Journal of Universal Computer Science, vol. 22, no. 6 (2016), 874-893
submitted: 30/11/15, accepted: 28/5/16, appeared: 1/6/16 © J.UCS

Improving Performance of the Differential Evolution Algorithm Using Cyclic Decloning and Changeable Population Size

Piotr Jędrzejowicz
(Gdynia Maritime University, Gdynia, Poland
pj@am.gdynia.pl)

Aleksander Skakovski
(Gdynia Maritime University, Gdynia, Poland
askakow@am.gdynia.pl)

Abstract: Differential evolution (DE) is a stochastic global optimization method that has been under continuous development during the past two decades. It has been recognized that preserving the diversification of population can significantly improve the performance of DE. Although, several results and approaches to population diversification have been proposed, it seems that this issue still has a potential for development. In this paper we have studied experimentally the possibility of increasing the performance of DE. Our investigation aims at identifying how the performance of DE depends on such factors as population diversity, size and number of fitness function evaluations carried out by DE to yield a solution. In our experiments we diversified the population in an intensive manner using the proposed decloning procedure carried out in cycles, and also through increasing the population size. The choice of how to preserve the diversification may depend on restrictions imposed on the population size, response time, and the quality of solutions that should be met by a specific implementation of the algorithm. The obtained results allowed us to propose a performance improvement policy that might noteworthy improve both the efficacy and response time of the algorithm. The discrete-continuous scheduling with continuous resource discretisation was used as the test problem.

Keywords: performance improvement, differential evolution, population diversification, decloning
Categories: D.1, G.1.6, I.6, I.2.8, J.0

1 Introduction

During search performed by the evolutionary algorithm (EA), it often comes to the point when individuals in the population reach a configuration such that evolutionary operators no longer produce offspring that can outperform their parents [Fogel, 94]. This phenomenon is known as convergence of EAs. The natural consequence of convergence is that in the population grows the number of similar or identical individuals. An important concern and shortcoming of EAs is premature convergence, i.e. convergence to a local optima. EA operating on the population with low diversity sticks in local optima, which impedes searching for global ones. Thus, ensuring high diversity of the population is viewed by researchers as one of the essential factors that affect EAs performance. To prevent getting trapped in local optimum, several

approaches have been proposed in the literature. Short review of these approaches can be found in [Section 2].

In this paper we study a special case of the evolutionary algorithm, which is differential evolution algorithm. Differential evolution is a stochastic direct search and global optimization algorithm proposed in [Storn, 97].

For our experimental research on preserving population diversity, we designed a decloning procedure which is supposed to cyclically replace genetically identical individuals (clones) with randomly generated ones, the detailed description of the procedure is given in [Section 5.2].

The goal of our research was to investigate the extent to which performance of the considered differential evolution algorithm depends on such parameters as the population diversification rate, the size of the population, and the number of fitness function evaluations carried out by the algorithm to yield a solution to the problem.

The goal of the experiments was to determine the most advantageous, in terms of the algorithm's performance, values of mentioned parameters. The results obtained through our experiments allow us to state, that the main contribution of our work is the increase the range of tools and methods for preserving diversity of the population undergoing DE, as well as proposing the performance improvement policy which takes advantage of experimentally determined relationships between the performance of DE and considered parameters. We also propose a decloning procedure, that was used in the experiments for cyclic population diversification. The procedure extends packing technique proposed in [Kureichick, 96], ROG technique proposed in [Rocha, 99], and $(\mu + 1)$ EA with genotype diversity proposed in [Friedrich, 09]. In the notation $(\mu + \lambda)$, μ stands for the size of the population and λ - for the number of offspring generated at a time from parents.

As a test problem, we used the discrete-continuous scheduling problem with continuous resource discretisation (DCSPwCRD), the brief definition of which is given in [Section 3], whereas detailed description might be found in [Różycki, 00] and [Jędrzejowicz, 14]. In order to conduct our tests, we adapted the differential evolution algorithm proposed in [Damak et al. 09] for solving the DCSPwCRD. In the paper, we denote this algorithm as DEA. A brief description of the DEA is given in [Section 4] and the extended description might be found in [Jędrzejowicz, 14]. The description of the policy for the DE's performance improvement, assumptions of the experiment, as well as discussion on the results and illustrating them charts are given in [Section 5]. [Section 6] includes conclusions and idea for the future research.

2 Research on Preventing Premature Convergence in Evolutionary and Genetic Algorithms

In the literature there have been proposed numerous approaches, methods, techniques and mechanisms to cope with the issue of premature convergence in evolutionary and genetic algorithms. A comparative review of approaches to prevent premature convergence in GA is given in [Pandey, 14] and an overview of methods maintaining diversity in genetic algorithms in [Gupta, 12]. In addition to the empirical work, there has also been conducted theoretical research on population diversification.

In [Friedrich, 09] it was investigated theoretically and empirically the global exploration capabilities of mutation-based algorithms for sublinear population sizes. Using a simple bimodal test function and rigorous runtime analyses, the authors compared well-known genotype and fitness diversity preserving mechanisms, deterministic crowding and fitness sharing with a plain $(\mu + 1)$ EA without diversification. It was shown that diversification is necessary for global exploration, but not all mechanisms succeed to the same extent. According to the authors, population is nearly useless for the algorithms with genotype and fitness diversity, as it was experienced the same performance as for simple hill climbers like local search or the (1 + 1) EA. On the other hand, algorithms with fitness sharing and deterministic crowding performed better than with genotype and fitness diversity due to their higher success probabilities. Their experimental results indicate a similar behavior of the diversity preserving mechanisms also for larger populations.

In [Oliveto, 15] it was conducted the theoretical analysis and empirical study on the $(\mu + 1)$ EA operating on the Balance Function. It follows from the experiments that linear population sizes are sufficient to make both the $(\mu + 1)$ EA without diversity and equipped with genotype diversity efficient for Balance function. Also there appeared to be a sharp threshold at population size cn for some $c > 0$ at which the expected performance of these two algorithms turns from exponential to polynomial. On the other hand, the $(\mu + 1)$ EA with fitness diversity appears to be effective for population sizes considerably smaller than the sublinear ones required for their proof to work. The authors stated also, that for larger population sizes the crowding mechanism becomes detrimental.

It follows from the last two cited works that the effectiveness of EA, depends not only on the diversification mechanisms used, but also on the size of the population evolved. It seems that the authors of these two works differently conclude on the effectiveness of genotype, fitness, and deterministic crowding mechanisms. For example, according to [Friedrich, 09] the algorithms without diversity or with genotype or fitness diversity preserving mechanisms were considered as weak and performed worse than with deterministic crowding, which is not entirely consistent with the results obtained in [Oliveto, 15]. Such inconsistency testifies to the fact that the efficiency of diversification mechanisms still remains an open issue and requires further theoretical and empirical research.

In [Kureichick, 96] it was proposed a Social Disasters Technique (SDT) where packing catastrophic operator was used for fitness diversity preserving by replacing individuals of the same fitness, except for one, with random individuals in the situations when the level of population diversity dropped to some preset limit. In [Rocha, 99] it was proposed a Random Offspring Generation (ROG) technique, aimed at genotype diversity preserving, by which the crossover is carried out only on parents with different genotypes, otherwise one or two offspring were generated at random. Although this technique prevents the formation of clone-offspring from clones-parents, it does not prevent the transition of clones from generation to generation, since each individual with an old population has the right to go to the next one with probability proportional to its quality. In [Friedrich, 09] it was proposed $(\mu + 1)$ EA with genotype diversity, which was based on (2 + 1) GA, described in [Storch, 04]. This algorithm does not allow an offspring to enter the population, if there already exists its genetic duplicate.

