BULGARIAN ACADEMY OF SCIENCES

PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS • 2020 • Vol. **74**, pp. 24-32
p-ISSN: 2738-7356; e-ISSN: ISSN: 2738-7364
https://doi.org/10.7546/PECR.74.20.03

Local Search, Brute-Force and Recursion for Selection Operator

Petar Tomov, Iliyan Zankinski, and Victor Danev

Institute of Information and Communication Technologies - Bulgarian Academy of Sciences
Acad. Georgi Bonchev Str., block 2, office 514, 1113 Sofia, Bulgaria
Emails: p.tomov@iit.bas.bg, iliyan@hsi.iccs.bas.bg, victor_danev@abv.bg

Abstract: Genetic algorithms are heuristics inspired by the processes in the biological evolution. They have a main application in the field of global optimization. The optimization process is organized into three common operations – crossover, mutation, and selection. The crossover and the mutation are used for recombination and do produce new individuals to the population. Selection is used for appointing better parents during the reproduction process [1]. Many different selection operators are well described in the literature. In the group of the most used are: Proportional Selection, Tournament Selection, Rank-Based Selection, Boltzmann Selection, Soft Brood Selection, Disruptive Selection, Nonlinear Ranking Selection, and Competitive Selection. This study proposes an improvement to recursive brute-force selection by the addition of local search on the level of the brute-force. In every level of recursive descent, all local individuals are recombined with each other (brute-force) as many times as there is an improvement (local search). Only the best-found individual after that is sent to the population of the higher level.

Keywords: *Local search, brute-force, recursion, selection operator*

1. Introduction

In the field of meta-heuristic global optimization, genetic algorithms are well known and widely applied. Genetic algorithms have stochastic nature because random or pseudo-random numbers are extensively used in common operators like crossover, mutation, and selection. Finding global optimum is rarely possible and, in most cases, genetic algorithms give sub-optimal solutions. Because of the stochastic nature found solutions can differ between different runs of the optimization process. At the same time, the possibilities for different runs are great option for parallel or distributed computing. Mechanisms in natural evolution have inspired the ideas involved in genetic algorithms. Meta-heuristics are most useful for optimization problems in multidimensional spaces with a high degree of non-linearity. A point in multidimensional space (for genetic algorithms it is the solutions space) is presented as vector of values. The set of vectors in the solutions space forms a population of individuals in terms of genetic algorithms. The size of the population is very problem dependent, and it is estimated empirically by trial and error approach [2]. The target function is applied to every one of the population. The result of this calculation gives the individual's fitness value. The fitness value gets better with approaching the optimum. Individuals with better fitness have better chances to mate and to form a new generation. The mating process consists of two common operators – crossover and mutation. Crossover contributes to the exploration of the solutions space when mutation contributes to the exploitation of the solutions space. Genetic algorithms are heavily researched in the last few decades, but there are still directions for improvements. Such direction is the selection operator. This research focuses on searching for optimum to set of benchmark functions by the addition of local search [3] in brute-force procedure when it is applied in recursive descent [4].

Selecting parents for mating can be done according many different strategies. The most important target of successful mating is to select as better individuals as possible more often with the hope that they will produce even better offspring. Selecting parents at random will lead to a very slow convergence of the optimization process [5]. The proportional selection [6] is one of the first improvements in the selection process. The idea behind it is that everyone receives a crossover chance proportional to its fitness value. Tournament selection [7] is another selection strategy in which by competition between a subset of individuals only winners are nominated for mating. Rank selection [8] on the other side uses ranking of individuals in order to determine their chances for mating. Boltzmann selection [9] uses simulated annealing. In this strategy, winners are nominated after a predetermined number of Boltzmann experiments.

