IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 8, 2022, accepted March 22, 2022, date of publication March 28, 2022, date of current version April 4, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3162634

A Multisize No Migration Island-Based Differential Evolution Algorithm With Removal of Ineffective Islands

ALEKSANDER SKAKOVSKI AND PIOTR JĘDRZEJOWICZ, (Member, IEEE)
Department of Information Systems, Gdynia Maritime University, 81-225 Gdynia, Poland
Corresponding author: Aleksander Skakovski (a.skakovski@wznj.umg.edu.pl)

This work was supported by Gdynia Maritime University under Grant WZNJ/2021/PZ/03.

ABSTRACT The paper is a continuation of our previous research where a novel concept of multisize island model was proposed. Such multisize approach facilitates the design of island-based algorithms and brings such benefits as: improved fitness dynamics throughout the entire time of operation even without the migration of solutions between the islands. The absence of migration eliminates the need to establish the topology and the policy of migration. It also makes the efficiency of multisize island-based algorithms independent of the particular islands' size and 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 multisize island model over the canonical one. In this paper we improved earlier proposed multisize island-based DE algorithm by adding to it the ability to automatically optimize the number of islands in operation. This feature enables the release of most computational units before the algorithm completes its operation in the case of concurrent execution of the algorithm on multiple computational units, or reduction of the algorithm running time in the case of its execution on a single computational unit. The proposed algorithm was tested by solving computationally difficult scheduling problem, which is the discrete-continuous scheduling with continuous resource discretization.

INDEX TERMS Evolutionary computation, multisize island model, differential evolution, discrete-continuous scheduling.

I. INTRODUCTION

The present work is a continuation of our previous research [1] on the island model of computing and the impact of population size on the efficiency of the DE algorithm. The size of the population is one of the parameters that have a significant impact on the efficiency of EAs. The publications on the impact of the size of the population on the efficiency of EAs have shown that determining the size of the population is not trivial and is one of the most important tasks of EA's parameter optimization. For this reason, this issue has been the subject of interest of many researchers since the emergence of EAs and has been called ''the curse of population size'' [2]. To achieve maximum performance, EA designers had to determine the optimal population size through a computational experiment. Unfortunately, the optimal population size depends on such factors as: the algorithm used [3], the problem at hand [2]–[7], including instance unique characteristics [8] and the dimensionality of the problem [9], [10], and the number of fitness function evaluations available or allowed [3], [11], [12]. So, every time when any of these factors has changed, the process of determining the optimal population size had to be repeated from the beginning. These were burdensome, but necessary implementation costs, which justified the high efficiency of the algorithm.

The problem of determining the optimal population size also occurs in the case of the implementation of the canonical island model. In the literature, this model of computing was often reported as more effective, than the search performed on a single population, e.g. [13]–[15]. In the island model, the overall population is represented by sub-populations (islands) of identical sizes. Sub-populations on the islands evolve autonomously, albeit periodically

The associate editor coordinating the review of this manuscript and approving it for publication was Asad Waqar Malik.

exchanging solutions between themselves (solutions are said to migrate between islands). Migration of solutions in a given island model takes place according to the interconnection topology and migration policy established for this model. So, in the case of the canonical island model, the volume of work related to the tuning of the EA's parameters increases, because not only the optimal size of the populations on the islands, but also the optimal number of islands, the interconnection topology and the migration policy need to be determined. All these implementation difficulties have motivated us to develop an efficient and easy-to-design island EA without the burden of parameter tuning. To design such an algorithm, we proposed in [1] a new concept of the island model – a multi-size island model. In this model, the islands are of different sizes and there is no migration between islands. These two features make a fundamental difference to the canonical island model. The proposed multi-size island model ensures ease of design because it is devoid of the problems that arise when designing algorithms based on the canonical island model. The model takes advantage of different convergence rates of algorithms operating on islands of different sizes to provide better dynamics of evolution than that of the canonical island model or any particular island used in the model. An EA based on the multi-size model can achieve efficiency which can be very close to the maximum size-dependent efficiency of the algorithm (MSDEA) operating on the islands. The term MSDEA refers to some ideal algorithm which ensures maximal possible efficiency as if at every moment of its operation the size of the population was optimal. With this property, the multi-size island model can be viewed as a method of dealing with the curse of population size. An example of very close to the MSDEA curve represented by the minimized fitness function values is shown in Fig. 1 as red dashed line and in Fig. 2 as curve composed of multiple segments.

The summary of advantages of the multi-size island model over the canonical model includes:

1. better dynamics of the fitness function - the curve of the fitness function throughout the entire runtime of the algorithm is very close to the optimum, which is unattainable in the case of the canonical island model; the model achieves better dynamics of the fitness function due to the fact that it uses different rates of algorithm convergence on different population sizes; the number of islands in the model determines the degree of proximity to the optimal curve - the more islands, the closer to the optimum,
2. no need to experimentally determine the size of the islands due to the specificity of the optimization problem to deal with - the diversity of island sizes ensures the highest efficiency for each type of the problem,
3. the quality of solutions is always the best for any available number of fitness function evaluations; the available number of fitness function evaluations has a direct impact on the quality of solutions and necessitates the adjustment of the population size; in the case of the multi-size model there is always an island on the archipelago whose size is close to optimal for a given number of fitness function evaluations,
4. no need to experimentally determine the interconnection topology between the islands and
5. no need to experimentally determine the migration policy, since there is no migration at all; there will always be an island with a similar efficiency to the canonical island algorithm with an optimally tuned sizes of the islands and the number of islands (this conclusion is based on the experiments described in [16].

The disadvantage of this algorithm is that large-sized islands are kept in use until the very end of the algorithm execution. Despite this fact, these islands are not always able to improve on the best current solution, which results in a costly waste of computational resources.

The motivation and goal of this work was to minimize the inefficient use of computational resources by the Multisize Island-Based DE Algorithm with Decloning and without Migration (IBDEA$^{\mathrm{X-md}}$), previously proposed in [1], by adding to it the ability to remove islands, which will not be able to improve the best current solution before the algorithm completes. The introduction of such functionality benefits in an earlier release of computational units, in the case of execution of the algorithm in a distributed system, or reduction of the overall algorithm operation time, in the case of its execution on a single computational unit. This has the advantage of using the released computational resources earlier for other purposes. To design the IBDEA$^{\mathrm{X-md}}$, the method of differential evolution was used. Differential evolution (DE), is a stochastic direct search and a global optimization method first proposed in [9]. We used 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 proposed in [17] and used to design the DE algorithm with decloning (DEA$^{\mathrm{d}}$). The DEA$^{\mathrm{d}}$ was used as the base search algorithm for designing IBDEA$^{\mathrm{X-md}}$ and is, therefore, inevitably used in the algorithm, proposed in this paper. The description of the proposed in this paper Multisize No Migration Island-Based Differential Evolution Algorithm with Removal of Ineffective Islands (IBDEA$^{\mathrm{Xr}}$) is provided in Section 4. The proposed algorithm was implemented and tested. Our experiments show that the IBDEA$^{\mathrm{Xr}}$ is able to reduce the number of operating islands before the algorithm terminates, and thus uses computational resources more efficiently than its predecessor.

The rest of the paper is organized as follows. In Section 2, the review of research on 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, a concept of multisize island model was described and IBDEA$^{\mathrm{Xr}}$ - a multisize island-based DE algorithm with removal of redundant islands was proposed. Section 5 contains the description of the computational experiment as well as a discussion on the

results. Section 6 includes the conclusion and an idea for future research.

