SYNCHRONOUS AND ASYNCHRONOUS MIGRATION IN ADAPTIVE DIFFERENTIAL EVOLUTION ALGORITHMS

*Petr Bujok**

Abstract: The influence of synchronous and asynchronous migration on the performance of adaptive differential evolution algorithms is investigated. Six adaptive differential evolution variants are employed by the parallel migration model with a star topology. Synchronous and asynchronous migration models with various parameters settings were experimentally compared with non-parallel adaptive algorithms in six shifted benchmark problems of dimension $D = 30$. Three different ways of exchanging individuals are applied in a synchronous island model with a fixed number of islands. Three different numbers of sub-populations are set up in an asynchronous island model. The parallel synchronous and asynchronous migration models increase performance in most problems.

Key words: *Differential evolution, parallel migration model, synchronous migration, asynchronous migration, benchmark problems, experimental comparison*

Received: July 15, 2012
Revised and accepted: November 29, 2012

1. Introduction

A lot of fields of research and industry hide problems that need to be optimized. Such optimization can be performed by several methods. In the case of problems where analytical solutions are not possible, biologically inspired optimization algorithms are often able to find an acceptable solution. A major group of biologically inspired algorithms are evolutionary algorithms (EA), which use several natural properties of the individuals. EA are typified by using randomization and, thus, EA can be successful in problems where a deterministic solution is not possible. On the other hand, solving problems by EA can last unexpectedly long. The search process of EA is managed by control parameters, and setting of the values of these control parameters is crucial for computational costs and precision of the algorithm.

Optimization in EA is generalized to search the global minimum of the problem, which is simply defined (1). The global minimization problem is formed as follows:

*Petr Bujok
Department of Informatics and Computers, University of Ostrava, petr.bujok@osu.cz

$$\text{minimize } f(\boldsymbol{x}),\ \ \boldsymbol{x} \in \Omega,\ \ f(\boldsymbol{x}) : \Omega \to \mathbb{R},\ \ \Omega \subseteq \mathbb{R}^D, \tag{1}$$

where $f(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)$ is the real-value objective function and D is the dimension of the problem.

The search space is boundary-constrained (2) for many continuous problems, i.e. the domain Ω is defined by:

$$\Omega = \prod_{i=1}^{D} [a_i, b_i],\ \ a_i < b_i\ , \tag{2}$$

and the goal is to find such a point $\boldsymbol{x}^*$ fulfilling condition:

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}),\ \ \forall \boldsymbol{x} \in \Omega\ . \tag{3}$$

In difficult optimization tasks where deterministic algorithms are not able to find the solutions with acceptable time demands, solutions can be searched heuristically. By application of stochastic algorithms, the area of potential solutions Ω is heuristically explored. One of the simplest and efficient stochastic algorithms is the differential evolution (DE) [20]. Standard DE and its adaptive variants are frequently used to solve very hard problems of practice. Although differential evolution is a very efficient algorithm, time requirements for a good solution in high-dimensional tasks are often too high, see e.g. [31]. Parallel models of evolutionary algorithms are able to distribute computations into several processes and thus reduce the total time demands of the search. Compared to standard non-parallel DE, migration applied in parallel models brings a new feature to the evolutionary process, which can increase the performance of the DE algorithm.

The aim of this paper is to study the influence of migration in various parallel models on the performance of adaptive DE algorithms. An increase of search performance is crucial in large-dimension problems [19, 33]. Any possible improvement of the search speed is important for getting an acceptable solution in reasonable time. That is why the influence of migration in EA is worth studying. We do not focus on possible increase of efficiency by parallel implementation but on the influence of migration. Thus, comparative experiments are carried out on a single-processor PC in a pseudo-parallel mode.

This paper is organized as follows. The differential evolution algorithm and its state-of-the-art adaptive variants are described in Section 2. Parallel models in evolutionary algorithms in brief and a novel asynchronous parallel island model of an adaptive differential evolution algorithm with a ring topology are summarized in Section 3. Experimental settings and test functions are presented in Section 4. Results of experimental comparison are summarized in Section 5. Benefits of the parallel synchronous and asynchronous island model are outlined in the last section.

2. Differential Evolution Algorithm

Differential evolution is one of the most frequently used evolutionary optimization techniques introduced in the 1990s by Storn and Price [26, 27]. In DE, the population of potential solutions is developed by evolutionary operators in order to

find the global minimum of the problem $\boldsymbol{x}^*$. A pseudo-code of the DE algorithm is depicted in Algorithm 1. The population of individuals P is randomly initialized in Ω and the 0 generation is completed. For next generations, for each individual $\boldsymbol{x}_i$ of the population a new trial vector $\boldsymbol{y}$ is created by using the evolutionary operators (*mutation* and *crossover*). A better point of the couple of $\{\boldsymbol{x}_i, \boldsymbol{y}\}$ is then selected to the new generation of population, i.e. a new trial point ($\boldsymbol{y}$) replaces the old one ($\boldsymbol{x}_i$) if $f(\boldsymbol{y}) \leq f(\boldsymbol{x}_i)$. The mutation is controlled by the input parameter F, $F \in (0, 2]$, another input parameter CR, $CR \in (0, 1)$, controls the intensity of crossover. A combination of mutation and crossover is called DE strategy, usually abbreviated by DE/m/n/c, where m stands for the kind of mutation, n for the number of differences in mutation, and c for the type of crossover.

