Explaining SOMA: the Relation of Stochastic Perturbation to Population Diversity and Parameter Space Coverage

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

Anezka Kazikova
Tomas Bata University in Zlin
Zlin, Czech Republic
kazikova@utb.cz

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

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

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

ABSTRACT

The Self-Organizing Migrating Algorithm (SOMA) is enjoying a renewed interest of the research community, following recent achievements in various application areas and renowned performance competitions. In this paper, we focus on the importance and effect of the perturbation operator in SOMA as the perturbation is one of the fundamental inner principles of SOMA. In this in-depth study, we present data, visualizations, and analysis of the effect of the perturbation on the population, its diversity and average movement patterns. We provide evidence that there is a direct relation between the perturbation intensity (set by control parameter *prt*) and the rate of diversity loss. The perturbation setting further affects the exploratory ability of the algorithm, as is demonstrated here by analysing the parameter space coverage of the population. We aim to provide insight and explanation of the impact of perturbation in SOMA for future researchers and practitioners.

CCS CONCEPTS

• **Computing methodologies** → **Continuous space search**; • **Mathematics of computing** → **Evolutionary algorithms**; • **Theory of computation** → **Bio-inspired optimization**.

KEYWORDS

Self-organizing migrating algorithm,SOMA, perturbation, diversity, parameter space coverage

ACM Reference Format:
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 *2021 Genetic and Evolutionary Computation Conference Companion(GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449726.3463211

GECCO '21, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07…$15.00
https://doi.org/10.1145/3449726.3463211

1 INTRODUCTION

The original self-organizing migrating algorithm (SOMA) was introduced by Zelinka in 2000 [17],[15]. It fits into the category of swarm intelligence [6] based metaheuristic algorithms alongside Particle swarm optimization [8] and Ant colony optimization [5], preceding the modern era boom of swarm algorithms [10] that has attracted criticism from part of the scientific community [13].

SOMA and its modifications have been proven very effective in various applications [9], including complex problems in the areas of discrete [2] and multi-objective [7] optimization attracting the general attention. In 2016 a book has been published [1] that details various applications of SOMA including chaos and complex systems, financial modeling or large-scale optimization.

More recently two SOMA variants, namely SOMA T3A [3] and SOMA PARETO [14] ranked 4th and 6th in the IEEE CEC 100 digit challenge [12] underlining the renaissance of SOMA algorithm.

Despite the renewed popularity of the algorithm and the broad spectrum of successful applications, the inner dynamics of SOMA have rarely been studied in detail. This study aims to fill part of this research gap by focusing on the perturbation operator and its relation to the population diversity, movement patterns, and overall fitness landscape coverage, providing useful insight for future researchers and users of SOMA.

The rest of the paper is structured as follows: In the following section, the SOMA algorithm is introduced. Section 3 details the perturbation operator in SOMA. The methodology for population diversity calculation is described alongside used benchmark functions in section 4. The main experimental part is presented in section 5. The paper concludes with the discussion of the results.

2 SELF-ORGANIZING MIGRATING ALGORITHM (SOMA)

SOMA is a population-based metaheuristic method that utilizes the traditional crossover and mutation operations in a modified manner simulating a social group of individuals.

In the original and most common SOMA variant called All-to-One [17],[15], [1], which is the focus of this study, the algorithm follows these steps: At the start of each iteration (called migration loop; *ML*) a Leader is selected based on the fitness. The fittest individual becomes the leader. Each remaining individual then moves in the direction towards the Leader in the search space. The

movement consists of jumps determined by the *Step* parameter until the individual reaches the final position given by the *PathLength* parameter.

Each step is evaluated using the objective function, and the best position (including the initial position of a given individual) is chosen as the new position of the individual in the next migration loop. The exact position of each step is calculated according to (1).

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

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 $PRTVector_j$ 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" towards the leader position or not. 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 (2), where the rand is a pseudo-random number from a uniform distribution within the range of 0 to 1.

$$PRTVector_j = \begin{cases} if\ rand_j < prt, & 1 \\ otherwise, & 0 \end{cases} \tag{2}$$

3 THE PERTURBATION OPERATOR IN SOMA

A perturbation is an essential operation in the SOMA equation (1). It represents the process of random mutation and is controlled by the predefined *prt* parameter. The value range of *prt* is from 0 to 1. The perturbation vector *PRTVector* is constructed according to the rules described in the previous section. However, in the initial SOMA proposal [17] the *PRTVector* was constructed for each individual at the start of its movement, leading to a linear trajectory of the movement. Illustration of the initial *PRTVector* implementation (in 2D) is given in Fig. 1.