3 Problem Formulation

The considered DCSPwCRD problem originates from a general Discrete-Continuous Scheduling Problem (DCSP), first described in [Józefowska, 98]. The DCSP is to find a sequence of jobs on machines and, simultaneously, a continuous resource allocation to the jobs, that minimizes given scheduling criterion. In order to simplify the allocation of a continuous resource to a feasible schedule the idea of continuous resource discretisation was proposed in [Różycki, 00]. We use the same approach in the paper. Namely, we assume that the number of possible continuous resource allocations to a task J_i is D_i, i.e. is fixed, and the amount of the continuous resource for each $l_i = 1, 2, \ldots, D_i$ is known in advance (in the original problem there was infinite number of the continuous resource allocations to a task and the amount of the continuous resource to be allocated was not known in advance). Because a different amount of the continuous resource is allocated to task J_i for each l_i, l_i is called a processing mode of task J_i. Such discretisation of the continuous resource allows to treat it as a discrete resource and define a problem Θ_Z.

We define the problem Θ_Z in the same way as in [Różycki, 00]. Namely, let $\boldsymbol{J} = \{J_1, J_2, \ldots, J_n\}$ be a set of nonpreemtable tasks, with no precedence relations and ready times $r_i = 0$, $i = 1, 2, \ldots, n$, and $\boldsymbol{P} = \{P_1, P_2, \ldots, P_m\}$ be a set of parallel and identical machines, and there is one additional renewable discrete resource in amount $U = 1$ available. A task J_i can be processed in one of the modes $l_i = 1, 2, \ldots, D_i$ (D_i – the number of processing modes of task J_i), for which J_i requires a machine from $\boldsymbol{P}$ and amount of the additional resource known in advance. The processing mode of J_i cannot change during the processing. For each task two vectors are defined: a processing times vector $\tau_i = [\tau_i^1, \tau_i^2, \ldots, \tau_i^{D_i}]$, where $\tau_i^{l_i}$ is the processing time of task J_i in mode $l_i = 1, 2, \ldots, D_i$ and a vector of additional resource quantities allocated in each processing mode $u_i = [u_i^1, u_i^2, \ldots, u_i^{D_i}]$. The problem is to find processing modes for tasks from $\boldsymbol{J}$ and their sequence on machines from $\boldsymbol{P}$ such that schedule length $Q = \max\{C_i\}$, $i = 1, \ldots, n$ is minimized.

The formulated problem Θ_Z is a particular case of more general Multi-Mode Resource Constrained Project Scheduling Problem (MMRCPSP), which is known to be NP-hard [Bartusch, 88].

4 Differential Evolution Algorithm

For the purpose of our research we adapted the differential evolution algorithm for solving the multi-mode resource-constrained project scheduling problem proposed in [Damak, 09] for solving the discrete-continuous scheduling problem with continuous resource discretisation (DCSPwCRD). In this section we give only a general idea of the DEA for solving DCSPwCRD, and the extended description of the algorithm the reader might find in [Jędrzejowicz, 14].

The DEA is an evolutionary algorithm that evolves population P^1_k of *target vectors* S_{tg} in cycles k. In every cycle k, for every target vector S_{tg} in P^1_k *a trial vector* T in P^2_k is created. This way P^2_k is filled with trial vectors T. The pseudo code of the DEA is given below.

```
The DEA:
 For every target vector S_tg in the current population P^1_k
 do:
  Create a mutant vector M from three vectors S_0, S_1, S_2
  randomly chosen from P^1_k, using formula:
  M = S_0 + A*r*(S_1 - S_2), where A > 0 - is a scale factor,
  that controls the evolution rate of the population and
  r ∈ [0, 1].
  Create a trial vector T in P^2_k applying crossover
  operator to each element of mutant vector M and
  corresponding element of target vector S_tg according to
  the rule:
   if the random number r ≤ C_r, C_r ∈ [0, 1]), then the
   trial element is inherited from mutant vector M,
   otherwise from target vector S_tg.
 Create a new population P^1_k+1 selecting the best vectors
 from P^1_k and P^2_k.
 Repeat evolution cycles until the stop criterion is met.
End.
```

The crossover constant C_r controls in the DEA probability, that trial vector T will inherit the element either from target vector S_{tg}, or mutant vector M.

5 Computational Experiments

5.1 Parameter Set Up

In our experiments on efficiency improvement, values of the parameters of the DEA were assumed to be the same as in [Damak, 09], namely the scale factor A which controls the evolution rate of the population was set to $A = 1{,}5$ and values of the variable $rand \in [0, 1]$. The crossover constants Cr_p and Cr_l which control the probability that the trial individual will receive the actual individual's tasks or modes were set $Cr_p = 0{,}2$ and $Cr_l = 0{,}1$, where p and l in the notations Cr_p and Cr_l stand for tasks positions and modes respectively. In our experiments, we considered the following population sizes: 20, 50, 100, 200, 1000, and the numbers of the fitness function evaluations necessary for the DEA to yield one solution: 37800, 450000, and 720000. An initial population of feasible individuals in the DEA was generated using the uniform distribution equal $1/n$ for the tasks, and $1/D$ for the task's modes. Our assumptions concerning the test problem are as follows. We considered three combinations of n x m: 10x2, 10x3, and 20x2, where n is the number of tasks and m is the number of machines, and three levels of the continuous resource discretisation D: 10, 20, 50. This way we considered nine sizes n x m x D of the problem Θ_Z: 10x2x10, 10x2x20, 10x2x50, 10x3x10, … , 20x2x50. For each of the sizes, we considered 6 instances of the problem Θ_Z, which makes a test set of 54 instances of the problem in total. This test set of 54 instances was used for testing each parameter configuration under investigation, and the considered parameters were: the decloning period T^d, population size x_P, and the number of fitness function evaluations carried out by the algorithm to yield a solution to the problem *#ev*.

All tests were carried out on a PC under 64-bit operating system Windows 7 Enterprise with Intel(R) Core(TM) i5-2300 CPU @ 2.80 GHz 3.00GHz, RAM 4GB compiled with aid of Borland Turbo Delphi for Win32. When *#ev* was set to 720000, mean time required by the DEA to find a solution for the problem sizes 10x2 and 10x3 for all discretisation levels was approximately 2 - 3s and for the problem size 20x2 for all discretisation levels approximately 5 – 6s. The total time taken by the DEA to process all 54 instances was approximately 206s.

5.2 Decloning Procedure

Our experiments on population diversification we conducted using the decloning procedure (DP) that was invoked in regular cycles with the duration of the period determined as the number of fitness function evaluations. The number of calls of the DP depended on the duration of the decloning period and *#ev* that were allowed for the DEA. We assumed, that the DP would remove 100% of identified clones, and replace them by randomly generated individuals (solutions). In these new solutions, the order of the tasks and task's execution modes were determined at random.

The thorough identification of clones in the population might be time consuming and therefore extend the time needed to execute the algorithm. In order to avoid that, we designed a simpler and quicker decloning procedure that identifies clones in an approximate way. We assumed, that a solution is a clone of another solution, if the following conditions hold:

- The fitness function value of both solutions is the same.
- There exists the same task in both solutions, that is executed in the same mode and that is placed at the same position, chosen at random from the second half of the solution's task list.
- The finish time of the task from the second condition in both solutions is the same.