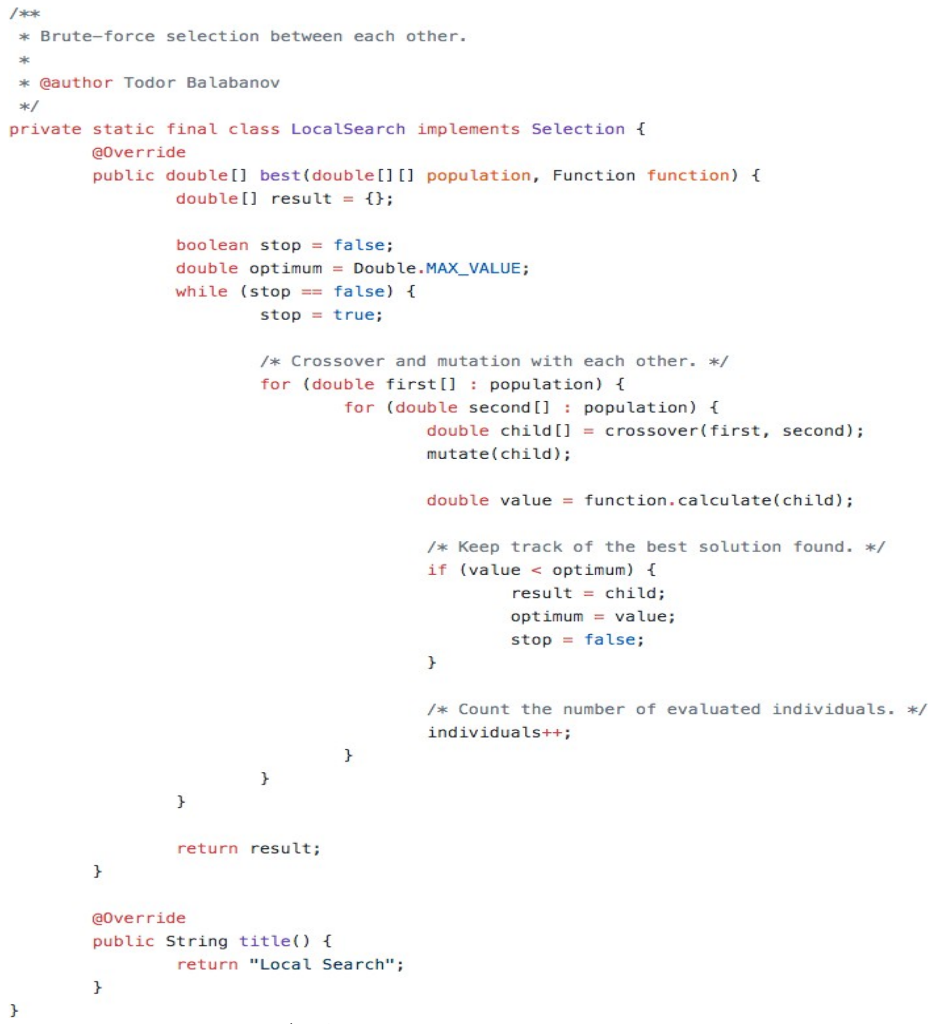
There are many hybrid implementations of the above listed selection strategies. This research proposes a local search extension of the brute-force element in the recursive selection proposed by Tomov, Zankinski, and Balabanov [10]. A hierarchy of sub-generations is generated, and the best-fitted individuals are selected with recursive descent. On each level of recursion single brute-force searching is applied and only one individual is promoted to a higher level. The newly proposed modification applies brute-force search as many times as there is an improvement of the findings, which in fact is a local search extension.

The rest of the paper is organized as follows: After the introductory part proposition for local search extension to recursive brute-force selection operator is presented; Some experiments and results are presented after that; Paper concludes with a summary of the research and with some directions for a further work.

2. Local Search Proposition

The selection operator described in [10] consists of recursive descent and brute-force. There is a predefined depth of recursion levels. On each level of recursion, there is a genetic algorithm sub-population. The size of sub-populations is equal, and it is also predefined. All individuals in the leaves of the recursive tree are randomly generated. The brute-force part of the algorithm consists of the fact that each individual mate with each other in the sub-population. Only the best-found individual is promoted to go in the higher sub-population. Each higher sub-population is fulfilled with individuals promoted from the lower levels.

The proposed improvement in this selection procedure states that brute-force should be returned as many times as there is an improvement of the best-found individual (Fig. 1).



```java
/**
 * Brute-force selection between each other.
 *
 * @author Todor Balabanov
 */
private static final class LocalSearch implements Selection {
	@Override
	public double[] best(double[][] population, Function function) {
		double[] result = {};

		boolean stop = false;
		double optimum = Double.MAX_VALUE;
		while (stop == false) {
			stop = true;

			/* Crossover and mutation with each other. */
			for (double first[] : population) {
				for (double second[] : population) {
					double child[] = crossover(first, second);
					mutate(child);

					double value = function.calculate(child);

					/* Keep track of the best solution found. */
					if (value < optimum) {
						result = child;
						optimum = value;
						stop = false;
					}

					/* Count the number of evaluated individuals. */
					individuals++;
				}
			}
		}

		return result;
	}

	@Override
	public String title() {
		return "Local Search";
	}
}
```

Fig. 1. Local search extended algorithm

The advantage of the proposed modification is that a much better investigation of the neighboring area of the search space is done. In this way, individuals with better properties are selected to mate in the evolutionary process.