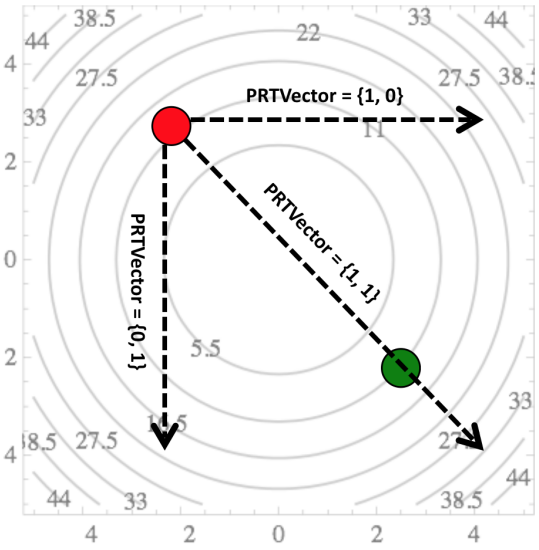


Figure 1: The original principle of perturbation in SOMA, (active individual red, leader green)

Later, the implementation of *PRTVector* has been changed [16]. A new *PRTVector* is constructed before each step of a given individual and further, as a rule, there has to be at least one zero in the vector. This change of implementation significantly altered the movement pattern of the individuals as is depicted in Fig. 2.

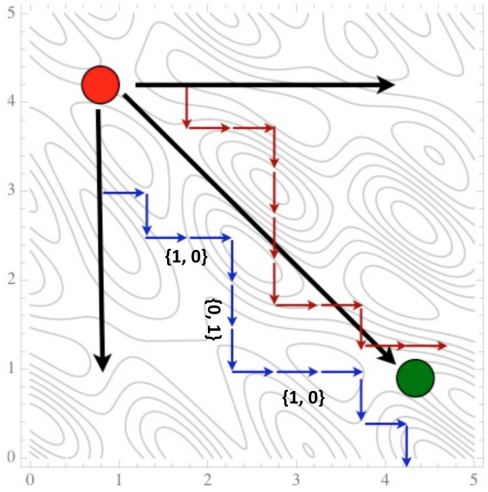


Figure 2: The modified principle of perturbation in SOMA, (active individual red, leader green)

With the updated *PRTVector* construction rule it is no longer possible for the individual to follow a direct linear path towards the Leader, but it is forced to explore the search space in series of steps in different directions while following the general direction vector towards the leader (as exampled by red and blue paths in Fig. 2.). It needs be stressed at this point, that the updated *PRTVector* rule

also significantly alters the impact of the *PathLength* parameter, as the individual trajectory is now less likely to cross the position of the leader and the distance between the leader and the last step of the individual movement trajectory is smaller. In other words, in original SOMA the individual's path would typically end way beyond the leader (for *PathLength = 3*) while in the updated scenario, it is not the case anymore.

The modified *PRTVector* implementation has led to significantly improved performance of SOMA [1]. The suggested default value for *prt* is near 0.1 as described in [15]. However, a higher value is suitable for certain problem types [1].

In Fig. 3 we illustrate an experiment of the area possibly covered by 100 repeated individual movements according to SOMA rules from a fixed start point (blue dot, positioned at {1,1}) towards a leader (red dot, positioned at {5,5}) for *prt* = 0.1. *Step* = 0.11 and *PathLength* = 3 (as suggested by the authors of SOMA [15], [1]). The figure depicts 100 paths in an overlay. Similar experiment for *prt* = 0.8 is illustrated in Fig. 4.

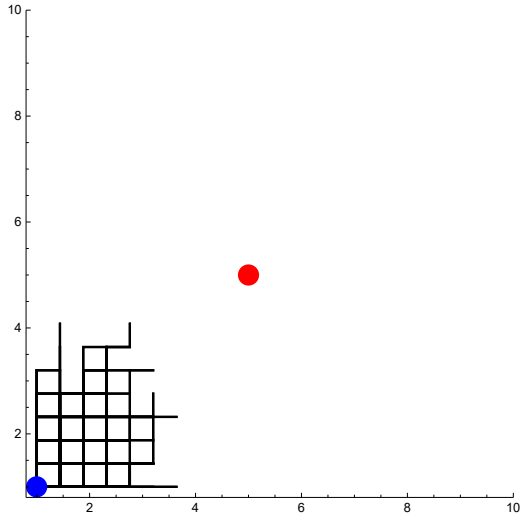


Figure 3: Area coverage simulation in 2D, *prt* = 0.1; 100 runs

The experiment shows that the area the individual can theoretically cover with its movement is significantly larger for the higher value of *prt*. The movement for *prt* = 0.1 is very limited. One of the reasons is that following the "at least one zero in *PRTVector*" condition it is extremely likely that the individual will not move at all in a given step (a zero-filled *PRTVector* is generated). With increasing the dimensionality of the problem, this situation becomes less likely. Therefore, it might be advisable to avoid small *prt* values for low-dimensional problems regardless of the complexity of the search space.

To illustrate this point, we present the same experiment in 3D. Given in Fig. 5 is the situation for *prt* = 0.1 and in Fig. 6 for *prt* = 0.8. With one extra dimension to use, the individuals seem to be missing fewer steps in the *prt* = 0.1 scenario. The area covered for *prt* = 0.8 nicely demonstrates the tendency of the individual to follow the general direction towards the leader.