If at least one of the conditions is not met, then both solutions are different. While the establishment of the first condition is obvious, the establishment of the second and third conditions can be justified by the need to increase the probability of clones identification. Considering tasks only from the second half of the solution in the second condition, on the one hand, simplifies the identification process, on the other hand, used together with the third condition, increases the probability that the solutions under consideration are clones. It is obvious, that such method does not ensure identification of all clones present in the population. In our experiments, the amounts of clones identified in the same population by the Decloning procedure run multiple times varied within 13% range.

5.3 Experiments on Decloning and Population Size

For the purpose of evaluating the effect of the considered parameters on DEA performance we introduce the parameter $sumC_{max}$, which is the total of C_{max} values obtained for the test set of 54 instances of the problem. Thus, in order to carry out a single test for a particular parameter configuration and, therefore, to obtain a single value of $sumC_{max}$, the DEA was run 54 times, each time processing one of 54 test instances. The total value of the obtained 54 schedule lengths created a single value of $sumC_{max}$. To ensure the credibility of results, the tests were always carried out by the

DEA with the same seed of the random number generator. Therefore, the only factor that could cause a change in the results was the value of the parameter that was tested. In cases, when the DEA had to be run with the randomized seed of random number generator, for evaluation and comparison purposes we used average of the $sumC_{max}$ values (AVG $sumC_{max}$) obtained in multiple tests. The need of such aggregated parameter is justified by the stochastic nature of the DEA, which causes yielding different results for the same input data when the algorithm is run repeatedly. The use of AVG $sumC_{max}$ as a reference value will make our observations and further conclusions more credible and allow for a reliable comparison of the algorithm's performance when run with different parameter settings.

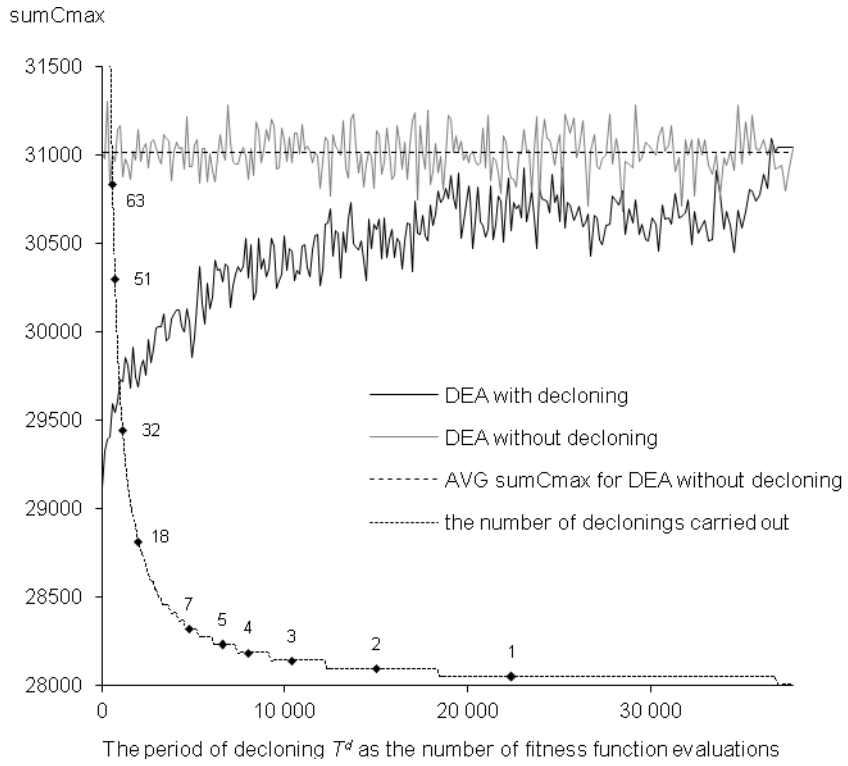

Figure 1: The effect of decloning on $sumC_{max}$, x_P = 20, #ev = 37800, $T^d \in [20, 37800]$

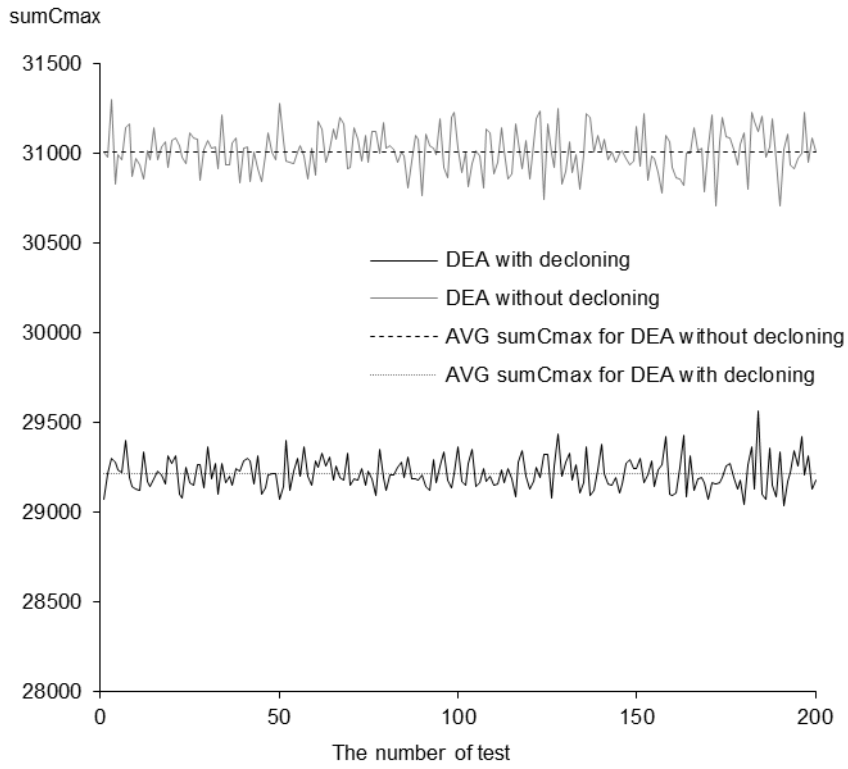

Figure 2: $sumC_{max}$ yielded by the DEA without and with decloning, $x_P = 20$, #ev = 37800, $T^d = 20$

An example of such comparison is given in [Fig. 2], where the values of AVG $sumC_{max}$ are shown for the cases when the algorithm was run with and without decloning with the randomized seed of random number generator. In this example, AVG $sumC_{max}$ values in both cases were calculated as the average of 200 $sumC_{max}$ values obtained in 200 tests. We will discuss the results of our tests using figures illustrating the influence of decloning and population size on the DEA's performance. In our experiment, results generated by DEA are considered to be better, the smaller are the corresponding values of $sumC_{max}$.

