Expert Systems With Applications 126 (2019) 308–320

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

An island-based differential evolution algorithm with the multi-size populations

Aleksander Skakovski*, Piotr Jędrzejowicz

Department of Information Systems, Gdynia Maritime University, Morska 83, 81-225 Gdynia, Poland

ARTICLE INFO

Article history:
Received 14 August 2018
Revised 19 February 2019
Accepted 20 February 2019
Available online 21 February 2019

Keywords:
Evolutionary computation
Island model
Multi-size populations
Differential evolution
Discrete-continuous scheduling

ABSTRACT

Computational intelligence methods can provide high-quality solutions to a variety of complex optimization problems where exact analytical solutions are impossible to obtain within a reasonable time and other resources used. The article proposes a novel concept of island model with islands of different sizes as well as differential evolution algorithms implementing this concept. Such a multi-size approach facilitates the design of island-based algorithms and brings a variety of benefits. Among them: improved fitness dynamics throughout the entire time of operation even without migration of solutions among the islands. The absence of migration eliminates the need to establish the topology and the policy of migration. It also makes the efficiency of multi-size island-based algorithms independent of the particular islands' size and practically eliminates the need of tuning the size of islands which is usually done in the case of the canonical island model. All these features indicate the superiority of the proposed multi-size island model over the canonical one. The efficiency of the proposed multi-size approach has been tested by solving one of the most difficult scheduling problems which is the discrete-continuous scheduling with continuous resource discretization.

1. Introduction

Computational intelligence involves a set of nature-inspired methodologies and approaches addressing complex real-world problems. Complexity and inherent uncertainty of such problems often make traditional mathematical modeling and analysis useless or not effective. Based on processes of natural selection, evolutionary algorithms are among important computational intelligence methodologies. Evolutionary algorithms can be seen as problem solvers or expert systems producing high-quality solutions to a variety of difficult and complex problems. In this paper, we discuss in depth one of the evolutionary algorithm kind, known as the island-based differential evolution algorithm.

The efficiency of the evolutionary algorithm (EA) to a great extent depends on its ability to cope with the obstacles that prevent or hinder the progress of the search. One of these obstacles is premature convergence, which results in getting stuck in a local optimum. Another one is a dimensionality curse, which is a natural obstacle on the way to finding the global optimum. One of the ways to cope with these difficulties is to maintain the diversity of the population at the appropriate level. In the literature, we find a variety of approaches and techniques designed for this purpose, see Pandey, Chaudharyb, and Mehrotra (2014). One of the simplest ways to diversify the population is to set an appropriate size of the population or perform the search in various distant regions separated from each other. In the latter case, performing the search in different regions of the search area might contribute to faster convergence towards the global optimum. This method of performing the search was the subject of interest of many researchers e.g., Alba and Troya (1999), Belding (1995), Cantú-Paz (2001), Cantú-Paz and Goldberg (2003). In the literature, the approach is known as the island model and is often reported as more effective, than the search performed on a single population, e.g., Mühlenbein (1991), Whitley and Starkweather (1990). In the canonical island model, the whole population of individuals is assumed to be divided into subpopulations of identical sizes. The island is created by associating a copy of the evolutionary algorithm at hand with each sub-population of the model. During the process of evolution, islands operate autonomously, cyclically exchanging solutions among each other. Such a process is called migration of individuals and is carried out in accordance with the interconnection topology and adopted migration policy. The migration policy specifies such details of migration as the frequency of migration, number of migrants, and the way in which migrants are selected from the source and introduced into the target population. Such a model is well suited for processing in parallel in distributed or

* Corresponding author.
E-mail addresses: askakow@am.gdynia.pl, manoadr@wp.pl (A. Skakovski), pj@am.gdynia.pl (P. Jędrzejowicz).

https://doi.org/10.1016/j.eswa.2019.02.027
0957-4174/

agent systems which may result in a more effective search and reduction of the response time. These benefits, however, impose on the designer the burden of multiple experiments required, during the fine-tuning phase, to determine the structure of the algorithm and the migration policy of the solutions. This might include determining the value of such parameters and factors as the number of islands, the size of the population on the island, the topology of connections between the islands, and migration policy which all together are expected to ensure the highest possible efficiency of the algorithm.

The motivation for this work was the idea to propose an approach that would provide competitive results as compared with the canonical approach and, at the same time, release the designer from most of the onerous tasks performed at the fine-tuning phase and facilitate the implementation of the island model. In order to achieve these goals, we propose an island model which is composed of islands of different sizes performing independently from each other without any communication among themselves. The idea is in several respects similar to one of sequential generating multiple populations of exponentially growing sizes proposed for the parameter-less GA in Harik and Lobo (1999), summarized in Section 2, however resulting in a better fitness dynamics and less demand for computational resources. The proposed model can achieve efficiency which can be very close to the maximum size-dependent efficiency of the algorithm operating on islands (the meaning of this term we explain in the next paragraph). It takes advantage of different convergence rates of algorithms operating on populations of different sizes and provides better evolution dynamics than that of the canonical island model or any single population algorithm used in the model.

The term *the maximum size-dependent efficiency of the algorithm* (MSDEA) which we will be using in the article requires a special explanation. We use the MSDEA to refer to the efficiency of evolutionary algorithms considered in the paper. It addresses the influence of the population size on the algorithm's efficiency *ceteris paribus*. The way how the MSDEA (or an upper bound for the size-dependent efficiency) can be determined, we explain as follows. Let us consider a collection of numerous programs each of which simultaneously and independently executes the same evolutionary algorithm, however, over populations of different sizes. Because sizes of the populations differ, the identical algorithms, operating on them, converge and perform also differently in different moments of time. The output of such collection at each moment of its operation is determined by a single solution which is the current best solution yielded by one of the programs from the collection. Thereby, the MSDEA is a hypothetical maximum efficiency which is determined at each moment of the operation of the algorithm used in the programs by the curve built on the current maximum values of the fitness function obtained by the programs from the collection. In other words, it is the efficiency of some hypothetical ideal algorithm operating over a population of a size which is optimal, with respect to the efficiency, at every moment of its operation. No single program from this collection can completely retrace such curve, if executed alone, regardless of the size of the population on which it would operate. At most, if at all, it can only retrace a part of the curve. An example of very close to the MSDEA curve represented by the minimized fitness function values is shown in Fig. 2 as red dashed line and in Fig. 4 as curve composed of multiple segments.

The main contribution of the paper is proposing the multi-size island model which improves the existing solutions in terms of efficiency and resource usage. We experimentally prove the superiority of the proposed model over the canonical one, showing that the approach, in addition to improved fitness dynamics and reduced demand for computational resources, manifests itself in the easier design of the island algorithm. Another important part of our contribution is the proposal of the concept of active islands, which allows reducing the number of islands operating simultaneously to 3 or even 2 islands, without a significant deterioration in the performance of the multi-size island algorithm. In addition, we investigated the possibility of applying the multi-size concept also to a single population DE algorithm to vary the size of its population. The details on the proposed multi-size island model, the concept of active islands, and the algorithms implementing it are provided in Section 5.

The proposed algorithms were implemented and validated in terms of their efficiency. Our experiments showed that the proposed multi-size approach to island model makes the effectiveness of the island-based algorithm independent from the particular population size which is identical for all islands in the canonical island model. It also eliminates the need to fine-tune the population size of identical islands to ensure high efficiency. Other important advantages of this approach include the elimination of the need of choosing topology of the migration and the need to establish a migration policy. Moreover, the proposed multi-size island-based algorithms are easier to implement and achieve efficiency nearly identical to the maximum size-dependent efficiency throughout the entire time of their operation. This property makes them superior over the canonical island-based algorithms which achieve their maximum size-dependent efficiency, if at all, only at certain stages of their operation. All of these features together indicate a superiority of the proposed multi-size approach over the single-population and the multi-population algorithms based on the canonical island model.

In the presented research, a special method of evolutionary search, namely the method of differential evolution, was used to design and validate the proposed multi-size population algorithms. Differential evolution (DE), is a stochastic direct search and a global optimization method described in Storn and Price (1997). Since then, this method has been intensively studied and widely used in science and engineering, decision making in business, economy and practical implementation in industry. Algorithms based on the DE are efficient and able to solve complex single and multi-objective optimization problems, including image processing, big data analytics, language processing, pattern recognition and others. It was only in the years 2017 and 2018 that more than a few thousand scientific articles were published with the proposal of new applications of the DE method. In periodically appearing surveys, the improvements and developments that have been made to DE algorithms are presented, e.g., Javaid (2019), Eltaeib and Mahmood (2018), Jebaraj et al. (2017), Das, Mullick and Suganthan (2016). Other surveys address parameter settings which are crucial for the performance the DE algorithms. The review (Piotrowski, 2017) deals with the impact of the population size on the performance of DE algorithms. In Dragoi and Dafinescu (2016), the replacement of manual control parameter setting with adaptive and self-adaptive methods and hybridization with other algorithms were considered. Due to its efficiency and high popularity, we will use the DE method in our research. We will use a classical scheme of the DE search enhanced with a decloning procedure, which cyclically replaces clones appearing in the population with new individuals. This procedure, was introduced in Jędrzejowicz and Skakovski (2016a) and used to design the DE algorithm with decloning (DEA^d). The DEA^d along with its island-based version ($IBDEA^d$) were used as the base for designing multi-size population algorithms proposed in this paper.

The rest of the paper is organized as follows. In Section 2, the review of research on the canonical island model and methods for population size management is provided. In Section 3, the discrete-continuous scheduling problem with continuous resource discretization (DCSPwCRD) used as a test problem is formulated. In Section 4, two algorithms: the DEA^d - a single popu-

lation DE algorithm with decloning and the IBDEA^{md} - an island-based DE algorithm implementing the canonical island model with islands of equal sizes, migration and decloning are described. In Section 5, the idea of the multi-size populations is proposed. Based on this idea, two island-based algorithms with a different number of active islands and a single multi-size population DE algorithm with decloning are proposed. Section 6, contains the description of the computational experiment as well as a discussion on the results. Section 7 includes the conclusion and an idea for future research.

2. Related work

In this Section, we make a brief review of the research on the canonical island model and population size management in evolutionary algorithms. At the end of the review, we discuss in detail a very similar to our model of multiple populations of different sizes proposed in Harik and Lobo (1999) and point out the differences, which make our model superior over the discussed one.

Most of the research on the island model of computation concerns the canonical model with islands of identical size, which cyclically exchange solutions among themselves according to some chosen topology and migration policy. The idea of increasing the population diversity by structuring it as a set of the autonomous and interacting with each other sub-populations was borrowed from the mathematical theory of population genetics, where isolation by distance theory, pioneered in Wright (1931, 1943), gave rise to different models of the population structure. The island model is viewed in evolutionary computation as an alternative to the single population evolutionary algorithm which may provide better efficiency when executed on a distributed system or also sequentially. It is known from the literature that most important factors of the island model are: island population size, number of islands (Cantú-Paz & Goldberg, 2003; Whitley, Rana, & Heckendorn, 1999) migration size and migration rate (Belding, 1995; Cantú-Paz, 2001; Tomassini, 2004; Krink, Mayoh, & Michalewicz, 1999), migration policy (Cantú-Paz, 2001) which can be performed synchronously or asynchronously (Alba & Troya, 1999; Hart, Baden, Belew, & Kohn, 1996) migration topology (Sekaj, 2004; Tanese, 1987), the heterogeneity of the island model and suitability for solving linearly separable and nonseparable functions (Whitley et al., 1999). Since all the above factors determine the efficiency of the island-based algorithms, the right choice of them is not an easy task and is a challenge for the researchers.

