ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2019.1.111

PARETO-BASED SELF-ORGANIZING MIGRATING ALGORITHM

Quoc Bao Diep[1,✉], **Ivan Zelinka**[1], **Swagatam Das**[2]

[1]Faculty of Electrical Engineering and Computer Science, VSB - Technical University of Ostrava, Czech Republic
[2]Electronics and Communication Sciences Unit, Indian Statistical Institute, India

diepquocbao@gmail.com✉, ivan.zelinka@vsb.cz, swagatam.das@isical.ac.in

Abstract

In this paper, we propose a new method named Pareto-based self-organizing migrating algorithm (SOMA Pareto), in which the algorithm is divided into the Organization, Migration, and Update processes. The important key in the Organization process is the application of the Pareto Principle to select the Migrant and the Leader, increasing the performance of the algorithm. The adaptive PRT, Step, and PRTVector parameters are applied to enhance the ability to search for promising subspaces and then to focus on exploiting that subspaces. Based on the testing results on the well-known benchmark suites such as CEC'13, CEC'15, and CEC'17, the superior performance of the proposed algorithm compared to the SOMA family and the state-of-the-art algorithms such as variant DE and PSO are confirmed. These results demonstrate that SOMA Pareto is an effective, promising algorithm.

Keywords: *self-organizing migrating algorithm, SOMA, single objective optimization, swarm intelligence.*

Received: 02 May 2019
Accepted: 31 May 2019
Published: 24 June 2019

1 Introduction

In recent years, swarm intelligence-based algorithms have been increasingly developed. It is attractive not only because of its simplicity and efficiency, but it can solve complex problems without many complex equations, refer to [11] for more details. Some of the most regarded-known algorithms such as particle swarm optimization [19], ant colony optimization [16], artificial bee colony algorithm [17], and another algorithm that are confirming its position, which is the self-organizing migrating algorithm.

The SOMA algorithm was inspired by the competitive-cooperative intelligent behavior of individuals in the population, through many migration loops, to find the global optimal solution of the problem [8, 33]. First introduced in 1999 [35], over two decades of development, SOMA has attracted many researchers and many different variants were proposed, besides the original methods of SOMA AllToOne and AllToAll. The most recent is the SOMA team to team adaptive - SOMA T3A [13]. It overcomes the disadvantages of the original versions and proposed a new method of selecting Migrants and Leader. It also introduced adaptive control parameters for each migration loop, the performance has been demonstrated through the CEC'13 and CEC'17 benchmark test suites. In addition, other variants of SOMA have also confirmed its superior performance compared to the original version such as SOMGA [10], C-SOMGA [9], CSOMA [28], SOMAQI [30], M-SOMAQI [29], mNM-SOMA [1], and HSOMA [24].

The SOMA algorithm is widely applied in many real-world applications such as computer games [34,36], real-time robot control in an unknown environment [3,15], or participating in single objective numerical optimization competitions [14, 37]. However, actual problems arise more and more complicated. On the one hand, the algorithm must strictly adhere to the given number of function evaluations because of the importance of time in many real-world scenarios. On the other hand, solving time is sometimes less important than having the correct answer. That makes the existing versions of SOMA gradually become obsolete and not to keep up with these actual contexts. Therefore, developing new methods is necessary to solve these complex problems.

In this paper, we propose a new method, named Pareto-based self-organizing migrating algorithm (SOMA Pareto). In Sec.2, the original version of SOMA is described. In Sec.3, the proposed algorithm SOMA Pareto is presented. Experiment setup and simulation results are described in Sec.4 and Sec.5, respectively. Finally, the paper concludes with Sec.6.

2 The Self-Organizing Migrating Algorithm

To solve increasingly complex problems, many variant versions of the swarm intelligent algorithms are constantly being improved and proposed, SOMA is not an exception. This section briefly describes the original version of SOMA as a background for the proposed algorithm.

Self-organizing migrating algorithm (SOMA), a stochastic optimization algorithm inspired by the intelligent behavior of natural creatures such as birds and fish. In numerical optimization field, the SOMA mission is to search globally optimal solutions. To accomplish that, the SOMA begins by generating a population containing a given number of individuals, each of which is a candidate solution to the problem. Other possible solutions were then created through many migration loops, better than the initial solutions based on the competition - cooperation between these individuals, a key feature of the swarm intelligent algorithm. And it is repeated until the given stop conditions of the algorithm are satisfied [8,33]. This process is described in detail below.

In the beginning, individuals are randomly generated in the entire search space, according to Eq. 1. Each individual is a candidate solution to the problem. They are then evaluated by the given fitness function.

$$P = x_j^{(lo)} + rand(x_j^{(hi)} - x_j^{(lo)}) \tag{1}$$

where:

- P: the initial population of the algorithm,
- $x_j^{(lo)}$: the lowest boundary value,
- $x_j^{(hi)}$: the highest boundary value,
- $rand$: random number from 0 to 1.

After that, the algorithm goes into migration loops. At each migration loop, an individual had the best fitness value is chosen to become the Leader, and the rest individuals become traveling individuals. Each of them will move step by step toward the Leader until the distance from its current to the initial position greater than the $PathLength$. The granularity of movement was specified by the $Step$ parameter. The $PathLength$ and $Step$ are the given number and they do not change their value throughout migration loops. Eq. 2 defines this moving process.

$$x_{n,j}^{ML+1} = x_{c,j}^{ML} + (x_{l,j}^{ML} - x_{c,j}^{ML})\ t\ PRTVector_j \tag{2}$$

where:

- $x_{n,j}^{ML+1}$: the new position in the next migration loop,
- $x_{c,j}^{ML}$: the position in current migration loop,
- $x_{l,j}^{ML}$: the leader position in current migration loop,
- t: jumping step, from 0, by $Step$, to $PathLength$.

To lead individuals jumping in the $N - k$ dimensional subspace instead of jumping directly toward the Leader, a random number vector is created and compared to the PRT threshold that given before. If it is smaller than PRT, the $PRTVector_j$ of that jump is set to 1 and vice versa, $PRTVector_j$ is set to 0, as Eq. 3.

$$if\ rand_j < PRT;\ PRTVector_j = 1;\quad else,\ 0. \tag{3}$$

When all individuals in the population finish the jumping process, a new migration loop will be started, and the algorithm will go on until the stop conditions are satisfied.

The strategy described above is SOMA AllToOne. In this strategy, all individuals move towards the Leader except itself. The convergence speed of the algorithm is fast. For simple problems and a small number of dimensions, the algorithm can find the global minima, but for complex problems and a larger number of dimensions, the algorithm can be trapped in the local minima [8].