Table 1 shows time consumption for achieving the results with a recursive level of 7 and population size at each recursive node of 11 individuals.

Table 1. Calculation time [ms] for recursion level of 7 and population size of 11.

Function / Algorithm	Michalewicz	Ackley	Schwefel	Rastrigin	Griewank
Local Search	1062492546	531027435	533232986	507650704	592370933
Brute-Force	403141211	166733879	175069174	159862955	218428047

Table 2 shows the number of fitness value calculation for achieving the results with a recursive level of 7 and population size at each recursive node of 11 individuals. Those numbers are not predefined, because the usage of brute-force procedure can vary in the number of new individuals created.

Table 2. Number of fitness calculations for recursion level of 7 and population size of 11.

Function / Algorithm	Michalewicz	Ackley	Schwefel	Rastrigin	Griewank
Local Search	641125639	641024362	640914978	641092606	640830762
Brute-Force	235794757	235794757	235794757	235794757	235794757

3. Experiments and Results

All experiments are done in 10000-dimensional solutions space. As benchmark functions are selected five very well-known functions: Michalewicz, Ackley, Schwefel, Rastrigin, and Griewank. Algorithms are tested with levels of recursive descent from 2 to 7 and with population size from 2 to 11. Table 1 shows the calculation time used in both algorithms for the configuration of recursive descent 7 and population size 11. Table 2 shows the total number of individuals evaluation (fitness value calculation) in both algorithms for the configuration of recursive descent 7 and population size 11. The exact numbers achieved in the experiments can be found at this URL (https://github.com/TodorBalabanov/FedCSIS-Conference-on-Computer-Science-and-Information-Systems-2020/tree/master/Local-Search-Brute-Force-and-Recursion-for-Selection-Operator).

Suboptimal values achieved by both algorithms are presented in Fig. 2 and Fig. 3. The presented information includes all experimental combinations of recursive descent levels and population sizes.

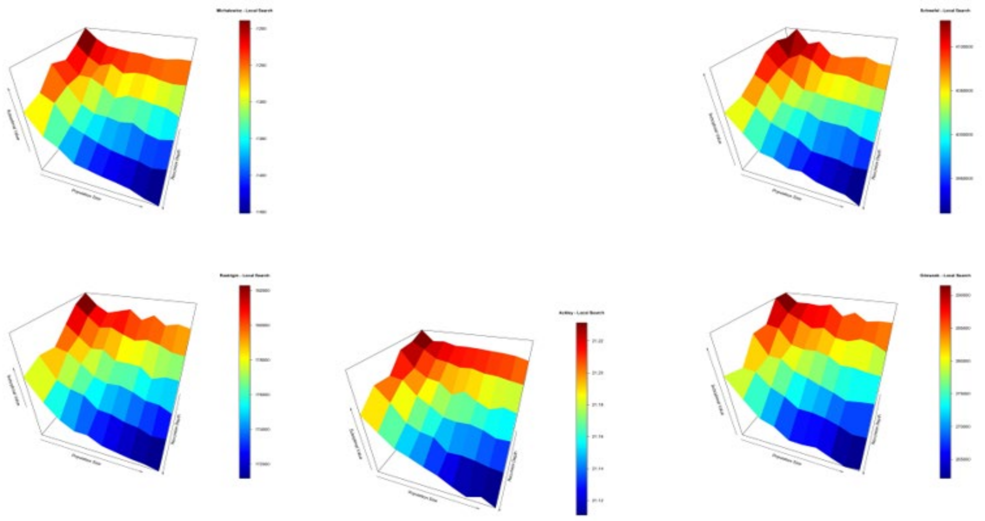


Fig. 2. Michalewicz, Ackley, Schwefel, Rastrigin,
Griewank Local Search – Suboptimal Values