Only a few studies have investigated the effect of population size on the efficiency of the canonical island-based algorithm. In Whitley et al. (1999), the experiments with the number of islands and island population size were conducted. It was concluded in Whitley et al. (1999) that when migration is introduced, the efficiency of the island-based genetic algorithm was approaching or exceeding that of a single population. In Jędrzejowicz and Skakovski (2016b), larger scale experiments with a single population, canonical island model, and island model without migration DE algorithms applied to discrete-continuous scheduling showed that all considered models can be equally effective when used with proper parameter settings. Although these studies gave a clue about the "efficient" structures of the population, they still need to prove that these results are universal and will be valid for other evolutionary algorithms and problems. In order to release the designer from the obligation to experimentally tune the size of the population, methods were proposed, by virtue of which the algorithm could independently change and automatically adjust the size to the needs of the current state of evolution. In Berntsson and Tang (2005), a population adapter which implemented an adaptive method for determining good combinations of island population sizes and the number of islands was proposed. The adapter automatically searches for good settings of island population sizes and the number of islands according to observed efficiency during the run of the GA. The authors conclude, that although the population adapter requires about three times the effort needed to solve the problem when the optimal population sizing is known beforehand, it eliminates the process of manual tuning of the considered parameters of the DGAs.

Most of the research on population sizing concerned genetic algorithms operating on a single population. The size of population in these algorithms was changed based on: fitness dynamics (Eiben, Marchiori, & Valkó, 2004), remaining lifetime (RLT) of individuals (Arabas, Michalewicz, & Mulawka, 1994; Bäck, Eiben, & van der Vaart, 2000), RLT and fitness of population (Rajakumar & Aloysius George, 2013), the diversity of the population and special manually set parameters (Chop & Calvert, 2005), or also the size was changed at random every *#ev* of fitness evaluations (Costa, Tavares, & Rosa, 1999), or adjusted by GA itself using evolutionary operators, where the size was the part of evolved target solution (Eiben, Schut, & de Wilde, 2006; Teo, 2006), or changed cyclically in a saw-tooth manner with restart of nearly entire population at the end of each cycle (Koumousis & Katsaras, 2006). In Auger and Hansen (2005), a restart covariance matrix adaptation (CMA) evolution strategy with increasing population size was proposed. This strategy relies on repeated successive restarts of the algorithm each time on a population of doubled size. The restart is performed when the solutions found by the algorithm do not meet the conditions imposed on their quality, the standard deviation of the normal distribution, and the condition number of the covariance matrix. The influence of population sizing on the performance of GA and EAs was investigated using practical problems, e.g., identification of a cultivation process model in Roeva, Fidanova, and Paprzycki (2013) and design of water networks using Pseudo-Genetic Algorithm (PGA), Particle Swarm Optimization (PSO), and Harmony Search (HS) in Mora-Melià, Martínez-Solanob, Iglesias-Reyb and Gutiérrez-Bahamondesa (2017). Exhaustive experiments to determine the optimal population size of GAs and EAs considered in these papers have shown that population sizing is not a trivial task and is one of the important issues of parameter optimization.

In Zhang, Chen, Xin, Cai and Chen (2011), differential evolution with adaptive population size (DEAPS) combining lifetime and extinction mechanisms was proposed. In DEAPS the size of a single population is regulated based on individuals' lifetime and extinction mechanisms. The lifetime mechanism (the idea was borrowed from Arabas et al. (1994)), eliminates the individuals whose ages exceed allowed lifetime. The clones of the best individuals are inserted instead. The extinction mechanism is activated if the evolution of population advances no more. The new population is created based on competition of the "old" individual and a new randomly generated one. The best of them would enter the new population. The proposed approach showed better performance than the DE operating on a population of fixed size and other DE variants with adaptive population size.

The general idea to vary population size during the operation of the algorithm proved to be a success in all of the above examples of its application. In the vast majority of cases, GAs with variable population size performed better than or at least the same as canonical ones or those with fixed population size. The same situation can be observed in cases where population size was varied in parallel or distributed EAs.

In Smorodkina and Tauritz (2007), the greedy population sizing method for evolutionary algorithms (GPS-EA) has been proposed. It evolves two populations of changing sizes by carrying out interchangeably sequential and parallel phases of evolution. The advantage of GPS-EA is that it eliminates the need for manually tuning the population size parameter, still being capable of finding good solutions. However, the disadvantage of this algorithm is the need

to perform a sequential phase which might require twice as much fitness function evaluations as requires a single EA with an "optimal" population size.

In the case of the parallel GA, proposed in Hu, Harding and Banzhaf (2010), the size of a single shared population was changed in cycles of D ms in a saw-tooth manner. In a distributed GA, proposed in Jumonji, Chakraborty, Mabuchi and Matsuhara (2007), the number of tiny two-individual islands was changing in time and depended on the convergence of solutions on currently operating islands.

In Goldman and Punch (2014), a parameter-less population pyramid (P3) was proposed. P3 was built of populations containing something similar to different generations of evolution with more optimized solutions located in higher levels. The size of the population on each level as well as the number of levels, which are not known beforehand, grow along with the propagation of evolutionary process and are controlled implicitly by the algorithm itself. The size of population P_0 is increased each time when a solution, which is unique to the total population, is yielded. The size and the number of the rest of populations are increased each time when a solution, which is propagating up to the top of the pyramid, has been improved and is unique to the total population. The drawback of this successful idea might be high demand for computational resources since there is no strict control over the increase in the number of levels and the sizes of populations in them. Because population P_{i+1} can participate in crossover only after it has received a solution from P_i, it looks like P3 is best suited for sequential or pipeline processing, rather than for distributed processing.

In Arellano-Verdejo, Godoy-Calderon, Alonso-Pecina, Arenas and Cruz-Chavez (2017), a new efficient entropy population-merging island model for evolutionary algorithms (PMIM) was proposed. In contrast to the canonical island model, the merging of islands is used instead of the migration of solutions. Merging is used not only to maintain the diversity of islands, but also to reduce the number of islands and, therefore, the size of the total population. Two islands with the lowest Shanon entropy are selected for merging and the resulting island is composed of best individuals from both of them. This model brings the benefit of eliminating the need to choose the interconnection topology between the islands and reduces the number of fitness function evaluations with the decrease in execution time.

Finally, the most relevant to our approach is the parameter-less GA (PLGA), introduced in Harik and Lobo (1999). The core idea is to run multiple populations of various sizes simultaneously and establish a race among them. The PLGA starts with a single small population with index 1 and then, after a fixed number of generations have been evolved, creates a new double-sized population with index 2. The moments of creating new populations or evolution of existing ones are determined by the moments of increment of a counter of base 4. The scheme of the operation of the PLGA can be described using the indexes of populations being evolved or created as follows: 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 3, 1, … The size of each newly created population is twice as large as the size of the population with the preceding index. According to this scheme, population i carries out twice the number of fitness function evaluations of population $i+1$. When the average fitness of a population is less than the average fitness of a larger population, then the smaller population is removed, and the counter is reset. It is assumed that the PLGA stops if the computer runs out of memory, or it is stopped by the user when the PLGA yields the solution of the desired quality. The PLGA can be executed sequentially or in parallel. The approach has, however, certain drawbacks. The first one is a risk of long computation times. Depending on the test problem, the PLGA requires 1-3 times more fitness evaluations as compared with the regular GA to yield a target solution. It was reported in Eiben et al. (2004), that the PLGA was much slower than the regular GA. Another drawback is low efficiency since populations of large and small sizes are not initiated at the same moment.

Since the idea introduced in PLGA is similar to ours, we will point out the differences which make our model competitive. Firstly, the most important difference is that our model is a collection of populations of different sizes which “exist” in the system and are introduced by design instead of being created on-the-fly. In our case, all populations start at the same moment to evolve concurrently. Starting evolution of all populations at the same moment eliminates delays in the evolution of larger populations and, in consequence, results in a higher efficiency which may be very close to the MSDEA. Such efficiency can never be achieved by the algorithm proposed in Harik and Lobo (1999) precisely because of the delayed initiation of larger populations. Secondly, our algorithms offer the possibility of controlling the use of computational resources, because the maximum number of populations being evolved is defined by the user and decreases along with the evolution process whereas in PLGA the number of populations is unknown. Thirdly, in our case, the number of populations being simultaneously evolved can be reduced, depending on the version of the algorithm, to a small number without a significant change in performance. All these differences make our model more attractive in terms of efficiency and the use of computational resources.

We have decided to use the discrete-continuous scheduling problem (DCSP) as a test-bed for our approach. There are several reasons for such a decision. DCSP is known to be one of the hardest problems in the scheduling practice (Józefowska & Węglarz, 1998). It has several important practical applications including scheduling production processes (Harjunkoski et al., 2014), chemical production processes (Lee & Maravelias 2018a; Lee & Maravelias 2018b) or processes with tasks requiring energy supply (Różycki & Węglarz 2015; Różycki & Węglarz 2018). One of the effective approaches to DCSP is discretization of the continuous resources required. The Discrete-Continuous Scheduling Problem with Continuous Resources Discretization (DCSPwCRD) was introduced in Józefowska, Mika, Różycki, Waligóra, and Węglarz (2000). Discretization of the continuous resources makes possible using metaheuristic algorithms to solve instances of the DCSPwCRD. Good quality solutions were obtained by applying tabu search and simulated annealing algorithms (Józefowska et al., 2000; Józefowska, Mika, Różycki, Waligóra, & Węglarz, 2001). Another possible approach to solving the discrete-continuous scheduling problems via the discretization of the continuous variables was discussed in Lee and Maravelias (2018a). The authors used the MIP solver to address industrial-scale problems. Such an approach allows obtaining solutions in a short time. Unfortunately, according to the above authors, the approach has one disadvantage, namely, the poor accuracy of the obtained solutions.

3. Test problem formulation

The proposed approach will be validated experimentally in the following Sections using one of the most computationally difficult optimization problems. It is the discrete-continuous scheduling problem with continuous resource discretization (DCSPwCRD). In this Section, we offer its formulation. The problem is denoted as Θ_Z and formulated in the same way as in Różycki (2000). Namely, let $\boldsymbol{J} = \{J_1, J_2, \ldots, J_n\}$ be a set of nonpreemtable tasks, with no precedence relations and release dates $r_i = 0$, $i = 1, 2, \ldots, n$, 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 $l_i = 1, 2, \ldots, W_i$ (W_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 $u_i^{l_i} = U/l_i$ known in advance. The processing mode

l_i cannot change during the processing of J_i. For each task two vectors are defined: a processing times vector $\boldsymbol{\tau}_i = [\tau_i^1, \tau_i^2, \ldots, \tau_i^{W_i}]$, where $\tau_i^{l_i}$ is the processing time of task J_i in mode $l_i = 1, 2, \ldots, W_i$ and a vector of additional resource quantities allocated in each processing mode $\mathbf{u}_i = [u_i^1, u_i^2, \ldots, u_i^{W_i}]$. The total amount of the continuous resource used by tasks J_i at any time t within a schedule cannot exceed U.

