA T3A Penalization
12	SOMA T3A No constraints

All experiments with three problem instances and 12 algorithm configurations (see Table 3) were repeated 30 times with maximum fitness function evaluations (*MAXFES*) set to 400 000 and population size (*NP*) of 100. The parameter settings (if required) of the SOMA variants were as follows: $PRT = 0.1$, $pathLength = 3.0$, $step = 0.11$, $Njump = 45$, $m = 10$, $n = 5$, $k = 10$, $gap = 2$, $adaptivePRT$ = True, $pathLength_L = 2$, and $step_L = 0.11$ These parameters have been chosen based on the recommendations of the Authors of SOMA variants, which were used in benchmarking competitions.

5 RESULTS

The results for three problem instances and 12 algorithm configurations are shown and organized in the following way: statistical data are presented in Tables 4–6, alongside with the box-whisker plots depicted in Figures 1–3, and convergence plots in Figures 4–6. In these figures, the line system is as follows: solid line: re-initialization, dashed line: penalization, dotted line: no-constraints. The feasibility of solutions for configurations 3, 6, 9, and 12 cannot be ensured. Thus those results are marked with an asterisk (*). The best feasible result was achieved for problem instance two and algorithm configuration 10 (SOMA T3A, random re-initialization, highlighted in bold). The parameter structure of this solution is: {6.56695459e-02, 2.75779657e+01, 1.42609578e+02, 2.43639544e+03, 5.96581118e+02, 1.24201482e+01, -6.54902690e+00, 1.64310256e+02}, confirming the problem of near bounds optimization problem (dimensions 1 and 4). Discussion about the revealed findings is presented in the next section.

6 DISCUSSION AND CONSLUSION

Building on the foundation established by previous studies [7, 14, 19], this research aims to enhance our understanding of metaheuristic algorithms in control engineering and shed light on their potential for optimizing the parameters of time delay system models. Rather than focusing on benchmarking various SOMA variants, the primary objective of this study was to conduct a comparative

Table 4: Results for problem instance 1 (bounds limits 1)

Conf.	min	max	mean	std.dev.
1	2.7751E-07	3.0534E-07	2.8440E-07	4.6720E-09
2	2.8065E-07	3.9946E-07	2.9599E-07	2.8050E-08
3*	2.7858E-07	5.4496E-05	2.0901E-06	9.8980E-06
4	2.7853E-07	6.4660E-05	2.4662E-05	3.0923E-05
5	2.7886E-07	6.4583E-05	1.7038E-05	2.8270E-05
6*	2.7983E-07	6.4660E-05	2.3247E-05	2.9926E-05
7	3.2117E-07	7.5444E-06	1.7705E-06	1.6090E-06
8	4.4355E-07	1.2137E-05	3.9110E-06	3.3825E-06
9*	3.7627E-07	4.1142E-05	6.0227E-06	8.2531E-06
10	2.5244E-07	2.8026E-07	2.7122E-07	7.3559E-09
11	2.5858E-07	2.8010E-07	2.7233E-07	6.0956E-09
12*	1.0759E-08	1.5207E-07	5.8109E-08	5.4885E-08

Table 5: Results for problem instance 2 (bounds limits 2)

Conf.	min	max	mean	std.dev.
1	2.8377E-07	6.6673E-05	3.1407E-05	3.2585E-05
2	2.8374E-07	6.8303E-05	2.6969E-05	3.2684E-05
3*	1.5842E-07	5.7268E-05	4.0895E-06	1.4432E-05
4	2.7990E-07	6.4689E-05	2.0093E-05	2.9648E-05
5	2.7690E-07	6.4605E-05	1.5071E-05	2.7273E-05
6*	2.7946E-07	6.4730E-05	9.5604E-06	2.2130E-05
7	9.6255E-07	6.8549E-05	2.5130E-05	2.1820E-05
8	4.1894E-06	1.0308E-04	3.9904E-05	2.7388E-05
9*	2.6173E-06	9.8096E-05	3.2716E-05	2.8043E-05
10	**2.3465E-07**	2.8074E-07	2.7184E-07	9.3330E-09
11	2.6700E-07	2.8042E-07	2.7598E-07	3.6866E-09
12*	4.2726E-08	5.2244E-05	6.2791E-06	1.0869E-05