5.3.1 Decloning

In order to reveal the influence of decloning on the results obtained by the DEA, we run the algorithm multiple times, each time processing the test set of 54 instances of the problem with different rate of decloning. The period of decloning T^d was defined as the number of fitness function evaluations that were carried out between the successive calls of the Decloning procedure. The largest improvement effect of

decloning on the results is observed when it is performed frequently, i.e. when decloning period T^d takes small values, [see Fig. 1, 2, 3, 4, 5, 6]. The curves corresponding to the DEA with decloning, were built on the points, each of which is a value of $sumC_{max}$ obtained for different decloning periods. In order to validate the effect of decloning we used AVG $sumC_{max}$ as the reference value, that was determined for presumably the best decloning period, e.g. [see Fig. 2], in which the curves for the DEA with and without decloning were obtained for $T^d = 20$. The improvement effect of decloning on the results yielded by the DEA given in percent is shown in [Fig. 8]. In [Fig. 8], we compare values of AVG $sumC_{max}$ determined for all considered population sizes, when the DEA had at its disposal $\#ev = 37800$, $\#ev = 450000$, and $\#ev = 720000$. It follows from [Fig. 8], that the most significant improvement effect of decloning on the results is observed when the DEA operates on small populations. For example, the curve labeled "20^d", which stands for the population size $x_P = 20$ in the DEA with decloning, shows the greatest improvement effect of decloning among all considered population sizes. The results yielded by the DEA with decloning and $x_P = 20$ were on average 5,79% ($\#ev = 37800$), 7,01% ($\#ev = 450000$), and 7,29% ($\#ev = 720000$) better, than the results yielded without decloning.

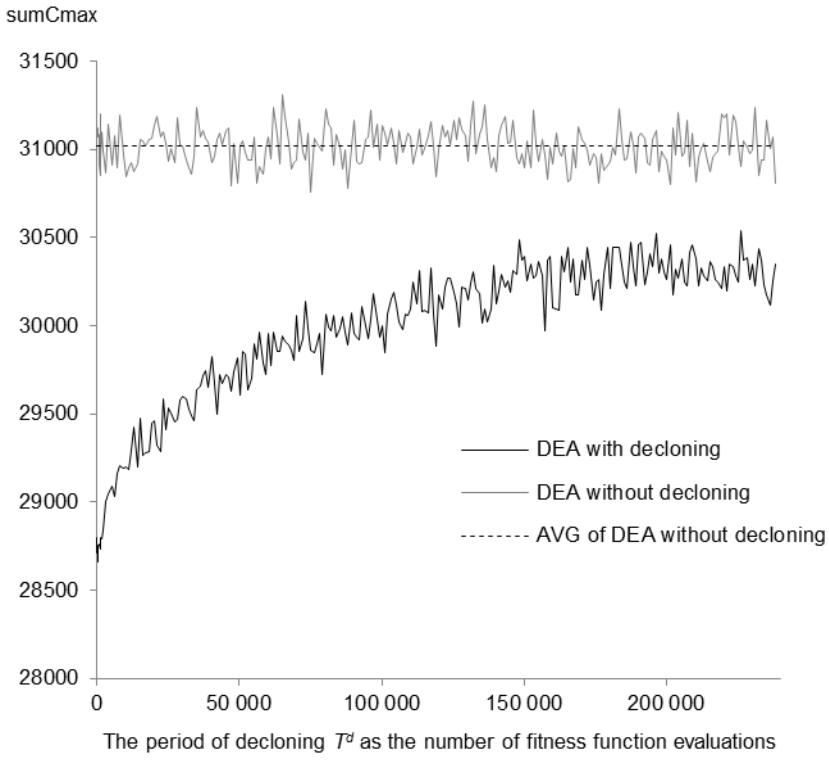

Figure 3: The effect of decloning on $sumC_{max}$, $x_P = 20$, $\#ev = 720000$, $T^d \in [20, 238200]$

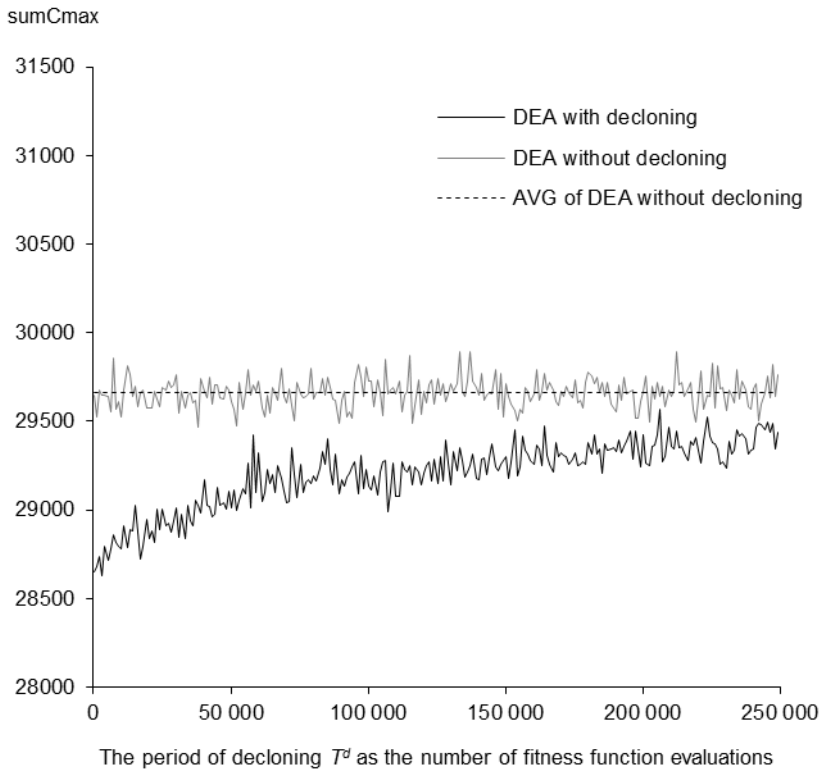

Figure 4: The effect of decloning on sumC_{max}, x_P = 50, #ev = 720000, $T^d \in [50, 249200]$

The improvement effect of decloning gradually decreases with the increase of population size resulting in no improvement at all when the DEA operates on large population. The above observation can be drawn from comparing curves labeled "50^{d}", "100^{d}", "200^{d}", and "1000^{d}" in [Fig. 8], where the labels denote the population sizes in the DEA with decloning. [Fig. 9 and 10] show the improvement effect of decloning on AVG *sum*C_{max}, where values AVG *sum*C_{max} of the DEA with decloning are compared to the values of AVG *sum*C_{max} yielded by the DEA without decloning (letters "nd" in the upper index of the label denote the cases without decloning). It should be also added, that in all considered cases, except for one, the DEA with decloning performed better than without decloning. The exception is $x_P = 200$ (*#ev* = 37800), where AVG *sum*C_{max} of the DEA without decloning were better by 0,02%. The decloning periods T^d, presumably most advantageous for the DEA's performance, used in our experiments are given in [Tab. 1].