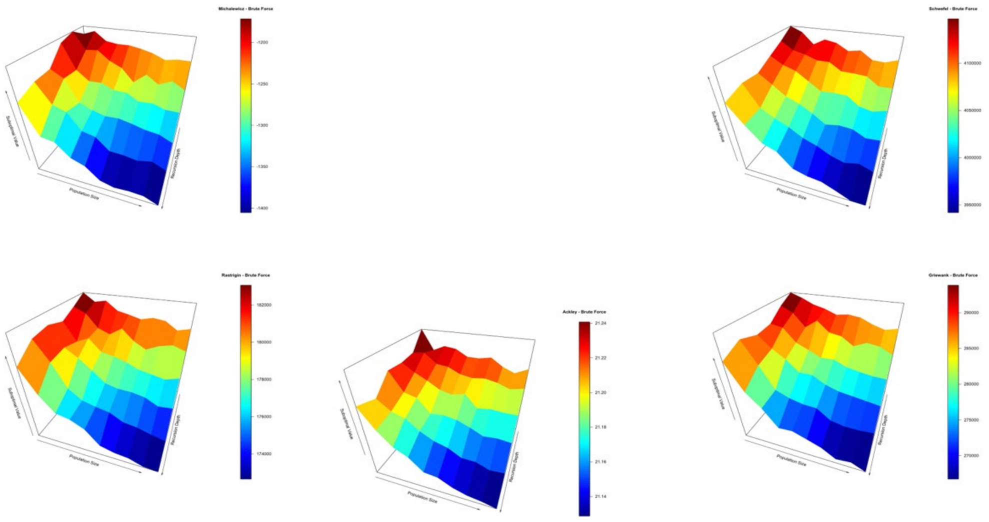


Fig. 3. Michalewicz, Ackley, Schwefel, Rastrigin,
Griewank Brute Force – Suboptimal Values

Fig. 4 and Fig. 5 illustrates the used of computational time. The graphs show that computation time rises with rising the size of the local node population size or the level of recursive descent. Such an increase is logical because when the local node population rises it means that much more brute-force recombinations will be possible. Also with deeper recursive descent, there would be much more local node populations to be evolved, which takes extra computation time.

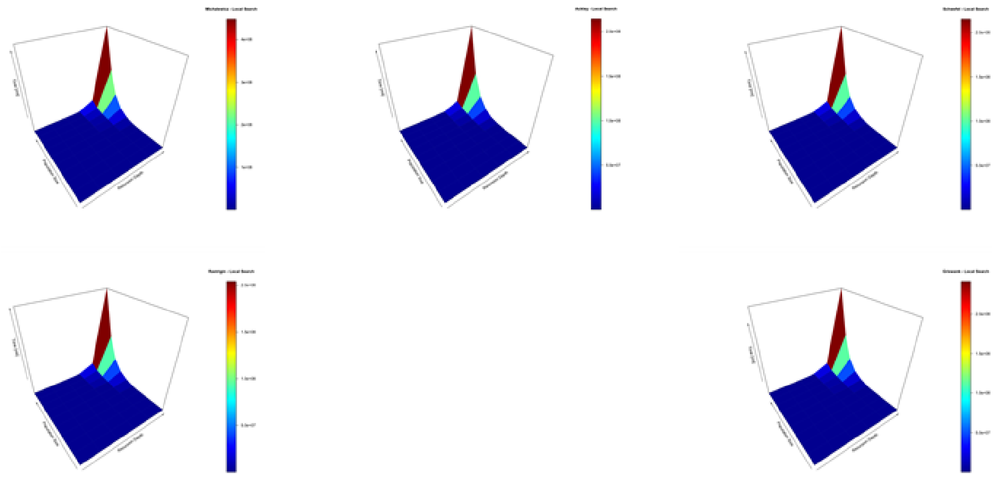


Fig. 4. Michalewicz, Ackley, Schwefel, Rastrigin,
Griewank Local Search – Local Search – Time [ms]