Algorithm 1 Differential evolution algorithm

```
initialize population P = {x_1, x_2, ..., x_N}
while stopping condition not reached do
  for i = 1, 2, ..., N do
    create a new trial vector y
    if f(y) ≤ f(x_i) then
      y → Q
    else
      x_i → Q
    end if
  end for
  P ← Q
end while
```

The choice of the DE strategy, population size N and $[F, CR]$ control parameters settings are crucial for the speed and the quality of the evolutionary process. Setting of DE control parameters by a trial-and-error method is time-consuming. Many studies and experimental comparisons have been carried out in order to find a proper control parameters setting for a wider class of problems [7, 12, 13, 20, 23, 27]. However, their results have not lead to unique recommendations that could be commonly applied. Due to this fact a great effort has been expended to develop some adaptive mechanisms of DE parameters to make the applications more effective, for a summary see [10]. Several efficient variants of DE were proposed or applied in [14, 15], three DE variants were successfully used for the artificial neural network learning in [4].

2.1 Adaptive differential evolution variants in comparison

Four self-adaptive DE variants (*jDE* [5], *JADE* [35], *SaDE* [21], and *EPSDE* [16]) are currently considered as the state-of-the-art DE variants. Along with these four variants, a variant of competitive DE [29], and a variant of DE based on composite trial vector generation strategies and control parameters [32] are also included in the algorithms for experimental comparison. The principles of the adaptive DE variants used in the experiments are described below and sorted according to the year of their publishing.

jDE [5]: DE strategy DE/rand/1/bin with self-adaptation of F and CR. Other control parameters are set to the values recommended by authors of the algorithm, i.e. the ranges of F and CR values are $[0.1,\ 0.9]$ and $[0,\ 1]$, respectively. Mutation probabilities of F and CR are set to $\tau_1 = \tau_2 = 0.1$.
Competitive b6e6rl [29]: Twelve strategies differing in the type of crossover or the values of F and CR from the strategy pool compete to be selected for the generation of a new trial vector. The probability of a strategy selection is proportional to the previous performance of the strategy. Additional parameters controlling the competition are set to their recommended values, $n_0 = 2$ and $\delta = 1/60$.
SaDE [21]: Four mutation strategies and the parameter CR are self-adapted, where the parameters performance in the previous LP generations influences their following values. Parameter F is generated randomly from the normal distribution with a mean and standard deviation of 0.5 and 0.3, respectively. The value of $LP = 50$ is used in our experiments.
JADE [35]: DE variant using a newly proposed current-to-pbest mutation with external archive and self-adaptation of the values of F and CR. The size of the archive is set to N, $\boldsymbol{x}_{\mathrm{pbest}}$ is randomly chosen from $100p\,\%$ best individuals. The value of $p = 0.05$ is used in our experiments, $c = 0.1$ is used for adaptation of the F and CR.
EPSDE [16]: Self-adaptive DE using an ensemble of mutation strategies and parameter values. The triplet (*strategy*, F, CR) is encoded along with each individual of the population. If the current vector produces a successful trial vector entering next generation, its triplet (*strategy*, F, CR) survives to next generation and the successful triplet is also stored in auxiliary memory. Otherwise, the triplet (*strategy*, F, CR) is randomly re-initialized from the respective pools or from the stored successful triplets. The length of memory with the successful triplets is set to $LP = N$.
CoDE [32]: Composite DE variant used in experiments published in [32]. Three mutation strategies with parameters assigned randomly from respective pools are used in each attempt to generate an individual for next generation. The best point of the triplet is used as a new trial point. A modified CoDE0 variant performing better than the original CoDE [30] is used in experiments.

3. Parallel Island Model Applied to Differential Evolution

The parallel models enable exploring the decision space and decrease the risk of trapping at the local minima. Over the past years several parallel models based on approaches of parallelization of evolutionary algorithms have been developed: *master-slave*, *neighborhood*, *island* and their *hybrid* variants [1].

Different types of topologies can be used on the island model for DE. The most frequently used topologies on the island model are *ring* and *star*. These topologies applied in the DE environment were compared in [6]. The control parameters of the island models with coarse-grained distributed population were studied in [22]. There are several parameters controlling migration of information among sub-po-

pulations: *number of migrants*, *selection policy*, *integration policy* (adopt policy), *migration criterion* and *selection the islands* [9, 11, 25].

The selection policy defines selection of the individuals in donor sub-population. The selected individuals are located according to the integration policy into the sub-population of the selected recipient island. The migration criterion defines the condition when the migration is performed. There are two types of migration determined by the migration criterion: *synchronous* and *asynchronous* migration.