For SOMA AllToAll, all individuals move toward each other instead of toward the Leader like AllToOne. One of the major drawbacks of this strategy is that there have many meaningless movements take place between bad individuals or from good to bad individual, as shown in Fig. 1, resulting in a lot of useless computing time and leading the algorithm face the stop condition before finding the global minima.

So we develop a better solution, named SOMA Pareto, as presented in the next section.

3 SOMA Pareto

The entire process of the SOMA Pareto is described in Fig. 2. In the beginning, a population containing candidate solutions to the problem is generated according to Eq. 1, similar to other versions of SOMA. The given fitness function then evaluates the population and the algorithm goes into the migration loop with three main processes, namely the Organization, Migration, and Update process.

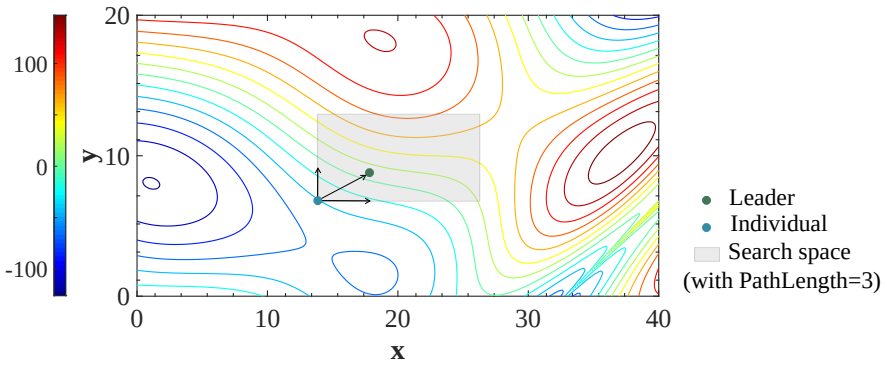

Figure 1: The meaningless move from the individual to the Leader having lower fitness value.

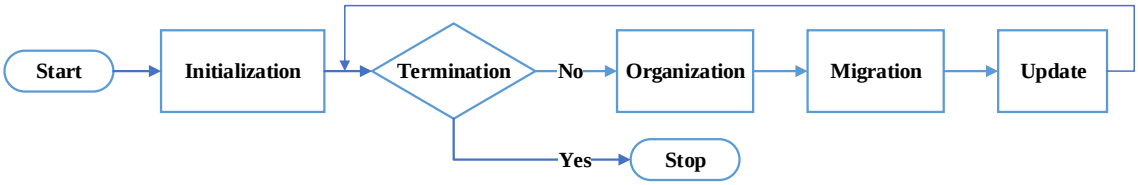

Figure 2: Flowchar of the algorithm.

3.1 The Organization Process

The Organization process determines which individuals in the population will interact with each other to create new candidate solutions that are different from the initial individuals. In other words, it is the process of selecting the Migrant and the Leader, and this Migrant will then move towards the Leader to search the better positions during the move.

The Organization process plays an essential role in exploring promising subspaces and then focusing on exploiting these promising subspaces. Therefore, an individual selected to become the Leader is an individual with good fitness value, but should not always be the best. Because if it is the best, the population will only focus on exploiting the subspace around the best Leader, ignoring other promising subspaces around other good individuals, which may contain global minima. And so the algorithm will converge quickly in local minima. The Migrant should be an individual with the worst fitness value than the Leader's fitness value to avoid the meaningless moves, as point out in Fig. 1.

To choose the Migrant and Leader as analyzed, we propose to apply the Pareto Principle [27]. Accordingly, 20% of the number of individuals (marked as A) hold 80% of the population value, and 80% of the number of individuals (marked as B) only hold 20% of the population value. That means the Leader should be in A and the Migrant should be in B. But not all individuals in B are Migrants, and not all individuals in A are Leaders. Instead, we propose to randomly select an individual in 20% of A (marked as C) as the Leader and randomly select an individual in 20% of B (marked as D) as the Migrant.

In other words, in each loop, the population will be sorted in increasing order of fitness value. And the population is then divided into 2 parts, the first part is A containing 20% of the number of individuals with the best fitness value, and part B containing 80% of the remaining individuals. After that, the best 20% of A will be selected and marked as C, and the best 20% of B will be selected and marked as D. And then, the algorithm randomly selects an individual from C to be the Leader and randomly selects an individual from D to be the Migrant. The entire process is described in Fig. 3.

3.2 The Migration Process

The Migration process searches for a better position by moving the Migrant towards the selected Leader. In the original version, individuals move in the $N - k$ dimensional subspace with the given fixed step until reaching the given $PathLength$. The ratio of each move is determined by the PRT. Using fixed parameters of $Step$, $PathLength$ and PRT lead to limiting the flexibility of the algorithm. Instead, it is better to start by exploring promising subspaces and then focus on exploiting these promising subspaces. That has been overcome in the SOMA T3A version [13], which is to use the adaptive PRT and $Step$ parameters and use the fixed number of jumps (N_{jump}) instead of the $PathLength$, but not yet thorough because $PRTVector$ still only is the fixed value, 0 or 1.

$$PRT = 0.5 + 0.45\ cos(T_1\pi\ \frac{FEs}{MaxFEs} + \pi) \tag{4}$$

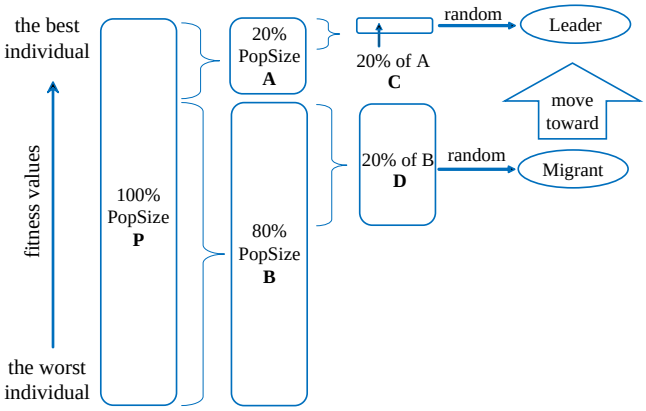

Figure 3: The Organization process.

where:

- FEs: the current function evaluations,
- $MaxFEs$: the maximun of function evaluations.