II. RELATED WORK

The population size curse is an inherent attribute of various types of population-based optimization algorithms, including: GA, EA, DE, Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Cuckoo Search (CS), Bat Algorithm (BA), Estimation of Distribution (EDA), Bayesian Optimization (BOA), and Covariance Matrix Adaptation (CMA) algorithms. Numerous publications on population size management, or *population sizing*, confirm the importance of this parameter for the effectiveness of all the above-mentioned types of algorithms. This section complements the overview of research on population sizing presented in [1].

As mentioned earlier in the introduction, the optimal population size depends on factors such as: the type of algorithm to be used, the characteristics of the problem to be solved and the available or allowed number of fitness function evaluations. The fact that population size depends on several varying and case-specific factors indicates that determining the optimal population size is not a trivial task. Therefore, attempts have been made to automate the sizing process.

There are two main approaches to the problem of population sizing in the literature. The first approach is to size a single population, while the second approach uses multiple multi-size populations.

In the case of a single population, various mechanisms, techniques and schema of population sizing have been developed. For example, population implosion, that is, a linear decrease in the population size [18]; population adaptive based immune algorithm (PAIA), which adjusts the population size to the problem being solved [19]; or deterministic population shrinkage using simple variable population sizing (SVPS) scheme based on a predetermined schedule, configured by a speed and a severity parameter [20]; a population-sizing model for entropy-based model building in discrete estimation of distribution algorithms [21], and a number of others that have already been mentioned in [1]. A review of the literature on adaptive population sizing schemes used in genetic algorithms, which were proposed until 2007, is provided in [22].

Since in our proposed algorithm we use DE and an approach that uses multiple multi-size populations, we will now pay more attention to similar research. In the literature, there are two main approaches to differentiate the size of multiple populations: the first - creating multiple populations with different, but constant sizes (we will refer to this as *a static approach*), and the second - dynamically adjusting the size and number of initial populations to changing conditions (we will refer to this as *a dynamic approach*). In the algorithm that we proposed, we used the first approach, i.e. static approach, so we will first discuss works that use multiple populations of different, albeit constant sizes.

The most important work, from the point of view of the algorithm we propose, is that of Harik and Lobo [23], in which they proposed a parameterless GA (PLGA). The core idea of the PLGA 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 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. Unfortunately, the protracted initiation of the populations extends the running time of the PLGA. 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. This is the price to be paid for the exemption from population sizing.

Due to its potential, not yet fully exploited, the work of Harik and Lobo gave an impulse for further research on using multi-size populations to increase EA performance. Below we present articles with new proposals based on this idea. In [24], a slightly different from PLGA scheme of creating multi-size populations was used to design a hierarchical Bayesian optimization algorithm (hBOA). In hBOA, all populations run at the same speed in terms of the number of evaluations. However, as a result of modifying the scheme of subpopulations creation, smaller populations carry out fewer evaluations, larger - more, and the latter are initiated earlier than in PLGA. This change in the way of operation of larger populations causes them to converge earlier than in PLGA. Which can be viewed as an advantage of hBOA over PLGA. However, the hBOA, like the PLGA, is designed to be executed sequentially, which results in a long running time, delayed initiation of the populations, and, therefore, including this features among its disadvantages. In [25], a sequential smart-restart compact GA (SRcGA) was proposed. The core idea of the algorithm is that the cGA is restarted cyclically with exponentially growing population size after each restart. The proposed algorithm has the advantage over the algorithm from [23] that it has ability to terminate inefficient runs caused by a genetic drift. To detect the genetic drift, the authors used the tight quantification of the genetic drift effect of the EDAs provided in [26]. A disadvantage of the algorithm is the sequentiality of the restarts, which causes a progressive delay in the initiation of larger populations, and thus the moment of obtaining the solutions found through them. In [27], a parallel-run cGA (PRcGA) was proposed in order to shorten the computation time. PRcGA cyclically

creates processes with exponentially growing populations. In each cycle, a new process with doubled population size is initiated and added to the pool of already existing processes. Although the processes run in parallel, they are nevertheless initiated sequentially and, as with Harik and Lobo, the larger the population, the later it will be initiated. Such a population initiation scheme, unfortunately, contributes to the elongation of the algorithm's running time and can be considered a disadvantage.

In the rest of this section we consider the second approach to population sizing, i.e. dynamic adjusting the size and number of initial populations to changing conditions.

In [28], a DE algorithm (MultiDE) operating on multiple independent subpopulations was proposed. In MultiDE, the number of initial populations varies, because the algorithm periodically reinitializes the already existing subpopulations, creates new ones and removes ineffective ones. There is no migration of solutions between the subpopulations, however MultiDE periodically saves each current global optimum in a special ''population 0''. Solutions from ''population 0'' are used to significantly reduce the probability of premature convergence to already found global optima and accelerate the rate of convergence to new ones. Although in MultiDE, the size of the global population is changing however the sizing of constituting populations is left to the designer. In [29], a distributed DE algorithm with explorative–exploitative population families (DDE-EEPF) is proposed. The DDE-EEPF consists of two interacting families of sub-populations. The first family (an explorative one), in its essence, is a canonical island model. In this family, the size of the sub-populations is fixed, and the best solutions migrate between islands according to the ring topology. The sub-populations of the second family of gradually reduced size (an exploitative family) are supposed to quickly detect solutions and deliver them to the first group. Same as in [28], in DDE-EEPF, the sizing of constituting populations is left to the designer.

In [30], the improvement of the effectiveness of the proposed algorithm was achieved due to the dynamic population sizing, which consisted in a progressive reduction of the population. Here, DE runs on sub-populations, the size of which is controlled over time using the success rate of evolution. When the algorithm works efficiently on a given sub-population, its size is reduced linearly. However, if the success rate of evolution is low, attempts are made to improve it using a set of novel size-dependent mutation strategies along with subpopulation size control. If the success rate remains low in the second half of the run, the current size of the sub-population is kept fixed. If the success rate remains low at the end of the run, the subpopulation is reduced to 1/6 of its initial size. In the proposed algorithm, the set of novel mutation strategies is applied to enhance the search efficiency. As soon as the success rate has improved, the population size is reduced linearly again.

In [31], a distributed DE with adaptive merging and splitting (DDE-AMS) of subpopulations was proposed to deal with large-scale optimization problems. The high efficiency of the algorithm is achieved through dynamic subpopulation restructuring with the use of merge and split operators, while maintaining a constant size of the entire population. The merge operator merges the best and worst subpopulations in order to move the search to promising regions. The split operator splits the merged subpopulation in half, if it no longer contributes to evolution. Thus, the merge-and-split strategy causes the algorithm to operate on a varying number of sub-populations of varying sizes. To prevent merging of all sub-populations into a single population, a minimum number of sub-populations has been established. There is a migration of individuals between sub-populations, which takes place with some probability. The DDE-AMS, like the multi-population algorithms discussed above, has the same disadvantage - sophisticated dynamic sub-population restructuring, and thus complex implementation.

Another approach to dynamic population sizing is based on dynamic restructuring of subpopulations according to their current status of evolution. In [32], an adaptive multi-population framework for locating and tracking multiple optima was proposed. Although the algorithms implemented into this framework proved to be highly effective in the tests, this result was paid for by the complex functionality of the framework. The high efficiency of the algorithms has been achieved thanks to an impressive arsenal of special components for controlling the course of the evolutionary process. These components include: a database of algorithm's behavior changes, heuristic clustering, adaptation of the number of populations according to their convergence, exclusion of overlapping populations, avoidance of explored peaks, population hibernation and wakening, movements for the best individual (a Brownian movement to track a moving or a better peak, or a Cauchy movement to transform stagnating population into a converging one). The need to use all these components together makes the implementation of this framework quite complicated, which can be considered a disadvantage.