In this work, the island model is applied and six different variants of adaptive differential evolution algorithms are used. The population of potential solutions on the island model is initialized randomly in Ω and divided into k independent sub-populations $P_1, P_2, \ldots, P_k$. Each sub-population is associated to one island. The population of N individuals is divided, which means that all the sub-populations are of the same size $N_p = N/k$. After initialization, each sub-population is developed independently by differential evolution until the condition for the migration is reached, the islands exchange information about their sub-populations. In this experimental comparison, two types of migration are used, i.e. a synchronous and an asynchronous type. After the migration, independent development of the sub-populations continues until next condition of the migration. These processes are repeated until the total stopping condition (4) is satisfied, in other words the evolution continues until the whole stopping condition is met. The total stopping condition in all the compared algorithms is considered as:

$$nfe < D \times maxnfe \quad \text{AND} \quad worst_{\mathrm{f}} - best_{\mathrm{f}} > \varepsilon, \tag{4}$$

where ($best_{\mathrm{f}}$) and ($worst_{\mathrm{f}}$) are the best and the worst function values in the whole model (unified all the sub-populations), *nfe* is the current number of function evaluations, ε and *maxnfe* are the input parameters. We suppose that such a distributed approach might allow to reduce the time demands of searching an acceptable approximation of the global minimum point.

3.1 Synchronous migration

In a synchronous migration, all islands send and adopt migrants at one moment, which happens after a specified number of generations of the evolutionary process. Migration in the synchronous mode is centrally controlled, i.e. all the islands wait until the slowest island finishes evolution of its sub-population. Such a model is very simple, but a drawback is that any slow-converged sub-population causes slowdown of the whole algorithm.

A pseudo-code of a synchronous island model of DE is depicted in Algorithm 2. At first, the whole pseudo-randomly initialized population is divided into k sub-populations placed to the islands. Several generations of each sub-population (*nde*) are performed, where *nde* is the input parameter and its value is crucial for the efficiency of a synchronous model. After *nde* generations all the islands exchange the selected individuals, i.e. the migration is performed and one epoch is done. In a synchronous model in our experiments, three different ways of selecting migrants on islands are compared and described in Section 4.

After the migration of all islands, an epoch is done (counter of the epochs is increased by one) and the development of the sub-populations by DE continues

Algorithm 2 Synchronous island model of DE with ring topology

```
initialize sub-populations P_j, j = 1, 2, ..., k
epoch = 1
while stopping condition (4) not reached do
  for j = 1, 2, ..., k do
    perform nde generations of each island by DE
  end for
  migrate selected individuals between islands by the unidirectional ring
  epoch = epoch + 1
end while
```

until the total stopping condition (4) is met. A synchronous parallel model of DE in the experiment is linked by the ring topology with an unidirectional communication Fig. 1a.

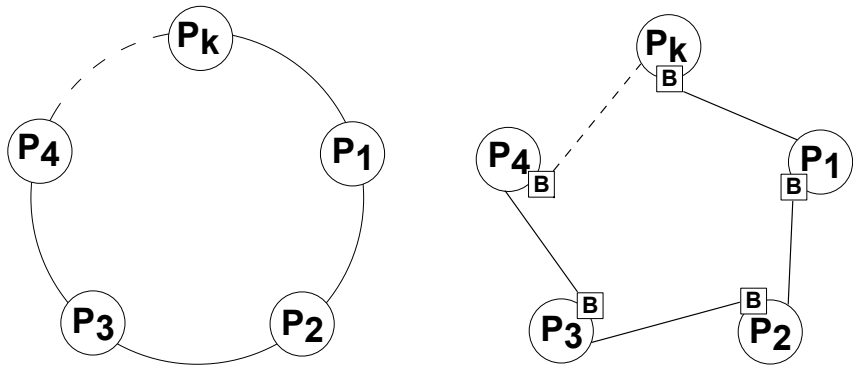

Fig. 1 *The ring topology of* **a)** *synchronous,* **b)** *asynchronous island model.*

3.2 Asynchronous migration

In an asynchronous migration, each island sends and adopts determined individuals if its own migration criterion is achieved. In other words, the individuals from the island migrate independently on the other islands and evolution processes of fast islands are not hindered by slow islands. The migration condition, i.e. definition of the evolution level when the island adopts and sends individuals from/to the specified islands, plays an important role in an asynchronous migration. Thus, migration condition provides each island with an ability to migrate if its sub-population is ready for migration independently of other islands. An island with a mature subpopulation does not wait for slower islands and this is a dynamic form of the communication. It has been shown that asynchronous migration operation can increase the convergence rate of evolutionary algorithms compared to a synchronous migration [2, 28].

A pseudo-code of the asynchronous island model tested in this study is shown in Algorithm 3. The initialized population is divided among k islands according

to the ring topology depicted in Fig. 1b. Contrary to synchronous migration, no centrally controlled migration of all the islands is performed. An island whose sub-population is sufficiently mature migrates (i.e. sends and adopts) the chosen individuals with other selected island. Maturity of a certain subpopulation is compared with the migration condition (5). For easier implementation of the asynchronous communication among the islands, an auxiliary *buffer* memory is used (depicted as small squares in Fig. 1b). This buffer is used to hold current migrants of all islands, i.e. buffer has size 'number of migrated individuals' × 'number of islands' and it contains only the last migrants of each island.