In this version, we propose the adaptive PRT and $Step$ as shown in Eq. 4 and Eq. 5. These parameters help maintain the diversity of the population, balances between two phases of exploration and exploitation, helping individuals move far away from the trapped area. In this case, $PathLength = Step \times N_{jump}$.

$$Step = 0.35 + 0.15\ cos(T_2\pi\ \frac{FEs}{MaxFEs}) \tag{5}$$

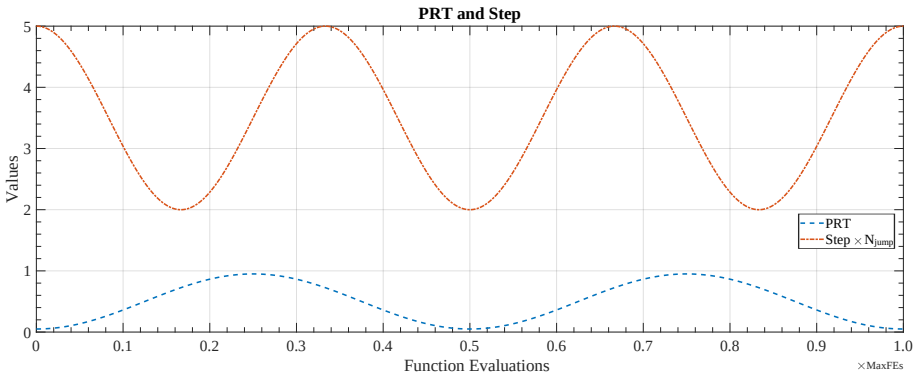

Figure 4: The control parameters.

Not only that, we propose to change the $PRTVector$ parameter. It not only receives two values 0 and 1 like all previous versions but also adapts to each migration loop. It means that the Migrant instead of moving in a $N - k$ dimensional subspace that is perpendicular to the original space, which will move in a narrow $N - k$ dimensional subspace gradually with each migration loop to focus on the Leader. $PRTVector$ is given in Eq. 6.

$$if\ rand_j < PRT;\ PRTVector_j = 1;\ \ else,\ PRTVector_j = \frac{FEs}{MaxFEs}. \tag{6}$$

3.3 The Update Process

The update process is the process of reviewing and deciding whether to replace the initial Migrant. To do that, all jumping positions are evaluated by the fitness function and then one of the best position is selected to compare with the initial position of that Migrant. If the new position is better, it will replace the initial position and become a new individual in the population. On the contrary, it is ignored, and that migration loop has no improvement.

The whole process of SOMA Pareto is described in Algorithm 1.

Algorithm 1 SOMA Pareto

```
Generate and evaluate the initial population
while stopping condition not reached do
    Update PRT and Step
    Sort the population
    Choose the Migrant and the Leader as Fig. 3
    The Migrant moves to the Leader
    Checking boundary
    Re-evaluate fitness function
    Updated the better position of the Migrant
end while
return
```

4 Experiment Setup

The performance of the SOMA Pareto was thoroughly evaluated on the three well-known benchmark suite of the IEEE Congress on Evolutionary Computation as shown below. They contain 73 functions divided into groups of unimodal, multimodal, hybrid, and composition functions.

The first suite consists of 28 functions from the IEEE CEC 2013 Special Session on RealParameter Single Objective Optimization (CEC'13, see detail at [23]);

The second suite consists of 15 functions from the IEEE CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization (CEC'15, see detail at [22]);

The last suite consists of 30 funciton from the IEEE CEC 2017 Special Session and Competition on Single Objective RealParameter Numerical Optimization (CEC'17, see detail at [2]).

These results obtained by SOMA Pareto were first compared to the original and the latest version of SOMA, and then were compared to the state-of-the-art algorithms, as listed below.

Algorithms were compared with SOMA Pareto:

- On the CEC'13:
  - Super-fit Multicriteria Adaptive Differential Evolution [6] (SMASE);
  - A CMA-ES Super-fit Scheme for the Re-sampled Inheritance Search [5] (CMAES-RIS);
  - Differential Evolution with Automatic Parameter Configuration [18] (DE-APC);
  - A Self-adaptive Heterogeneous Particle Swarm Optimization [25] (fk-PSO).
- On the CEC'15:
  - Tuning Maturity Model of Ecogeography-Based Optimization [38] (TEBO);
  - An Improved Covariance Matrix Leaning and Searching Preference Algorithm [7] (ICMLSP);
  - A Self-adaptive Dynamic Particle Swarm Optimizer [21] (SaDPSO);
  - Dynamic Search Fireworks Algorithm with Covariance Mutation [32] (dynFWACM).
- On the CEC'17:
  - Self-Organizing Migrating Algorithm Original [33] (SOMA AllToOne and AllToAll);
  - Self-Organizing Migrating Algorithm Team to Team Adaptive [13] (SOMA T3A);
  - A Version of IPOP-CMA-ES Algorithm with Midpoint [4] (RB-IPOP-CMA-ES);
  - Proactive Particles in Swarm Optimization: a Settings-Free Algorithm [31] (PPSO);
  - Teaching Learning Based Optimization with Focused Learning and its Performance [20] (TLBO-FL);
  - Self-Adaptive Multi Population Elitist Jaya [26] (SAMPE-Jaya).

In the above algorithms, there are 11 algorithms that have participated in the CEC Competition in the corresponding years, except for SAMPE-Jaya and SOMA. Control parameter settings of these algorithms have been used exactly the same as original papers in the citations without any changes. For SOMA AllToOne and AllToAll, $PopSize = 30$ and $PopSize = 20$ respectively; $Step = 0.11$; $PRT = 0.1$; $PathLength = 3$.

The number of dimension of all 73 test functions was set at $30D$ with the same search range $[-100, 100]^D$ and $MaxFEs = 10000 * D$. For each function, each algorithm was independently repeated 51 times. Error values will be taken as 0 if it is smaller than 10^{-8}, as experimental settings requested in [23], [22], and [2]. The

Wilcoxon rank-sum test was applied at the 5% significance level to evaluate whether the differences between the results are significant [12].

SOMA Pareto setting: $PopSize = 100$, $N_{jump} = 10$, PRT and $Step$ as in Eq. 4 and Eq. 5 with $T_1 = T_2 = 1$ for all problems. [1]

5 Simulation Results

5.1 Out-performance of SOMA Pareto compares to other SOMA

The comparison results between SOMA Pareto and other algorithms are presented in form tables, in which each row is the mean and standard deviation errors of run 51 consecutive trials corresponding to each function. The sign (+), (−), and (≈) respectively indicate that the comparing algorithm has significantly better results, significantly worse results, and not significantly better or worse results compared to SOMA Pareto using the Wilcoxon rank-sum test (WRT) at 5% significance level [12]. The best results without statistical tests on each row was marked in bold.

Table 1: Comparison of SOMA Pareto with SOMA AllToOne, AllToAll, and SOMA T3A on the CEC'17 benchmark functions (30 dimensions, 51 runs).