Following this initial experiment that served mainly to illustrate the movement patterns and area coverage of individuals under

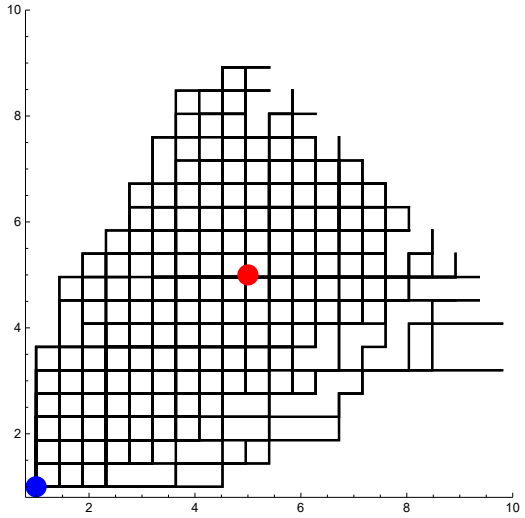


Figure 4: Area coverage simulation in 2D, *prt* = 0.8; 100 runs

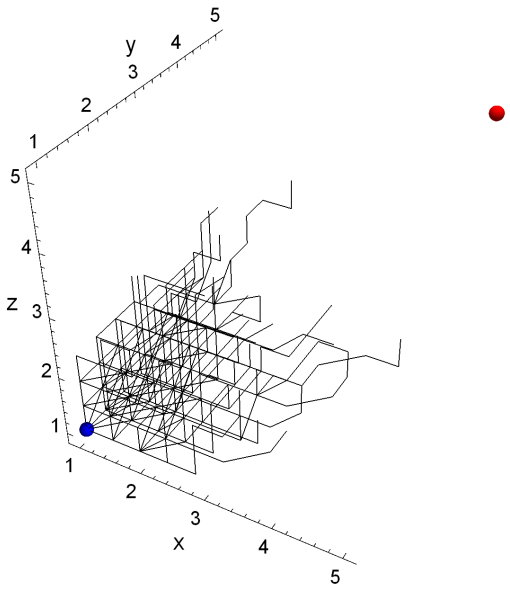


Figure 5: Area coverage simulation in 3D, *prt* = 0.1; 100 runs

significantly different *prt* values, we continue with a more detailed experiment, focusing on the diversity of the population.

4 METHODOLOGY

In order to proceed with further experiments, it is necessary to establish in this section the methodology used in this paper, notably the method of diversity quantification and benchmark functions used for the simulations.

4.1 Diversity measure

We use the measure introduced in [11] to quantify the population diversity. The diversity value is based on the sum of deviations (3)

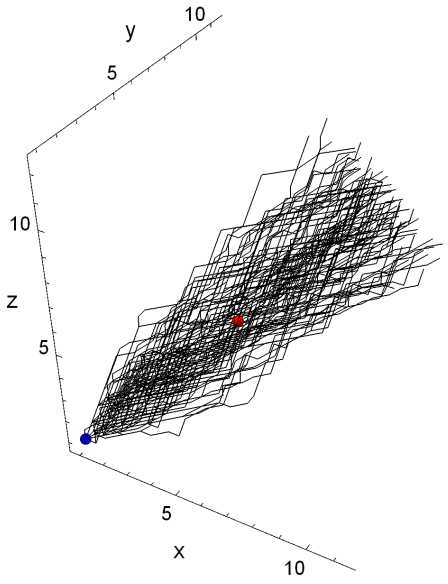


Figure 6: Area coverage simulation in 3D, *prt* = 0.8; 100 runs

of individual's components from their corresponding means (4). For further clarity, in the experimental section, the value is presented as a relative percentage to a theoretical maximal value of the diversity (subject to dimensionality and bounds of the search space).

$$PD = \sqrt{\frac{1}{NP}\sum_{i=1}^{NP}\sum_{j=1}^{D}(x_{i,j} - \overline{x_j})^2} \tag{3}$$

$$\overline{x_j} = \frac{1}{NP}\sum_{i=1}^{NP} x_{i,j} \tag{4}$$

Where i is the population member iterator, and j is the vector component iterator.

4.2 Fitness landscapes

A set of four very well established test functions [4] was used in the following experiments. The Sphere function represents a simple unimodal problem, while Rosenbrock function is a more challenging fitness landscape in a higher dimension and further Rastrigin and Schwefel functions represent more complex highly multi-modal landscapes.

Given that this study is not focused on the performance of the algorithm, using more complex functions (e.g. the IEEE CEC benchmark functions, with rotations, shifts etc.) would be disadvantageous. Using the above described simple functions allows better understanding of the influence of the control parameter change on the population dynamic.