When migration of the island is completed, the current epoch of the island ends and the counter of epochs of the island is increased by one. It is obvious that the number of the DE generations in one epoch can vary from island to island AND the number of the DE generations on the island in the epochs may be different as well. The epoch on the jth island is finished if the following condition is fulfilled:

$$\begin{gathered} worst_{\text{new}} - best_{\text{new}} < max(\tfrac{1}{exp(epoch_j - 1)}, 1 \times 10^{-4}) \text{ AND } best_{\text{old}} - best_{\text{new}} > \tfrac{\varepsilon}{100} \\ \text{OR} \\ newgen_j - oldgen_j \geq N, \end{gathered} \tag{5}$$

where $worst_{\text{new}}$ and $best_{\text{new}}$ are function values of the currently worst and the currently best individuals, $best_{\text{old}}$ is the function value of the best individual in the preceding generation, ε is an input parameter, difference of $(newgen_j - oldgen_j)$ means the number of generations completed till now in the current epoch, and $epoch_j$ is the value of the epoch counter. All the values except for ε and N are related to the jth island.

Condition (5) defines two ways how to finish the epoch on the jth island:

a) the maximum difference in the function values on the island has become small and there is at least a small progress in the quality of the sub-population development when two last generations are compared,

b) or the number of generation in the current epoch is large without reaching any significant progress in the sub-population development.

Algorithm 3 Asynchronous island model of DE with ring topology

```
initialize sub-populations P_j, j = 1, 2, ..., k
initialize migration counter of length k, epoch = (1, 1, ..., 1)
while stopping condition (4) not reached do
  for j = 1, 2, ..., k do
    while migration condition (5) of jth sub-population is not reached do
      perform generation of jth island by DE
    end while
    select best and replace old best on buffer
    adopt best of neighbor from buffer
    epoch_j = epoch_j + 1
  end for
end while
```

4. Experiments

Synchronous and asynchronous parallel algorithms are compared with six state-of-the-art non-parallel adaptive DE. One hundred runs are performed for all the algorithms and test functions. The main control parameters for all DE variants are set up: population size $N = 60$, $maxnfe = 20000$ and $\varepsilon = 1 \times 10^{-6}$. The settings of parameters controlling the parallel models are mostly based on author's previous experiments [6, 8]. Synchronous parallel variants used in the experiments differ in various ways of the migrants' selection. Asynchronous variants differ in the number of islands resulting in different sub-population sizes.

In the synchronous variants, the individuals are distributed by an unidirectional ring topology to 6 islands and the sub-populations sizes are $N/6 = 10$. A fixed number of DE generations $nde = 5$ is performed before the migration occurs. Three labels in Tab. I specify how the individuals are selected for migration between the islands in the unidirectional ring. In the first (synch1) variant, the best individuals are exchanged between next immediate islands where they replace the worst individuals and, moreover, mig randomly chosen individuals are exchanged, where $mig = 4$ is the input parameter. In (synch2) variant, the best individuals are also exchanged between immediate individuals where they replace the worst individuals. Moreover, the function values of the best individuals in the neighboring islands are compared and a better island sends $mig+1$ randomly chosen individuals to the worse island and the worse one island sends $mig - 1$ randomly chosen individuals to the better one. In the last (synch3) synchronous variant, a half of individuals in the neighboring sub-populations are randomly chosen and exchanged.

In the asynchronous algorithms, the individuals are distributed to islands linked into an unidirectional ring. Asynchronous migration is implemented using a buffer memory. Algorithms with three different numbers of islands and sub-populations sizes are compared, the same condition of migration (5) is used in all algorithms. In these three variants, individuals are distributed to the islands in the following manner: two islands 30 individuals per island (this variant is called asynch1 hereafter), three islands, 20 individuals per island (asynch2) and four islands, 15 individuals per island (asynch3), see also Tab. I.

When the asynchronous migration occurs, the best individual of the migrated island is copied to the buffer and the worst individual is replaced by the best individual from the previous island in the unidirectional ring. In more detail, the best individual from the migrated island is sent to the buffer memory for the previous island and the best individual of the next island is adopted from the buffer memory to the worst position in the migrated island. In other words, the migrants are selected according to the following rules:

- The individual with the least function value in migrating sub-population on the jth island replaces the individual on jth position in the buffer and waits to be adopted by the $(j-1)$th island.

- The individual on the $(j+1)$th position in the buffer is sent to the currently migrating island (j), where it replaces the worst individual.

The numbering of islands by j is considered in a circular manner.

synchronous		asynchronous	
abbr.	description	abbr.	description
synch1	1 best and 4 random	asynch1	2 islands × 30 individuals
synch2	1 best and 5 or 3 random	asynch2	3 islands × 20 individuals
synch3	half population random	asynch3	4 islands × 15 individuals

Tab. I *Labels of parallel DE variants used in the presentation of results.*

In this study, six non-parallel, three synchronous parallel and six asynchronous parallel adaptive DE algorithms are compared. The abbreviated labels of algorithms used in the presentation of results are shown in Tab. I. Thus, we can clarify if the parallel model is able to reduce time demands with keeping the level of reliability.