F	SOMA Pareto Mean (Std Dev)	SOMA AllToOne Mean (Std Dev)	SOMA AllToAll Mean (Std Dev)	SOMA T3A Mean (Std Dev)
F_1	2.49e−08 (1.48e−07)	1.48e+03 (2.41e+03)−	5.52e+02 (1.14e+03)−	**0.00e+00 (0.00e+00)**+
F_2	**1.67e+03 (9.51e+03)**	3.61e+08 (2.47e+09)−	9.10e+04 (5.96e+05)−	2.97e+09 (1.46e+10)−
F_3	**3.81e−05 (1.45e−04)**	1.54e+04 (4.40e+03)−	9.89e+03 (3.34e+03)−	1.90e−02 (6.43e−02)−
F_4	**3.77e+00 (9.58e+00)**	8.46e+01 (2.78e+01)−	8.54e+01 (2.19e+01)−	5.12e+01 (3.12e+01)−
F_5	**3.00e+01 (7.20e+00)**	6.81e+01 (6.19e+00)−	4.99e+01 (9.35e+00)−	5.20e+01 (1.70e+01)−
F_6	1.41e−03 (1.83e−03)	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	4.22e−04 (5.76e−04)+
F_7	**5.13e+01 (7.60e+00)**	1.06e+02 (7.50e+00)−	8.27e+01 (8.37e+00)−	7.77e+01 (1.48e+01)−
F_8	**2.85e+01 (7.89e+00)**	7.07e+01 (6.25e+00)−	5.46e+01 (8.48e+00)−	5.65e+01 (1.48e+01)−
F_9	6.37e+00 (5.03e+00)	**7.17e−01 (1.25e+00)**+	3.05e+00 (4.67e+00)+	4.14e+00 (4.32e+00)+
F_{10}	2.84e+03 (5.85e+02)	3.08e+03 (2.21e+02)−	**2.35e+03 (2.98e+02)**+	2.51e+03 (4.67e+02)+
F_{11}	2.80e+01 (1.94e+01)	6.00e+01 (2.77e+01)−	**1.75e+01 (1.32e+01)**+	2.35e+01 (2.15e+01)+
F_{12}	1.13e+04 (5.68e+03)	3.96e+05 (2.70e+05)−	5.09e+05 (3.26e+05)−	**1.04e+04 (5.79e+03)**≈
F_{13}	**7.26e+01 (5.11e+01)**	1.31e+04 (1.39e+04)−	8.30e+03 (7.59e+03)−	1.63e+02 (2.24e+02)−
F_{14}	1.21e+02 (3.14e+02)	4.28e+04 (3.36e+04)−	8.75e+04 (1.14e+05)−	**6.86e+01 (7.42e+01)**+
F_{15}	1.49e+02 (3.22e+02)	7.45e+03 (7.70e+03)−	2.12e+03 (2.42e+03)−	**2.52e+01 (1.77e+01)**≈
F_{16}	6.93e+02 (2.43e+02)	7.83e+02 (1.27e+02)−	5.89e+02 (1.72e+02)+	**5.61e+02 (1.66e+02)**+
F_{17}	**8.78e+01 (8.76e+01)**	2.34e+02 (7.51e+01)−	1.45e+02 (8.89e+01)−	9.63e+01 (7.21e+01)≈
F_{18}	**1.15e+04 (6.72e+03)**	2.09e+05 (1.09e+05)−	2.04e+05 (1.17e+05)−	1.24e+04 (1.34e+04)≈
F_{19}	4.29e+01 (4.28e+01)	7.99e+03 (8.78e+03)−	2.92e+03 (3.60e+03)−	**1.91e+01 (9.23e+00)**+
F_{20}	1.75e+02 (8.22e+01)	2.91e+02 (9.03e+01)−	1.85e+02 (8.71e+01)≈	**1.57e+02 (8.33e+01)**≈
F_{21}	**2.28e+02 (8.31e+00)**	2.79e+02 (8.92e+00)−	2.50e+02 (2.86e+01)−	2.45e+02 (4.41e+01)−
F_{22}	**1.45e+02 (3.21e+02)**	5.43e+02 (1.02e+03)−	6.45e+02 (1.08e+03)−	3.85e+02 (8.73e+02)−
F_{23}	**3.82e+02 (8.72e+00)**	4.26e+02 (9.35e+00)−	4.04e+02 (1.05e+01)−	4.01e+02 (1.70e+01)−
F_{24}	**4.52e+02 (7.26e+00)**	5.49e+02 (1.40e+01)−	5.12e+02 (4.29e+01)−	4.74e+02 (1.79e+01)−
F_{25}	3.88e+02 (3.32e+00)	3.87e+02 (1.09e+00)+	**3.87e+02 (9.42e−01)**+	3.88e+02 (1.11e+00)+
F_{26}	1.41e+03 (2.01e+02)	1.18e+03 (6.92e+02)≈	1.00e+03 (5.68e+02)+	**6.61e+02 (5.68e+02)**+
F_{27}	5.34e+02 (6.75e+00)	5.20e+02 (6.22e+00)+	**5.12e+02 (6.48e+00)**+	5.12e+02 (6.49e+00)+
F_{28}	**3.04e+02 (2.05e+01)**	4.04e+02 (1.14e+01)−	4.02e+02 (4.98e+00)−	3.23e+02 (4.25e+01)−
F_{29}	**5.20e+02 (9.75e+01)**	6.67e+02 (7.73e+01)−	5.29e+02 (7.50e+01)−	5.63e+02 (9.71e+01)−
F_{30}	**3.22e+03 (2.87e+02)**	7.12e+03 (2.83e+03)−	4.60e+03 (9.44e+02)−	4.34e+03 (2.00e+03)−
	+	4	8	11
	−	25	21	14
	≈	1	1	5

Table 1 shows the performance of SOMA Pareto compare to other versions of SOMA. For SOMA AllToOne, SOMA Pareto had 25 out of 30 cases having significantly better results (win), 4 cases having significantly worse results (lose), and one case un-confirm significantly better or worse results (draw) using WRT. Compared to SOMA AllToAll, SOMA Pareto wins 21, lose 8 and draw 1. For SOMA T3A, SOMA Pareto wins 14, lose 11 and draw 5. These results demonstrate that the proposed algorithm completely better than the original versions, as well as the latest version of SOMA.

5.2 Promising results compare to the state-of-the-art algorithms

Mean and standard deviation of SOMA Pareto, SMADE, CMAES-RIS, DE-APC, and fk-PSO tested on 28 functions of CEC'13 are presented in Table 2. Three last rows show the comparison results from SOMA Pareto to the rest. In this experiment, SOMA Pareto achieved competitive results compared to the SMADE and

[1] The source code is publicly available at https://www.mathworks.com/matlabcentral/fileexchange/71724-soma-pareto.

CMAES-RIS when SOMA Pareto won SMADE 9, lose 11 and won CMAES-RIS 10, lose 9. Compare to DE-APC and fk-PSO, the proposed algorithm shows much better performance when reaching 14 wins, only losing 10 and 7 respectively.