Sphere function:

$$f(x) = \sum_{i=1}^{\dim} x_i^2 \tag{5}$$

Search Range: $[-100,100]^D$; Glob. opt. pos.: $[0]^D$

Rosenbrock function:

$$f(x) = \sum_{i=1}^{\dim-1} 100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2 \tag{6}$$

Search Range: $[-10,10]^D$; Glob. opt. pos.: $[0]^D$

Rastrigin function:

$$f(x) = 10\dim + \sum_{i=1}^{\dim} x_i^2 - 10\cos(2\pi x_i) \tag{7}$$

Search Range: $[-5.12,5.11]^D$;; Glob. opt. pos.: $[0]^D$

Schwefel function:

$$f(x) = \sum_{i=1}^{\dim} -x_i \sin(\sqrt{|x|}) \tag{8}$$

Search Range: $[-512,511]^D$; Glob. opt. pos.: $[420.96]^D$

5 THE EXPERIMENTS

One of the key aspects of modern swarm algorithm designs is the ability to maintain the population diverse and therefore keep the ability to create new (different) solutions by the crossover or similar operations. Many of the swarm intelligence heuristic optimizers suffer from fast premature convergence into sub-optima. The speed of diversity loss is, therefore, an essential characteristic of such methods. One of the main purposes of perturbation operator in SOMA is to keep the population diverse, that is at least in the initial phases of the search. We investigated the impact of different *prt* values on the diversity of the population.

5.1 Diversity

In the following experiment, we optimized the four above described benchmark functions, and observed the diversity of the population for different *prt* values, starting at 0.1 and ending at 1 by step 0.1. For each setting, 50 independent runs with random population initialization were performed. The experiments were performed for two different dimensionality settings to investigate if the dimensionality affects the population diversity development for different *prt* values.

With accordance to general recommendations [16] and the focus of this study, the algorithm was set up as follows: *Population size:* 30; *Migration loops:* 100; *Step* = 0.11; *Path Length* = 3; *Dimension* = 10, 100;

The average from 50 runs relative population diversity values for different *prt* settings are given in Figs. 7 - 10. Please note that after uniform random initialization, the relative diversity of the population (according to the used measure) is between 55% and 60%.

For the Sphere function, the population diversity loss is very fast. The simple unimodal surface of a spherical function is ideal for fast local-search. The difference in diversity loss for *prt* = 1 and 0.9 is almost indistinguishable. With any other setting than *prt* = 0.1 the population diversity is well under 15% after just 20 iterations of the algorithm for both dimensional settings. This might not be an issue for a unimodal problem. However, it might prove very troublesome in higher dimensions.

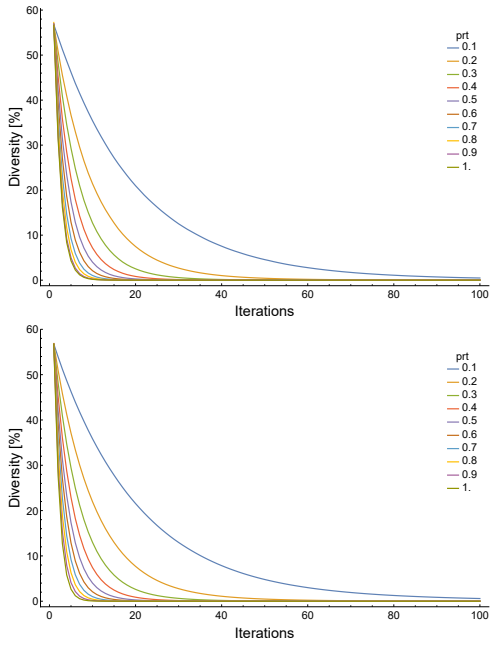


Figure 7: Average population diversity history - Sphere function - 50 runs, dim: 10 upper, dim: 100 lower

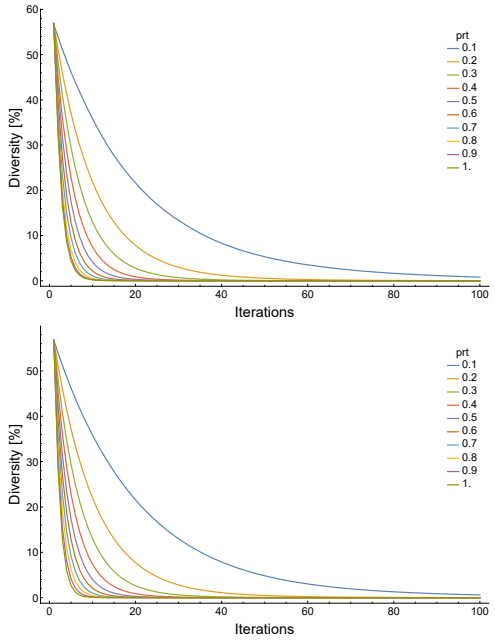


Figure 8: Average population diversity history - Rosenbrock function - 50 runs, dim: 10 upper, dim: 100 lower

The results for Rosenbrock function depict very similar diversity loss. The diversity reaches near-zero values for the majority of the settings during the first third of the provided migration loops. With *prt* <0.5, it takes less than 10 migration loops for the diversity to reach 10%. The dimensionality of the problem seems to have only a limited effect on the diversity loss rate.