The goal 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 detailed discussion on problem Θ_Z and the DCSP might be found in Ratajczak-Ropel and Skakovski (2018), Różycki (2000). Our test problem is a particular case of a more general Multi-Mode Resource-Constrained Project Scheduling Problem (MMR-CPSP), which is known to be NP-hard (Bartusch, Rolf, & Radermacher, 1988).

4. The DEA^d and the IBDEA^{md}

In order to solve problem Θ_Z, the DEA^d and the IBDEA^{md} are used. In the case of the DEA^d, the DE search is performed on a single population, and, in the case of the IBDEA^{md} - on multiple autonomous sub-populations. The considered algorithms were used to carry out computational experiments which helped to reveal properties, owing to which more effective algorithms have been proposed. The remainder of this chapter contains the description of the DEA^d and the IBDEA^{md}.

4.1. The DEA^d - a single population DE algorithm with decloning

The DEA^d is a combination of the DEAnd and the Decloning Procedure (DP). The DP was used to improve the efficiency of the DEAnd by preserving the diversity of the population at some appropriate level so that the algorithm can work effectively. It cyclically replaces clones appearing in the population with new randomly generated solutions. The procedure does not remove all clones from the population, because clones are not harmful to the exploration, on the contrary, they are even desirable. What actually limits the exploration is their quantity. Too few clones - too slow convergence, too many clones - stagnation at the local optimum. Thus, the goal of the DP is to revive the population by replacing redundant clones with new solutions so that the algorithm can work effectively. The proposed DP 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, which is 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 identifying clones. 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 DP run multiple times varied within 13% range.

In this Section, only general descriptions of the DEAnd and the DEA^d are given.

In the DEAnd used in the DEA^d, the population P consists of two halves P^1 and P^2. The DEAnd performs DE search on population P^1 of target vectors S_{tg} in cycles c. In every cycle c, for every target vector S_{tg} in ${P^1}_c$, a trial vector T in ${P^2}_c$ is created. This way ${P^2}_c$ is filled with trial vectors T. The new population ${P^1}_{c+1}$ is created by selecting the best vectors from ${P^1}_c$ and ${P^2}_c$, and the next cycle $c+1$ begins. The general course of the DEAnd operation is given below.

Algorithm 1 DEAnd.

1. Assume the population of individuals P_c consists of two halves ${P^1}_c$ and ${P^2}_c$, i.e. $P_c = {P^1}_c + {P^2}_c$, and $|P_c| = 2 \cdot x_P$, $|{P^1}_c| = x_P$, $|{P^2}_c| = x_P$;
2. **for** every target vector S_{tg} in the current population ${P^1}_c$ **do**:
3. Create a mutant vector M from three vectors S_0, S_1, S_2 randomly chosen from ${P^1}_c$, using the equation: $M = S_0 + A^*r^*(S_1 - S_2)$, where $A > 0$ - is a scale factor, that controls the evolution rate of the population and $r \in [0, 1]$;
4. Create a trial vector T in ${P^2}_c$ applying the crossover operator to each element of mutant vector M and the corresponding element of target vector S_{tg} according to the rule:
5. **if** the random number $r \leq C_r$, $C_r \in [0, 1]$, then the trial element is inherited from mutant vector M, otherwise from target vector S_{tg}; **end if**;
6. **end for;**
7. Create a new population ${P^1}_{c+1}$ selecting the best vectors from ${P^1}_c$ and ${P^2}_c$;
8. Repeat steps 2 - 7 until the stop criterion is met;

End

In the DEAnd's population, the individuals are generated in such a way, that the position of a task in a solution S, as well as the task's processing mode, are chosen at random with the uniform distribution. The crossover constant C_r controls probability, that trial vector T will inherit the element either from target vector S_{tg}, or mutant vector M. The computational complexity of the DEAnd is determined by the complexity of the sorting algorithm (merge sort) which it uses and is $O(n \log n)$.

The general course of operation of the DEA^d is given below.

Algorithm 2 DEA^d.

1. Set the values of the parameters required to carry out the DEAnd;
2. Set the value of the period of decloning T^d, which is most advantageous for the size of the population being evolved;
3. Use the DEAnd to evolve the population. While carrying out the DEAnd, apply the decloning procedure in cycles determined by T^d to the population being evolved;

End

The value of the period of decloning T^d is given as the number of fitness function evaluations to be carried out or the number of generations to be created between the subsequent invocations of the decloning procedure. The value of the period of decloning T^d which is most advantageous for the size of the population x_P being evolved and the number of the fitness function evaluations $\#ev$ allowed had been determined experimentally. The decloning procedure used in DEA^d replaces genetically identical individuals (clones) with randomly generated ones in cycles determined by T^d. Although the decloning procedure does not remove all clones present in the population, however, it provides the appropriate level of diversification of solutions and improves the efficiency of the DEAnd. Since the decloning procedure does not increase the size of the population, the complexity of the DEA^d is $O(n \log n)$.

Graphs, showing the improvement of solutions yielded by the DEAnd and the DEA^d for different population sizes with respect to the number of fitness function evaluations $\#ev$ carried out, are

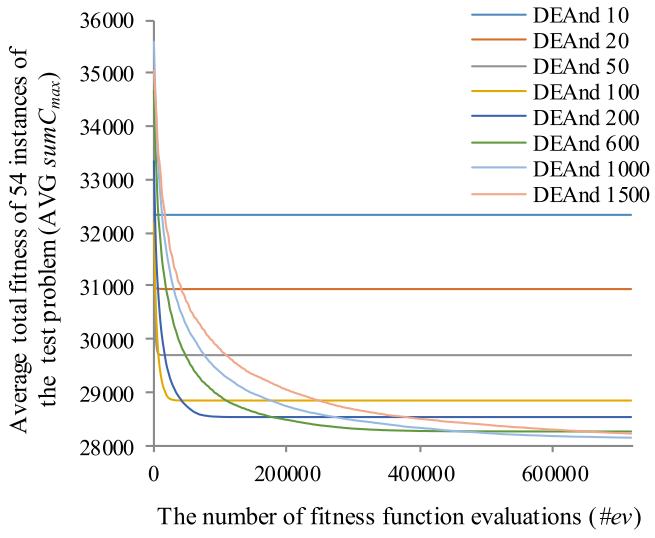

Fig. 1. AVG $sumC_{max}$ yielded by the DEAnd for different population sizes w.r.t. #*ev*.

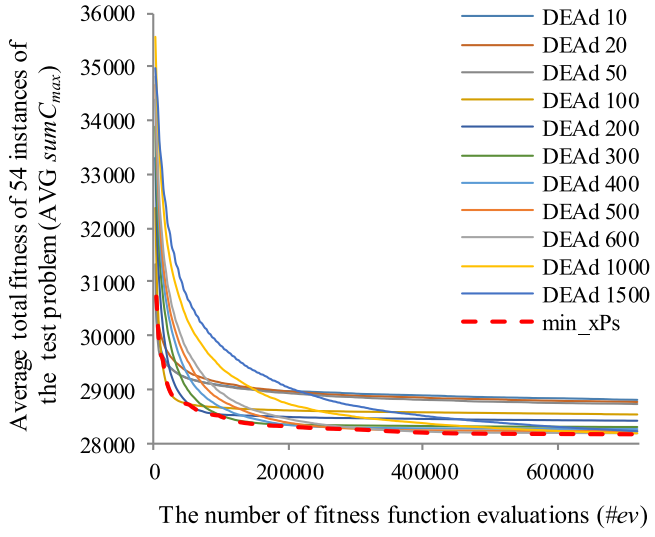

Fig. 2. AVG $sumC_{max}$ yielded by the DEA^{d} for different population sizes w.r.t. #*ev*. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

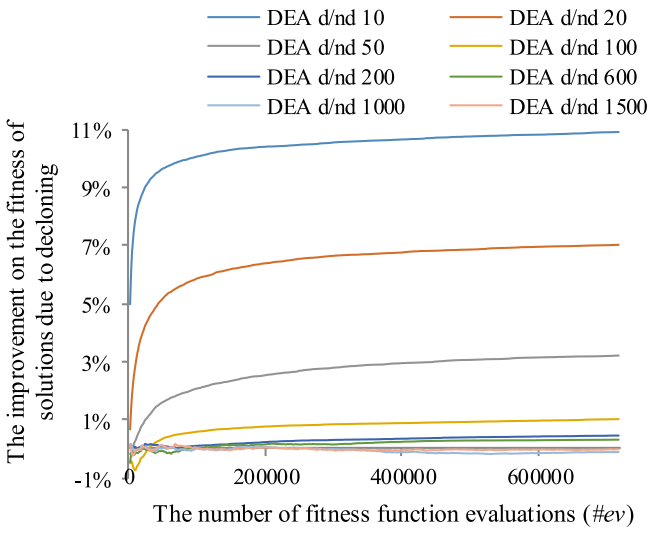

Fig. 3. The improvement on the fitness of solutions yielded by the DEA^{d} w.r.t. the solutions yielded by the DEAnd for different population sizes x_P, in percent.

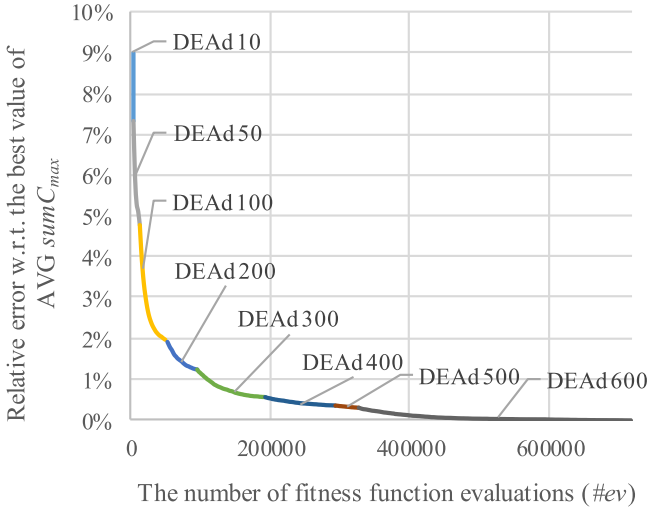

Fig. 4. A min_x_Ps curve showing relative error of x_Ps' minima (AVG*$sumC_{max}$) w.r.t. the best value of AVG $sumC_{max}$ across all x_Ps (AVG**$sumC_{max}$).

shown in Figs. 1 and 2, respectively. In these graphs, the quality of solutions is represented by the values of parameter AVG $sumC_{max}$ which is the average total fitness of 54 instances of the test problem (the details of AVG $sumC_{max}$ calculation are provided in Section 6.1). The quality of solutions is inversely related to the value of AVG $sumC_{max}$, i.e. the smaller the value of AVG $sumC_{max}$, the higher the quality of the solutions found.

The improvement ($RE^{Imp.}$) on the quality of solutions yielded by the DEA^{d} as compared with solutions yielded by the DEAnd for different population sizes x_P, is shown in percent in Fig. 3. The improvement was calculated according to (1):

$$RE^{Imp} = \left(Q - Q'\right)/Q, \tag{1}$$

where Q = AVG $sumC_{max}$ of the DEAnd and Q' = AVG $sumC_{max}$ of the DEA^{d}.