Table 6: Results for problem instance 3 (bounds limits 3)

Conf.	min	max	mean	std.dev.
1	2.8375E-07	6.6268E-05	2.9037E-05	3.2477E-05
2	2.8081E-07	6.7234E-05	3.3529E-05	3.2563E-05
3*	2.5709E-07	5.7731E-05	4.7722E-06	1.2639E-05
4	2.7982E-07	6.4710E-05	1.8542E-05	2.8387E-05
5	2.7822E-07	6.4585E-05	1.9311E-05	2.9572E-05
6*	2.7981E-07	6.4671E-05	2.5818E-05	3.1815E-05
7	1.1684E-06	7.7289E-05	3.4576E-05	2.5457E-05
8	8.0244E-06	1.8836E-04	5.3389E-05	3.9135E-05
9*	3.0885E-06	1.2325E-04	5.2677E-05	3.4290E-05
10	2.5898E-07	7.3259E-06	5.0954E-07	1.2874E-06
11	2.5616E-07	7.9723E-06	6.2715E-07	1.4860E-06
12*	4.2251E-08	1.6375E-05	4.0056E-06	4.7001E-06

analysis of the algorithm behavior, with a particular emphasis on adaptive mechanisms and constraint handling. This approach allows for a deeper exploration of the intricacies involved in applying metaheuristic algorithms to control engineering applications.

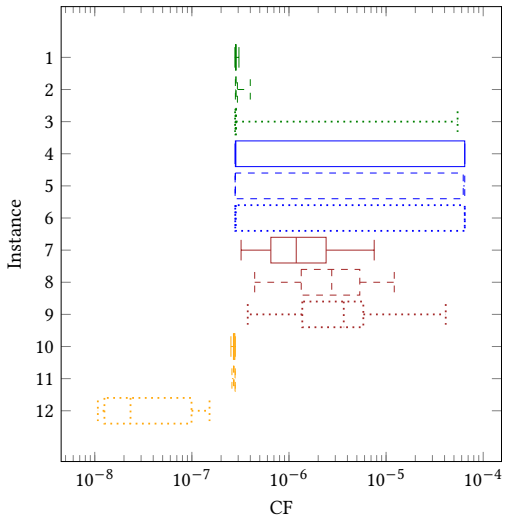

Figure 1: Box plots test 1

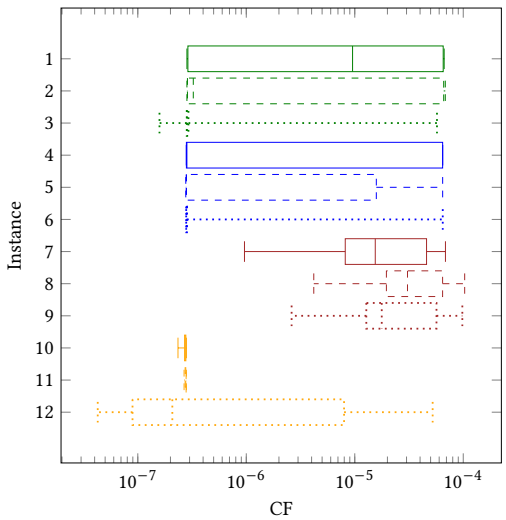

Figure 2: Box plots test 2

The results support the need to pay careful attention to the configuration of the metaheuristic algorithm, the choice of its adaptive techniques, and the setting of its hyperparameters. Focusing on the individual adaptive features, it can be seen from the convergence plots 4–6 and box-whisker plots 1–3 that the baseline SOMA only shows promising results for a smaller range of model parameters (problem instance 1). However, the adaptive mechanisms for data-driven population analysis (clustering), switching exploration and exploitation strategies (even with *PRT* control similar to SOMA T3A - see below), and the basic ensemble SOMA do not achieve good results in all instances. The chosen constraint treatment strategy did not matter much for these configurations.