The idea to use multiple populations of varying sizes is also present in an Adaptive Multi-Population Optimization Algorithm for Global Continuous Optimization (AMPO) proposed in [33]. The AMPO consists of five sub-populations to which different search strategies have been assigned. During the search, the sizes of three subpopulations change dynamically. The algorithm showed high performance by borrowing some useful operations from evolutionary algorithms and swarm intelligence techniques and using them in a multi-population manner. The disadvantage of this algorithm is not only the large number of control parameters mentioned by the authors, but also its complex implementation. In addition to the basic search algorithm, AMPO must also implement 5 additional algorithms to perform its basic operations.

There is more research where an idea of partitioning global population into multiple subpopulations is used to improve the efficiency of different types of algorithms. A study of population partitioning techniques on efficiency of

swarm algorithms is provided in [34]. Many nature-inspired algorithms [35]–[42] and various types of optimization algorithms [43]–[46] use static and dynamic multipopulation design to improve their efficiency.

However, the implementation of the algorithms proposed under these two approaches requires extra work in addition to the implementation of the basic search algorithm. In the case of a dynamic approach, additionally sophisticated subpopulation management should be implemented. And in the case of a static approach, time-consuming experimentation must be carried out to determine the number of subpopulations, their size, the best interconnection topology among subpopulations, the policy, rate and frequency of migration in order to obtain the best performance of the algorithm.

To address the above issues, this paper proposes a Multisize No Migration Island-Based Differential Evolution Algorithm with Removal of Ineffective Islands (IBDEA$^{\mathrm{Xr}}$). The proposed algorithm is very simple in construction and devoid of disadvantages occurring in the dynamic and static approaches described above.

To summarize the review of related work, it should be stated that the population sizing has a significant impact on the efficiency of EAs and should be included among the basic functionalities of this type of algorithms.

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 [47]. It has several important practical applications including scheduling production processes [48], chemical production processes [49], [50] or processes with tasks requiring energy supply [51], [52]. 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 [53]. Discretization of the continuous resources makes possible using metaheuristic algorithms to solve instances of the DCSPwCRD.

III. TEST PROBLEM FORMULATION

The Discrete-Continuous Scheduling Problem with Continuous Resources Discretization (DCSPwCRD) is denoted as Θ_Z and formulated in the same way as in [53]. 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 $\tau_i = [\tau_i^1, \tau_i^2, \ldots, \tau_i^{W_i}]$, 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.

Our test problem is a particular case of a more general Multi-Mode Resource-Constrained Project Scheduling Problem (MMRCPSP), which is known to be NP-hard [54].

IV. THE MULTISIZE ISLAND MODEL

This Section discusses the concept of a multi-size island model. 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 $x_{Pk} < x_{Pk+1}$. Then the total primary population should be decomposed into K sub-populations (islands) of 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. Dividing a population into the set of subpopulations of different sizes 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. In the following sub-section, an algorithm which implements the described concept is proposed.

A. IBDEA$^{\mathrm{Xr}}$ - MULTISIZE ISLAND-BASED DIFFERENTIAL EVOLUTION ALGORITHM WITH REMOVAL OF INEFFECTIVE ISLANDS

Based on the concept of multi-size population, there is a proposal for a multi-size island-based DE algorithm with decloning, without migration, and with removal of ineffective islands. The algorithm denoted as IBDEA$^{\mathrm{Xr}}$ enhances its predecessor - a multi-size island-based DE algorithm IBDEA$^{\mathrm{X-md}}$, proposed in [1], with the ability to optimize the number of islands by removing ineffective ones. Both algorithms take advantage of the different convergence rates which are characteristic to different population sizes. The main idea of the IBDEA$^{\mathrm{Xr}}$, is to create an archipelago of K islands (sub-populations) of different sizes. All sub-populations are initiated at the same moment and are evolved independently by a copy of the DEA$^{\mathrm{d}}$ assigned to each island. There is no migration of solutions between the islands. Concurrently to the main evolution process, two procedures monitor the situation on all islands. The first one - Small Island Removal Procedure (SIRP) monitors small islands and removes them from the archipelago if the algorithms on small islands start converging and do not offer better solutions than on the larger islands. The second one - Large Island Removal Procedure (LIRP) monitors large islands and removes them from the archipelago if the algorithms running on them will not offer better solutions than on the smaller operating islands before the algorithm stops. The details of how exactly operate SIRP and LIRP are given below. In the case of the distributed implementation of the IBDEA$^{\mathrm{Xr}}$, when the DEA$^{\mathrm{d}}$

stops on a particular island a computing resource, on which the island has been implemented, is released. Below, the general course of operation of the IBDEA$^{\mathrm{Xr}}$ and description of its components are given:

Algorithm 1: IBDEA$^{\mathrm{Xr}}$

1. Set the maximum number of fitness function evaluations *max_#ev* and calculate the size of the largest population using (1).
2. Define the set of population sizes $X_P = \{x_{P\mathrm{k}}\}$, where $k = 1, 2, \ldots, K$.
3. Generate K populations of sizes $x_{Pk} \in X_P, k = 1, 2, \ldots, K$.
4. Create an archipelago of K islands, where an island is comprised by a population of size x_{Pk} and a copy of the DEA$^{\mathrm{d}}$ that will evolve this population.
5. On each of K islands start DEA$^{\mathrm{d}}$. Set $K^{op} = K$, K^{op} – the number of islands in operation.
6. Start the small islands removal procedure (**SIRP**).
7. After *#sr* small islands have been removed from the archipelago start the large islands removal procedure (**LIRP**).
8. Output the best solution among the remained islands.

End of IBDEA$^{\mathbf{Xr}}$.

The number of islands and the size of individual islands can be derived from the size x_{PK} of the largest island I_K. The size x_{PK} we calculate as the square root of the number of the fitness function evaluations *max_#ev* available for the algorithm:

$$x_{PK} = \sqrt{max\_\#ev} \tag{1}$$

Having calculated the size of the largest island, we set the sizes of the smaller islands either arbitrarily or by decrementing x_{PK} with some coefficient c, which can either be constant or increase as the size of the islands decreases. Regardless of the method of determining the size of the islands, it should be remembered that the smaller the c-factor, the more accurate the LIRP will work. We recommend that the sizes of the larger islands do not differ more than 1.5 times, and that the sizes of the smaller islands do not differ more than 2 times. For the suggested algorithm we propose two islands removal procedures:

1) SIRP - SMALL ISLANDS REMOVAL PROCEDURE

The general course of operation of the SIRP is given below.

Procedure 1: SIRP (Small Islands Removal Procedure)

1. Stop DEA$^{\mathrm{d}}$ on island k at moment *#ev* either if the best solution yielded on this island is worse than the best solution yielded on any other larger island, or $\#ev = max\_\#ev$.
2. Remove island k from the archipelago if DEA$^{\mathrm{d}}$ on k was stopped before carrying out *max_#ev*.
3. Update $K^{op} = K^{op}$ - 1.

End of SIRP.

