Abdelbasset Essabri

Mariem Gzara

Taïcir Loukil

A STUDY OF DISTRIBUTED EVOLUTIONARY ALGORITHMS FOR MULTI-OBJECTIVE OPTIMISATION

Abstract

Most popular Evolutionary Algorithms for single multi-objective optimisation are motivated by the reduction of the computation time and the resolution larger problems. A promising alternative is to create new distributed schemes that improve the behaviour of the search process of such algorithms. In the multi-objective optimisation problems, more exploration of the search space is required to obtain the whole or the best approximation of the Pareto front. Almost all proposed Parallel Multi-Objective Evolutionary Algorithms (PMOEAs) are based on the specialisation concept which means dividing the objective and/or the search space then assigning each part to a processor. One processor called the organiser or the coordinator is usually charged to direct the whole algorithm. In this paper, we present a new parallel scheme of multi-objective evolutionary algorithms which is based on a clustering technique. This new parallel algorithm is implemented and compared to three PMOEAs which are cone-separation [1], Divided Range Multi-Objective Genetic Algorithm (DRMOGA) [8] and a Parallel Strength Pareto Evolutionary Algorithm (PSPEA) based on the island model without migration.

Keywords

Parallel computing, multi-objective optimisation, evolutionary algorithms, parallel genetic algorithms, clustering algorithms.

Introduction

Evolutionary Algorithms (EAs) are adaptive methods that have proven successful in the resolution of several optimisation problems. They are based on the genetic evolution process of Darwin. EAs make evolve a set of solutions, called population of individuals. A new population is produced while selecting parents among the most excellent individuals of the "present generation" to perform crossover and mutation. The new population will contain a bigger proportion of features from the best individuals of the previous generation. The search is thus guided towards the most promising regions of the search space. In Multi-objective Optimisation Problems (MOP), a set of conflicting criteria have to be simultaneously optimised. The aim is to find a set of non dominated solutions rather than one solution in the single objective optimisation case. EAs are particularly suitable to solve MOP. They perform well global search, since they simultaneously explore different regions of the search space. To obtain a set of diversified non dominated solutions, Multi-Objective Evolutionary Algorithms (MOEAs) integrate techniques such as elitism and diversity (crowding, sharing) (NSGA-II [4], SPEA-II [15], NPGA-II [6]). These techniques have proven successful in deriving good Pareto optimum solutions. However, their high computing time constitutes a major drawback. Parallel computing has been applied to MOEAs so as to accelerate solving problems [2]. Moreover, parallelism offers a best exploration of the search space by the cooperation between populations evolving with different genetic operators. Several approaches have been proposed to parallelise both EAs and MOEAs. In the single objective case, Parallel Evolutionary Algorithms (PEAs) exploit the intrinsic parallelism in the algorithm. In fact, fitness evaluation, crossover and mutation operators can be performed independently on different individuals and thus can be easily distributed. In the multi-objective case, most Parallel Multi-Objective Evolutionary Algorithms (PMOEAs) are based on the algorithmic "divide to conquer" principle: divide the objective/search space among available processors while favouring migration between sub-populations. Each processor will concentrate its search in a specific region of the search space (specialisation).

In this paper, we propose a new parallel evolutionary algorithm for multi--objective optimisation named „[…] parallel multi-objective evolutionary algorithm with Multi-Front Equitable Distribution" (MFED). MFED, which is based on the island model, uses a clustering technique to divide the global population into sub-populations. MFED is implemented and compared to three

PMOEAs: cone-separation [1], Divided Range Multi-Objective Genetic Algorithm (DRMOGA) [8] and a Parallel Strength Pareto Evolutionary Algorithm (PSPEA) based on the island model without migration.

This paper is organised as follows: Section 1 gives common parallelisation approaches of EAs. Section 2 deals with PMOEAs. Then, in Section 3, the MFED is described. Test results and comparisons are presented and analysed in Section 4. Finally, conclusions will be highlighted.

1. Parallel evolutionary algorithms

Parallel architectures permit to get some very satisfactory results in parallelising EAs. Parallel genetic algorithms are roughly classified into three categories: master-slave population model, island model, and cellular model [2].

Master-slave model

The master-slave model is based on a simple parallelisation of the fitness calculation stage or of the recombination/evaluation steps. In the first case, a single station (called master) manages the algorithm itself (selection/replacement and genetic operators), and sends the performance computation to other stations (called slaves). In the second case, the master centralises the population and manages the selection and the replacement steps. It sends the sub-populations to the workers that execute the recombination and evaluation steps. This model is useful only for a small number of processors and high fitness computing time [1].

Island model

The island model, which is also called coarse-grained model or distributed model, divides the population into small sub-populations. Each of them will evolve in a certain processor, following a traditional diagram to which a stage of migration is added. In other words, every sub-population transmits its good individuals towards the neighbouring sub-populations in a “common pool” (this choice depends on the relative cost of communications between processors). Then, every sub-population receives individuals which are sent by neighbours, or which already exist in the central pool.