Notable results were achieved by SOMA T3A, which includes adaptive functionality for linear control of the *PRT* parameter and specific population organization that can support the exploratory

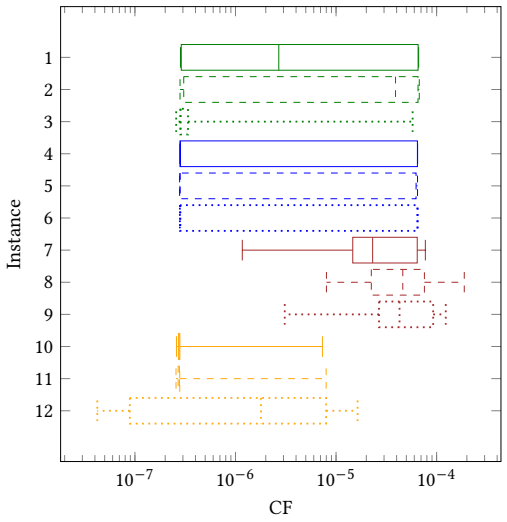

Figure 3: Box plots test 3

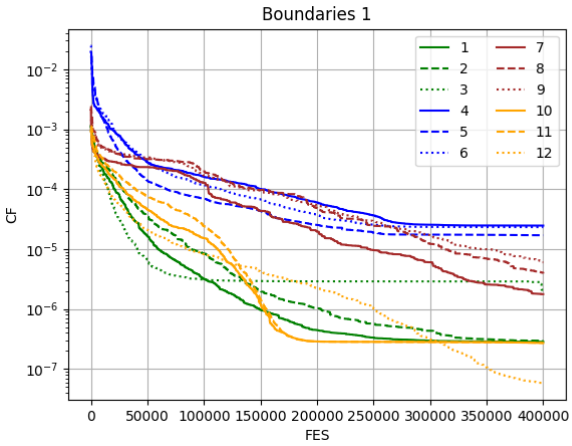

Figure 4: Convergence plot - problem instance 1

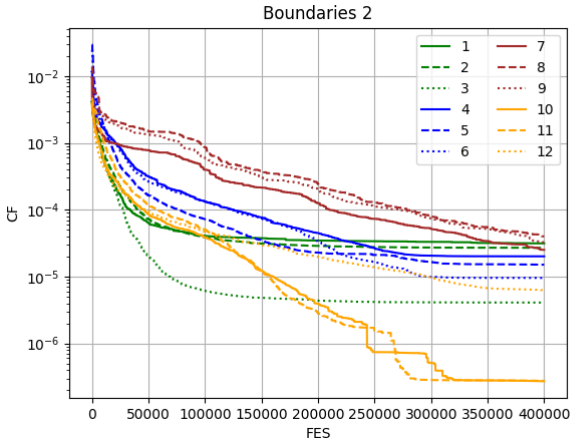

Figure 5: Convergence plot - problem instance 2

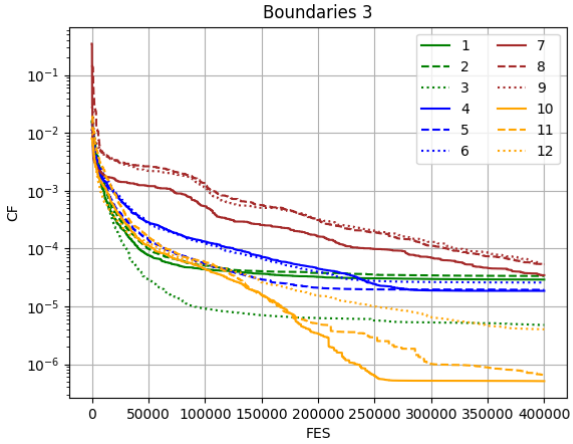

Figure 6: Convergence plot - problem instance 3

capabilities of the algorithm and local search in the later stages of optimization.