2) LIRP - LARGE ISLANDS REMOVAL PROCEDURE

The procedure removes large islands from the archipelago with use of a linear tend function $\hat{y}(d_k)$ which predicts the difference d_k between the quality of solutions found on two adjacent islands I_k and I_{k+1}. We use the difference d_k as an indirect parameter of the algorithm's convergence on the larger island I_{k+1} to obtain a linear trend line $\hat{y}(d_k)$, which then tells us whether to remove I_{k+1} from the archipelago or not. We define the difference d_k at point *#ev* as:

$$d_k = sumC_{max}(k+1) - sumC_{max}(k) \tag{2}$$

for $sumC_{max}$ see Sect. 5. The reason we have chosen the difference d_k instead of the values of fitness function itself to predict island convergence is because of the greater accuracy of the prediction. The convergence on neighboring islands is similar, therefore the behavior of the difference d_k is more linear than the behavior of fitness function, and the linear trend is easier to predict. The smaller the difference between the population sizes, the more linear the behavior of d_k and the more accurate the prediction.

The moment of starting the removal of large islands should be determined in the advanced stages of the algorithm's operation. By this point, the smaller "faster" islands have already been removed, and the larger "slower" islands are still working. Some of these still functioning large islands may not be able to "outperform" their smaller predecessors before the algorithm terminates, and therefore will not improve the results. In such a situation, they can be considered ineffective and removed from the archipelago. To remove all ineffective islands, find an island that fails to improve the result of its smaller predecessor before the algorithm terminates. Let us denote such an island by I_r. If I_r has been found, then remove I_r and all larger islands from the archipelago. The removal of all islands larger than I_r can be justified by the fact that since island I_r does not outperform its smaller predecessor before the algorithm terminates, then each next larger island will not do so, because the larger the island, the slower it converges. The removal of large islands can be started e.g. after the first 4 smaller islands has been already removed from the archipelago. From that moment on, an x-intercept point of linear trend function $\hat{y}(d_k)$ for all pairs (I_k, I_{k+1}) of still operating adjacent islands should be cyclically checked with a step of e.g. $5\%{\cdot}max\_\#ev$, where I_{k+1} - is the larger island in a pair, $k = q, q+1, \ldots, K^{op}-1$, I_q is the smallest of all islands still in operation, and K^{op} – the number of islands still in operation. A larger island I_{k+1} in the pair (I_k, I_{k+1}) can be removed only when the linear trend function $\hat{y}(d_k)$ becomes zero at some x-intercept point $\#ev_c > max\_\#ev$. The larger island I_{k+1} in such a pair is marked as I_r and removed from the archipelago together with all larger islands $I_{r+1}, I_{r+2}, \ldots, I_{K^{op}}$. On the other hand, if $\hat{y}(d_k)$ becomes zero at the moment $\#ev_c < max\_\#ev$, it means that larger island in the pair may outperform the smaller one before the algorithm stops and the smaller one can be removed from the archipelago. However, the removal of smaller islands is better to entrust to SIRP as more accurate. As it may happen, that only some of the large islands still in operation are removed, the process of removing large islands should continue until there is only one working island left, or until $\#ev = max\_\#ev$. The linear trend line $\hat{y}(d_k)$ can be

created using Least squares method by applying (3) - (5) to e.g. {10%-15%}·*max_#ev* values of difference d_k, preceding point $\#ev_c$:

$$a = \frac{n\sum_{i=1}^{n} y_i t_i - \sum_{i=1}^{n} t_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} t_i^2 - \left(\sum_{i=1}^{n} t_i\right)^2} \tag{3}$$

$$b = \bar{y} - a \cdot \bar{t} \tag{4}$$

$$\hat{y} = a \cdot t + b \tag{5}$$

where y – the observed value, $\bar{y}$ – the mean value of all observed values, $\hat{y}$ – the predicted value of y, t – the index of current observation, $\bar{t}$ – the mean value of all indices t, a – the slope of the function and of the line, b - the initial value of y, n – the number of observations.

The general course of operation of the LIRP is given below.

Procedure 2: LIRP (Large Islands Removal Procedure)

1. **While** the number of islands still in operation $K^{op} > 1$ and $\#ev <$ *max_#ev* for all pairs of islands (I_k, I_{k+1}) still in operation keep checking, with the period of 5%·*max_#ev*, the value of ev_{ip} at x-intercept point of linear trend function $\hat{y}(d_k)$.
2. If there exists a pair of islands (I_k, I_{k+1}) for which $ev_{ip} >$ max_#ev, then remove islands $I_{k+1}, I_{k+2}, \ldots, I_K^{op}$ from the archipelago and update $K^{op} = K^{op} - K^{rr}$, K^{rr} – the number of removed ineffective islands. If there is no such pair, then go to step 1.

End of LIRP.

The computational complexity of the IBDEA^{Xr} is the same as that of the DEA^{d} and is $O(n \log n)$. The test results of the IBDEA^{Xr} are presented and discussed in Section 5.

3) DEA^{D} - A SINGLE POPULATION DIFFERENTIAL EVOLUTION ALGORITHM WITH DECLONING

The DEA^{d} is a combination of the DEAnd and the Decloning Procedure (DP) first proposed in [17]. 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. 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.

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.

Since the decloning procedure does not increase the size of the population, the complexity of the DEA^{d} is $O(n \log n)$.

Algorithm 2: DEAnd

1. Assume the population of individuals P_c consists of two halves P_c^1 and P_c^2, i.e. $P_c = P_c^1 + P_c^2$, and $|P_c| = 2 \cdot x_P$, $|P_c^1| = x_P$, $|P_c^2| = x_P$;
2. **for** every target vector S_{tg} in the current population P_c^1 **do**:
3. Create a mutant vector M from three vectors S_0, S_1, S_2 randomly chosen from P_c^1, 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_c^2 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_{c+1}^1 selecting the best vectors from P_c^1 and P_c^2;
8. Repeat steps 2 - 7 until the stop criterion is met;

End of DEAnd.

Algorithm 3: 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};

End of DEA^{d}.

V. COMPUTATIONAL EXPERIMENT

In the experiments, the values of the parameters of the DEA^{d} were assumed to be the same as in [1]. The parameters necessary to carry out the differential evolution algorithm were set to the same values as in [55], namely the scale factor F (in [55], "F" is denoted as "A") which controls the evolution rate of the population was set $F = 1{,}5$ and the values of the variable $rand \in [0, 1]$. The crossover constants Cr_p and Cr_m which control the probability that the trial individual will receive the actual individual's genes were set $Cr_p = 0{,}2$ and $Cr_m = 0{,}1$, where p and m in the notations Cr_p and Cr_m stand for tasks' positions and modes. Problem instances used in all of the reported experiments are available by e-mail 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%. 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%.

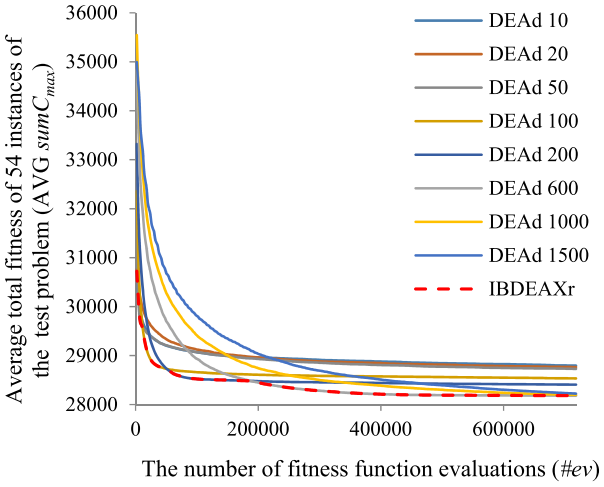


FIGURE 1. AVG sumCmax yielded by the DEA^{d} for different population sizes and by IBDEA^{Xr} w.r.t. #ev.