#ev	**20^d**	**50^d**	**100^d**	**200^d**	**1000^d**
37800	20 (1)	50 (1)	100 (1)	17400 (87)	19000 (19)
450000	420 (21)	50 (1)	100 (1)	200 (1)	21000 (21)
720000	440 (22)	3200 (64)	1000 (10)	2400 (120)	51000 (51)

Table 1: Decloning periods T^d most advantageous for the DEA's performance, given as the number of fitness function evaluations and the number of generations in parentheses respectively

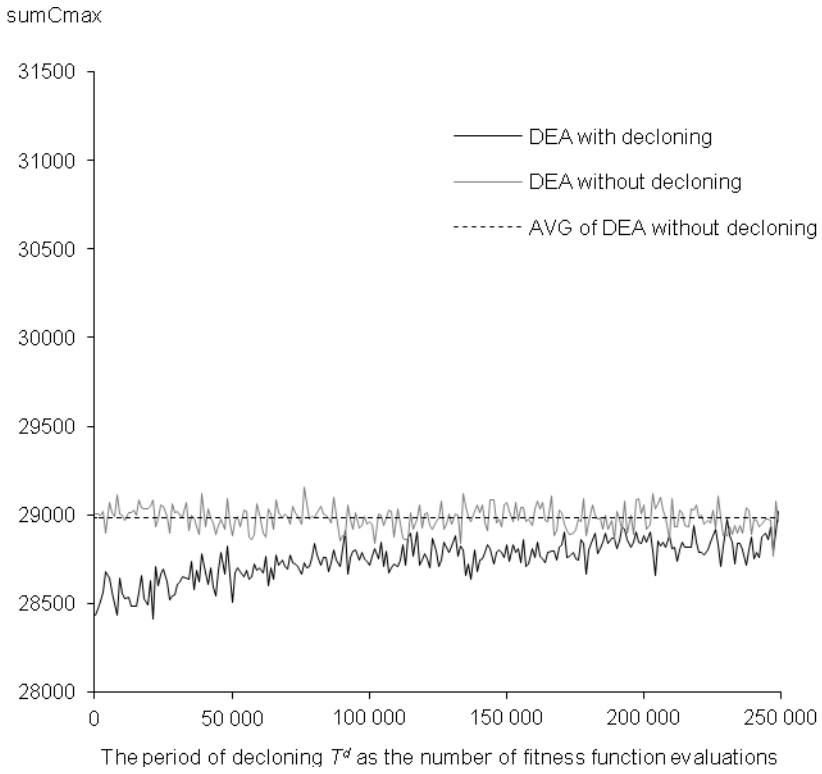

Figure 5: The effect of decloning on $sumC_{max}$, $x_P = 100$, #ev = 720000, $T^d \in [100, 249200]$

5.3.2 Population Size

As it has been already observed, increasing the population size weakens the improvement effect of the decloning. Nonetheless, population size is the second important factor contributing to the results improvement. The general observation is that with the population size growth, the DEA's results become better. When the population size was large, e.g. $x_P = 1000$, [see Fig. 7], the results obtained were better in comparison to the cases with the smaller population sizes, [see Fig. 3, 4, 5, 6]. It might seem at this point, that the population size is a principal factor affecting the results improvement, and that there is no need of decloning at all, since it is enough to increase population size to achieve better results. However, this is not always true. It turns out, that there are circumstances in which decloning ensures better results than increasing population size. In order to determine the most beneficial strategy for the results improvement, the third factor, namely *#ev* should be taken into consideration.

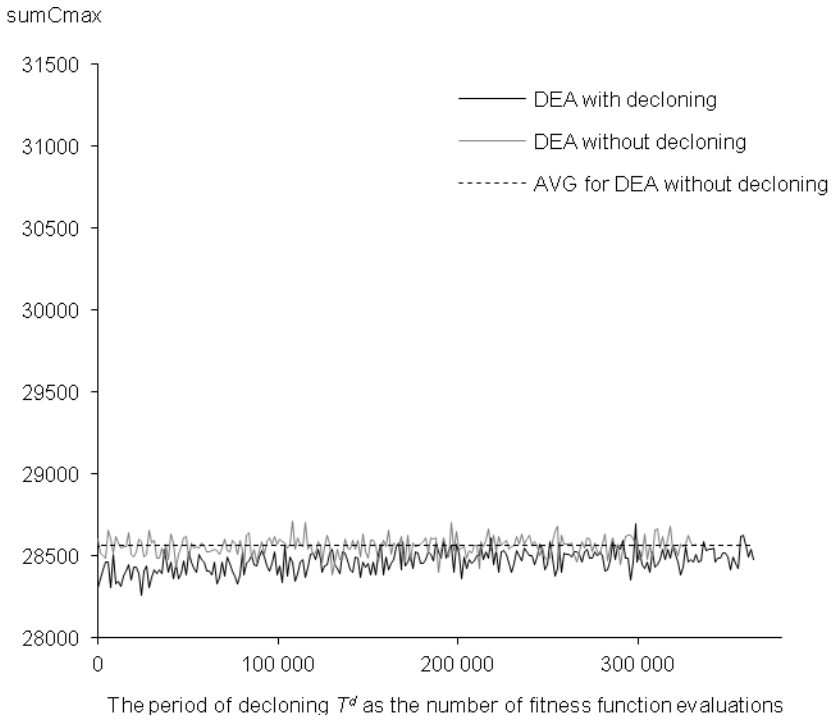

Figure 6: The effect of decloning on $sumC_{max}$, $x_P = 200$, #ev = 720000, $T^d \in [200, 364200]$

5.3.3 The Number of Fitness Function Evaluations

Our experiments show, that *#ev* might also contribute to the algorithm's performance improvement. When the DEA without decloning operated on a large population, e.g. $x_P = 1000^{nd}$, the results were improved over 8% merely by the increase of *#ev* from 37800 to 720000, [see Fig. 11]. The same figure shows an improvement of about 1,3% for $x_P = 200^{nd}$ and no improvement at all for the population sizes $x_P \in \{20^{nd}, 50^{nd}, 100^{nd}\}$. When the DEA operated with decloning, the improving effect, caused by the increase of *#ev*, was observed for all population sizes, [see Fig. 12]. Although, the greatest improvement effect is observed again for $x_P = 1000^{d}$, we must remind, that for this population size decloning does not contribute to the results improvement, as the curves $x_P = 1000^{nd}$ and $x_P = 1000^{d}$ in [Fig. 9] are identical.

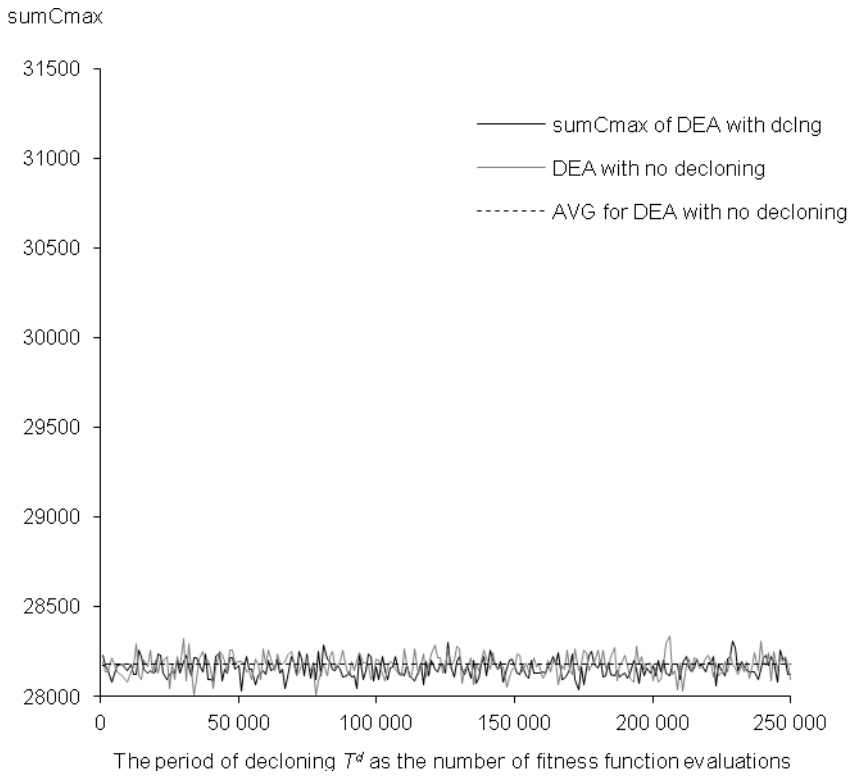

Figure 7: The effect of decloning on $sumC_{max}$, x_P = 1000, #ev = 720000, $T^d \in [1000, 250000]$