Regarding the constraint treatment techniques, for this specific case, characterized as "optimum near the domain boundary," it seems more convenient to use random reinitialization rather than penalization. However, it has to be acknowledged that by not treating the constraints (again, due to the specific nature of the constraints - fitness can be evaluated, but from a stability perspective and in the frequency domain, the solution may be infeasible), the algorithms found the lowest fitness values and thus the evolutionary process was probably better supported without interfering with the population by reinitializing individuals or penalizing. Yet, subsequent post-hoc analysis showed that very few of these solutions were feasible, and in terms of fitness, these few feasible solutions have worse fitness than, e.g., the best solution of other algorithm configurations. Thus, it can be argued that for this type of real-world problem, it is advantageous to use an algorithm configuration with an organizing process or sub-populations (even with partial restarting or injection of individuals) and to control the exploration-to-exploitation transition. Future research may then include methods for bound constraints and, of course, identifying the most important algorithm components for using autoconfiguration frameworks.

ACKNOWLEDGMENTS

The research presented in this paper was partially supported by: NCN project no: 2020/39/I/ST7/02285, Polish Ministry of Education and Science funds assigned to AGH University of Science and Technology. It was also supported by Czech Science Foundation (GACR) project no: GF21-45465L, the Internal Grant Agency of the Tomas Bata University in Zlin - IGA/CebiaTech/2023/004, and resources of A.I.Lab at the Faculty of Applied Informatics, Tomas Bata University in Zlin (ailab.fai.utb.cz).

REFERENCES

[1] Donald Davendra and Ivan Zelinka. 2016. Self-organizing migrating algorithm. *New optimization techniques in engineering* (2016). Publisher: Springer.

[2] Jesica de Armas, Eduardo Lalla-Ruiz, Surafel Luleseged Tilahun, and Stefan Voß. 2022. Similarity in metaheuristics: a gentle step towards a comparison methodology. *Natural Computing* 21, 2 (2022), 265–287.

[3] Quoc Bao Diep, Ivan Zelinka, Swagatam Das, and Roman Senkerik. 2020. SOMA T3A for Solving the 100-Digit Challenge. In *Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing (Communications in Computer and Information Science)*, Aleš Zamuda, Swagatam Das, Ponnuthurai Nagaratnam Suganthan, and Bijaya Ketan Panigrahi (Eds.). Springer International Publishing, Cham, 155–165.

[4] Absalom E Ezugwu, Amit K Shukla, Rahul Nath, Andronicus A Akinyelu, Jeffery O Agushaka, Haruna Chiroma, and Pranab K Muhuri. 2021. Metaheuristics: a comprehensive overview and classification along with bibliometric analysis. *Artificial Intelligence Review* 54 (2021), 4237–4316.

[5] E. Fridman. 2014. *Introduction to Time-Delay Systems: Analysis and Control*. Springer International Publishing.

[6] Wenjuan Gu, Yongguang Yu, and Wei Hu. 2017. Artificial bee colony algorithm-based parameter estimation of fractional-order chaotic system with time delay. *IEEE/CAA Journal of Automatica Sinica* 4, 1 (2017), 107–113.

[7] Hubert Guzowski, Maciej Smołka, Aleksander Byrski, Libor Pekar, Zuzana Kominkova Oplatkova, Roman Senkerik, Radek Matusu, and Frantisek Gazdos. 2022. Effective Parametric Optimization of Heating-Cooling Process with Optimum near the Domain Border. In *2022 IEEE 11th International Conference on Intelligent Systems (IS)*. IEEE, 1–6.

[8] Nikolaus Hansen. 2009. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. *GECCO (Companion)* (07 2009). https://doi.org/10.1145/1570256.1570333

[9] Nikolaus Hansen. 2016. The CMA Evolution Strategy: A Tutorial. https://doi.org/10.48550/ARXIV.1604.00772

[10] Shintaro Ikeda and Ryozo Ooka. 2015. Metaheuristic optimization methods for a comprehensive operating schedule of battery, thermal energy storage, and heat source in a building energy system. *Applied energy* 151 (2015), 192–205.