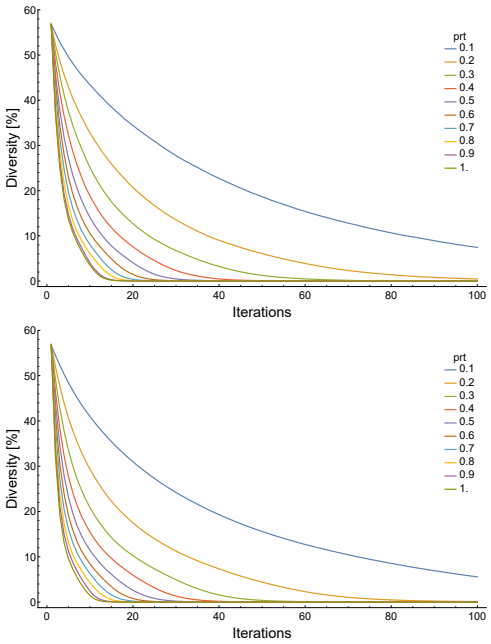


Figure 9: Average population diversity history - Rastrigin function - 50 runs, dim: 10 upper, dim: 100 lower

The Rastrigin function represents a periodical, highly multimodal fitness landscape. The diversity loss is significantly slower than in previous cases. With the *prt* value set to 0.1, the population maintains diversity over or around 10% through the whole optimization. Therefore the fitness landscape is clearly a factor in diversity loss. There is a significant gap between the diversity loss rate for *prt* set to 0.1 and 0.2.

The Schwefel function manages to produce the most significant difference in the speed of the population diversity loss for different values of *prt*. Again, setting the *prt* to 0.1 helps the population to avoid complete loss of diversity in the given time-frame of 100 migration loops. For all other tested settings of *prt* and dimensionality, the diversity eventually drops to near-zero values.

5.2 Dimensional convergence and optima proximity

Another important characteristic of a population-based method is how the individuals avoid a notorious "two steps forward, one step back" problem. The problem represents a phenomenon when the individual moves towards a more feasible solution in some dimensions but moves another way in others.

In this experiment, we illustrate the convergence pattern and optima proximity of the population in 10-dimensional space using the Box and Whiskers plot that depicts the statistical characteristics of the population parameter pool (median, maximum, minimum, 25% and 75% quantiles), the y-axis value is given by the corresponding parameter value (in the individual position vector). The aim is to observe the population dynamics in multi-dimensional space.

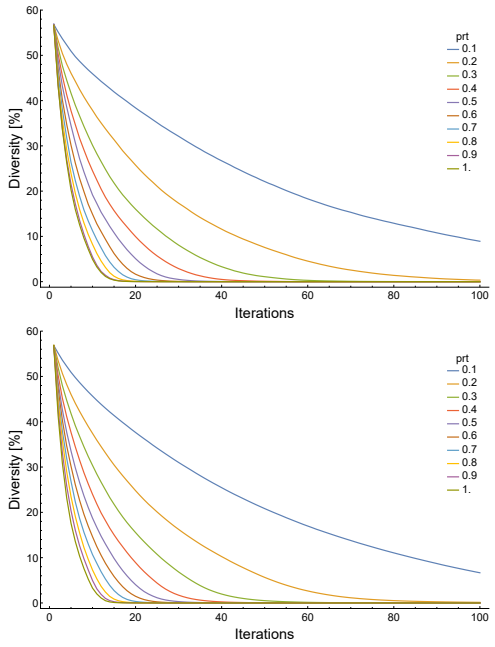


Figure 10: Average population diversity history - Schwefel function - 50 runs, dim: 10 upper, dim: 100 lower

The statistic is based on 50 repeated runs with *pop. size* = 30 that equals 1500 data samples for each plot. Each data sample is a vector representing the ten parameters of the individual. We observe the population at the start, after 10, 20, 50 and 100 migration loops. As an example, *prt* is set to 0.1 and 0.8 similarly to the initial experiment. The Rastrigin function is used as the fitness landscape for this experiment; the global optima location is a vector of zeros.

At the start (migration loop 1), the population is evenly spread over the search space as is depicted in Fig. 11a), but converges rapidly after just ten iterations (See Fig. 11b)). The majority of the population parameter pool (possible parameter values for crossover) is between -1 and 1.

In an ideal (and simplified) case, the population parameters will converge evenly in all dimensions toward the zero (position of optima), however, given the stochasticity of the search process, irregularities start to develop in the parameter pool across dimensions, despite that the data are acquired from 50 independent runs of the algorithm. As is depicted in Fig. 11c), and more notably in Fig. 11d), the convergence in some dimensions is faster than in others. More importantly, for the *prt* value set to 0.1, the convergence continues during the majority of the optimization process. In Fig. 11e), the state at the last migration loop is visualized.

In contrast, after 10 migration loops, when *prt* is set to 0.8 the population diversity pool (initialized randomly (Fig. 12a))) converges rapidly but less evenly than in the previous case. There is no further development in the parameter pool and therefore the Fig. 12b) depicts the state after 10, 20, 50 and 100 migration loops. The explanation is that in all 50 runs the algorithm converged into local sub-optima before 20 migration loops were finished.

In the above presented experiment, we observed that higher *prt* value (exampled on *prt = 0.8*) leads to very fast convergence into sub-optima in all dimensions of the problem. As the dimensions are independent of each other, there is different distribution of population parameters in each dimension. Despite that in the case of *prt = 0.1* the convergence is much slower and more even amongst the dimensions at first, in the final stage the same disproportion among the dimensions is observable. Therefore it seems that the *prt* value has no effect on this phenomenon (as is it caused mainly by the stochastic nature of the optimization process).