5.4 Performance Improvement Policy

The main goal of our experiments was to find out how to improve the ability of the DEA to yield better solutions. We tried to achieve this goal by making use of decloning and determining most advantageous T^d, x_P, and *#ev*. As it follows from the experiment, all three factors, might improve the results, provided that they have been assigned the most advantageous values. [Tab. 2] below shows in percent relative difference between the actual AVG $sumC_{max}$, for particular *#ev* and x_P, and the best among all values of AVG $sumC_{max}$ (obtained for $\#ev = 720000$ and $x_P = 1000$), provided, that in each considered case, the decloning period T^d was assigned the values from [Tab. 1]. Thus, the relative difference between AVG $sumC_{max}$ for the case $\#ev = 37800$, $x_P = 20$ and the best AVG $sumC_{max}$ is 3,77%. The smallest relative differences for particular *#ev* are given in [Tab. 2] in bold font, with the purpose to indicate values of x_P that are most preferrable in terms of increasing the performance of the DEA. Therefore, in order to achieve better results, x_P should be determined according to the *#ev* used. It follows from [Tab. 2], that if *#ev* increases, then most advantageous values of x_P increase as well.

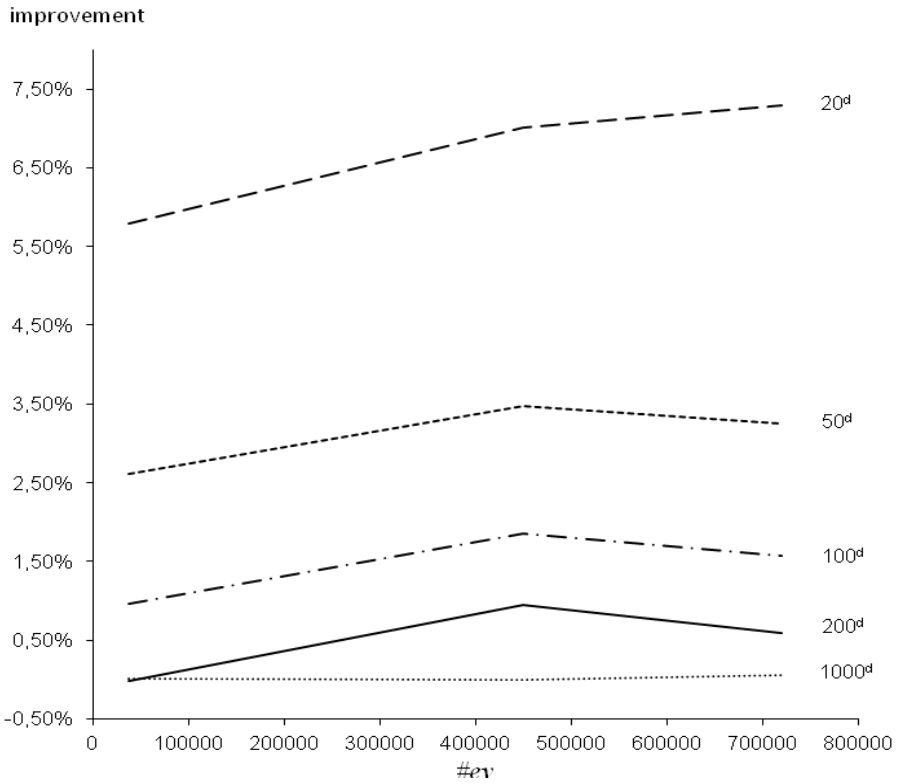

Figure: 8 The improvement of the results due to decloning for different population sizes x_P compared to the case without declining

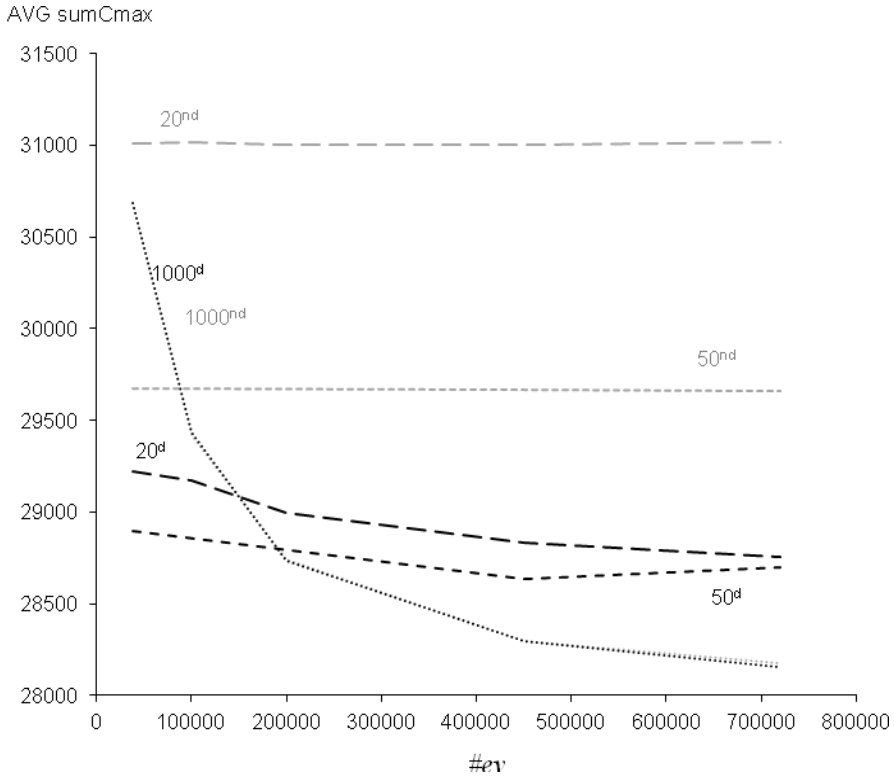

Figure 9: The difference between AVG sumC_{max} values obtained by the DEA with and without decloning for population sizes x_P = 20, x_P = 50, x_P = 1000

#ev	**20^d**	**50^d**	**100^d**	**200^d**	**1000^d**
37800	3,77%	2,62%	**1,91%**	2,83%	8,98%
450000	2,39%	1,69%	1,02%	**0,48%**	0,50%
720000	2,12%	1,91%	1,32%	0,83%	**0,00%**

Table 2: The relative difference between AVG sumC_{max} obtained for particular values of #ev and x_P, and the best AVG sumC_{max}, obtained for #ev = 720000 and x_P = 1000