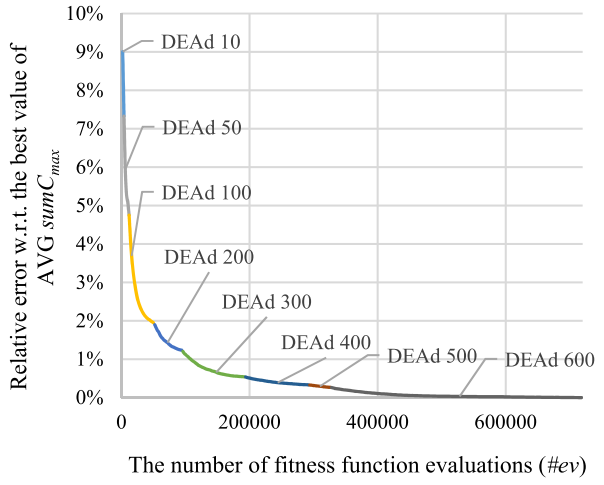
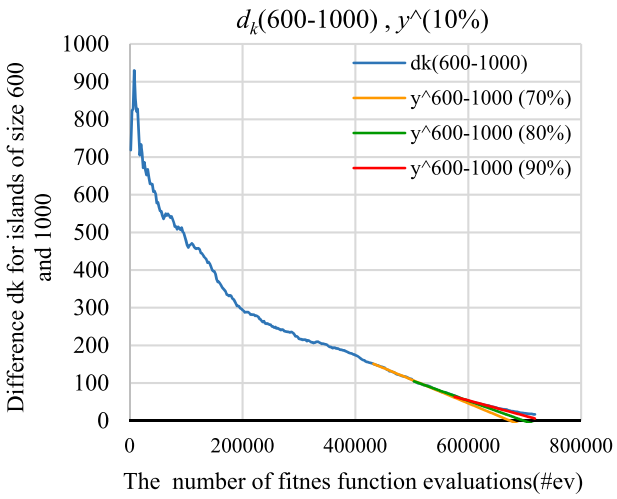
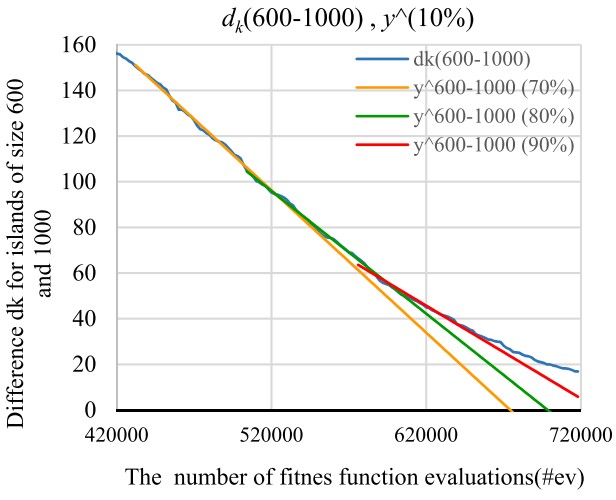


FIGURE 2. A min_x$_P$s curve showing relative error of x$_P$s' minima (AVG*sumC$_{max}$) w.r.t. AVG**sumC$_{max}$ - the best value of AVGsumC$_{max}$ across all x$_P$s.



(a)



(b)

FIGURE 3. (a) difference d_k between islands of sizes 600 and 1000 and the linear trend lines based on 10%·max_#ev of points, preceding points {70%, 80%, 90%}·max_#ev. (b) the detailed view of the trend lines.

In the experiments, we considered the following population sizes: $X_p = \{10, 20, 50, 100, 200, 300, 400, 500, 600, 1000, 1500\}$, and the number of the fitness function evaluations $\#ev$ available for the algorithms to yield a solution to the problem was set to $\#ev = 720000$.

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 the number of fitness function evaluations was set to 720000, 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 – 3s and for the problem size 20×2 for all discretization levels approximately 5 – 6s. The total time taken by the DEAnd to process all 54 instances was approximately 206s.

A. EXPERIMENT RESULTS

Below we present graphical results obtained during our tests with the proposed IBDEA^{Xr} and its constituent parts, such as DE algorithm with decloning (DEA^{d}), and Small and Large Island Removal Procedures (SIRP and LIRP respectively). In these graphs, the quality of solutions is represented by the values of parameter AVG $sumC_{max}$ (the details of AVG $sumC_{max}$ calculation are provided above in Section 5). 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.

Fig. 1 shows curves of AVG $sumC_{max}$, an average of total fitness of 54 test instances of the test problem, obtained for all tested population sizes. The red dashed line in it, marked with IBDEA^{Xr}, shows a hypothetical nearly ideal efficiency curve of the DEA^{d}. As it can be seen from Fig. 1, no single curve from this collection can completely retrace such curve, regardless of the size of the population on which DEA^{d} would operate. At most, if at all, only a part of the curve can be retraced. Fig. 2 shows the same nearly ideal curve, however,

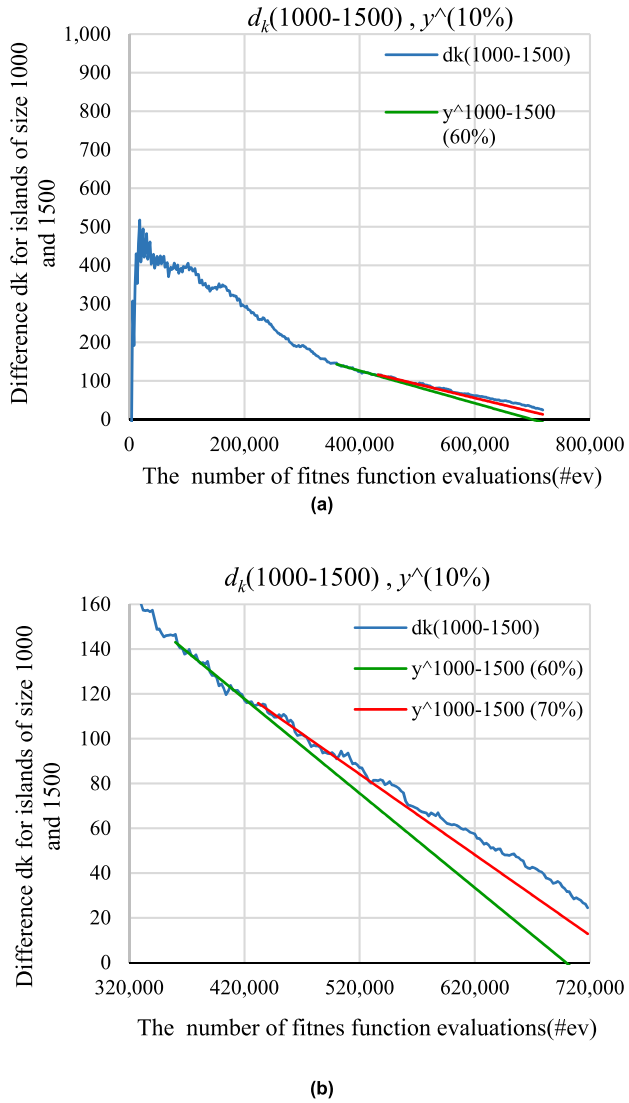


FIGURE 4. (a) difference d_k between islands of sizes 1000 and 1500 and the linear trend lines based on 10%·max_#ev of points, preceding points {60%, 70%}·max_#ev. (b) the detailed view of the trend lines.

composed of multiple segments obtained by Small Island Removal Procedure (SIRP). Fig. 3(a) and (b) show difference d_k between islands of sizes 600 and 1000 and the linear trend lines based on 10%·*max_#ev* points preceding points {70%, 80%, 90%}·*max_#ev*. The detailed view of the trend lines is shown in Fig. 3(b). Similarly, Fig. 4(a) and (b) show difference d_k between islands of sizes 1000 and 1500 and the linear trend lines based on 10%·*max_#ev* points preceding points {60%, 70%}·*max_#ev*.