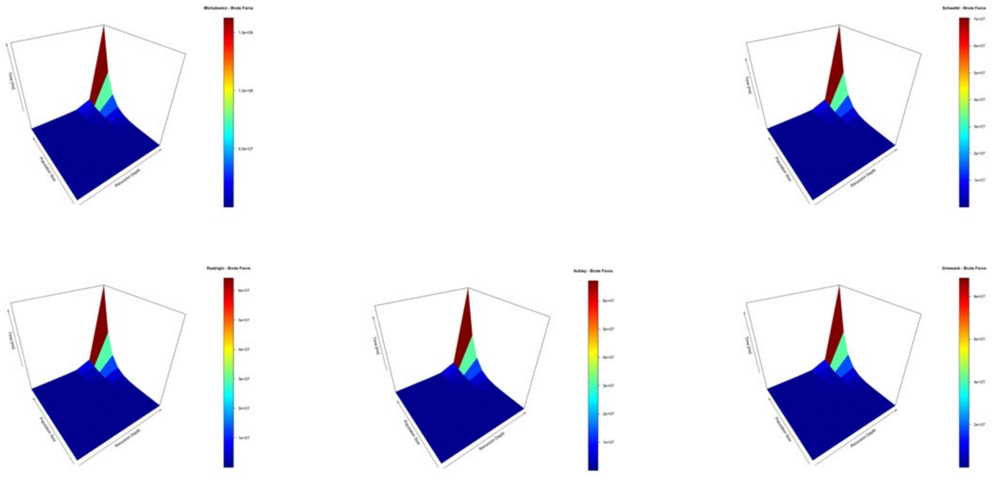


Fig. 5. Michalewicz, Ackley, Schwefel, Rastrigin,
Griewank Brute Force – Time [ms]

Fig. 6 and Fig. 7 show number of evaluations done during optimization process. Graphs show that local search has a deterministic predefined number of local node population number of individuals' evaluation. On the opposite side, brute-force has more local node population of individuals' evaluation, but better individuals are promoted for mating in the upper levels of the recursive descent.

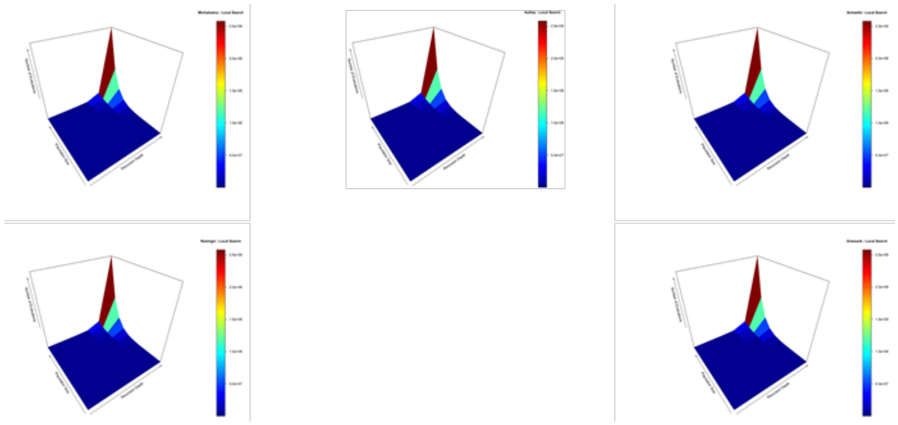


Fig. 6. Michalewicz, Ackley, Schwefel, Rastrigin, Griewank Local Search – Fitness Evaluations

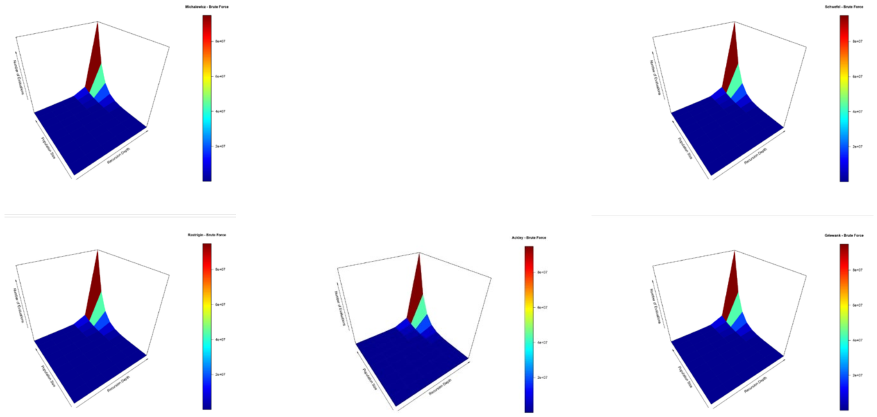


Fig. 7. Michalewicz, Ackley, Schwefel, Rastrigin, Griewank Brute Force – Fitness Evaluations

In Table 3 are shown the suboptimal values found by both algorithms for the configuration of recursive descent 7 and population size 11.