Compared to ICMLSP and SaDPSO tested on CEC'15, SOMA Pareto continues to show competitive performance when winning 8 and 7 respectively, losing 7 and 6 respectively. Despite showing the superiority performance to dynFWACM, SOMA Pareto has not yet gained the good when compared with TEBO. These results are shown in Table 3.

Table 4 reports the comparison result between SOMA Pareto and the other algorithms: RB-IPOP-CMA-ES, PPSO, TLBO-FL, and SAMPE-Jaya that implemented on the CEC'17 benchmark suites. In this test suite, SOMA Pareto exhibits completely superior performances compared to other algorithms, when winning 25, 26, 22 out of 30 cases respectively, except RB-IPOP-CMA-ES algorithm.

An important point to note is that 11 of the 12 algorithms used to compare (except SAMPE-Jaya) have been carefully tweaked to attend the CEC Competition in its respective years. For SOMA Pareto there is only one setting to run for 73 functions of 3 benchmark suites. It is believed that the comparison results will be more raised if SOMA Pareto is set individually for each benchmark suite. However, we have proposed the same setting for all issues to keep generality of the proposed algorithm, giving a visual perspective to the reader.

Table 2: Comparison of SOMA Pareto with state-of-the-art algorithms on the CEC'13 benchmark functions (30 dimensions, 51 runs).

F	**SOMA Pareto** Mean (Std Dev)	**SMADE** Mean (Std Dev)	**CMAES-RIS** Mean (Std Dev)	**DE-APC** Mean (Std Dev)	**fk-PSO** Mean (Std Dev)
F_1	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
F_2	8.79e+04 (3.68e+04)	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	1.75e+05 (1.33e+05)−	1.59e+06 (8.11e+05)−
F_3	3.60e+07 (5.49e+07)	9.82e+03 (4.99e+04)+	**2.24e+03 (1.10e+04)**+	3.21e+06 (1.19e+07)+	2.40e+08 (3.75e+08)−
F_4	7.43e+02 (1.12e+03)	**0.00e+00 (0.00e+00)**+	**0.00e+00 (0.00e+00)**+	2.20e−01 (6.03e−01)+	4.78e+02 (1.98e+02)≈
F_5	**0.00e+00 (0.00e+00)**	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈	**0.00e+00 (0.00e+00)**≈
F_6	1.71e+01 (1.95e+01)	2.67e+00 (7.92e+00)+	**6.94e−04 (2.01e−03)**+	9.35e+00 (2.06e+00)+	2.99e+01 (1.78e+01)−
F_7	3.74e+01 (7.86e+00)	3.25e+01 (1.63e+01)≈	4.48e+01 (2.96e+01)≈	**2.18e+01 (1.89e+01)**+	6.39e+01 (3.12e+01)−
F_8	2.09e+01 (4.89e−02)	2.10e+01 (4.85e−02)≈	2.09e+01 (8.19e−02)+	**2.09e+01 (5.24e−02)**≈	2.09e+01 (6.34e−02)≈
F_9	2.35e+01 (4.33e+00)	2.23e+01 (3.61e+00)≈	2.37e+01 (1.95e+00)≈	3.07e+01 (9.41e+00)−	**1.85e+01 (2.72e+00)**+
F_{10}	3.00e−01 (1.48e−01)	1.84e−02 (1.35e−02)+	**8.31e−03 (5.46e−03)**+	6.42e−02 (4.82e−02)+	2.29e−01 (1.33e−01)+
F_{11}	1.55e+01 (4.86e+00)	1.09e+01 (4.23e+00)+	2.54e+01 (6.36e+00)−	**3.08e+00 (4.50e+00)**+	2.36e+01 (8.84e+00)−
F_{12}	3.57e+01 (8.45e+00)	5.72e+01 (1.72e+01)−	7.94e+01 (4.39e+01)−	**3.17e+01 (8.51e+00)**+	5.64e+01 (1.52e+01)−
F_{13}	8.06e+01 (2.32e+01)	1.28e+02 (3.53e+01)−	1.56e+02 (5.42e+01)−	**7.55e+01 (2.65e+01)**≈	1.23e+02 (2.21e+01)−
F_{14}	1.26e+03 (4.01e+02)	**1.33e+02 (1.28e+02)**+	7.92e+02 (2.21e+02)+	3.84e+03 (3.85e+02)−	7.04e+02 (2.40e+02)+
F_{15}	3.34e+03 (6.66e+02)	4.10e+03 (8.55e+02)−	**3.13e+03 (4.57e+02)**≈	4.14e+03 (1.08e+03)−	3.42e+03 (5.21e+02)≈
F_{16}	1.87e+00 (3.60e−01)	1.31e−01 (7.65e−02)+	**1.07e−01 (6.78e−02)**+	2.46e+00 (4.45e−01)−	8.48e−01 (2.23e−01)+
F_{17}	5.22e+01 (5.37e+00)	**3.48e+01 (1.54e+00)**+	5.50e+01 (5.24e+00)−	5.92e+01 (5.56e+00)−	5.26e+01 (7.18e+00)≈
F_{18}	**5.25e+01 (8.29e+00)**	8.33e+01 (2.08e+01)−	1.89e+02 (2.73e+01)−	6.04e+01 (9.80e+00)−	6.81e+01 (9.78e+00)−
F_{19}	2.78e+00 (7.47e−01)	2.55e+00 (5.23e−01)≈	2.80e+00 (6.42e−01)≈	**2.30e+00 (6.24e−01)**+	3.12e+00 (9.93e−01)−
F_{20}	**9.86e+00 (6.76e−01)**	1.05e+01 (8.15e−01)−	1.43e+01 (5.75e−01)−	1.26e+01 (7.40e−01)−	1.20e+01 (9.36e−01)−
F_{21}	3.28e+02 (7.87e+01)	3.27e+02 (8.73e+01)+	**1.86e+02 (4.01e+01)**+	2.67e+02 (6.58e+01)+	3.11e+02 (8.00e+01)+
F_{22}	1.30e+03 (4.06e+02)	**1.79e+02 (4.54e+01)**+	1.17e+03 (2.93e+02)≈	4.56e+03 (6.08e+02)−	8.59e+02 (3.13e+02)+
F_{23}	**3.48e+03 (6.40e+02)**	4.22e+03 (8.83e+02)−	4.03e+03 (5.43e+02)−	4.18e+03 (9.21e+02)−	3.57e+03 (5.96e+02)≈
F_{24}	**2.27e+02 (4.44e+00)**	2.32e+02 (2.60e+01)−	2.59e+02 (1.76e+01)−	2.92e+02 (1.90e+01)−	2.48e+02 (8.20e+00)−
F_{25}	2.78e+02 (8.97e+00)	2.78e+02 (1.00e+01)≈	2.82e+02 (8.50e+00)≈	2.99e+02 (6.86e+00)−	**2.49e+02 (7.89e+00)**+
F_{26}	2.00e+02 (2.11e−03)	2.15e+02 (5.30e+01)−	**1.97e+02 (1.21e+01)**≈	3.28e+02 (5.47e+01)−	2.95e+02 (7.13e+01)−
F_{27}	**6.38e+02 (8.02e+01)**	6.47e+02 (1.39e+02)≈	7.49e+02 (1.87e+02)−	1.19e+03 (1.86e+02)−	7.76e+02 (7.18e+01)−
F_{28}	3.00e+02 (3.27e−13)	3.88e+02 (3.27e+02)−	5.39e+02 (1.33e+03)−	**3.00e+02 (0.00e+00)**+	4.01e+02 (3.51e+02)−
	+	11	9	10	7
	−	9	10	14	14
	≈	8	9	4	7