Six well-known scalable test functions [3, 18, 20, 27] shown in Tab. II are used as benchmark at level of problem dimension $D = 30$. In the case of four following test functions their global optimum point in a non-shifted form is in the center of solution area Ω, $\boldsymbol{x}^* = (0, 0, \ldots, 0)$, which makes the search for solution easier for many stochastic algorithms. Therefore, we use them in their shifted version. The shifted function is evaluated at the point $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}$, $\boldsymbol{o} \in \Omega$, $\boldsymbol{o} \neq (0, 0, \ldots, 0)$. The shift $\boldsymbol{o}$ is generated randomly from uniform D-dimensional distribution before each run of algorithm. The global optimum of the test problem is $\boldsymbol{x}^* = \boldsymbol{o}$ for all the shifted functions.

problem	abbr.	function	decision space
Ackley	f_1	$-20\,\exp\left(-0.2\,\sqrt{\frac{1}{D}\sum_{j=1}^{D}(x_j - o_j)^2}\right) -$ $-\exp\left(\frac{1}{D}\sum_{j=1}^{D}\cos 2\pi(x_j - o_j)\right) + 20 + \exp(1)$	$[-30, 30]^D$
DeJong	f_2	$\sum_{j=1}^{D}(x_j - o_j)^2$	$[-5.12, 5.12]^D$
Griewank	f_3	$\sum_{j=1}^{D}\frac{(x_j - o_j)^2}{4000} - \prod_{j=1}^{D}\cos\left(\frac{(x_j - o_j)}{\sqrt{j}}\right) + 1$	$[-400, 400]^D$
Rastrigin	f_4	$10\,D + \sum_{j=1}^{D}\left[(x_j - o_j)^2 - 10\cos(2\pi(x_j - o_j))\right]$	$[-5.12, 5.12]^D$
Rosenbrock	f_5	$\sum_{j=1}^{D-1}\left[100(x_j^2 - x_{j+1})^2 + (1 - x_j)^2\right]$	$[-2.048, 2.048]^D$
Schwefel	f_6	$418.98288727\;D - \sum_{j=1}^{D} x_j \sin(\sqrt{\mid x_j \mid})$	$[-500, 500]^D$

Tab. II *Test problems.*

5. Results

The results of the comparison are in Tab. III and Tab. IV. Each table contains the results of the non-parallel adaptive DE algorithms and the adequate parallel island algorithms for a better comparison.

		non-parallel DE			synch1			synch2			synch3		
alg	fcn	nfe	fnr	R	nfe_r	fnr_r	R	nfe_r	fnr_r	R	nfe_r	fnr_r	R
b6e6rl	f_1	230179	213310	100	**-8**	**-7**	**100**	**-9**	**-7**	**100**	**-11**	**-9**	**98**
CoDE0	f_1	600000	0	0	**-10**		**92**	**-11**		**88**	**-13**		**84**
EPSDE	f_1	600000	212880	0	0		0	0		0	0		0
jDE	f_1	600000	0	0	0		0	0		0	0		0
JADE	f_1	302861	279637	95	**-14**	**-10**	**96**	**-17**	**-13**	**94**	**-19**	**-16**	**96**
SaDE	f_1	600000	0	0	0		0	0		0	0		0
b6e6rl	f_2	37472	27004	100	**-27**	**-28**	**100**	**-27**	**-28**	**100**	**-30**	**-30**	**100**
CoDE0	f_2	47670	34928	100	**-68**	**-68**	**100**	**-68**	**-68**	**100**	**-68**	**-68**	**100**
EPSDE	f_2	23818	17255	100	19	19	100	9	**-71**	**100**	16	16	100
jDE	f_2	32559	23194	100	**-20**	**-19**	**100**	**-20**	**-18**	**100**	**-21**	**-20**	**100**
JADE	f_2	13470	9832	100	20	19	100	19	19	100	21	21	100
SaDE	f_2	20947	15611	100	10	9	100	10	9	100	11	10	100
b6e6rl	f_3	51934	41482	100	**-29**	**-30**	**98**	**-28**	**-29**	**100**	**-30**	**-32**	**96**
CoDE0	f_3	68010	55261	100	**-68**	**-69**	**98**	**-69**	**-69**	**100**	**-69**	**-69**	**100**
EPSDE	f_3	32438	25851	100	15	14	100	7	-77	92	14	13	96
jDE	f_3	43690	34458	100	**-20**	**-19**	**98**	**-21**	**-20**	**98**	**-21**	**-21**	**98**
JADE	f_3	22759	15977	93	22	13	98	7	10	96	31	15	96
SaDE	f_3	28312	19833	87	10	25	100	6	23	100	11	28	92
b6e6rl	f_4	73402	62829	100	**-10**	**-8**	**100**	**-11**	**-8**	**100**	**-13**	**-10**	**100**
CoDE0	f_4	268062	255236	100	**-67**	**-67**	**100**	**-67**	**-67**	**100**	**-67**	**-67**	**100**
EPSDE	f_4	251678	245034	100	102	104	100	85	77	100	96	98	100
jDE	f_4	137343	125573	98	**-3**	0	**100**	**-3**	1	**100**	**-5**	**-1**	**100**
JADE	f_4	67801	60428	100	9	6	100	9	7	100	8	6	100
SaDE	f_4	79901	73594	100	85	105	28	88	108	22	86	108	30
b6e6rl	f_5	147185	134946	100	17	20	98	19	23	100	33	37	94
CoDE0	f_5	359860	338333	100	**-57**	**-57**	**100**	**-57**	**-57**	**100**	**-54**	**-54**	**100**
EPSDE	f_5	163082	151356	100	59	60	98	65	12	100	70	71	100
jDE	f_5	377216	332957	97	59		0	59		0	59		0
JADE	f_5	76440	72055	93	85	85	96	83	82	100	99	97	98
SaDE	f_5	241504	210633	95	148		0	148		0	148		0
b6e6rl	f_6	64243	53669	100	**-18**	**-17**	**100**	**-18**	**-16**	**100**	**-19**	**-17**	**100**
CoDE0	f_6	149849	137044	100	**-68**	**-68**	**100**	**-68**	**-68**	**100**	**-68**	**-68**	**100**
EPSDE	f_6	74555	67297	99	68	74	100	48	33	100	63	70	100
jDE	f_6	60093	50229	99	**-13**	**-10**	**100**	**-12**	**-9**	**100**	**-14**	**-11**	**100**
JADE	f_6	57994	52400	77	17	13	100	16	12	96	15	10	98
SaDE	f_6	53004	46724	100	22	26	96	20	23	100	22	25	100