5.3 Parameter space coverage

Finally, to provide another insight into the population dynamics, we present a Parameter space coverage visualization example in this section. Selecting a random run of the algorithm on Rastrigin function (search space bounds -5, +5) in *dim*=10, each individual in Fig. 13 is visualized as a colored line connecting 10 parameter values, corresponding to the position of the individual in the search space. We further highlight the parameter space coverage borders of the population (min, max parameter value in each dimension in the whole population) by black lines.

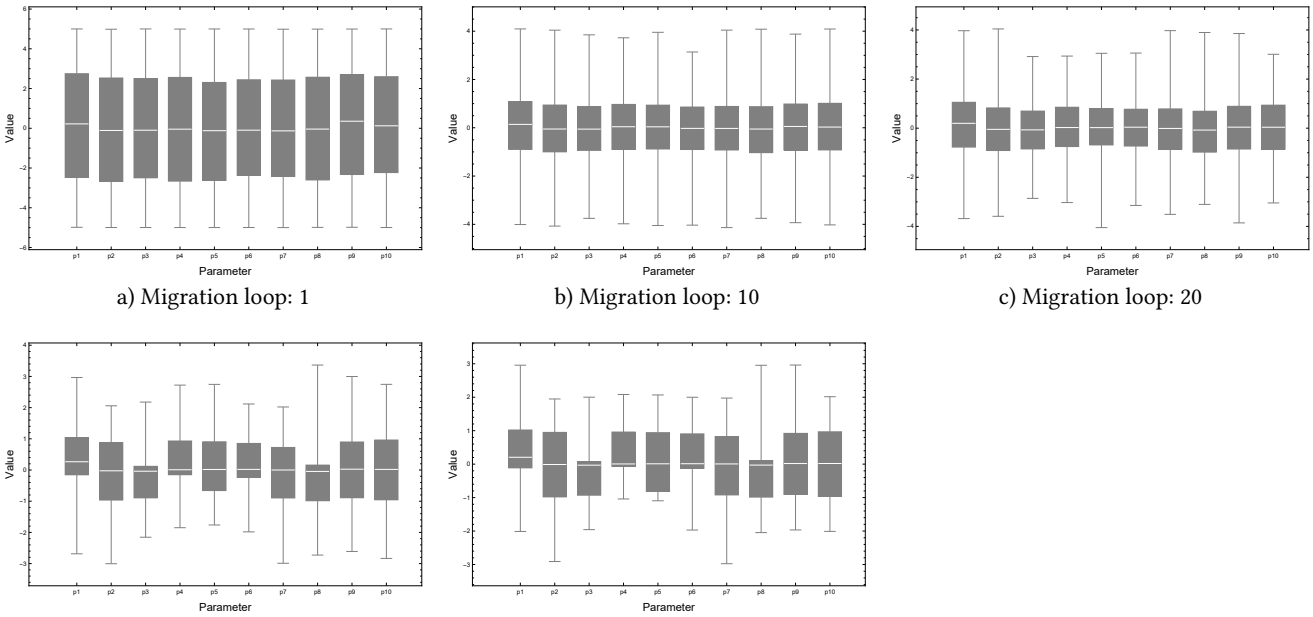
To maximize the successfully crossover potential, the population should be able to cover the whole area between the function bound values (-5, +5). In Fig 13a) the state of the population after initialization for *prt* = 0.1 is depicted.

The parameters of the individuals are randomly distributed. In fast converging algorithms like SOMA All-to-One, individuals only very rarely leave the area between the black lines. As presented in Fig 13b) - e), with the convergence of the population, the area covered by the population is shrinking rapidly. The process is much faster for *prt* = 0.8 (Fig. 14).

The added benefit of this type of visualization is that it allows us to observe all individuals and their changes in time. Generally, if all the lines follow the same general shape, the crossover operator is dominant, and in the case of SOMA algorithm, the Leader shapes the whole population (fast convergence). If there are different shapes of the lines, there is more stochasticity (successful mutation) in the population. In the above presented examples, all individuals converge fast into the same general shape and the population is unable to produce new successful offspring given the low diversity of the population parameter pool and falls into stagnation (premature convergence). The speed of the premature convergence varies by the *prt* setting, however neither setting is able to avoid it, providing another piece of evidence that static setting of *prt* should be avoided.