6 Conclusion

The paper proposed a new method named SOMA Pareto, in which the Pareto Principle was applied to the Migration process, overcoming the disadvantages of the original versions, limiting meaningless moves. Besides, the adaptive PRT, $Step$ and $PRTVector$ parameters were introduced, varying with each migration loop, significantly increasing the performance of the SOMA algorithm compared to previous existing versions. Not only that, the results obtained from SOMA Pareto are competitive when compared with well-known algorithms on the CEC'13, CEC'15, and CEC'17 benchmark suites, proving the effectiveness of the proposed algorithm.

Table 3: Comparison of SOMA Pareto with state-of-the-art algorithms on the CEC'15 benchmark functions (30 dimensions, 51 runs).

F	SOMA Pareto Mean (Std Dev)	TEBO Mean (Std Dev)	ICMLSP Mean (Std Dev)	SaDPSO Mean (Std Dev)	dynFWACM Mean (Std Dev)
F_1	1.11e+04 (8.30e+03)	3.69e+02 (7.71e+02)+	**0.00e+00 (0.00e+00)**+	1.93e−02 (8.42e−02)+	6.17e+05 (2.49e+05)−
F_2	3.14e−01 (1.10e+00)	**4.54e−07 (3.02e−06)**+	4.05e−05 (1.44e−04)+	2.88e+02 (1.09e+03)−	3.31e+03 (3.59e+03)−
F_3	2.08e+01 (9.95e−02)	2.00e+01 (6.32e−02)+	2.00e+01 (7.57e−03)+	2.00e+01 (4.15e−05)+	**2.00e+01 (5.75e−06)**+
F_4	**2.95e+01 (6.53e+00)**	4.41e+01 (1.06e+01)−	2.31e+02 (5.66e+01)−	4.25e+01 (9.31e+00)−	1.30e+02 (3.80e+01)−
F_5	2.59e+03 (5.71e+02)	**1.96e+03 (6.32e+02)**+	4.03e+03 (6.56e+02)−	2.52e+03 (3.58e+02)≈	3.38e+03 (6.98e+02)−
F_6	5.47e+03 (5.55e+03)	**6.98e+02 (6.52e+02)**+	1.47e+03 (4.08e+02)+	1.38e+03 (6.04e+02)+	2.69e+04 (1.90e+04)−
F_7	**4.00e+00 (9.00e−01)**	4.42e+00 (1.41e+00)≈	2.07e+01 (1.45e+01)−	9.52e+00 (1.93e+00)−	1.46e+01 (2.57e+00)−
F_8	5.71e+03 (5.02e+03)	**1.21e+02 (1.48e+02)**+	9.42e+02 (2.50e+02)+	1.62e+03 (1.35e+03)+	2.40e+04 (1.32e+04)−
F_9	**1.03e+02 (1.61e−01)**	1.08e+02 (1.22e+00)−	1.63e+02 (1.32e+02)−	1.03e+02 (1.86e−01)−	1.08e+02 (9.01e−01)−
F_{10}	4.55e+03 (4.50e+03)	**6.21e+02 (9.31e+01)**+	1.43e+03 (3.36e+02)+	6.52e+03 (4.66e+03)−	3.15e+04 (2.01e+04)−
F_{11}	**3.14e+02 (5.70e+01)**	4.81e+02 (1.95e+02)−	1.12e+03 (2.72e+02)−	3.20e+02 (8.88e+00)−	6.72e+02 (1.54e+02)−
F_{12}	**1.04e+02 (4.13e−01)**	1.06e+02 (1.04e+00)−	1.61e+02 (4.11e+01)−	1.05e+02 (4.90e−01)−	1.17e+02 (1.23e+01)−
F_{13}	1.03e+02 (5.72e+00)	9.87e+01 (5.46e+00)+	8.53e−02 (1.26e−01)+	1.01e+02 (4.06e+00)≈	**2.62e−02 (7.46e−03)**+
F_{14}	3.27e+04 (4.54e+02)	3.45e+04 (4.04e+03)−	4.21e+04 (4.67e+03)−	**1.87e+04 (5.27e+03)**+	4.49e+04 (1.02e+03)−
F_{15}	1.00e+02 (2.59e−13)	**1.00e+02 (0.00e+00)**+	1.27e+02 (1.62e+01)−	1.00e+02 (1.19e−13)+	**1.00e+02 (0.00e+00)**+
	+	9	7	6	3
	−	5	8	7	12
	≈	1	0	2	0

Table 4: Comparison of SOMA Pareto with state-of-the-art algorithms on the CEC'17 benchmark functions (30 dimensions, 51 runs).