Table 3. Suboptimal values found for recursion level of 7 and population size of 11.

Function / **Algorithm**	**Michalewicz**	**Ackley**	**Schwefel**
Local Search	-1484.7137949531716	21.09334816052046	3877924.0971615044
Brute-Force	-1439.2296970724608	21.114702255301292	3919318.729777085
Function / **Algorithm**	**Rastrigin**	**Griewank**	
Local Search	170204.87849875208	259918.15469527297	
Brute-Force	171780.33307271387	262621.61053178157	

Values achieved with the local search extension are closer to the global optimums. The results in Table 3 show that Local Search gives better suboptimal solutions than the Brute-Force algorithm for all experimented functions. This improvement comes at the price of the higher computational time used.

4. Conclusion

This research extends the brute-force part of the recursive descent procedure in genetic algorithms selection operator with local search. The experimental results show very good efficiency, but it comes with the price of higher computational time. Because of the higher computational time, the proposed selection can be used for tasks that have no strict time limitations.

The advantage of the proposed modification is that it gives a better individual to be promoted for mating in the higher levels of the recursive descent, which indirectly gives better sub-optimal solutions. The disadvantage of the proposed local search modification is that it takes extra time and the number of local individuals' evaluations is not deterministic. The proposed modification and the selection operator as recursive descent may find application in a wide variety of combinatorial optimization problems, which can be represented in the terms of the genetic algorithms.

As further research, it will be interesting the same selection operator to be implemented in algorithms like differential evolution [11] with different sizes of the sub-populations [12] on different recursive levels like small sizes on low levels and bigger sizes on higher levels.

Acknowledgment

This research is funded by Velbazhd Software LLC and it is partially supported by the Bulgarian Ministry of Education and Science (contract D01–205/23.11.2018) under the National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)", approved by DCM # 577/17.08.2018.

References

1. Matsui, K.: New selection method to improve the population diversity in genetic algorithms. In: IEEE International Conference on Systems, Man, and Cybernetics, Japan, https://doi.org/10.1109/ICSMC.1999.814164, (1999).
2. Alander, J.: On optimal population size of genetic algorithms. Computer Systems and Software Engineering, https://doi.org/10.1109/CMPEUR.1992.218485, (1992).
3. Fellows, M., Fomin, F., Lokshtanov, D., Rosamond, F., Saurabh, S., Villanger, Y.: Local search: Is Brute-Force avoidable. Journal of Computer and System Sciences 78(3), 707–719, https://doi.org/10.1016/j.jcss.2011.10.003, (2012).

4. Gelfand, S., Mitter, S., Recursive stochastic algorithms for global optimization in Rd. SIAM Journal on Control and Optimization, https://doi.org/10.1137/0329055, (1991).
5. Wang, Q.: The genetic algorithm and its application to calibrating conceptual rainfall-runoff models. Water resources research 27(9), 2467–2471, https://doi.org/10.1029/91WR01305, (1991).
6. Back, T.: Self-adaptation in genetic algorithms. In: Proc. of First European Conference on Artificial Life, pp. 263–271, (1992).
7. Miller, B., Goldberg D.: Genetic algorithms, tournament selection, and the effects of noise. Complex Systems 9, 193–212 (1995).
8. Grefenstette, J.: Rank-based selection. Evolutionary Computation, (2000).
9. Goldberg, D.: A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. Complex Systems 4, 445–460 (1990).
10. Tomov, P., Zankinski, I., Balabanov, T.: Genetic algorithm selection operator based on recursion and Brute-Force. In: 14th Annual Meeting of the Bulgarian Section of SIAM, (2019).
11. Surender Reddy, S., Bijwe, P. R.: Differential evolution-based efficient multi- objective optimal power flow. Neural Computing and Applications 31, 509–522, https://doi.org/10.1007/s00521-017-3009-5, (2019).
12. Koumousis, V.K., Katsaras, C.P.: A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Transactions on Evolutionary Computation 10(1), 19–28, https://doi.org/10.1109/TEVC.2005.860765, (2006).