Fig. 3 shows, that the most significant effect of decloning on results improvement is observed when the DEA^{d} operates on small populations. For example, the curve labeled "$10^{d/nd}$", which stands for the population size $x_P = 10$, shows the greatest improvement effect of decloning among all considered population sizes. The results yielded by the DEA^{d} with $x_P = 10$ were on average about 5% to 11% better, than the results yielded by the DEAnd. The improvement effect of decloning, however, gradually decreases with the increase of the population size, resulting in no improvement at all when the DEA^{d} operates on large populations, e.g. $x_P = 1000$ and $x_P = 1500$. Another observation is drawn from Fig. 2, that none of the sizes x_P ensures the highest efficiency of the DEA^{d} at every moment of its operation. It can be observed in Fig. 2 that at the beginning of its operation, e.g. when #*ev* ≤ 100 000, the DEA^{d} performs better on small populations, i.e. finds a better solution in shorter time, than on large populations. The situation reverses along with the increase of #*ev*. The DEA^{d} finds better solutions when it operates longer on large populations. Thus, the curve which would determine the limit of the efficiency of the DEA^{d}, that can be achieved due to operating on populations of different sizes x_P, would copy the minima among all curves corresponding to particular sizes of populations at moment #*ev*. Such an x_Ps' minima curve (denoted as min_x_Ps) is shown in Fig. 2 as a red dashed line. It is also shown separately in Fig. 4 as the line of relative error (RE) of AVG $sumC_{max}$ values belonging to min_x_Ps curve (which we denoted as AVG*$sumC_{max}$) with regard to the best among all AVG

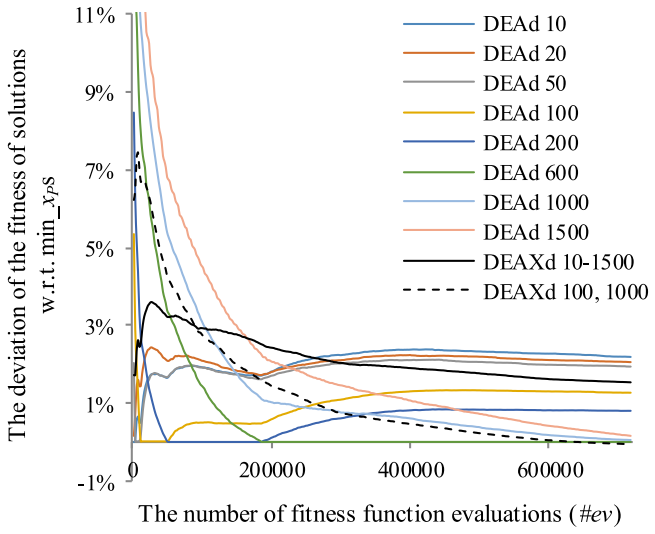

Fig. 5. The deviation (rel. error) of the fitness of solutions yielded by the DEA^{d} and DEA^{Xd} over populations of different sizes w.r.t. to min_x_Ps at moment *#ev*, in percent.

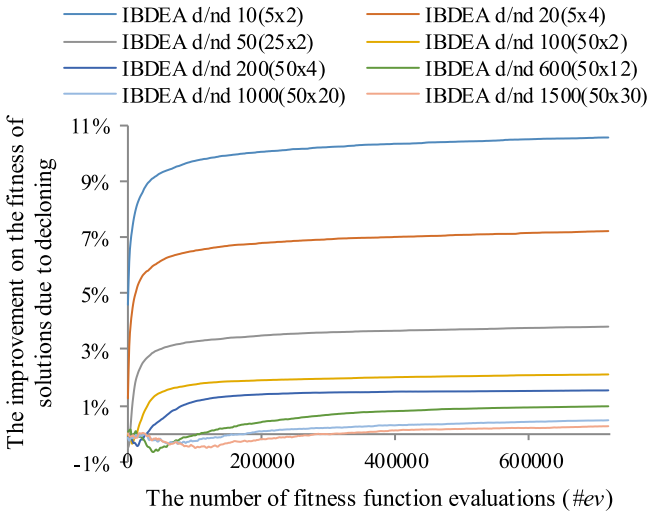

Fig. 6. The improvement on the fitness of solutions yielded by the IBDEA^{md} w.r.t. the solutions yielded by the IBDEA^{mnd} considered for different population sizes x_A and x_P, in percent.

$sumC_{max}$ values yielded by the DEA^{d} at the end of computing on all considered x_Ps (this the best value we denoted as AVG**$sumC_{max}$). The RE was calculated using (2):

$$RE = (Q - Q')/Q', \quad (2)$$

where Q = AVG*$sumC_{max}$ and Q' = AVG**$sumC_{max}$.

In Fig. 5, the RE of AVG $sumC_{max}$ values with respect to min_x_Ps is shown. In other words, this figure shows the deviation of the solutions found by the DEA^{d} while operating on populations of particular sizes x_P from the solutions situated on min_x_Ps curve at moment *#ev* in percent. It also indicates the ranges of *#ev* within which the DEA^{d} finds the best solutions among all ones obtained for considered population sizes. These ranges, in turn, indicate the most favorable population size on which the algorithm should work. For example, within the interval of *#ev* ∈ [52,000, 186,000] the DEA^{d} operating on the population of size x_P = 200 yields the best solutions among all ones obtained for considered population sizes. The value of RE was calculated according to (2), in which Q = AVG $sumC_{max}$ of the DEAnd and Q' = AVG*$sumC_{max}$. The meaning of curves labeled as "DEA^{Xd} 10–1500″ and "DEA^{Xd} 100–1000″ in Fig. 5 is explained in Section 6.4.

4.2. The IBDEA^{md} - an island-based DE algorithm with islands of equal sizes, migration, and decloning

The IBDEA^{md}, considered in this section, implements the canonical island model with the DEA^{d} operating on every island. The only difference between the algorithms is that the IBDEA^{md} uses the DEA^{d} instead of the DEAnd. The general course of operation of the IBDEA^{md} is given below.

Algorithm 3 IBDEA^{md}.

1. 1. Set $K \geq 2$ (K - the number of islands);
4. Assign DEA$^{d}_{k}$ (the kth copy of the DEA^{d}) to island k, $k = 1, 2, \ldots, K$;
5. Improve population on the island k with DEA$^{d}_{k}$, cyclically exchanging best individuals among randomly chosen pairs of islands;
6. Stop after max_*#ev* number of fitness function evaluations have been carried out on the archipelago;
7. Output the best solution on the archipelago;

End

The solution exchange among the islands occurs cyclically, after each ex << max_*#ev* of fitness function evaluations have been carried out on every island. The pairs of islands, chosen at random from the archipelago, exchange their best solutions between themselves. The random interconnection topology among islands was chosen as the most efficient one, which was determined by the experiment. The computational complexity of the IBDEA^{md} is the same as that of the DEA^{d} and is O (n log n). Fig. 6 shows the improvement on the fitness of solutions yielded by the IBDEA^{md} w.r.t. the solutions yielded by the IBDEA^{mnd} considered for different population sizes x_P in percent.

5. The concept of a multi-size population

This Section discusses the concept of a multi-size population, which can be applied both to a compound population consisting of the multiple sub-populations as well as to a single population. According to the concept, a set X_P of population sizes should be defined, where $X_P = \{x_{P1}, x_{P2}, \ldots, x_{PK}\}$, and $x_{Pk} < x_{Pk+1}$. In the case of a compound population, the total primary population should be decomposed into K sub-populations (islands) of different sizes $x_{Pk} \in X_P$. This partition of the primary population into sub-populations is fundamentally different from the partition used in the canonical island model where subpopulations of identical sizes are obtained. In the case of a single population, the size of the population should be increased over time from x_{P1} to x_{PK} according to some defined policy. In the following sub-sections, algorithms which implement the described concept are proposed. Dividing a population into the set of subpopulations of different sizes or designing a single population of variable size could be based on some strategy or could be set in an arbitrary manner. In the reported experiments we use the second option assuring a fair distribution of the different population sizes. This distribution of sizes x_{Pk}, $k = 1,\ldots, K$, can be reproduced by using the following approximation: $x_{Pk} \approx -1{,}3131k^3 + 22{,}608k^2 - 23{,}698k + 12{,}857$. However, in practice, it can be assumed for simplicity that $x_{Pk+1} \approx 2 \cdot x_{Pk}$.

5.1. The multi-size island model

5.1.1. The IBDEA$^{X\text{-}md}$ - a multi-size island-based DE algorithm with decloning and without migration

Based on the concept of multi-size population proposed above, there is a straightforward proposition for a multi-size island-based DE algorithm with islands of different sizes, decloning and without migration. The algorithm denoted as IBDEA$^{X\text{-}md}$ takes advantage of

the different convergence rates which are characteristic to different population sizes. The main idea of the IBDEA$^{X\text{-}md}$, is to create an archipelago of islands (sub-populations) of different sizes. The sub-populations are evolved independently by a copy of the DEA^{d} assigned to each island. There is no migration of solutions among the islands. The DEA^{d} stops on an island when either the best current solution yielded on this island is worse than the best current solution yielded on any other larger island, or *#ev* (the number of fitness function evaluations) carried out has reached its limit. Moreover, in the case of the distributed implementation of the IBDEA$^{X\text{-}md}$, when the DEA^{d} stops on a particular island a computing resource, on which the island has been implemented, should be released. The general course of operation of the IBDEA$^{X\text{-}md}$ is given below:

Algorithm 4 IBDEA$^{X\text{-}md}$.

1. Define the set of population sizes $X_P = \{x_{P1}, x_{P2}, \ldots, x_{PK}\}$;
2. Generate K populations of sizes $x_{Pk} \in X_P$, $k = 1, 2, \ldots, K$;
3. Create an archipelago of K islands, where an island is comprised by a population of size x_{Pk} and a copy of the DEA^{d} that will evolve this population;
4. Evolve population on island k until either the best current solution yielded on this island is worse than the best current solution yielded on any other larger island, or *#ev* has reached its limit;
5. Output the best solution among the remained islands;

End

The computational complexity of the IBDEA$^{X\text{-}md}$ is the same as that of the DEA^{d} and is $O(n \log n)$. The test results of the IBDEA$^{X\text{-}md}$ are presented and discussed in Section 6.2.

5.1.2. The IBDEA^{Xamd} - a multi-size island-based DE algorithm with active islands, migration, and decloning

The design of the IBDEA^{Xamd} is motivated by the idea of reducing the demand for computational resources required to run the previously proposed IBDEA$^{X\text{-}md}$ in a distributed system, still maintaining the highest possible efficiency. Although, the number of computing units is gradually decreasing towards the end of the IBDEA$^{X\text{-}md}$'s operation, however, K computing units to implement K islands are required to start it. In the case of the IBDEA^{Xamd}, implemented in a distributed system, the maximum number of computing units, required to start it, is reduced from K to K^a, where K^a – the number of active islands, $K^a \leq 3$, and gradually decreases to 1 towards the end of its operation. Although the proposed algorithm carries out a migration of solutions among islands, however, it is not performed in regular cycles, but only when active island with the smallest index is deactivated. The general course of operation of the IBDEA^{Xamd} is given below:

The computational complexity of the IBDEA^{Xamd} is the same as that of the DEA^{d}, i.e. $O(n \log n)$. The policies of using active islands and migration, as well as the values of parameters used in the computational experiment are provided in Section 6.3.