F	SOMA Pareto Mean (Std Dev)	RB-IPOP-CMA-ES Mean (Std Dev)	PPSO Mean (Std Dev)	TLBO-FL Mean (Std Dev)	SAMPE-Jaya Mean (Std Dev)
F_1	**2.49e−08 (1.48e−07)**	3.15e−08 (2.75e−08)−	7.48e+02 (6.06e+02)−	3.51e+03 (3.61e+03)−	9.95e+09 (5.01e+09)−
F_2	1.67e+03 (9.51e+03)	**0.00e+00 (0.00e+00)**+	5.32e+01 (6.91e+01)+	8.52e+16 (5.81e+17)−	6.87e+35 (3.43e+36)−
F_3	3.81e−05 (1.45e−04)	**0.00e+00 (0.00e+00)**+	1.13e+00 (4.83e−01)−	2.99e+03 (1.08e+03)−	5.09e+04 (1.35e+04)−
F_4	**3.77e+00 (9.58e+00)**	5.53e+01 (1.65e+01)−	4.39e+01 (3.19e+01)−	9.01e+01 (2.37e+01)−	1.01e+03 (5.95e+02)−
F_5	**3.77e+00 (9.58e+00)**	5.53e+01 (1.65e+01)−	4.39e+01 (3.19e+01)−	9.01e+01 (2.37e+01)−	1.01e+03 (5.95e+02)−
F_6	1.41e−03 (1.83e−03)	**1.21e−07 (3.97e−08)**+	2.03e+01 (4.15e+00)−	4.87e−01 (4.24e−01)−	3.59e+01 (6.23e+00)−
F_7	5.13e+01 (7.60e+00)	**3.43e+01 (1.28e+00)**+	1.35e+02 (1.63e+01)−	1.39e+02 (4.75e+01)−	3.44e+02 (1.33e+02)−
F_8	2.85e+01 (7.89e+00)	**1.76e+00 (1.65e+00)**+	8.10e+01 (1.04e+01)−	3.67e+01 (1.84e+01)−	2.17e+02 (2.59e+01)−
F_9	6.37e+00 (5.03e+00)	**0.00e+00 (0.00e+00)**+	1.36e+03 (2.82e+02)−	3.45e+01 (2.71e+01)−	1.01e+02 (7.18e−14)−
F_{10}	2.84e+03 (5.85e+02)	1.44e+03 (5.83e+02)+	3.13e+03 (3.46e+02)−	6.69e+03 (2.77e+02)−	**6.42e−01 (0.00e+00)**+
F_{11}	**2.80e+01 (1.94e+01)**	4.11e+01 (4.76e+01)≈	8.43e+01 (1.84e+01)−	8.16e+01 (4.14e+01)−	**1.43e+02 (1.51e+02)**−
F_{12}	1.13e+04 (5.68e+03)	1.09e+03 (2.81e+02)+	2.77e+04 (8.55e+03)−	5.75e+04 (8.99e+04)−	**5.56e+02 (6.89e−13)**+
F_{13}	**7.26e+01 (5.11e+01)**	1.19e+02 (4.00e+02)−	3.21e+03 (2.88e+03)−	2.02e+04 (1.79e+04)−	4.97e+07 (4.44e+07)−
F_{14}	1.21e+02 (3.14e+02)	**9.08e+01 (5.62e+01)**≈	2.32e+03 (1.52e+03)−	7.10e+03 (5.85e+03)−	5.57e+04 (5.92e+04)−
F_{15}	**1.49e+02 (3.22e+02)**	2.18e+02 (1.84e+02)≈	2.13e+03 (1.63e+03)−	2.16e+04 (2.27e+04)−	3.83e+05 (9.94e+04)−
F_{16}	6.93e+02 (2.43e+02)	5.02e+02 (2.54e+02)+	8.46e+02 (1.53e+02)−	**4.92e+02 (3.53e+02)**+	1.23e+03 (2.15e+02)−
F_{17}	**8.78e+01 (8.76e+01)**	1.32e+02 (9.54e+01)−	3.31e+02 (1.13e+02)−	1.41e+02 (6.59e+01)−	2.66e+02 (6.98e+01)−
F_{18}	1.15e+04 (6.72e+03)	**1.60e+02 (1.14e+02)**+	6.99e+04 (3.06e+04)−	3.67e+05 (1.67e+05)−	8.26e+05 (9.54e+05)−
F_{19}	**4.29e+01 (4.28e+01)**	1.15e+02 (6.59e+01)−	1.71e+03 (1.69e+03)−	1.07e+04 (1.10e+04)−	2.22e+07 (6.23e+07)−
F_{20}	**1.75e+02 (8.22e+01)**	2.97e+02 (1.19e+02)−	3.48e+02 (9.16e+01)−	2.21e+02 (1.25e+02)−	3.40e+02 (1.38e+02)−
F_{21}	2.28e+02 (8.31e+00)	2.09e+02 (1.67e+01)+	3.05e+02 (3.30e+01)−	2.34e+02 (1.16e+01)−	**1.70e+02 (5.74e−14)**+
F_{22}	1.45e+02 (3.21e+02)	6.72e+02 (7.63e+02)−	**1.00e+02 (5.05e−07)**+	1.01e+02 (1.94e+00)+	8.78e+02 (6.51e+02)−
F_{23}	3.82e+02 (8.72e+00)	**3.39e+02 (4.93e+01)**+	6.81e+02 (3.79e+01)−	3.96e+02 (1.62e+01)−	6.19e+02 (3.88e+01)−
F_{24}	4.52e+02 (7.26e+00)	**4.19e+02 (3.06e+00)**+	7.39e+02 (4.57e+01)−	4.69e+02 (1.62e+01)−	6.95e+02 (3.88e+01)−
F_{25}	3.88e+02 (3.32e+00)	**3.87e+02 (1.47e−02)**+	3.85e+02 (1.77e+00)+	4.02e+02 (1.76e+01)−	6.17e+02 (1.76e+02)−
F_{26}	1.41e+03 (2.01e+02)	**3.94e+02 (2.17e+02)**+	2.04e+03 (1.73e+03)≈	1.42e+03 (4.69e+02)≈	4.87e+02 (6.31e−13)+
F_{27}	5.34e+02 (6.75e+00)	5.12e+02 (1.13e+01)+	7.08e+02 (5.42e+01)−	5.32e+02 (2.07e+01)≈	**3.87e+02 (4.59e−13)**+
F_{28}	3.04e+02 (2.05e+01)	3.09e+02 (2.96e+01)−	3.27e+02 (3.17e+01)−	4.30e+02 (2.67e+01)−	**2.87e+02 (4.02e−13)**+
F_{29}	5.20e+02 (9.75e+01)	4.93e+02 (1.04e+02)+	7.80e+02 (1.21e+02)−	6.15e+02 (9.09e+01)−	**1.87e+02 (1.15e−13)**+
F_{30}	3.22e+03 (2.87e+02)	2.84e+03 (1.94e+03)+	3.32e+03 (3.89e+02)≈	2.57e+04 (2.78e+04)−	**8.72e+01 (0.00e+00)**+
	+	19	3	2	8
	−	8	25	26	22
	≈	3	2	2	0

Acknowledgement: The following grants are acknowledged for the financial support provided for this research: Grant of SGS No. SP2019/137, VSB-Technical University of Ostrava.

References

[1] Agrawal, S. and Singh, D. 2017. Modified Nelder-Mead self organizing migrating algorithm for function optimization and its application. *Applied Soft Computing* 51, pp. 341–350.

[2] Awad, N. H., Ali, M. Z., Liang, J. J., Qu, B. Y., and Suganthan, P. N. 2016. *Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization.* Technical Report, Nanyang Technological University, Singapore.

[3] Bao, D. Q. and Zelinka, I. 2019. Obstacle Avoidance for Swarm Robot Based on Self-Organizing Migrating Algorithm. *Procedia Computer Science* 150, pp. 425–432.