6 CONCLUSION

We have presented a detailed study on how the perturbation operator works in SOMA and how the setting of *prt* control parameter relates to population diversity, movement patterns, and convergence in multi-dimensional space. The perturbation operator simulates the random mutation, and it is the main stochastic element in SOMA. As was presented, the population dynamic of SOMA is significantly sensitive to the setting of its control parameter *prt*. The lower *prt* value slows the diversity loss, prolongs the convergence phase and limits the individuals' movement. The area explored by



a) Migration loop: 1 b) Migration loop: 10 c) Migration loop: 20

d) Migration loop: 50 e) Migration loop: 100

Figure 11: Population parameter pool; *prt*: 0.1; 1500 samples

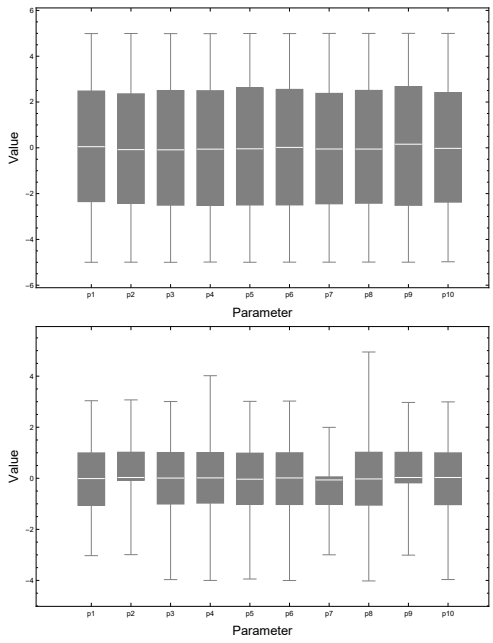


Figure 12: Population parameter pool; *prt*: 0.8; 1500 samples, Migration loop: 1 (upper), Migration loop: 10, 20, 50 and 100 (lower)

the individuals is also smaller when compared to higher *prt* values in the same migration loops limit.

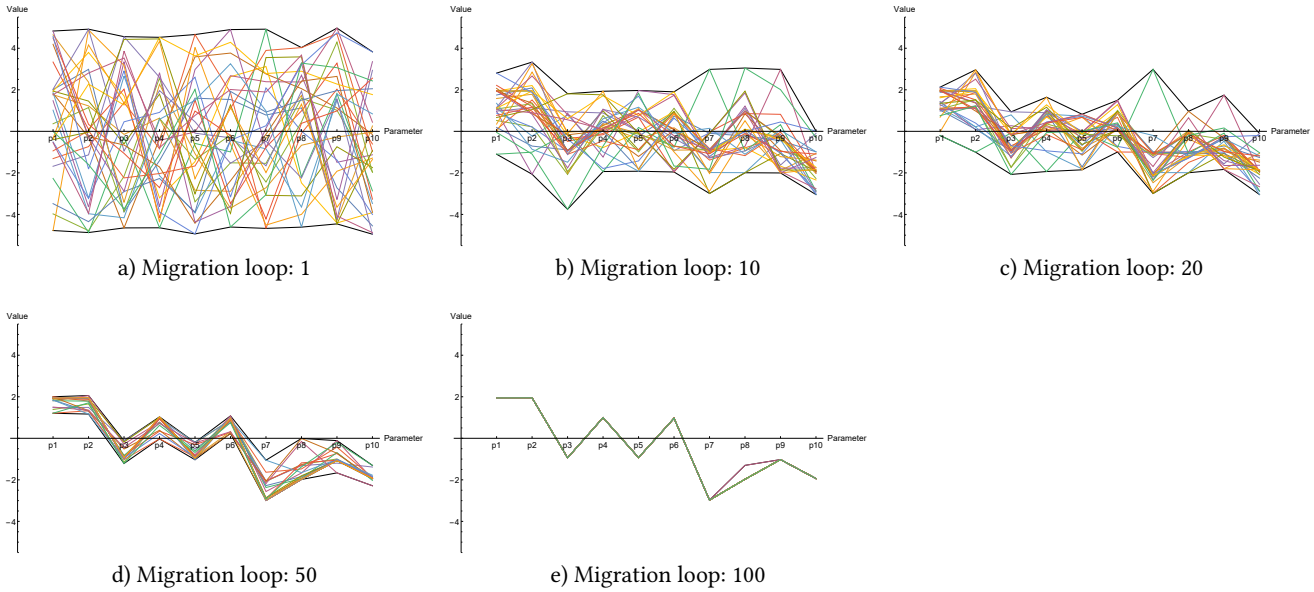
As the inner dynamic of SOMA is completely self-adaptive, the behavior patterns associated with particular *prt* value scale similarly regardless of the problem dimensionality, or the search space size.

Even though the results are affected by various factors, the modality of the fitness landscape included, the general behavior characteristics like the diversity loss ratios are consistent.

Further, the convergence in higher-dimensional space is much more consistent for small *prt* value (0.1). On the other hand, for low-dimensional problems, a low *prt* value presents a significant obstacle for the movement of the individuals, and such a setting should be well re-considered.