Tab. III *Results of experimental comparison of synchronous variants with non-parallel DE.*

The values of non-parallel DE in the tables:

- Reliability rate (R) is the number of the algorithm runs, where the function value of the best find solution $\boldsymbol{x}_{\min}$ is: $f(\boldsymbol{x}_{\min}) < 1 \times 10^{-4}$.
- Number of the function evaluations in non-parallel DE when the algorithm firstly approximates to the global minima point under the value 1×10^{-4} (fnr).
- Total function evaluations of non-parallel DE (nfe).

For the parallel models, the relative number of function evaluations (in percent) with respect to non-parallel DE, computed by equation (6) is given in nfe_r column.

$$nfe_r = \frac{nfe_p - nfe_s}{nfe_s} \times 100, \tag{6}$$

where nfe_p is the number of function evaluations for the parallel version and nfe_s for the non-parallel one. The relative number of function evaluations (fnr_r) for the parallel variants when the algorithm firstly approximates to the function value in the global minimum point with difference less than 1×10^{-4} is computed by equation (7).

$$fnr_r = \frac{fnr_p - fnr_s}{fnr_s} \times 100, \tag{7}$$

where fnr_p denotes the number of the function evaluations of parallel version DE and fnr_s for the non-parallel DE. Other measures for the comparison of stochastic methods are proposed in work [17].

The results of the parallel models are compared with the corresponding non-parallel variants in Tab. III and Tab. IV. The results of parallel models are emphasized as follows. If the parallel variant keeps or increases the reliability R of the non-parallel variant, the parallel R value is underlined. If the speed of the parallel variant (equations (6) and (7)) is better than of the non-parallel variant, the parallel values (nfe_r, fnr_r or both) are underlined. When the parallel variant outperforms non-parallel in speed and value of reliability R is at least the same, the values of the parallel variant are printed in bold. For easier evaluation of the migration benefit, the number of cases in which the parallel models achieve a higher speed and reliability than the non-parallel variants (complete results are presented in Tab. III and Tab. IV) are summarized in Tab. V. The parallel models achieve a higher performance than the non-parallel variants in almost half of all the test problems.