5.2. The DEA^{Xd} - a single multi-size population DE algorithm with decloning

Based on the results obtained for the DEA^{d} and the IBDEA^{md}, it is also reasonable to propose a single population DE algorithm which would take advantage of the different rates of convergence which are characteristic to different population sizes. For this purpose, a single multi-size population DE algorithm with decloning, denoted as DEA^{Xd}, was designed. The DEA^{Xd} gradually increases the size of the population at the moments when no further improvement of the fitness function is observed. The DEA^{Xd} uses DEA^{d} algorithm which is described in Section 4.1. The general course of operation of the DEA^{Xd} is presented below:

Algorithm 5 IBDEA^{Xamd}.

1. Define the set of population sizes $X_P = \{x_{P1}, x_{P2}, \ldots, x_{PK}\}$;
2. Generate K populations of sizes $x_{Pk} \in X_P$, $k = 1, 2, \ldots, K$;
3. Create an archipelago of K islands, where an island comprises a population of size x_{Pk} and a copy of the DEA^{d} that will evolve this population;
4. Define the policy of solution migration;
5. Define the policy of using active islands;
6. Set the number of active islands K^a according to the policy of using active islands;
7. Set values for max_*#ev*, period Δ*#ev*, and define a set of values *#ev*′ according to the policy of using active islands, where *#ev*′ determines the moment when K^a is to be changed;
8. Evolve populations on active islands K^a only, checking with period Δ*#ev*, whether the best solution on the active island with the smallest index is worse than the best solution at least on one from the rest of active islands. If so, then perform migration according to the defined policy, increase, if possible, the indexes of active islands by one, and continue in the same way, until *#ev*′ have been carried out on the archipelago;
9. Set a new value for *#ev*′ and K^a according to the policy of using active islands, and repeat steps 8 and 9 until max_*#ev* has been carried out on the archipelago;
10. Output the best solution among active islands;

End

Algorithm 6 DEA^{Xd}.

1. Define the set of population sizes $X_P = \{x_{P1}, x_{P2}, \ldots, x_{PK}\}$;
2. Generate a population of size x_{PK};
3. Set the maximum number of generations with no improvement $\max\_n_{gni}$;
4. Set $k = 1$;
5. Apply DEA^{d} to the first x_{Pk} of the x_{PK} individuals until the number of generations with no improvement exceeds $\max\_n_{gni}$, $x_{Pk} \in X_P$, $k = 1, 2, \ldots, K$;
6. Set $k = k + 1$, $k \leq K$;
7. Repeat steps 5 and 6 until the stop criterion is met;

End

The computational complexity of the DEA^{Xd} is the same as that of the DEA^{d}, i.e. $O(n \log n)$. The test results of the DEA^{Xd} are presented and discussed in Section 6.3.

6. Computational experiment

6.1. Parameter set-up

In the experiments, the values of the parameters of the DEAnd were assumed to be the same as in Damak, Jarboui, Siarry and Loukil (2009), namely, the scale factor A which controls the evolution rate of the population was set to $A = 1{,}5$ and values of the variable $r \in [0, 1]$. The crossover constants Cr_p and Cr_l which control the probability that the trial individual will receive the targets individual's tasks or modes were set $Cr_p = 0{,}2$ and $Cr_l = 0{,}1$, where p and l in symbols Cr_p and Cr_l stand for task's position and mode, respectively. The initial population of feasible individuals in the DEAnd was generated using the uniform distribution equal $1/n$ for the tasks, and $1/W$ for the task's modes. The assumptions concerning the test problem are as follows. Three combinations of n x m have been considered: 10×2, 10×3, and 20×2, where n is the number of tasks and m is the number of machines. Three levels of the continuous resource discretization W: 10, 20, 50 were considered. This way, nine sizes $n \times m \times W$ of the problem Θ_Z has been considered: $10 \times 2 \times 10$, $10 \times 2 \times 20$, $10 \times 2 \times 50$, $10 \times 3 \times 10$, …, $20 \times 2 \times 50$. For each of the sizes, we have considered 6 instances of the problem Θ_Z, which makes a test set of 54 instances of the problem in total. Problem instances used in all of the reported experiments are available by e-mail

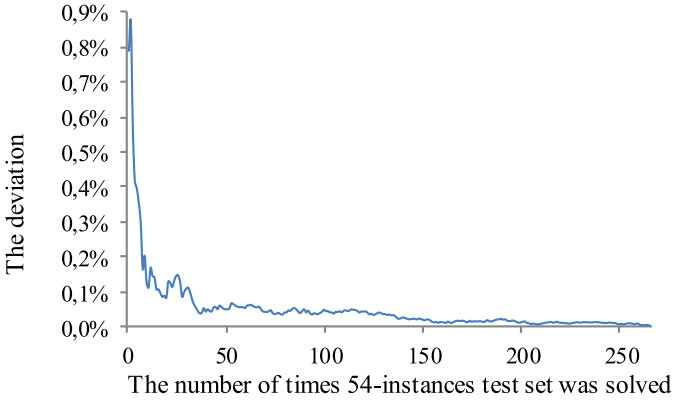

Fig. 7. The deviation of AVG $sumC_{max}$ of the 54-instances test set solved a given number of times from AVG $sumC_{max}$ of the set solved 266 times.

or at http://kpisk.am.gdynia.pl/as/Instances_of_DCSPwCRD/. In order to evaluate the efficiency of the considered algorithms, parameters $sumC_{max}$ and AVG $sumC_{max}$ were introduced. A single value of $sumC_{max}$ was calculated as the total of 54 C_{max} values obtained by solving the test set of 54 instances of the problem. To ensure the credibility of results, the test set of 54 instances was solved 10 times, which allowed to calculate

AVG $sumC_{max}$ as the average of 10 values of $sumC_{max}$. Such AVG $sumC_{max}$ differs from the average of 266 values of $sumC_{max}$ by only about 0,2%, which can be observed in Fig. 7. It was assumed that the deviation of 0,2% from the average of 266 does not prevent the correct evaluation of the results obtained. In the cases when higher precision was required, the test set of 54 instances was solved 100 times, which reduced the deviation to about 0,05%.

In most experiments, we considered the following population sizes: $X_p = \{10, 20, 50, 100, 200, 600, 1000, 1500\}$, and the number of the fitness function evaluations $\#ev$ available for the algorithms to yield a solution to the problem was set to $\#ev = 720{,}000$.

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 3.00 GHz, RAM 4 GB compiled with aid of Borland Turbo Delphi for Win32. When the number of fitness function evaluations was set to 720,000, mean time required by the DEAnd to find a solution for the problem sizes 10×2 and 10×3 for all discretization levels was approximately 2 – 3 s and for the problem size 20×2 for all discretization levels approximately 5 – 6 s. The total time taken by the DEAnd to process all 54 instances was approximately 206 s.

6.2. Experiment results - the IBDEA$^{X\text{-}md}$

In the experiment, sizes of populations on the islands of the IBDEA$^{X\text{-}md}$ have been determined as $X_p = \{10, 50, 100, 200, 300, 400, 500, 600\}$. These subpopulations were evolved by the algorithm independently and without migration of solutions among the islands. The efficiency of the IBDEA$^{X\text{-}md}$ is determined by the minima among all values of AVG $sumC_{max}$ yielded on all islands at each moment $\#ev$. These values formed the curve min_x_Ps shown as the red dashed line in Fig. 2, which is also shown separately in Fig. 4. Due to the multiple sizes of subpopulations, the IBDEA$^{X\text{-}md}$ finds solutions just as fast or faster and of the same or better quality than DEA^{d} executed alone with any of considered x_Ps at any moment $\#ev$. Thus, the multi-size island model makes the effectiveness of the IBDEA$^{X\text{-}md}$ independent of one specific population size and shows efficiency which is nearly identical to the maximum size-dependent one throughout the entire time of its operation. An important advantage of the IBDEA$^{X\text{-}md}$ is that in contrast to the DEA^{d} and the IBDEA^{md} there is no need to tune the population size in order to achieve higher efficiency. Moreover, because the algorithm does not use solution migration, there is no need to specify neither migration topology, nor migration policy, which in general is laborious and time-consuming. All of these advantages together indicate the superiority of the proposed multi-size approach over the single-population and the canonical island model algorithm. The advantage of the multi-size IBDEA$^{X\text{-}md}$ w.r.t. the canonical IBDEA^{md} shows Fig. 8. Since the IBDEA$^{X\text{-}md}$ is viewed as a collection of several DEA^{d}s operating over populations of different sizes, the advantage is represented in Fig. 8 as the relative error of solutions yielded by the IBDEA^{md} w.r.t. the curve min_x_Ps of the DEA^{d}. Although, generally the IBDEA$^{X\text{-}md}$ performs better than the IBDEA^{md}s, however over some sizes IBDEA^{md} yields slightly better results. To clarify the difference, we created Fig. 9, which shows the maximal difference that was observed in our experiments between fitness yielded by the multi-size IBDEA$^{X\text{-}md}$ w.r.t. the canonical IBDEA^{md} when the latter operated over different population sizes. Fig. 9 shows that at advanced stages of operation, where the quality of solutions is already quite high, the difference is only about 0,6%. The difference of about 1% at the initial stage of operation ($\#ev < 50{,}000$) can be ignored, because at this stage the solutions are not mature enough, since their quality is about 3%–18%

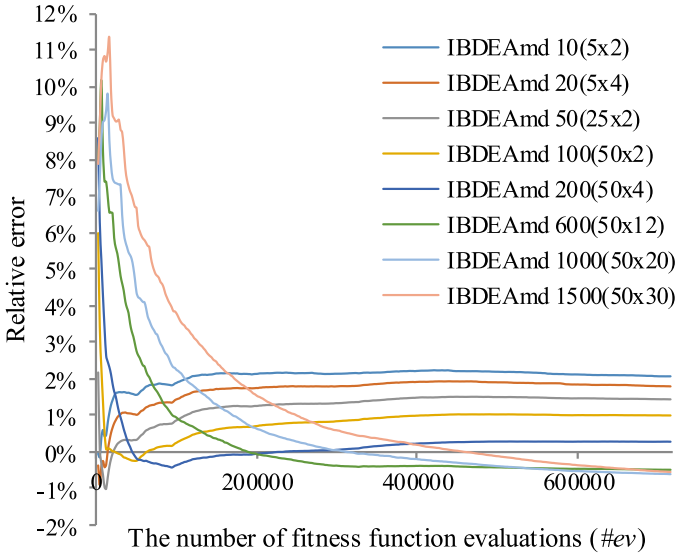

Fig. 8. The relative error of solutions yielded by the IBDEA^{md} with regard to the curve min_x_Ps of the DEA^{d} considered for different population sizes x_P, $x_A = x_P$, in percent.