[4] Biedrzycki, R. 2017. A version of IPOP-CMA-ES algorithm with midpoint for CEC 2017 single objective bound constrained problems. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 1489–1494.

[5] Caraffini, F., Iacca, G., Neri, F., Picinali, L., and Mininno, E. 2013. A CMA-ES super-fit scheme for the re-sampled inheritance search. In *2013 IEEE Congress on Evolutionary Computation*. IEEE, pp. 1123–1130.

[6] Caraffini, F., Neri, F., Cheng, J., Zhang, G., Picinali, L., Iacca, G., and Mininno, E. 2013. Super-fit multicriteria adaptive differential evolution. In *2013 IEEE Congress on Evolutionary Computation*. IEEE, pp. 1678–1685.

[7] Chen, L., Peng, C., Liu, H.-L., and Xie, S. 2015. An improved covariance matrix leaning and searching preference algorithm for solving CEC 2015 benchmark problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 1041–1045.

[8] Davendra, D. and Zelinka, I. 2016. *Self-organizing migrating algorithm*. New Optimization Techniques in Engineering, Studies in Computational Intelligence, Springer.

[9] Deep, K. and Dipti, G. 2008. A self-organizing migrating genetic algorithm for constrained optimization. *Applied Mathematics and Computation* 198, 1, pp. 237–250.

[10] Deep, K. et al. 2007. A new hybrid self organizing migrating genetic algorithm for function optimization. In *IEEE Congress on Evolutionary Computation, 2007. CEC 2007.* IEEE, pp. 2796–2803.

[11] Del Ser, J. et al. 2019. Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation* 48, pp. 220–250.

[12] Derrac, J., García, S., Molina, D., and Herrera, F. 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation* 1, 1, pp. 3–18.

[13] Diep, Q. B. 2019. Self-Organizing Migrating Algorithm Team To Team Adaptive – SOMA T3A. In *The 2019 IEEE Congress on Evolutionary Computation, Wellington, New Zealand*. IEEE, In Press.

[14] Diep, Q. B., Zelinka, I., and Das, S. 2019. Self-Organizing Migrating Algorithm for the 100-Digit Challenge. In *Proceedings of the Genetic and Evolutionary Computation Conference 2019 (GECCO '19)*. ACM, New York, NY, USA.

[15] Diep, Q. B., Zelinka, I., and Senkerik, R. 2019. An algorithm for swarm robot to avoid multiple dynamic obstacles and to catch the moving target. In *International Conference on Artificial Intelligence and Soft Computing*. Springer, pp. 666–675.

[16] Dorigo, M. and Birattari, M. 2010. *Ant colony optimization*. Springer.

[17] Dorigo, M., Maniezzo, V., Colorni, A. et al. 1996. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, man, and cybernetics, Part B: Cybernetics* 26, 1, pp. 29–41.

[18] Elsayed, S. M., Sarker, R. A., and Ray, T. 2013. Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization. In *2013 IEEE Congress on Evolutionary Computation*. IEEE, pp. 1932–1937.

[19] Kennedy, J. 2010. Particle swarm optimization. In *Encyclopedia of machine learning*. Springer, pp. 760–766.

[20] Kommadath, R. and Kotecha, P. 2017. Teaching learning based optimization with focused learning and its performance on CEC2017 functions. In *2017 IEEE congress on evolutionary computation (CEC)*. IEEE, pp. 2397–2403.

[21] Liang, J. J., Guo, L., Liu, R., and Qu, B. Y. 2015. A self-adaptive dynamic particle swarm optimizer. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 3206–3213.

[22] Liang, J. J., Qu, B. Y., Suganthan, P. N., and Chen, Q. 2014. *Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization.* Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore.

[23] Liang, J. J., Qu, B. Y., Suganthan, P. N., and Hernández-Díaz, A. G. 2013. *Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization.* Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report.

[24] Lin, Z. and Wang, L. J. 2014. Hybrid self-organizing migrating algorithm based on estimation of distribution. In *2014 International Conference on Mechatronics, Electronic, Industrial and Control Engineering (MEIC-14)*. Atlantis Press. DOI: 10.2991/meic-14.2014.56

[25] Nepomuceno, F. V. and Engelbrecht, A. P. 2013. A self-adaptive heterogeneous pso for real-parameter optimization. In *2013 IEEE congress on evolutionary computation*. IEEE, pp. 361–368.

[26] Rao, R. V. 2019. *Jaya: an advanced optimization algorithm and its engineering applications.* Springer.

[27] Reh, F. J. 2005. Pareto's principle-The 80-20 rule. *Business Credit* 107, 7, pp. 76.

[28] Coelho, L. S. and Mariani, V. C. 2010. An efficient cultural self-organizing migrating strategy for economic dispatch optimization with valve-point effect. *Energy Conversion and Management* 51, 12, pp. 2580–2587.

[29] Singh, D. and Agrawal, S. 2015. Hybridization of self organizing migrating algorithm with quadratic approximation and non uniform mutation for function optimization. In *Proceedings of Fourth International Conference on Soft Computing for Problem Solving*. Springer, pp. 373–387.

[30] Singh, D. and Agrawal, S. 2016. Self organizing migrating algorithm with quadratic interpolation for solving large scale global optimization problems. *Applied Soft Computing* 38, pp. 1040–1048.

[31] Tangherloni, A., Rundo, L., and Nobile, M. S. 2017. Proactive particles in swarm optimization: A settings-free algorithm for real-parameter single objective optimization problems. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 1940–1947.

[32] Yu, C., Kelley, L. C., and Tan, Y. 2015. Dynamic search fireworks algorithm with covariance mutation for solving the CEC 2015 learning based competition problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 1106–1112.

[33] Zelinka, I. 2004. SOMA – Self-organizing Migrating Algorithm. In *New optimization techniques in engineering*. Springer, pp. 167–217.

[34] Zelinka, I. and Bukacek, M. 2016. SOMA swarm algorithm in computer games. In *International Conference on Artificial Intelligence and Soft Computing*. Springer, pp. 395–406.

[35] Zelinka, I. and Lampinen, J. 2000. SOMA – Self-Organizing Migrating Algorithm Mendel. In *6th International Conference on Soft Computing, Brno, Czech Republic*.

[36] Zelinka, I. and Sikora, L. 2015. StarCraft: Brood War-Strategy powered by the SOMA swarm algorithm. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, pp. 511–516.

[37] Zelinka, I. and Tomaszek, L. 2016. Competition on learning-based real-parameter single objective optimization by SOMA swarm based algorithm with SOMARemove strategy. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 4981–4987.

[38] Zheng, Y.-J. and Wu, X.-B. 2015. Tuning maturity model of ecogeography-based optimization on CEC 2015 single-objective optimization test problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 1018–1024.