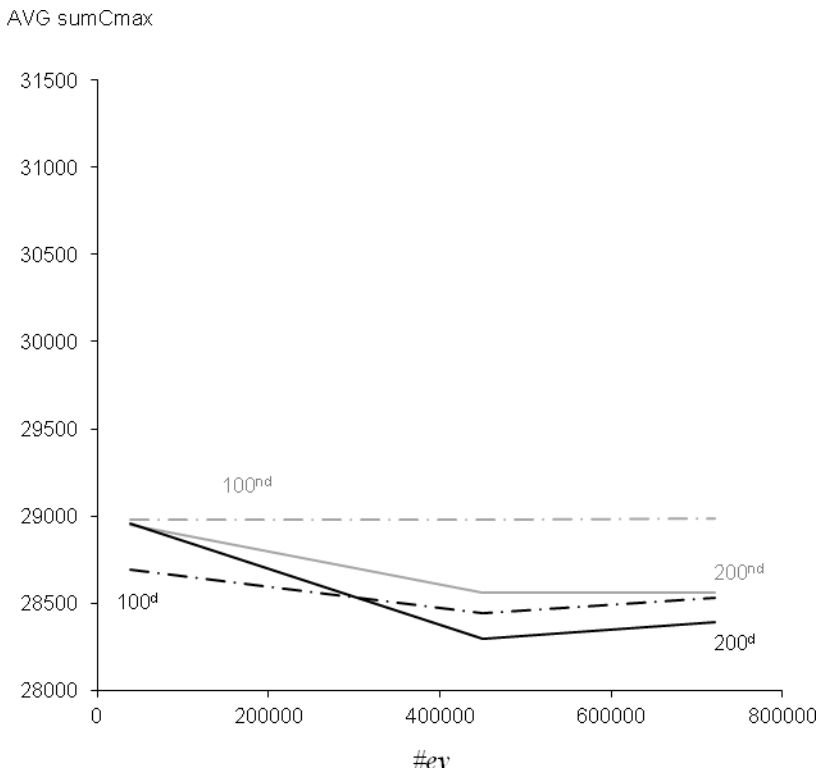

Figure 10: The difference between AVG sumC_{max} values obtained by the DEA with and without decloning for population sizes $x_P = 100$, $x_P = 200$

Based on the above observations, we propose a policy for improving the DEA's performance in terms of the quality of results and, when possible, in terms of response time. In our view the DEA's performance can be improved by adjusting x_P to the available *#ev*, such that will ensure the best results, e.g. if *#ev* is limited to 37800, then $x_P = 100$, if *#ev* = 450000, then $x_P = 200$, and if *#ev* = 720000, then $x_P = 1000$, [see Tab. 2].

The response time of the DEA can be improved as follows. Suppose, the DEA is searching for a solution having at its disposal *#ev* = 720000, which, according to [Tab. 2], assumes $x_P = 1000$ for the maximal DEA's performance. Although, $x_P = 1000$ ensures the best results at the end of the computing, however, setting $x_P = 100$ instead of 1000 ensures better AVG $sumC_{max}$ value within *#ev* = 37800 (compare 1,91% vs. 8,98% in [Tab. 2] and lines 100^d and 1000^d in [Fig. 12] respectively). The DEA with $x_P = 1000$ yields the same AVG $sumC_{max}$ value only after *#ev* ≈ 200000 (observe two auxiliary perpendicular grey dashed lines in [Fig. 12]). Therefore, setting $x_P = 100$ instead of 1000 for the first *#ev* = 37800 is an

opportunity for shortening the response time of the DEA. Thus, after carrying out $\#ev = 37800$, $x_P = 100$ should be changed into $x_P = 1000$, and the rest of the computing would take $\#ev = 720000 - 200000 \approx 520000$. Now, the same AVG $sumC_{max}$ as for $\#ev = 720000$ would be yielded by the DEA carrying out only $\#ev \approx 37800 + 520000 = 557800$, i.e. $720000/557800 \approx 1{,}29$ times faster. If the DEA had at its disposal only $\#ev = 450000$, the speed-up would be $\approx 450000/287800 = 1{,}56$. At this point, we wish to draw attention to the fact that the difference in the quality of the results obtained after the $ev = 720000$ and $ev = 450000$ is only 0,48%, [see Tab. 2]. This fact can be seen as an opportunity to shorten the response time of the algorithm. If a loss of 0,48% on the quality of the results can be tolerated, then carrying out only $ev = 450000$ instead of $ev = 720000$, could shorten the response time of the algorithm $720000/450000 = 1{,}6$ times.

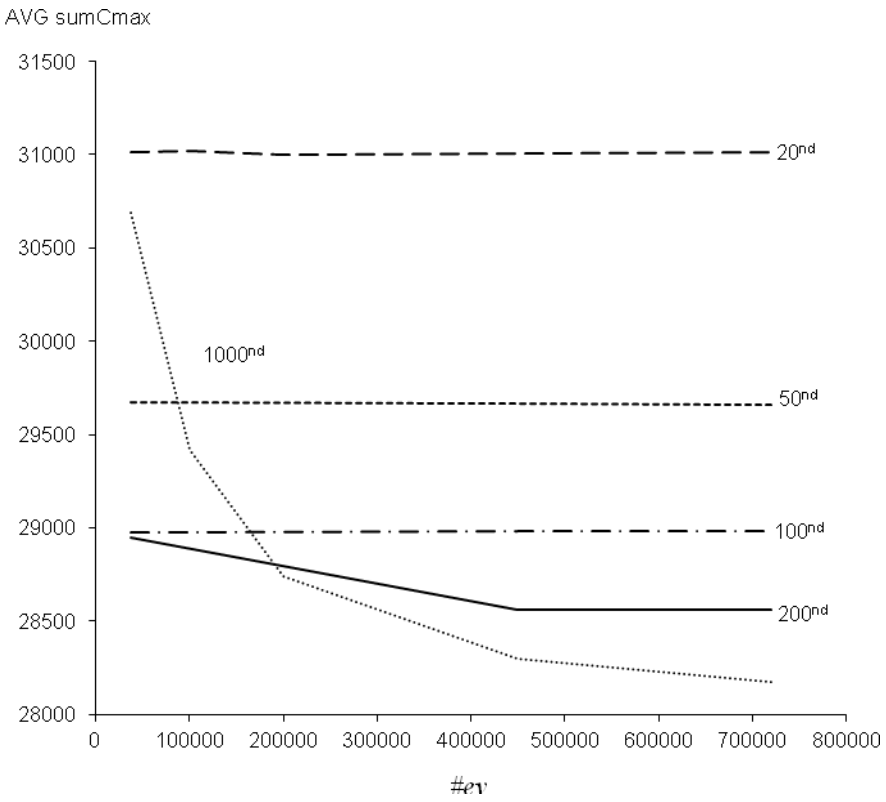

Figure 11: AVG $sumC_{max}$ of the DEA without decloning, considered for different x_P and #ev