Based on the acquired knowledge, several points and recommendations for future research are listed below:

(1) Low values of *prt* lead to better population dispersion on the fitness landscape in later phases of the optimization. However, it does limit the movement significantly for lower dimensional problems. For low dimensions, settings of *prt* near 0.1 should be avoided in favor of higher values.
(2) The perturbation is an essential factor for maintaining the diversity of the population, yet the static setting of *prt* might not be favorable for general black-box problems. An adaptive or ensemble mechanism for *prt* is a favorable possibility and subject of future research.
(3) In SOMA, maintaining higher diversity does not guarantee a convergence into global optima but merely helps the population to avoid local sub-optima for a longer time and prolongs the active search time. Another modification is probably necessary to improve the final optima proximity of the population.



a) Migration loop: 1 b) Migration loop: 10 c) Migration loop: 20

d) Migration loop: 50 e) Migration loop: 100

Figure 13: Parameter space coverage visualization; *prt*: 0.1; 30 individuals

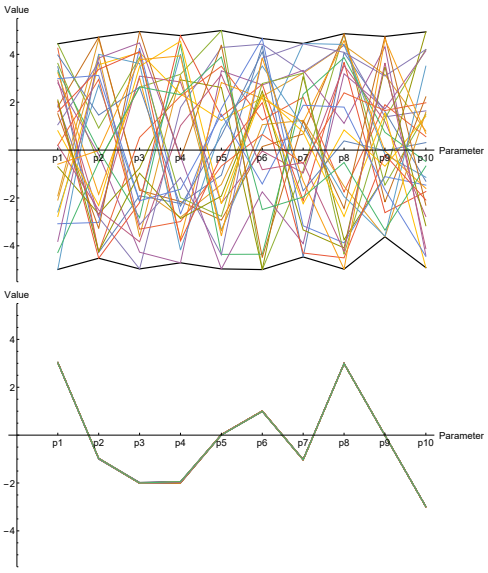


Figure 14: Parameter space coverage visualization; *prt*: 0.8; 30 individuals, Migration loop: 1 (upper), Migration loop: 10, 20, 50 and 100 (lower)

(4) The original All-to-One variant of SOMA is subject to very fast population convergence, that can only by a limited amount be slowed down by the *prt* setting.

This study aimed to provide an insight into the perturbation operator in SOMA and to illustrate the main issues of various settings of the *prt* control parameter. We believe that the provided evidence and visualization will prove useful for researchers working with the algorithm or with population-based methods and swarm algorithms in general. In the future, we will aim to perform similar studies on other SOMA variants and present the results to the scientific community and SOMA users.

ACKNOWLEDGMENTS

This work was supported by the Internal Grant Agency of Tomas Bata University under the Projects no. IGA/CebiaTech/2021/001, and further by the 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] Donald Davendra, Ivan Zelinka, Magdalena Bialic-Davendra, Roman Senkerik, and Roman Jasek. 2013. Discrete Self-Organising Migrating Algorithm for flow-shop scheduling with no-wait makespan. *Mathematical and Computer Modelling* 57, 1 (Jan. 2013), 100–110. https://doi.org/10.1016/j.mcm.2011.05.029

[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] Johannes M. Dieterich and Bernd Hartke. 2012. Empirical review of standard benchmark functions using evolutionary global optimization. *arXiv preprint arXiv:1207.4318* (2012).

[5] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. 2006. Ant colony optimization. *IEEE computational intelligence magazine* 1, 4 (2006), 28–39. ISBN: 1556-603X Publisher: IEEE.

[6] Russell C. Eberhart, Yuhui Shi, and James Kennedy. 2001. *Swarm intelligence*. Elsevier.

[7] Petr Kadlec and Zbyněk Raida. 2011. A Novel Multi-Objective Self-Organizing Migrating Algorithm. *Radioengineering* 20, 4 (2011). ISBN: 1210-2512.

[8] James Kennedy and Russell Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, Vol. 4. IEEE, 1942–1948.

[9] L. Nolle, I. Zelinka, A. A. Hopgood, and A. Goodyear. 2005. Comparison of an self-organizing migration algorithm with simulated annealing and differential evolution for automated waveform tuning. *Advances in Engineering Software* 36, 10 (Oct. 2005), 645–653. https://doi.org/10.1016/j.advengsoft.2005.03.012

[10] Rafael S. Parpinelli and Heitor S. Lopes. 2011. New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation* 3, 1 (2011), 1–16. ISBN: 1758-0366 Publisher: Inderscience Publishers.

[11] R. Poláková, J. Tvrdík, P. Bujok, and R. Matoušek. 2016. Population-size adaptation through diversity-control mechanism for differential evolution. In *MENDEL, 22th International Conference on Soft Computing*. 49–56.

[12] K. Price, N. H. Awad, M. Z. Ali, and P. Suganthan. 2019. *The 2019 100-Digit Challenge on Real-Parameter, Single Objective Optimization: Analysis of Results*. Technical Report. Technical Report 2019.

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

[14] Thanh Cong Truong, Quoc Bao Diep, Ivan Zelinka, and Roman Senkerik. 2020. Pareto-Based Self-organizing Migrating Algorithm Solving 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, 13–20.

[15] Ivan Zelinka. 2004. SOMA—self-organizing migrating algorithm. In *New optimization techniques in engineering*. Springer, 167–217.

[16] Ivan Zelinka. 2016. SOMA—self-organizing migrating algorithm. In *Self-Organizing Migrating Algorithm*. Springer, 3–49.

[17] Ivan Zelinka and Jouni Lampinen. 2000. SOMA – Self–Organizing Migrating Algorithm. In *Proceedings of the 6th International Conference on Soft Computing Mendel 2000*. Brno.