The island model modifies the basic genetic algorithm and introduces some new parameters (i.e. the migration strategies and the topology of the network) [2]. It becomes particularly interesting when the number of processors is lower than the population size [1].

This model is better adapted than the precedent one to parallel machines. It is a very popular model since it is very easy to apply on a local network with standard workstations. Moreover, it offers the possibilities that every sub-population can evolve using some different parameters. {?}

Cellular model

The cellular model distributes a unique population among several processors (in general on massively parallel machines). On each processor some individuals (often only one) evolve. Then, the selection, replacement and crossover operations are outperformed between "neighbourhood individuals" for the topology of the processor network. This model is particularly suitable for massively parallel computers with a fast local intercommunication network.

A detailed discussion of parallelisation approaches for EAs is found in [1, 3].

2. Parallel multi-objective evolutionary algorithms

MOEAs look for a whole set of Pareto-optimal solutions. Therefore, they require more exploration of the search space and more computation to characterise the Pareto-optimal front than the single objective EAs. For that reason, several works ([11], [5], [1], [8], etc.) discuss the manner to parallelise them. Since the resolution of MOP aims at finding a set of different trade-off solutions between the objectives, the most natural parallel scheme was to assign different parts of the search/fitness space to different processors. Each process will focus its computation on a specific region (explore one region) of the search space to characterise one area of the global Pareto optimal front. That's why most proposed PMOEAs are based on the island model and deal with the manner of dividing the search space and/or the fitness space between the different processors. In this section we limit our focus on DRMOGA [8] and Cone Separation [1] in order to compare them to our new parallel model.

DRMOGA

Hiroyasu et al. [8] have developed a Divided Range Multi-Objective Genetic Algorithm (DRMOGA) based on an island model where the Pareto--optimum solutions, which are close to each other, are collected by one sub-population. All individuals are gathered in the master process and are again

divided among the different processors. Each sub-population receives a set of N/m individuals (N is the population size and m is the number of processors) selected according to the value of the objective function considered f_i (the objectives are considered in turn). Figure 1 shows the division of the population according to the objective f1 in the case of a bi-objective optimisation problem and three processors. An improvement version of DRMOGA uses a sharing operation for Pareto-optimum solutions when the number of the frontier solutions exceeds a given size.

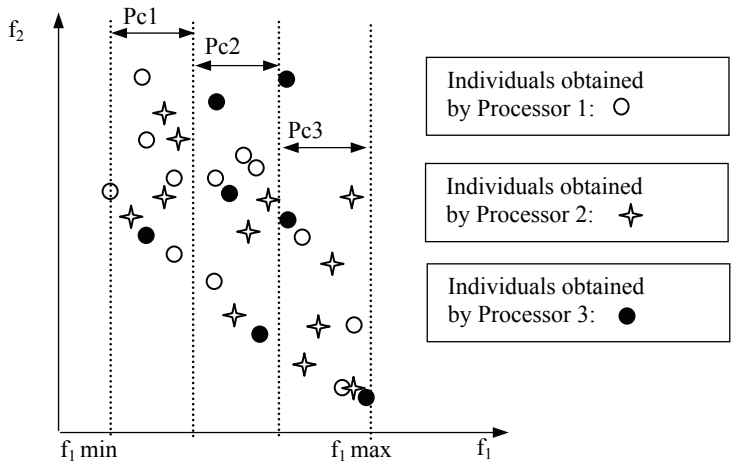

Figure 1. DRMOGA: distribution of the population according to the first objective with 3 processors

Cone separation

Branke et al. [1] normalise the fitness values and then they partition the fitness space into equal cones. In the bi-objective case, the population is within the unit square after normalisation. They start from the reference point (1,1) and divide the 90° angle that encompasses the non-dominated front into equal parts (see Figure 2). The fitness space is renormalised at regular intervals leading to a migration step of individuals to processors specialised on the cone to which they belong.

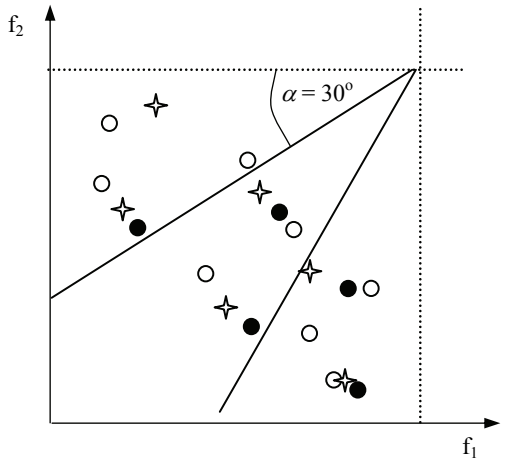

Figure 2. Cone-separation with 2 objectives functions problem and 3 processors

3. Parallel Multi-Objective Evolutionary Algorithms with multi-front equitable distribution

Most PMOEAs are based on the specialisation concept where neighbouring solutions in the search/objective space live and progress on the same island. Thus, these models authorise only crossover of individuals close to each other. This fact may result in a lack of diversity among sub-populations, in a stagnation of the search and thus in a rapid convergence. Moreover, in the case of a multi-objective optimisation problem for which the Pareto Optimal Front (POF) is discontinuous, individuals of the same sub-population may belong to extreme zones of two continuous portions of the Pareto Front (PF).