[11] Tomas Kadavy, Michal Pluhacek, Roman Senkerik, and Adam Viktorin. 2019. The ensemble of strategies and perturbation parameter in self-organizing migrating algorithm solving CEC 2019 100-digit challenge. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 372–375.

[12] Tomas Kadavy, Michal Pluhacek, Adam Viktorin, and Roman Senkerik. 2020. Self-organizing migrating algorithm with clustering-aided migration. In *Proceedings of the 2020 genetic and evolutionary computation conference companion*. 1441–1447.

[13] Tomas Kadavy, Michal Pluhacek, Adam Viktorin, and Roman Senkerik. 2021. SOMA-CLP for competition on bound constrained single objective numerical optimization benchmark: a competition entry on bound constrained single objective numerical optimization at the genetic and evolutionary computation conference (GECCO) 2021. In *Proceedings of the genetic and evolutionary computation conference companion*. 11–12.

[14] Piotr Kipiński, Hubert Guzowski, Aleksandra Urbańczyk, Maciej Smołka, Marek Kisiel-Dorohinicki, Aleksander Byrski, Zuzana Kominkova Oplatkova, Roman Senkerik, Libor Pekar, Radek Matusu, et al. 2022. Socio-cognitive Optimization of Time-delay Control Problems using Evolutionary Metaheuristics. In *2022 IEEE 11th International Conference on Intelligent Systems (IS)*. IEEE, 1–7.

[15] Guo-Han Lin, Jing Zhang, and Zhao-Hua Liu. 2018. Hybrid particle swarm optimization with differential evolution for numerical and engineering optimization. *International Journal of Automation and Computing* 15, 1 (2018), 103–114.

[16] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[17] Wim Michiels and S.-I Niculescu. 2014. *Stability, control and computation or time-delay systems. An eigenvalue based approach*. SIAM.

[18] Daniel Molina, Javier Poyatos, Javier Del Ser, Salvador García, Amir Hussain, and Francisco Herrera. 2020. Comprehensive taxonomies of nature-and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis recommendations. *Cognitive Computation* 12 (2020), 897–939.

[19] Mateusz Nabywaniec, Hubert Guzowski, Aleksandra Urbańczyk, Maciej Smolka, Marek Kisiel-Dorohinicki, Aleksander Byrski, Zuzana Kominkova Oplatkova, Roman Senkerik, Libor Pekar, Radek Matusu, et al. 2022. Socio-cognitive optimization of time-delay control problems using agent-based metaheuristics. In *2022 IEEE 11th International Conference on Intelligent Systems (IS)*. IEEE, 1–7.

[20] John A. Nelder and Roger Mead. 1965. A simplex method for function minimization. *Computer Journal* 7 (1965), 308–313.

[21] Libor Pekař, Mengjie Song, Subhransu Padhee, Petr Dostálek, and František Zezulka. 2022. Parameter identification of a delayed infinite-dimensional heat-exchanger process based on relay feedback and root loci analysis. *Scientific Reports* 12, 1 (03 Jun 2022), 9290. https://doi.org/10.1038/s41598-022-13182-5

[22] Michal Pluhacek, Anezka Kazikova, Tomas Kadavy, Adam Viktorin, and Roman Senkerik. 2021. Explaining SOMA: the relation of stochastic perturbation to population diversity and parameter space coverage. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1944–1952.

[23] Lenka Skanderova. 2023. Self-organizing migrating algorithm: review, improvements and comparison. *Artificial Intelligence Review* 56, 1 (2023), 101–172.

[24] Kenneth Sörensen. 2015. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research* 22, 1 (2015), 3–18. ISBN: 0969-6016 Publisher: Wiley Online Library.

[25] Guohua Wu, Xin Shen, Haifeng Li, Huangke Chen, Anping Lin, and Ponnuthurai N Suganthan. 2018. Ensemble of differential evolution variants. *Information Sciences* 423 (2018), 172–186.

[26] G Zames, NM Ajlouni, NM Ajlouni, NM Ajlouni, JH Holland, WD Hills, and DE Goldberg. 1981. Genetic algorithms in search, optimization and machine learning. *Information Technology Journal* 3, 1 (1981), 301–302.