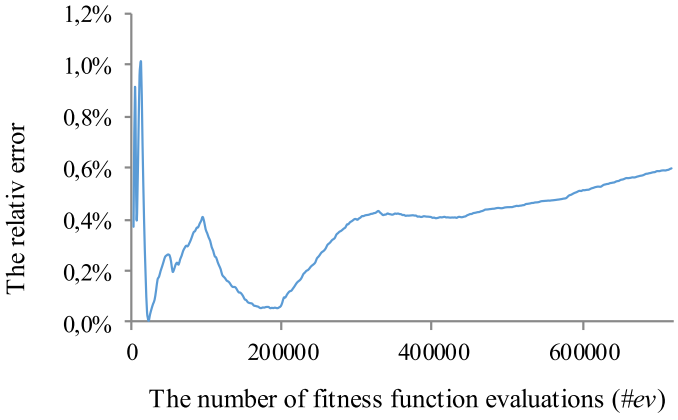

Fig. 9. The deviation (rel. error) of curve min_x_Ps of the DEA^{d} from curve min_x_As of the IBDEA^{md} in percent, $x_A = x_P$.

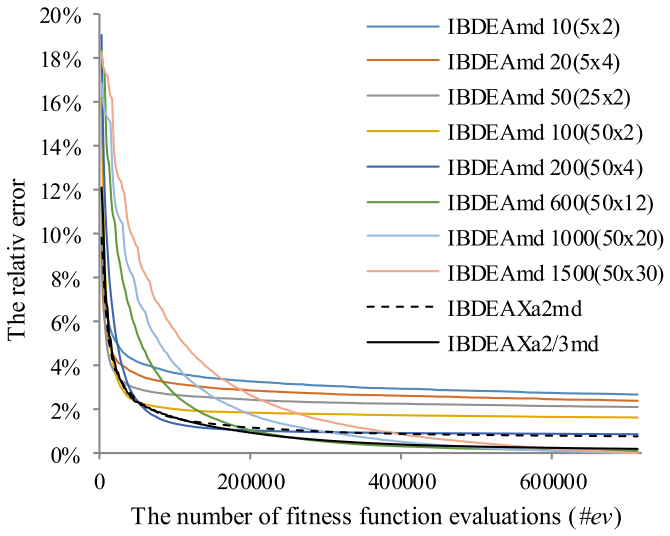

Fig. 10. The relative error of AVG $sumC_{max}$ values yielded by the IBDEA^{md}, the IBDEA^{Xa2md}, and the IBDEA$^{Xa2/3md}$ w.r.t. the best value AVG$^{**}sumC_{max}$, in percent.

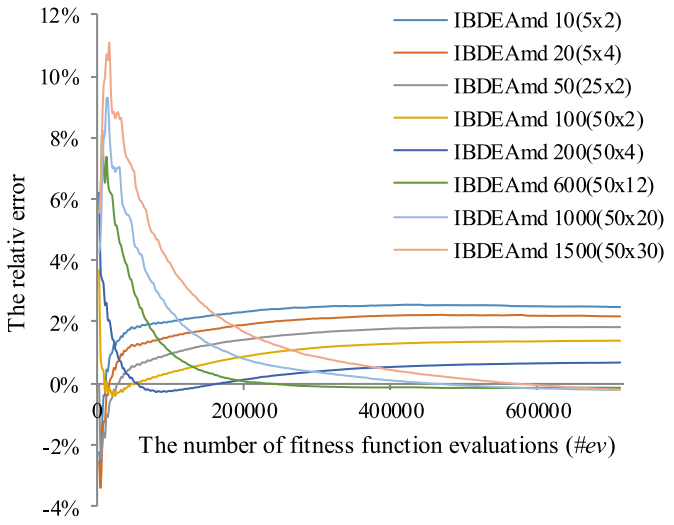

Fig. 11. The relative error of AVG $sumC_{max}$ yielded by the IBDEA^{md} run on populations of particular sizes w.r.t. AVG $sumC_{max}$ yielded by the IBDEA$^{Xa2/3md}$, in percent.

lower than in the final stage, see Fig. 10. To calculate the relative error shown in Fig. 9, the values min_x_As of the IBDEA^{md} were determined in the same way as min_x_Ps, i.e. as the minima among all AVG $sumC_{max}$ values which can be obtained while running simultaneously several IBDEA^{md}s, each one on the archipelago of one of the considered sizes x_A at moment #ev, where $x_A = x_P$.

Additionally, experiments to compare the performance of the proposed algorithm with some other method were carried out. For this purpose, an algorithm based on a tabu search (TS) method was chosen, since it is known in the literature as the one that yields good quality solutions (Józefowska et al., 2001). In order to evaluate the performance of both algorithms, rigorous parametric statistical Student's t-test and non-parametric Friedman test were carried out. Both algorithms were run on the same 54 problem instances 5 times. The tests were applied to 2 respective sets of 54 data each, where each data was a mean of 5 runs of the respective algorithm. The paired two sample for means parametric one-tailed Student's t-test was carried out using significance level $\alpha = 0{,}05$. The results of the t-test showed significant statistical difference between the IBDEA$^{X\text{-}md}$ and TS, since the obtained $p\text{-}value = 1{,}83875E\text{-}15$ is significantly less than $\alpha = 0{,}05$. Non-parametric Friedman test confirmed this conclusion resulting with the right tailed distribution of χ^2 equal 1,01656E-11 which is also significantly less than $\alpha = 0{,}05$ (also $\chi^2 = 46{,}2962963 > \chi^2{}_{critical} = 3{,}84146$). Moreover, the Kendall's coefficient of concordance (Kendall's W) was calculated. The Kendall's W was used for evaluating the degree of agreement among the obtained assessments. Kendall's W ranges from 0 (no agreement) to 1 (complete agreement). The intermediate values of W indicate a greater or lesser degree of unanimity among the various assessments. The value of $W = 0{,}857$, calculated for the results obtained in our Friedman test, indicates a high degree of unanimity among the assessments. We also carried out the comparison of relative errors: minimum (RE_{min}), average (RE_{avg}), and maximum (RE_{max}) obtained in 43 runs. The results of comparison testify in favor of the IBDEA$^{X\text{-}md}$. The considered relative errors were calculated with respect to the best known solutions and are compiled in Table 1, where minimal REs of all categories are given in bold font. In cases with negative RE_{min}, the IBDEA$^{X\text{-}md}$ improved the best known solutions. It can also be observed that in 17 cases out 27 the IBDEA$^{X\text{-}md}$ yielded smaller REs than that of TS's. And TS yielded the smallest REs only in 8 cases.

6.3. Experiment results – the IBDEA^{Xamd}

Experiments on the IBDEA^{Xamd} were carried out on its 19 versions with different policies of using active islands, policies of migration, and different parameter values. The algorithm demonstrated high efficiency when the policy of using active islands was to increase K^a from 2 to 3 at experimentally determined moment $\#ev' = 40{,}000$. The migration policy was to replace the worst solution on the last island on the archipelago K by the best solution from each active island, which was deactivated at the moment of increase of the active islands' indexes. This version of the IBDEA^{Xamd} will be referred to as the IBDEA$^{Xa2/3md}$. The values of the rest of the parameters were: $K = 4$, $X_P = \{50, 100, 200, 600\}$, $\#ev' = \{40{,}000\}$, $\Delta\#ev = 2400$, max_$\#ev = 720{,}000$. Results obtained by the IBDEA$^{Xa2/3md}$ were compared with the efficiency of the IBDEA^{md} implemented as the canonical island model. The IBDEA^{md} was run over populations of total sizes $x_A = \{10, 20, 50, 100, 200, 600, 1000, 1500\}$ composed of $K = \{2, 4, 12, 20, 30\}$ islands with sub-populations of sizes $x_P = \{5, 25, 50\}$. The migration rate of individuals among the islands $ex = \{1, 10, 11, 15\}$ and period of decloning $T^d = \{1, 2, 20\}$ have been adjusted to the size of the sub-populations on the islands x_P in order to ensure the highest possible efficiency of the IBDEA^{md}. The values of migration rate ex and the period of decloning T^d were adjusted by experiment. The advantage of the IBDEA$^{Xa2/3md}$ with respect to the IBDEA^{md} run over populations of different sizes can be observed in Figs. 10 and 11. Fig. 10 shows that throughout the entire time of its operation the results yielded by the IBDEA$^{Xa2/3md}$ are very close to the best among all values achieved at any moment by the IBDEA^{md} while operating on populations of different sizes. The REs of the algorithms were calculated with respect to the best overall value AVG$^{**}sumC_{max}$, and the black solid line represents the IBDEA$^{Xa2/3md}$. Fig. 11 specifies explicitly the advantage of the IBDEA$^{Xa2/3md}$ with respect to particular sizes over which the IBDEA^{md} was run. The relative error RE was calculated using (2) in which Q = AVG $sumC_{max}$ of the IBDEA^{md} and Q' = AVG $sumC_{max}$ of the IBDEA$^{Xa2/3md}$.

As can be seen in Fig. 11, the RE of the IBDEA^{md} with respect to the IBDEA$^{Xa2/3md}$ largely depends on two factors: the size of the population on which the IBDEA^{md} operates and the number of fitness function evaluations it has carried out. At the very beginning of its operation, the IBDEA^{md} gives better results than the

Table 1
Relative errors of the IBDEA$^{X\text{-}md}$ and TS w.r.t. to the best known solutions.

Problem size		Discretization level of continuous resource					
		$D=10$		$D=20$		$D=50$	
	REs	IBDEA$^{X\text{-}md}$	TS	IBDEA$^{X\text{-}md}$	TS	IBDEA$^{X\text{-}md}$	TS
$10\times2\times D$	RE_{min}	0,01%	0,01%	0,00%	0,00%	**−0,53%**	0,00%
	RE_{avg}	3,86%	**3,49%**	2,86%	**2,36%**	3,22%	**3,09%**
	RE_{max}	11,52%	**9,91%**	10,72%	**7,66%**	11,86%	**8,79%**
$10\times3\times D$	RE_{min}	**−0,10%**	0,07%	**−0,04%**	0,00%	**−1,12%**	0,06%
	RE_{avg}	**4,29%**	5,36%	**3,18%**	4,95%	**3,02%**	5,72%
	RE_{max}	**13,31%**	17,72%	**17,16%**	18,28%	**16,80%**	18,16%
$20\times2\times D$	RE_{min}	**0,33%**	0,68%	**−0,20%**	0,53%	**−0,03%**	1,05%
	RE_{avg}	**4,93%**	6,26%	**4,18%**	5,09%	**5,24%**	6,19%
	RE_{max}	**11,91%**	12,09%	11,37%	**10,81%**	13,22%	**12,65%**

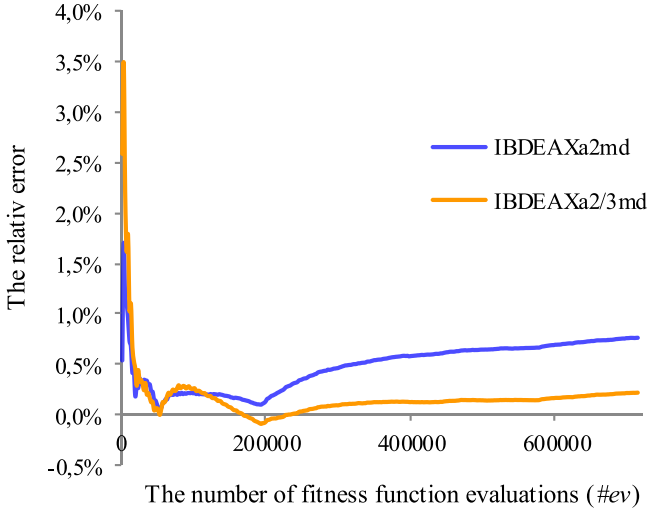

Fig. 12. The relative error of AVG $sumC_{max}$ yielded by the IBDEA^{Xa2md} and the IBDEA$^{Xa2/3md}$ w.r.t. min_x_As values (AVG*$sumC_{max}$), in percent.