		non-parallel DE			asynch1			asynch2			asynch3		
alg	fcn	nfe	fnr	R	nfe_r	fnr_r	R	nfe_r	fnr_r	R	nfe_r	fnr_r	R
b6e6rl	f_1	230179	213310	100	-56	-46	45	-59	-39	18	-66	-44	6
CoDE0	f_1	600000	0	0	**-55**		**30**	**-59**		**27**	**-66**		**4**
EPSDE	f_1	600000	212880	0	-17		0	-32		0	**-82**	**-27**	**1**
jDE	f_1	600000	0	0	-49		0	-57		0	-40		0
JADE	f_1	302861	279637	95	**-30**	**-29**	**89**	-61	-37	2	-25		0
SaDE	f_1	600000	0	0	-21		0	-40		0	-50		0
b6e6rl	f_2	37472	27004	100	3	**-1**	**100**	3	0	100	3	1	100
CoDE0	f_2	47670	34928	100	**-69**	**-67**	**100**	**-68**	**-66**	**100**	**-67**	**-65**	**100**
EPSDE	f_2	23818	17255	100	20	25	100	17	23	100	20	29	100
jDE	f_2	32559	23194	100	**-17**	**-12**	**100**	**-19**	**-14**	**100**	**-15**	**-7**	**100**
JADE	f_2	13470	9832	100	7	11	100	17	23	100	34	44	100
SaDE	f_2	20947	15611	100	5	11	100	24	26	100	32	37	100
b6e6rl	f_3	51934	41482	100	95	6	100	60	9	100	40	5	100
CoDE0	f_3	68010	55261	100	**-67**	**-69**	**100**	**-65**	**-68**	**100**	**-64**	**-68**	**100**
EPSDE	f_3	32438	25851	100	23	23	99	21	23	97	28	31	97
jDE	f_3	43690	34458	100	**-16**	**-13**	**100**	**-16**	**-13**	**95**	**-9**	**-4**	**100**
JADE	f_3	22759	15977	93	27	**-31**	**98**	42	19	97	69	41	100
SaDE	f_3	28312	19833	87	12	34	90	29	47	98	38	60	100
b6e6rl	f_4	73402	62829	100	**-23**	**-32**	**100**	**-22**	**-31**	**100**	**-21**	**-30**	**100**
CoDE0	f_4	268062	255236	100	**-69**	**-70**	**100**	**-70**	**-70**	**100**	**-72**	**-73**	**100**
EPSDE	f_4	251678	245034	100	-19	-19	56	-25	-23	34	-30	-30	30
jDE	f_4	137343	125573	98	**-19**	**-17**	**100**	**-21**	**-19**	**97**	-26	-25	79
JADE	f_4	67801	60428	100	**-2**	**-5**	**100**	**-2**	**-5**	**100**	**-1**	**-5**	**100**
SaDE	f_4	79901	73594	100	59	71	8	34	46	14	17	28	11
b6e6rl	f_5	147185	134946	100	308		0	308		0	308		0
CoDE0	f_5	359860	338333	100	**-58**	**-58**	**100**	**-50**	**-48**	**100**	**-42**	**-40**	**99**
EPSDE	f_5	163082	151356	100	46	49	99	87	93	99	129	137	99
jDE	f_5	377216	332957	97	59		0	59	66	4	59	72	1
JADE	f_5	76440	72055	93	53	50	94	124	111	97	200	185	100
SaDE	f_5	241504	210633	95	91	115	68	148		0	148		0
b6e6rl	f_6	64243	53669	100	**-13**	**-21**	**100**	**-12**	**-19**	**100**	**-11**	**-18**	**100**
CoDE0	f_6	149849	137044	100	**-68**	**-69**	**100**	**-68**	**-69**	**100**	**-69**	**-70**	**100**
EPSDE	f_6	74555	67297	99	36	38	99	33	35	98	34	36	99
jDE	f_6	60093	50229	99	**-11**	**-8**	**100**	**-13**	**-10**	**100**	**-7**	**-4**	**98**
JADE	f_6	57994	52400	77	9	3	90	16	4	1	13	5	97
SaDE	f_6	53004	46724	100	25	31	98	11	8	97	10	5	99

Tab. IV *Results of experimental comparison of asynchronous variants with non-parallel DE.*

model	type 1	type 2	type 3
synchronous	16	17	16
asynchronous	16	13	13

Tab. V *Summary of variants with higher performance of migration models.*

6. Conclusion

The experimental results show that the new proposed parallel island models are able to increase the performance of adaptive variants of DE. On the other hand, no algorithm employing heuristics methods solves all problems better than the remaining algorithms [34]. Based on the results in Tab. III, Tab. IV and Tab. V, we can conclude:

- Synchronous parallel variants perform better on average than asynchronous migration variants.
- Compared to non-parallel variants, the highest increase in performance for both the synchronous and the asynchronous variants was achieved in the case of CoDE0 algorithm. The parallel variants of jDE, b6e6rl and JADE algorithms exhibited better performance compared to non-parallel versions only in a part of test problems.
- There is no significant difference in the performance among three synchronous migration variants.
- There is no relevant difference in the performance among three asynchronous migration variants but variant async1 outperforms the non-parallel variants more frequently than the others.
- Increase of the performance by migration occurs frequently in five test problems out of six. The performance of parallel models in Rosenbrock problem is better only in the case of CoDE0 algorithm. It may be caused by the nature of Rosenbrock function that is very hard for many search algorithms, which was found in many previous studies. Rosenbrock problem represents a case of a non-separable function with the narrow valley in which the gradient is very small. The fact that the function is not uni-modal for $D > 3$ has been proved recently [24].

Acknowledgement

I would like to thank doc. Ing. Josef Tvrdík, CSc. for his guidance during the research reported here. This work was supported by the University of Ostrava from the project SGS13/PrF/2012.

References

[1] Alba E., Cotta C.: Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation, **6**, 5, 2002, pp. 443–462.

[2] Alba E., Troya J. M.: Analyzing synchronous and asynchronous parallel distributed genetic algorithms. Future Gener. Comput. Syst., **17**, 4, 2001, pp. 451–465.

[3] Ali M. M., Törn A.: Population set based global optimization algorithms: Some modifications and numerical studies. Computers and Operations Research, **31**, 2004, pp. 1703–1725.

[4] Boštík J., Kukal J., Macek K., Van Tran Q.: Learning ann via differential evolution. In: R. Matoušek (ed.) 16th International Conference on Soft Computing MENDEL 2011, Mendel, Brno University of Technology, Brno, 2010, pp. 15–21.

[5] Brest J., Greiner S., Boškovič B., Mernik M., Žumer V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation, **10**, 2006, pp. 646–657.

[6] Bujok P.: Parallel models of adaptive differential evolution based on migration process. In: Aplimat, 10th International Conference on Applied Mathematics, Bratislava, 2011, pp. 357–364.

[7] Bujok P., Tvrdík J.: A comparison of various strategies in differential evolution. In: R. Matoušek (ed.) MENDEL 2011, 17th International Conference on Soft Computing, University of Technology, Brno, 2011, pp. 48–55.