Figures 4(a) and (b) also illustrate the process of removal of ineffective islands from the archipelago, which were islands of sizes 1000 and 1500. The island of size 1500 was removed after the DEA^{d} carried out 70%·*max_#ev*. It took more time, namely 90%·*max_#ev*, to remove island of size 1000. This way IBDEA^{Xr} gradually reduced the number of processing units from 11 at the beginning to 1 after 90%·*max_#ev* have been carried out. For the last 10%·*max_#ev*, the algorithm was operating on one island of population of size 600.

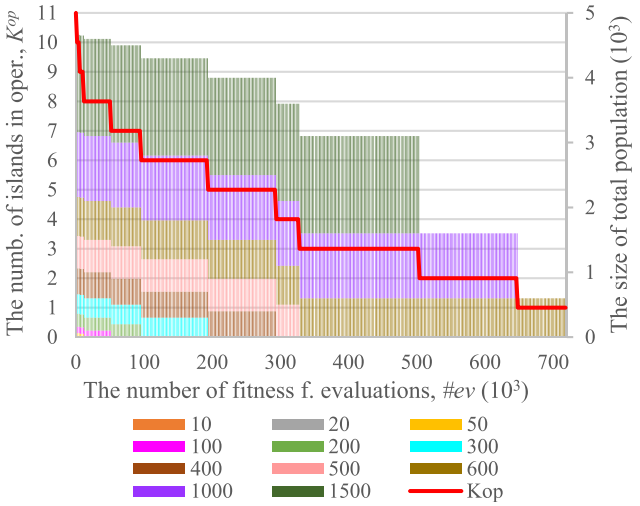


FIGURE 5. An example of reducing the number of islands in operation and the size of the total population by IBDEA^{Xr} for the case K = 11, X_P = {10, 20, 50, 100, 200, 300, 400, 500, 600, 1000, 1500}.

Figure 5 illustrates how the number of operating islands and the size of the total population have changed over time.

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. This ability makes it possible to release the computational unit earlier, in the case of concurrent execution of the algorithm on multiple computational units, or to shorten the algorithm's execution time, in the case of its execution on a single computational unit. The proposed algorithm was tested by solving computationally difficult scheduling problem, which is the discrete-continuous scheduling with continuous resource discretization. The experiment proved the ability to reduce the number of processing units. It reduced the number of islands from 11 at the beginning to 1 after 90% of given number of fitness function evaluations (*max_#ev*) have been carried out. For the last 10%·*max_#ev*, the algorithm was operating only on one island.

In addition to the above-described experiment with removal of ineffective islands, tests were carried out in order to show the effect of applying the multi-size island model without migration on the efficiency of the standard DE algorithm. For this purpose, the standard DE algorithm proposed by Storn and Price in [9] was implemented and experimentally tested. This algorithm is described in subsection IV.A.3 as "Algorithm 2: DEAnd". DEAnd was used as a search algorithm to construct a multi-size island-based algorithm without migration - IBDEA^{ndXr}, consisting of $K = 8$ islands, with population sizes $X_P = \{10, 20, 50, 100, 200, 600, 1000, 1500\}$. The results of the experiments are presented in Fig. 6. In this figure, the solid colored curves show the dynamics of the fitness function on the particular islands, and the red dashed line shows the dynamics of the resulting fitness function curve of the solutions found by IBDEA^{ndXr}. Comparing the curves in Fig. 1 and Fig. 6, it can be seen how unfavorably for DEAnd, w.r.t. DEA^{d}, differ the

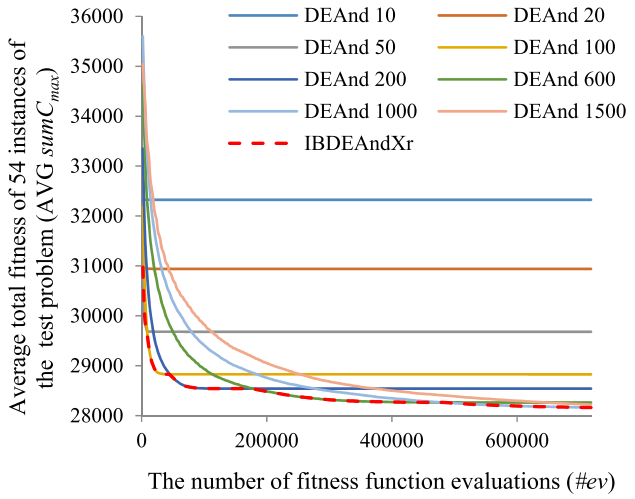


FIGURE 6. AVG sumCmax yielded by the DEAnd for different population sizes and by IBDEA^{ndXr} w.r.t. #ev.

curves corresponding to populations with smaller sizes $X_P = \{10, 20, 50\}$. On the other hand, the resulting IBDEA^{ndXr} curve is practically as good as the IBDEA^{Xr}'s curve in Fig. 1. This fact shows that owing to the multi-size island model, IBDEA^{ndXr} is able to find very good solutions and demonstrate good dynamics of the fitness function, despite the poorer quality of solutions on some islands.

VI. CONCLUSION

The paper proposes a multi-size island-based DE algorithm with decloning, without migration, and with removal of ineffective islands. The algorithm denoted as IBDEA^{Xr} implements a novel concept of multi-size island model which facilitates the design of island-based algorithms and brings such benefits as: improved fitness dynamics throughout the entire time of operation even without the migration of solutions between the islands. The absence of migration, which eliminates the need to establish the topology and the policy of migration. The proposed IBDEA^{Xr} enhances its predecessor - a multi-size island-based DE algorithm IBDEA^{X-md}, proposed in [1], with the ability to optimize the number of islands by removing ineffective ones.

Future research will involve validating the multi-size island approach using a wider spectrum of computationally hard optimization problems and implementing and validating the multi-size islands framework using different types of population-based algorithms.