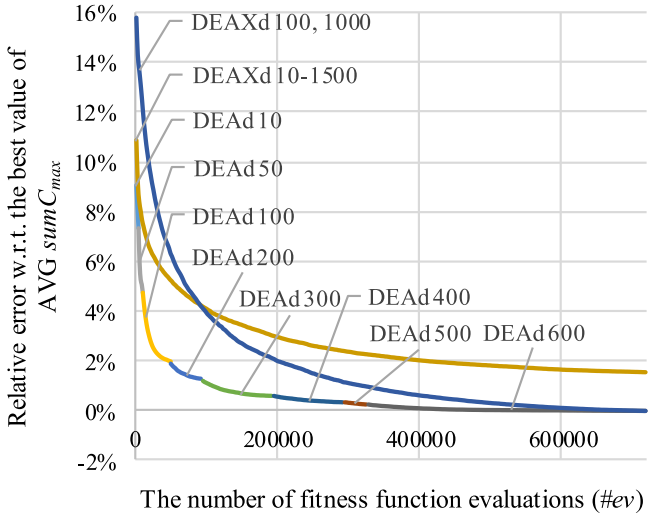

Fig. 13. The relative error of min_x_Ps (AVG*$sumC_{max}$) and of AVG $sumC_{max}$ values yielded by the DEA^{Xd} for the cases $X_P=\{10, 20, 50, 100, 200, 600, 1000, 1500\}$ (denoted as "DEA^{Xd} 10–1500") and $X'_P=\{100, 1000\}$ (denoted as "DEA^{Xd} 100, 1000") w.r.t. the best value AVG**$sumC_{max}$, in percent.

IBDEA$^{Xa2/3md}$ when it works on small populations, e.g. $x_P=\{10, 20, 50\}$. However, this advantage of the IBDEA^{md} is of no practical value, because, at this stage of operation the quality of results is about 9%−12% lower than the best yielded AVG**$sumC_{max}$, see Fig. 10. At the following stages of its operation, the situation reverses, and the results yielded by the IBDEA^{md} are about 1,5%−2,5% worse than those found by the IBDEA$^{Xa2/3md}$. When the IBDEA^{md} operates on larger populations, e.g. $x_P=\{100, 200, 600, 1000, 1500\}$, the situation is inverse to the one with small x_Ps. Namely, the RE is high at the beginning, about 3,5%−11%, and gradually decreases to about −0,2% with the advance of operating. The exception is the case with $x_P=\{100, 200\}$, when RE drops to about −0,3% at the beginning and slowly increases to about 1,35% and 0,65% respectively towards the end of operation. To sum up, although the IBDEA$^{Xa2/3md}$ uses fewer islands, its efficiency is very high and similar to that of the IBDEA$^{X\text{-}md}$. This conclusion confirms Fig. 12 which shows the relative error of AVG $sumC_{max}$ yielded by the IBDEA$^{Xa2/3md}$ with respect to min_x_As values (AVG*$sumC_{max}$), in percent. Although in the initial stage of operation when solutions are immature the RE is high, however in the further, more significant, stages the RE is reduced rapidly to at most 0,3%.

In another version of the IBDEA^{Xamd} (denoted as IBDEA^{Xa2md}) with efficiency very similar to that of the IBDEA$^{Xa2/3md}$, the policy of using active islands was to keep $K^a=2$ constant, until island $K-1$ was deactivated. The migration was carried out at the moments of increase of active islands' indexes by replacing the worst solution on the island just activated by the best solution from the deactivated one. The values of the rest of the parameters were: $K=5$, $X_P=\{10, 50, 100, 200, 600\}$, $\Delta\#ev=1800$, and max_$\#ev=720{,}000$. Although the IBDEA^{Xa2md} needs only two active islands to operate, it performs very similarly to the IBDEA$^{Xa2/3md}$. The relative error of AVG $sumC_{max}$ yielded by the IBDEA^{Xa2md} with respect to min_x_As values (AVG*$sumC_{max}$) in percent is shown in Fig. 12. As can be observed, the RE of the IBDEA^{Xa2md} in the final stages of operation is slightly greater than that of the IBDEA$^{Xa2/3md}$ and is at most 0,8%.

To sum up, both versions of the IBDEA^{Xamd} use fewer islands than the IBDEA$^{X\text{-}md}$ and their efficiency is very high and similar to that of the IBDEA$^{X\text{-}md}$. If the relative error of the order of 0,3% or 0,8% can be tolerated, then they are a good alternative to the more computationally demanding IBDEA$^{X\text{-}md}$. They need fewer islands than the IBDEA$^{X\text{-}md}$ does and can also be implemented sequentially on a single computing unit still maintaining high efficiency. All presented results were obtained for the sequential execution of the IBDEA$^{Xa2/3md}$ and the IBDEA^{Xa2md} with max_$\#ev=720{,}000$.

6.4. Experiment results – the DEA^{Xd}

The DEA^{Xd} was tested on the set of population sizes $X_P=\{10, 20, 50, 100, 200, 600, 1000, 1500\}$ and its subsets. Experiment results show that the efficiency of the DEA^{Xd} is quite similar to that of the DEA^{d} run with a constant population size. Fig. 13 and Fig. 5 confirm this observation. Fig. 5 confronts the results yielded by the DEA^{Xd} with the results yielded by the DEA^{d}. In this figure,

the black curves show the relative error of AVG $sumC_{max}$ values obtained by the DEA^{Xd} run with X_P (black solid line) and $X'_P = \{100, 1000\}$ (black dashed line) with regard to AVG*$sumC_{max}$ values comprising curve min_x_Ps at moment #*ev*. The DEA$^{Xd}(X_P)$ yields better results than those of the DEA$^{Xd}(X'_P)$ in the initial stage of its operation. However, very quickly after the start, the DEA$^{Xd}(X'_P)$ is ahead of the DEA$^{Xd}(X_P)$ and ends with the relative error of about 0,0%. On the contrary, the best results yielded by the DEA$^{Xd}(X_P)$ differs from the best ones by about 1,5%. Because the results obtained for other tested subsets of X_P were located in between two just discussed cases, they were omitted in both figures.

7. Conclusions

This article deals with the island model, which is often used in distributed evolutionary systems to increase the efficiency of computing. The concept of the multi-size populations proposed in the paper can be applied to the primary population in two ways. One way is to divide the primary population into multiple subpopulations of different sizes. Another way is to keep the primary population as a whole but to change its size over time. The case in which the primary population is divided into the multi-size subpopulations resembles the canonical island model. However, the partition of the primary population proposed in the article is fundamentally different from the partition which is usually done in the canonical island model where subpopulations of identical size are obtained.

The proposed multi-size approach to the population partition problem is advantageous over the canonical island model. It makes the effectiveness of the multi-size island-based algorithm independent of the particular population size. This feature successfully eliminates the need to fine-tune the population size of identical islands to ensure high efficiency, which as a rule, has to be done in the case of the canonical island-based algorithms.

Another important advantage of the multi-size island model is the ability to assure efficiency which is very close to the maximum size-dependent efficiency of the algorithm used in the model even without migration of solutions among islands. Migration of solutions among islands is one of the fundamental features determining the efficiency of the canonical island-based algorithms. The use of migration requires establishing the topology of migration and the migration policy, which both, in general, are laborious and time-consuming tasks. Because the proposed multi-size island model performs efficiently without migration (see the IBDEA$^{X\text{-}md}$ algorithm in Section 5.1.1), there is no need to define the topology and the policy of migration, which is an obvious advantage of the proposed approach.

The proposed multi-size island-based DE algorithm with decloning and without migration (the IBDEA$^{X\text{-}md}$) demonstrates efficiency which is nearly identical to the MSDEA throughout the entire time of its operation. This is another advantage over the canonical island-based algorithms which achieve their highest efficiency only at certain stages of operations. Based on our experimental results it can be concluded that the proposed multi-size approach is easier to implement and more attractive from the practical point of view than the single-population and the canonical island-based algorithm based on subpopulations of identical size.

The disadvantages of the proposed algorithm (IBDEA$^{X\text{-}md}$) may include an increased initial demand for computational resources. Although such a demand is gradually decreasing towards the end of the operation, the large-scale problems would require execution in the distributed environment, since IBDEA$^{X\text{-}md}$, if executed sequentially, could possibly need an excessive runtime. In order to overcome this drawback, a multi-size island-based DE algorithm - the IBDEA^{Xamd}, with active islands, migration, and decloning based on the idea of the multi-size population has been proposed. In its distributed implementation, the maximum number of computing units has been reduced, depending on its version, IBDEA$^{Xa2/3md}$ or IBDEA^{Xa2md}, to only 3 or 2 respectively. Such a simplification allows for the effective execution of the algorithm also in a sequential manner, which has been confirmed experimentally. Thus, both versions, the IBDEA$^{Xa2/3md}$ and IBDEA^{Xa2md}, use fewer islands than the IBDEA$^{X\text{-}md}$, but their efficiency remains high and similar to that of the IBDEA$^{X\text{-}md}$ even when the algorithms are executed sequentially.

Finally, the proposed concept of the multi-size population was also used to design a single-population DE algorithm with a variable population size and decloning (DEA^{Xd}). The algorithm takes advantage of the different convergence rates which are characteristic of different population sizes.

Future research will involve the following directions:

- Validating the approach using a wider spectrum of computationally hard combinatorial optimization problems.
- Implementing and validating the multi-size islands concept using different population-based algorithms paradigms.
- Evaluating the effectiveness of the approach versus different competitive metaheuristics.
- Identifying scalability constraints and barriers of the approach.
- Integrating the proposed approach with analytical methods used to solve sub-problems in case such a possibility exists.

Credit authorship contribution statement

Aleksander Skakovski: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Visualization, Writing - original draft. **Piotr Jędrzejowicz:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Supervision, Validation, Writing - review & editing.

References

Alba, E., & Troya, J. (1999). Analysis of synchronous and asynchronous parallel distributed genetic algorithms with structured and panmictic islands. In *Proceedings of the tenth symposium on parallel and distributed processing* (pp. 248–256).

Arabas, J., Michalewicz, Z., & Mulawka, J. (1994, June). GAVaPS - a genetic algorithm with varying population size. In *Proceedings of the first IEEE conference on evolutionary computation* (pp. 73–78). IEEE Press.

Arellano-Verdejo, J., Godoy-Calderon, S., Alonso-Pecina, F., Arenas, A. G., & Cruz-Chavez, M. A. (2017). A new efficient entropy population-merging parallel model for evolutionary algorithms. *International Journal of Computational Intelligence Systems, 10*(1), 1186–1197. doi:10.2991/ijcis.10.1.78.