[8] Bujok P., Tvrdík J.: Parallel migration model employing various adaptive variants of differential evolution. In: Lecture Notes in Computer Science, 7269, Springer-Verlag, Berlin Heidelberg, 2012, pp. 39–47.

[9] Cantú-Paz E.: Migration policies, selection pressure, and parallel evolutionary algorithms. Journal of Heuristics, **7**, 4, 2001, pp. 311–334.

[10] Das S., Suganthan P. N.: Differential evolution: A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation, **15**, 2011, pp. 27–54.

[11] Dorronsoro B., Bouvry P.: Improving classical and decentralized differential evolution with new mutation operator and population topologies. IEEE Transactions Evolutionary Computation, **15**, 1, 2011, pp. 67–98.

[12] Feoktistov V.: Differential Evolution in Search of Solution. Springer, 2006.

[13] Gämperle R., Müller S. D., Koumoutsakos P.: A parameter study for differential evolution. In: A. Grmela, N. E. Mastorakis (eds.) Advances in Intelligent Systems Fuzzy Systems, Evolutionary Computing, WSEAS Press, Athens, 2002, pp. 293–298.

[14] Hrstka O., Kučerová A.: Improvements of real coded genetic algorithms based on differential operators preventing premature convergence. Advances in Engineering Software, **35**, 3-4, 2004, pp. 237–246.

[15] Kordík P., Koutník J., Drchal J., Kovářík O., Čepek M., Šnorek M.: Meta-learning approach to neural network optimization. Neural Networks, **23**, 4, 2010, pp. 568–582.

[16] Mallipeddi R., Suganthan P. N., Pan Q. K., Tasgetiren M. F.: Differential evolution algorithm with ensemble of parameters and mutation strategies. Applied Soft Computing, **11**, 2011, pp. 1679–1696.

[17] Mojzeš M., Kukal J., Van Tran Q., Jablonský J.: Performance comparison of heuristic algorithms via multi-criteria decision analysis. In: R. Matoušek (ed.) 17th International Conference on Soft Computing MENDEL 2011, Mendel, Brno University of Technology, Brno, 2011, pp. 244–251.

[18] Molga M., Smutnicki C.: Test functions for optimization needs, 2005. http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf

[19] Olorunda O., Engelbrecht A. P.: Differential evolution in high-dimensional search spaces. In: IEEE Congress on Evolutionary Computation, 2007, pp. 1934–1941.

[20] Price K. V., Storn R., Lampinen J.: Differential Evolution: A Practical Approach to Global Optimization. Springer, 2005.

[21] Qin A., Huang V., Suganthan P.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation, **13**, 2009, pp. 398–417.

[22] Rivera W.: Scalable parallel genetic algorithms. Artificial Intelligence Review, **16**, 2001, pp. 153–168.

[23] Ronkkonen J., Kukkonen S., Price K. V.: Real-parameter Optimization with Differential Evolution. In: IEEE Congress on Evolutionary Computation, 2005, pp. 506–513.

[24] Shang Y. W., Qiu Y. H.: A note on the extended Rosenbrock function. Evolutionary Computation, **14**, 1, 2006, pp. 119–126.

[25] Skolicki Z., De Jong K.: The influence of migration sizes and intervals on island models. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, GECCO '05, ACM, New York, NY, USA, 2005, pp. 1295–1302.

[26] Storn R., Price K. V.: Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. TR-95-012. 1995, Available [ONLINE] http://www.icsi.berkeley.edu/ storn/litera.html

[27] Storn R., Price K. V.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimization, **11**, 1997, pp. 341–359.

[28] Tongchim S., Chongstitvatana P.: Asynchronous migration in parallel genetic programming. In: Proceedings of the 5th Asian Computing Science Conference on Advances in Computing Science, ASIAN '99, Springer-Verlag, London, UK, 1999, pp. 388–389.

[29] Tvrdík J.: Self-adaptive variants of differential evolution with exponential crossover. Analele of West University Timisoara, Series Mathematics-Informatics **47**, 2009, pp. 151–168. URL http://www1.osu.cz/∼tvrdik/ Reprint available [ONLINE].

[30] Tvrdík J.: Modifications of differential evolution with composite trial vector generation strategies. In: Proceedings of SOCO (accepted). Springer, 2012.

[31] Tvrdík J., Poláková R., Bujok P.: A comparison of adaptive differential evolution variants for single-objective optimization. In: R. Matoušek (ed.) 18th International Conference on Soft Computing Mendel 2010, Brno University of Technology, 2012, pp. 132–137.

[32] Wang Y., Cai Z., Zhang Q.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Transactions on Evolutionary Computation, **15**, 2011, pp. 55–66.

[33] Weber M., Neri F.: Contiguous binomial crossover in differential evolution. In: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, J. M. Zurada (eds.) ICAISC (SIDE-EC), Lecture Notes in Computer Science, vol. 7269, Springer, 2012, pp. 145–153.

[34] Wolpert D. H., Macready W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, **1**, 1997, pp. 67–82.

[35] Zhang J., Sanderson A. C.: JADE: Adaptive Differential Evolution with Optional External Archive. IEEE Transactions on Evolutionary Computation, **13**, (2009.)