REFERENCES

[1] A. Skakovski and P. Jędrzejowicz, "An island-based differential evolution algorithm with the multi-size populations," *Expert Syst. Appl.*, vol. 126, pp. 308–320, Jul. 2019, doi: 10.1016/j.eswa.2019.02.027.
[2] K. G. Dhal, A. Das, S. Sahoo, R. Das, and S. Das, "Measuring the curse of population size over swarm intelligence based algorithms," *Evolving Syst.*, vol. 12, no. 3, pp. 779–826, Dec. 2019, doi: 10.1007/s12530-019-09318-0.
[3] A. P. Piotrowski, "Review of differential evolution population size," *Swarm Evol. Comput.*, vol. 32, pp. 1–24, Feb. 2017, doi: 10.1016/j.swevo.2016.05.003.
[4] R. E. Smith and E. Smuda, "Adaptively resizing populations: Algorithm, analysis, and first results," *Complex Syst.*, vol. 9, no. 1, pp. 47–72, 1995.
[5] D. E. Goldberg, *The Design of Innovation: Lessons From and for Competent Genetic Algorithms*. Boston, MA, USA: Kluwer, 2002.
[6] F. G. Lobo and C. F. Lima, "Revisiting evolutionary algorithms with on-the-fly population size adjustment," in *Proc. 8th Annu. Conf. Genetic Evol. Comput. (GECCO)*, 2006, pp. 1241–1248, doi: 10.1145/1143997.1144192.
[7] V. K. Koumousis and C. P. Katsaras, "A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 19–28, Feb. 2006, doi: 10.1109/TEVC.2005.860765.
[8] O. Roeva, S. Fidanova, and M. Paprzycki, "Influence of the population size on the genetic algorithm performance in case of cultivation process modelling," in *Proc. Federated Conf. Comp. Sci. Inf. Syst.*, 2013, pp. 371–376.
[9] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997, doi: 10.1023/A:1008202821328.
[10] N. Hansen and S. Kern, "Evaluating the CMA evolution strategy on multimodal test functions," in *Proc. PPSN* (Lecture Notes in Computer Science), vol. 3242. Berlin, Germany: Springer, 2004, pp. 282–291, doi: 10.1007/978-3-540-30217-9_29.
[11] T. Chen, K. Tang, G. Chen, and X. Yao, "A large population size can be unhelpful in evolutionary algorithms," *Theor. Comput. Sci.*, vol. 436, no. 2, pp. 54–70, Jun. 2012, doi: 10.1016/j.tcs.2011.02.016.
[12] R. A. Sarker and M. F. A. Kazi, "Population size, search space and quality of solution: An experimental study," in *Proc. Congr. Evol. Comput. (CEC)*, 2003, pp. 2011–2018, doi: 10.1109/CEC.2003.1299920.
[13] H. Ma, S. Shen, M. Yu, Z. Yang, M. Fei, and H. Zhou, "Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey," *Swarm Evol. Comput.*, vol. 44, pp. 365–387, Feb. 2019, doi: 10.1016/j.swevo.2018.04.011.
[14] H. Muhlenbein, "Evolution in time and space: The parallel genetic algorithm," in *Foundations of Genetic Algorithms FOGA*, vol. 1. Burlington, MA, USA: Morgan Kaufmann, 1991, pp. 316–337.
[15] D. Whitley and T. Starkweather, "GENITOR II: A distributed genetic algorithm," *J. Exp. Theor. Artif. Intell.*, vol. 2, no. 3, pp. 189–214, Jul. 1990, doi: 10.1080/09528139008953723.
[16] P. Jódrzejowicz and A. Skakovski, "Properties of the island-based and single population differential evolution algorithms applied to discrete-continuous scheduling," in *Proc. KES-IDT*, Puerto de la Cruz, Spain, 2016, pp. 349–359.
[17] P. Jódrzejowicz and A. Skakovski, "Improving performance of the differential evolution algorithm using cyclic decloning and changeable population size," *J. CUS*, vol. 22, no. 6, pp. 874–893, 2016, doi: 10.3217/jucs-022-06-0874.
[18] S. Luke, G. C. Balan, and L. Panait, "Population implosion in genetic programming," in *Proc. GECCO* (Lecture Notes in Computer Science), vol. 2724. Berlin, Germany: Springer, 2003, pp. 1729–1739, doi: 10.1007/3-540-45110-2_65.
[19] J. Chen and M. Mahfouf, "A population adaptive based immune algorithm for solving multi-objective optimization problems," in *Artificial Immune Systems (ICARIS)* (Lecture Notes in Computer Science), vol. 4163. Berlin, Germany: Springer, 2006, pp. 280–293, doi: 10.1007/11823940_22.
[20] J. L. J. Laredo, C. Fernandes, J. J. Merelo, and C. Gagné, "Improving genetic algorithms performance via deterministic population shrinkage," in *Proc. 11th Annu. Conf. Genetic Evol. Comput. (GECCO)*, New York, NY, USA, 2009, pp. 819–826, doi: 10.1145/1569901.1570014.
[21] T.-L. Yu, K. Sastry, D. E. Goldberg, and M. Pelikan, "Population sizing for entropy-based model building in discrete estimation of distribution algorithms," in *Proc. 9th Annu. Conf. Genetic Evol. Comput. (GECCO)*, New York, NY, USA, 2007, pp. 601–608, doi: 10.1145/1276958.1277080.
[22] F. G. Lobo and C. F. Lima, "Adaptive population sizing schemes in genetic algorithms," *Parameter Setting in Evolutionary Algorithms* (Studies in Computational Intelligence), vol. 54. Berlin, Germany: Springer, 2007, pp. 185–204, doi: 10.1007/978-3-540-69432-8_9.
[23] G. R. Harik and F. G. Lobo, "A parameter-less genetic algorithm," in *Proc. GECCO*, Orlando, FL, USA, 1999, pp. 258–265.
[24] M. Pelikan and T.-K. Lin, "Parameter-less hierarchical BOA," in *Proc. GECCO* (Lecture Notes in Computer Science), vol. 3103. Berlin, Germany: Springer, 2004, pp. 24–35, doi: 10.1007/978-3-540-24855-2_3.
[25] B. Doerr and W. Zheng, "From understanding genetic drift to a smart-restart parameter-less compact genetic algorithm," in *Proc. Genetic Evol. Comput. Conf.*, New York, NY, USA, Jun. 2020, pp. 805–813, doi: 10.1145/3377930.3390163.