Auger, A., & Hansen, N. (2005, September). A restart CMA evolution strategy with increasing population size. In *Proceedings of the IEEE congress on evolutionary computation, CEC* Edinburgh, Scotland, UK.

Bäck, T., Eiben, A. E., & van der Vaart, N. A. L. (2000). An empirical study on GAs "without parameters. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, & H.-P. Schwefel (Eds.), Proceedings of the sixth conference on parallel problem solving from nature, *LNCS: 1917* (pp. 315–324). Berlin: Springer.

Bartusch, M., Rolf, H. M., & Radermacher, F. J. (1988). Scheduling project networks with resource constraints and time windows. *Annals of Operations Research, 16*(1), 199–240.

Belding, T. C. (1995). The distributed genetic algorithm revisited. In *Proceedings of the sixth international conference on genetic algorithms* (pp. 114–121).

Berntsson, J., & Tang, M. (2005). Adaptive sizing of populations and number of islands in distributed genetic algorithms. In *Proceedings of the genetic and evolutionary computation conference GECCO'05* (pp. 1575–1576).

Cantú-Paz, E. (2001). Migration policies, selection pressure, and parallel evolutionary algorithms. *Journal of heuristics, 7*(4), 311–334.

Cantú-Paz, E., & Goldberg, D. E. (2003). Are multiple runs of genetic algorithms better than one? In *Proceedings of the genetic and evolutionary computation conference* (pp. 801–812).

Chop, N. E., & Calvert, D. (2005, November). The chopper genetic algorithm: A variable population genetic algorithm. In *Proceedings of the artificial neural networks in engineering conference, ANNIE 2005: 15*. St Louis, MO, USA: Asme Press.

Costa, J., Tavares, R., & Rosa, A. (1999, October). An experimental study on dynamic random variation of population size. In *Proceedings of the IEEE systems, man and cybernetics conference: 6* (pp. 607–612). IEEE Press.

Damak, N., Jarboui, B., Siarry, P., & Loukil, T. (2009). Differential evolution for solving multi-mode resource-constrained project scheduling problems. *Computers & Operations Research, 36*(9), 2653–2659.

Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution - An updated survey. *Swarm And Evolutionary Computation, 27*, 1–30. doi:10.1016/j.swevo.2016.01.004.

Dragoi, E.-N., & Dafinescu, V. (2016). Parameter control and hybridization techniques in differential evolution: a survey. *Artificial Intelligence Review, 45*(4), 447–470. doi:10.1007/s10462-015-9452-8.

Eiben, A. E., Marchiori, E., Valkó, V. A., et al. (2004). Evolutionary algorithms with on-the-fly population size adjustment. In X. Yao, et al. (Eds.). In *Parallel problem solving from nature - PPSN VIII: 3242* (pp. 41–50). Berlin, Heidelberg: Springer. PPSN 2004. LNCS.

Eiben, A. E., Schut, M. C., & de Wilde, A. R. (2006). Is self-adaptation of selection pressure and population size possible? – A case study. In T. P. Runarsson, H-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, & X. Yao (Eds.). In *Parallel problem solving from nature - PPSN ix: 4193* (pp. 900–909). Berlin, Heidelberg: Springer. LNCS.

Eltaeib, T., & Mahmood, A. (2018). Differential Evolution: A Survey and Analysis. *Applied Sciences, 8*(10), 1945. doi:10.3390/app8101945.

Goldman, B. W., & Punch, W. F. (2014, July). Parameter-less population pyramid. In *Proceedings of the annual conference on genetic and evolutionary computation, GECCO '14* (pp. 785–792). Vancouver, BC, Canada. doi:10.1145/2576768.2598350.

Harik, G. R., & Lobo, F. G. (1999). A parameter-less genetic algorithm. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, & R. E. Smith (Eds.), *Proceedings of the genetic and evolutionary computation conference: 1* (pp. 258–265). Orlando, FL, USA Morgan Kaufmann.

Harjunkoski, I., Maravelias, C. T., Bongers, P., Castro, P. M., Engell, S., Grossmann, I. E., et al. (2014). Scope for industrial applications of production scheduling models and solution methods. *Computers & Chemical Engineering, 62*, 161–193.

Hart, W. E., Baden, S., Belew, R. K., & Kohn, S. (1996). Analysis of the numerical effects of parallelism on a parallel genetic algorithm. In *Proceedings of the IPPS: 96* (pp. 606–612).

Hu, T., Harding, S., & Banzhaf, W. (2010). Variable population size and evolution acceleration: A case study with a parallel evolutionary algorithm. *Genetic Programming and Evolvable Machines, 11*(2), 205–225.

Javaid, N. (2019). Differential Evolution: An Updated Survey. In L. Barolli, N. Javaid, M. Ikeda, & M. Takizawa (Eds.), *Advances in Intelligent Systems and Computing* (772, pp. 681–691). Springer.

Jebaraj, L., Venkatesan, C., Soubache, I., & Rajan, C. C. A. (2017). Application of differential evolution algorithm in static and dynamic economic or emission dispatch problem: A review. *Renewable & Sustainable Energy Reviews, 77*, 1206–1220. doi:10.1016/j.rser.2017.03.097.

Jędrzejowicz, P., & Skakovski, A. (2016a). Improving performance of the differential evolution algorithm using cyclic decloning and changeable population size. *Journal of Universal Computer Science (J.UCS), 22*(6), 874–893.

Jędrzejowicz, P., & Skakovski, A. (2016b). Properties of the island-based and single population differential evolution algorithms applied to discrete-continuous scheduling. In *Proceedings of the eighth KES international conference on intelligent decision technologies (KES-IDT 2016)* (pp. 349–359).

Józefowska, J., & Węglarz, J. (1998). On a methodology for discrete-continuous scheduling. *European Journal of Operational Research, 107*(2), 338–353.

Józefowska, J., Mika, M., Różycki, R., Waligóra, G., & Węglarz, J. (2000). Solving the discrete-continuous project scheduling problem via its discretization. *Mathematical Methods of Operations Research, 52*(3), 489–499.

Józefowska, J., Mika, M., Różycki, R., Waligóra, G., & Węglarz, J. (2001). Solving discrete-continuous scheduling problems by tabu search. In *Proceedings of the fourth metaheuristics international conference MIC'2001* (pp. 667–671).

Jumonji, T., Chakraborty, G., Mabuchi, H., & Matsuhara, M. (2007, September). A novel distributed genetic algorithm implementation with variable number of islands. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation* (pp. 4698–4705). IEEE.

Koumousis, V., & Katsaras, C. P. (2006). 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.

Krink, T., Mayoh, B. H., & Michalewicz, Z. (1999). A PATCHWORK model for evolutionary algorithms with structured and variable size populations. In *Proceedings of the genetic and evolutionary computation conference: 2* (pp. 1321–1328).

Lee, H., & Maravelias, C. T. (2018a). Combining the advantages of discrete- and continuous-time scheduling models: Part 1. Framework and mathematical formulations. *Computers & Chemical Engineering, 116*, 176–190.

Lee, H., & Maravelias, C. T. (2018b). Combining the advantages of discrete- and continuous-time scheduling models: Part 2. Systematic methods for determining model parameters. *Computers & Chemical Engineering* in Press.

Mora-Melià, D., Martínez-Solanob, F. J., Iglesias-Reyb, P. L., & Gutiérrez-Bahamondesa, J. H. (2017). Population size influence on the efficiency of evolutionary algorithms to design water networks. In *Proceedings of the XVIII international conference on water distribution systems analysis, WDSA2016, procedia engineering: 186* (pp. 341–348).

Mühlenbein, H. (1991). Evolution in time and space: The parallel genetic algorithm. In G. Rawlins (Ed.). In *Foundations of genetic algorithms FOGA: 1* (pp. 316–337). Burlington, MA: Morgan Kaufmann.

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

Piotrowski, A. P. (2017). Review of Differential Evolution population size. *Swarm And Evolutionary Computation, 32*, 1–24. doi:10.1016/j.swevo.2016.05.003.

Rajakumar, B. R., & Aloysius George (2013). APOGA: An adaptive population pool size based genetic algorithm. In Proceedings of the AASRI Conference on Intelligent Systems and Control (AASRI Procedia*): 4* (pp. 288–296). Elsevier.

Ratajczak-Ropel, E., & Skakovski, A. (2018). Population-based approaches to the resource-constrained and discrete-continuous scheduling. In J. Kacprzyk (Ed.). In *Studies in systems, decision and control: 108* (pp. 101–236). Cham, Switzerland: Springer.

Roeva, O., Fidanova, S., & Paprzycki, M. (2013). Influence of the population size on the genetic algorithm performance in case of cultivation process modelling. In *Proceedings of the federated conference on computer science and information systems* (pp. 371–376).

Różycki, R. (2000). *Zastosowanie algorytmu genetycznego do rozwiązywania dyskretno-ciągłych problemów szeregowania. Ph.D. dissertation in Polish*. Poland: Faculty of Computing, Poznań University of Technology, Poznań.

Rozycki, R., & Weglarz, J. (2015). Solving a power-aware scheduling problem by grouping jobs with the same processing characteristic. *Discrete Applied Mathematics, 182*, 150–161.

Rozycki, R., & Weglarz, J. (2018). Improving the efficiency of scheduling jobs driven by a common limited energy source. In *Proceedings of the twenty-third international conference on methods & models in automation & robotics (MMAR)* (pp. 932–936).

Sekaj, I. (2004). Robust parallel genetic algorithms with re-initialisation. In *Proceedings of the parallel problem solving from nature – PPSN VIII, eighth international conference* (pp. 316–337).

Smorodkina, E., & Tauritz, D. (2007). Greedy population sizing for evolutionary algorithms. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 2181–2187).

Storn, R., & Price, K. (1997). Differential evolution: A simple and efficient heuristic for global optimization over continuous space. *Journal of Global Optimization, 11*(4), 341–359.

Tanese, R. (1987). Parallel genetic algorithms for a hypercube. In *Proceedings of the second international conference on genetic algorithms and their application* (pp. 177–183).

Teo, J. (2006). Exploring dynamic self-adaptive populations in differential evolution. *Soft Computing, 10*(8), 673–686.

Tomassini, M. (2004, June). Spatially structured EAs. *Presented at Genetic and Evolutionary Computation Conference GECCO'04 Tutorials* Available http://www.cs.bham.ac.uk/~wbl/biblio/gecco2004/TUT030.pdf2004.

Whitley, D., & Starkweather, T. (1990). GENITOR II: A distributed genetic algorithm. *Journal of Experimental & Theoretical Artificial Intelligence, 2*(3), 189–214.

Whitley, D., Rana, S., & Heckendorn, R. B. (1999). The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology, 7*(1), 33–47.

Wright, S. (1931). Evolution in Mendelian populations. *Genetics, 16*(2), 97–159.

Wright, S. (1943). Isolation by distance. *Genetics, 28*(2), 114–138.

Zhang, C., Chen, J., Xin, B., Cai, T., & Chen, C. (2011). Differential evolution with adaptive population size combining lifetime and extinction mechanisms. In *Proceedings of the eighth asian control conference (ASCC)* (pp. 1221–1226). (May).