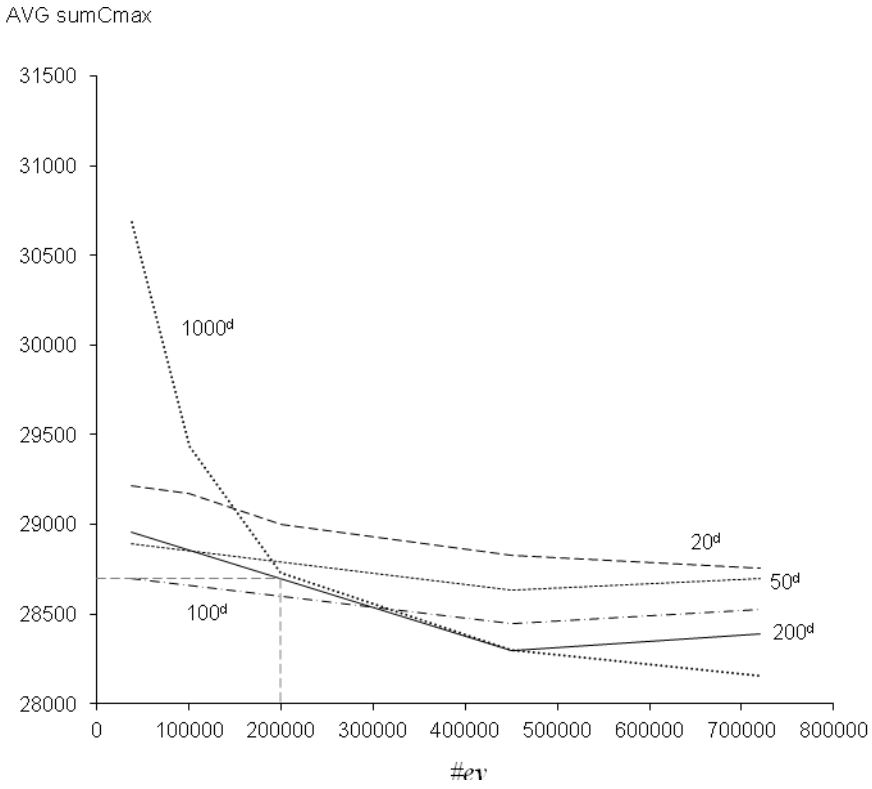

Figure 12: The effect of decloning on AVG sumC_{max}, considered for different x_P and #ev

If we additionally apply the proposed policy to the case *ev* = 450000, this would shorten the response time even 720000/287800 ≈ 2,5 times. However, applying the policy to the case with *#ev* = 720000 as well, would reduce the speed-up to 557800 / 287800 ≈ 1,9 times. It follows from the above discussion, that the final choice of the values of *#ev* and, therefore, x_P should allow to meet someone's expectations as to the quality of results and the response time of algorithm.

Thus, to summarize the above discussion, the performance improvement policy would consist of determining the intervals of *#ev* and corresponding most advantageous x_P such, that the desired quality of the results and response time of the algorithm is assured. The approximate intervals of #ev and x_P can be determined using the approach as in [Tab. 2] and [Fig. 12], which has been described earlier. At this stage of research, validity of the proposed policy should be proved experimentally in each case.

6 Conclusions

In the paper, we investigated the extent to which performance of the considered differential evolution algorithm - the DEA depends on such parameters as the population diversification rate, the size of the population, and the number of fitness function evaluations to yield a solution. In the experiments, the most advantageous values of these parameters, in terms of the algorithm's performance, have been determined, and the improvement policy was proposed. Population diversification was carried out cyclically using the proposed decloning procedure. As the test problem, the discrete-continuous scheduling problem with continuous resource discretisation was used.

The obtained results allowed us to propose a performance improvement policy that might noteworthy improve both the efficacy and response time of the DEA. The idea is to choose the diversification rate, the population size and the number of fitness function evaluations to yield a solution using [Tab. 1 and 2], given in [Sections 5.3.1 and 5.4] respectively. This might ensure the expected quality of the results and the response time of the algorithm.

The results of our experiments show that the diversification of the population can be preserved in an intensive manner, i.e. using dedicated diversifying mechanisms and procedures, e.g. decloning, or extensive one, by increasing the size of the population. The choice of how to preserve the diversification may depend on the restrictions imposed on the population size, response time, and quality of solutions that should be met by a specific implementation of the algorithm.

Our population diversification technique differs from the one proposed in [Kureichick, 96] since it uses packing catastrophic operator for replacing genetic, not fitness duplicates. Also, unlike ROG technique proposed in [Rocha, 99], it prevents transition of clones to the next generation due to their fitness, which is allowed in ROG, and unlike in $(\mu + 1)$ EA with genotype diversity proposed in [Friedrich, 09], a new random individual is introduced every time when a clone is identified, which is not carried out in $(\mu + 1)$ EA. Finally, decloning, i.e. population diversification, can be carried out less frequently than every generation, which is the case in ROG and $(\mu + 1)$ EA.

Our experiments also show that if a compromise between the quality of the results and the response time of the algorithm can be allowed, then it is possible to significantly reduce the response time using the performance improvement policy proposed by us while only slightly losing on the quality of the results, e.g. if one can accept a deterioration in the quality of results by 0,48%, then the response time of the algorithm might be reduced, depending on the assumptions made, from 1,29 to 2,5 times, see [Section 5.4].

The main conclusion which results from our research is that the performance of algorithm does not depend on any single factor considered in the paper, but from all of them, combined together, i.e. is a function of several arguments rather than one. Only the selection of appropriate values of these arguments could meet the expectations as to the effectiveness and the response time of the algorithm. An attempt to mathematically describe how the performance depends on the factors examined in the paper might lay foundations for future work.

References

[Bartusch, 88] Bartusch, M., Rolf, H. M., Radermacher, F. J.: Scheduling Project Networks with Resource Constraints and Time Windows, Annals of Operations Research, 16 (1988), 201-240.

[Damak, 09] Damak, N., Jarboui, B., Siarry, P., Loukil, T.: Differential Evolution for Solving Multi-Mode Resource-Constrained Project Scheduling Problems, Computers & Operations Research, 36, 9 (2009), 2653-2659.

[Fogel, 94] Fogel, D. B.: An Introduction to Simulated Evolutionary Optimization, IEEE Transactions on Neural Networks, 5, 1 (1994), 3-14.

[Friedrich, 09] Friedrich, T., Oliveto, P. S., Sudholt, D., Witt, C.: Analysis of Diversity-Preserving Mechanisms for Global Exploration, Evolutionary Computation, 17, 4 (2009), 455-476.

[Gupta, 12] Gupta, D., Ghafir, S.: An overview of methods maintaining diversity in genetic algorithms, International Journal of Emerging Technology and Advanced Engineering, 2, 5 (2012), www.ijetae.com

[Jędrzejowicz, 14] Jędrzejowicz, P., Skakovski, A.: Island-based Differential Evolution Algorithm for the Discrete-continuous Scheduling with Continuous Resource Discretisation, Procedia Computer Science, 35 (2014), 111–117.

[Józefowska, 98] Józefowska, J., Węglarz, J.: On a methodology for discrete-continuous scheduling, European Journal of Operational Research, 107, 2 (1998), 338-353.

[Kureichick, 96] Kureichick, V. M., Melikhov, A. N., Miaghick, V. V., Savelev, O. V., Topchy, A. P.: Some New Features in the Genetic Solution of the Traveling Salesman Problem, Proc. ACEDC'96, Plymouth (1996).

[Oliveto, 15] Oliveto, P. S., Zarges, C.: Analysis of diversity mechanisms for optimisation in dynamic environments with low frequencies of change, Theoretical Computer Science, 561, A (2015), 37-56.

[Pandey, 14] Pandey, H. M., Chaudharyb, A., Mehrotra, D.: A comparative review of approaches to prevent premature convergence in GA, Applied Soft Computing, 24 (2014), 1047–1077.

[Rocha, 99] Rocha, M., Neves, J.: Preventing Premature Convergence to Local Optima in Genetic Algorithms via Random Offspring Generation, LNAI (Lecture Notes in Artificial Intelligence), 1611 (1999), 127-136.

[Różycki, 00] Różycki, R.: Zastosowanie algorytmu genetycznego do rozwiązywania dyskretno-ciągłych problemów szeregowania, PhD diss, Poznań University of Technology, Poland (2000).

[Storch, 04] Storch, T., Wegener, I.: Real Royal Road Functions for Constant Population Size, Theoretical Computer Science, 320, 1 (2004): 123–134.

[Storn, 97] Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, 11, 4 (1997), 341-359.