[26] B. Doerr and W. Zheng, "Sharp bounds for genetic drift in estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 24, no. 6, pp. 1140–1149, Dec. 2020, doi: 10.1109/TEVC.2020.2987361.
[27] B. Doerr, "The runtime of the compact genetic algorithm on jump functions," *Algorithmica*, vol. 83, no. 10, pp. 3059–3107, Oct. 2021, doi: 10.1007/s00453-020-00780-w.
[28] Z. V. Hendershot and F. W. Moore, "MultiDE: A simple, powerful differential evolution algorithm for finding multiple global optima," in *Proc. FLAIRS Conf.*, Miami Beach, FL, USA, 2004, pp. 368–373.
[29] M. Weber, F. Neri, and V. Tirronen, "Distributed differential evolution with explorative–exploitative population families," *Genetic Program. Evolvable Mach.*, vol. 10, no. 4, pp. 343–371, Oct. 2009, doi: 10.1007/s10710-009-9089-y.
[30] Y. Fu, H. Wang, and M.-Z. Yang, "An adaptive population size differential evolution with novel mutation strategy for constrained optimization," 2018, *arXiv:1805.04217*.
[31] Y.-F. Ge, W. J. Yu, Y. Lin, Y. J. Gong, and Z. H. Zhan, "Distributed differential evolution based on adaptive mergence and split for large-scale optimization," *IEEE Trans. Cybern.*, vol. 48, no. 7, pp. 2166–2180, Jul. 2018, doi: 10.1109/TCYB.2017.2728725.
[32] C. Li, T. T. Nguyen, M. Yang, M. Mavrovouniotis, and S. Yang, "An adaptive multipopulation framework for locating and tracking multiple optima," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 590–605, Aug. 2016, doi: 10.1109/TEVC.2015.2504383.
[33] Z. Li, V. Tam, and L. K. Yeung, "An adaptive multi-population optimization algorithm for global continuous optimization," *IEEE Access*, vol. 9, pp. 19960–19989, 2021, doi: 10.1109/ACCESS.2021.3054636.
[34] R. Chaudhary and H. Banati, "Study of population partitioning techniques on efficiency of swarm algorithms," *Swarm Evol. Comput.*, vol. 55, Jun. 2020, Art. no. 100672.
[35] H. Chen, S. Li, A. Asghar Heidari, P. Wang, J. Li, Y. Yang, M. Wang, and C. Huang, "Efficient multi-population outpost fruit fly-driven optimizers: Framework and advances in support vector machines," *Expert Syst. Appl.*, vol. 142, Mar. 2020, Art. no. 112999.
[36] B. H. Abed-alguni, A. F. Klaib, and K. N. Nahar, "Island-based whale optimisation algorithm for continuous optimisation problems," *Int. J. Reasoning-Based Intell. Syst.*, vol. 11, no. 4, pp. 319–329, 2019.
[37] M. A. Al-Betar and M. A. Awadallah, "Island bat algorithm for optimization," *Expert Syst. Appl.*, vol. 107, pp. 126–145, Oct. 2018.
[38] M. A. Awadallah, M. A. Al-Betar, A. L. Bolaji, I. A. Doush, A. I. Hammouri, and M. Mafarja, "Island artificial bee colony for global optimization," *Soft Comput.*, vol. 24, pp. 13461–13487, Feb. 2020.
[39] B. H. Abed-alguni and M. Barhoush, "Distributed grey wolf optimizer for numerical optimization problems," *Jordanian Jour. Comput. Inf. Technol.*, vol. 4, no. 3, pp. 130–149, 2018.
[40] B. H. Abed-alguni and D. Paul, "Island-based cuckoo search with elite opposition-based learning and multiple mutation methods for solving discrete and continuous optimization problems," in *Review at Soft Computing*. Cham, Switzerland: Springer, 2021, doi: 10.21203/rs.3.rs-773831/v1.
[41] S.-S. Guo, J.-S. Wang, and X.-X. Ma, "Improved bat algorithm based on multipopulation strategy of island model for solving global function optimization problem," *Comput. Intell. Neurosci.*, vol. 2019, Aug. 2019, Art. no. 6068743, doi: 10.1155/2019/6068743.
[42] M. A. Al-Betar, M. A. Awadallah, I. Abu Doush, A. I. Hammouri, M. Mafarja, and Z. A. A. Alyasseri, "Island flower pollination algorithm for global optimization," *J. Supercomput.*, vol. 75, no. 8, pp. 5280–5323, Aug. 2019.
[43] M. A. Al-Betar, M. A. Awadallah, A. T. Khader, and Z. A. Abdalkareem, "Island-based harmony search for optimization problems," *Expert Syst. Appl.*, vol. 42, no. 4, pp. 2026–2035, Mar. 2015.
[44] L. de Lima Corrêa and M. Dorn, "A multi-population memetic algorithm for the 3-D protein structure prediction problem," *Swarm Evol. Comput.*, vol. 55, Jun. 2020, Art. no. 100677, doi: 10.1016/j.swevo.2020.100677.
[45] M. Łącki, "An adaptive island model of population for neuroevolutionary ship handling," *Polish Maritime Res.*, vol. 28, no. 4, pp. 142–150, Jan. 2022, doi: 10.2478/pomr-2021-0056.
[46] R. Vafashoar and M. R. Meybodi, "A multi-population differential evolution algorithm based on cellular learning automata and evolutionary context information for optimization in dynamic environments," *Appl. Soft Comput.*, vol. 88, Mar. 2020, Art. no. 106009, doi: 10.1016/j.asoc.2019.106009.
[47] J. Józefowska and J. Węglarz, "On a methodology for discrete–continuous scheduling," *Eur. J. Oper. Res.*, vol. 107, no. 2, pp. 338–353, Jun. 1998, doi: 10.1016/S0377-2217(97)00346-9.
[48] I. Harjunkoski, C. T. Maravelias, P. Bongers, P. M. Castro, S. Engell, I. E. Grossmann, J. Hooker, C. Méndez, G. Sand, and J. Wassick, "Scope for industrial applications of production scheduling models and solution methods," *Comput. Chem. Eng.*, vol. 62, pp. 161–193, Mar. 2014, doi: 10.1016/j.compchemeng.2013.12.001.
[49] H. Lee and C. T. Maravelias, "Combining the advantages of discrete- and continuous-time scheduling models: Part 1. Framework and mathematical formulations," *Comput. Chem. Eng.*, vol. 116, pp. 176–190, Aug. 2018, doi: 10.1016/j.compchemeng.2017.12.003.
[50] H. Lee and C. T. Maravelias, "Combining the advantages of discrete- and continuous-time scheduling models: Part 2. Systematic methods for determining model parameters," *Comput. Chem. Eng.*, vol. 128, pp. 557–573, Sep. 2019, doi: 10.1016/j.compchemeng.2018.10.020.
[51] R. Różycki and J. Węglarz, "Solving a power-aware scheduling problem by grouping jobs with the same processing characteristic," *Discrete Appl. Math.*, vol. 182, pp. 150–161, Feb. 2015, doi: 10.1016/j.dam.2013.11.003.
[52] R. Różycki and J. Węglarz, "Improving the efficiency of scheduling jobs driven by a common limited energy source," in *Proc. MMAR*, Międzyzdroje, Poland, 2018, pp. 932–936, doi: 10.1109/MMAR.2018.8486126.
[53] J. Joózefowska, M. Mika, R. Rózycki, G. Waligóra, and J. Weglarz, "Solving the discrete-continuous project scheduling problem via its discretization," *Math. Methods Oper. Res.*, vol. 52, no. 3, pp. 489–499, Dec. 2000, doi: 10.1007/s001860000094.
[54] M. Bartusch, R. H. Möhring, and F. J. Radermacher, "Scheduling project networks with resource constraints and time Windows," *Ann. Oper. Res.*, vol. 16, no. 1, pp. 199–240, Dec. 1988, doi: 10.1007/BF02283745.
[55] N. Damak, B. Jarboui, P. Siarry, and T. Loukil, "Differential evolution for solving multi-mode resource-constrained project scheduling problems," *Comput. Oper. Res.*, vol. 36, pp. 2653–2659, Sep. 2009.

ALEKSANDER SKAKOVSKI received the M.S. degree in computer science from Kaunas Technical University, Kaunas, Lithuania, in 1983, and the Ph.D. degree in computer science from the Technical University of Gdańsk, Gdańsk, Poland, in 1997.

He is currently an Assistant Professor with the Department of Information Systems, Gdynia Maritime University, Gdynia, Poland. He is the coauthor of a book and about 30 articles. His research interests include artificial intelligence, software reliability engineering, task scheduling and assignment problems, and parallel and distributed computing.

Dr. Skakovski is a member of KES International Scientific Community.

PIOTR JĘDRZEJOWICZ (Member, IEEE) received the Ph.D. and Habilitation degrees in economics and operational research from Gdańsk University, Poland.

He is currently a Full Professor and the Chair of the Information Systems with Gdynia Maritime University, Gdynia, Poland. He has authored several books, numerous book chapters, and over 200 papers published in the international scientific journals and proceedings. During his career, he has been a Visiting Professor in Germany, U.K., China, Sweden, and a Research Fellow with the School of Computer Science, McGill University, Montreal. His research interests include machine learning, software reliability engineering, and decision support systems.

Prof. Jędrzejowicz is an Elected Member of the Committee of Computer Science, Polish Academy of Science. During the present term, he is chairing the Polish Chapter, IEEE SMC Society.