Our new algorithm is an island model. As the previous PMOEAs, it is based on the algorithmic principle “divide to conquer”. Yet, it is different from the others in the fact that no sub-population focuses its computation on a specific region of the search/fitness space. Each processor focuses its computation over all promising regions of the search space that are already discovered within the global population. In fact, MFED functions as a multi-start optimisation procedure where an elitist MOEA evolves on each processor with

its proper starting sub-population. These sub-populations cooperate through a recombination/distribution mechanism. This parallel model divides the global population among the available processors so that each one receives a representative set of solutions from the global population. Each sub-population uses its proper genetic operators (crossover and mutation). In this way, diversification among sub-populations will be maintained.

The main algorithm consists of several elitist MOEAs. One processor, the organiser, has the responsibility of collecting individuals from the other processors and then redistributing them.

Every processor k (k=0, 1,…, p) constitutes its first n Pareto fronts found (n is a parameter) as follows: the non-dominated individuals of the sub-population P_k of the processor k constitute the first front F_{1k}. The set of non-dominated individuals in $P_k \setminus F_{1k}$ makes up the second front F_{2k}. This scheme is repeated until the first n PFs are created. After that the processor sends them to the organiser.

The organiser gathers individuals sent by all processors (the organiser and the others) in order to create the first n global Pareto fronts. The redistribution mechanism is described as follows: each global Pareto front GF_i ($i \in \{1,...,n\}$) is first partitioned into NC clusters (NC is a parameter). After that, every cluster is redistributed with equity between the available processors. Each processor will receive at least one individual from each cluster.

Let's consider a cluster CL of |CL| individuals. If p≤|CL| then the processor k (k=0,1, ..., p-1) receives from CL all individuals j (j=0, ..., |CL|-1) such that j%p = k. Otherwise the processor k will receive the individual j such that k%|CL|=j.

In this way, each processor receives a good approximation of the first n global Pareto fronts which is different from the sets received by the others. We maintain diversity in each sub-population and especially in the Pareto front. After the redistribution process, the size of each sub-population will be increased by crossover so that this sub-population will be equal to its initial size N/p, N is the population size.

We have used an agglomerative clustering algorithm that begins with each individual representing a single cluster. At each step, the distance between each two different clusters is calculated as the maximal Euclidean distance between two individuals from these clusters. Then, the nearest two clusters with respect to the distances calculated are merged. These steps are repeated until the number NC of desired clusters is obtained.

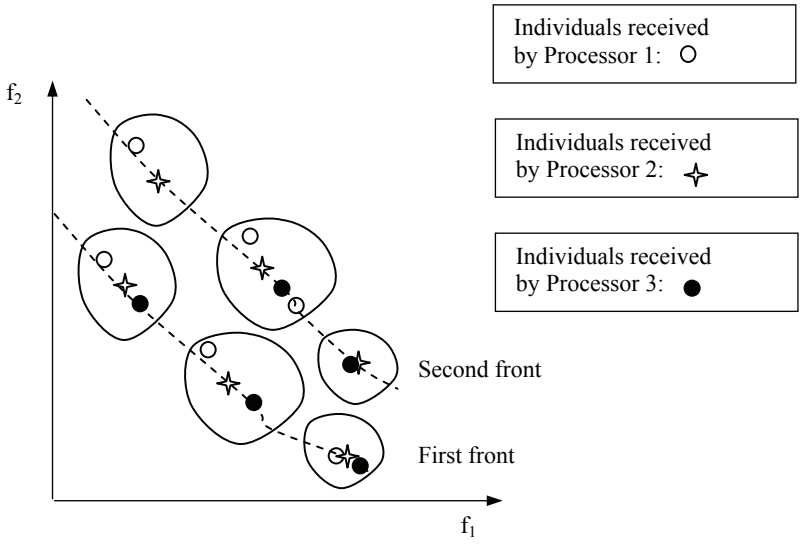

Figure 3. Equitable distribution of the first and the second front

The pseudo-code of the algorithm is the following:

I. **Parameters:**

N: population size,
NC: number of clusters,
T: maximum number of generations,
C: interval of migration,
n: number of fronts to distribute.

II. **Step 1. Initialization:** randomly generate an initial sub-population P_0. The global population size is N. The size of each sub-population is N/P (p is the number of processors)

Set t=0.

III. **Step 2. Evolutionary computation:**

1. If t≠0 then increase the size of the sub-population by crossover.
2. Run a Multi-objective genetic algorithm for C generations.

IV. **Step 3. Recombination/redistribution of the first *n* global fronts:**

1. Each process sends its first *n* Pareto fronts to the organiser.
2. The organiser process:

- Creates the first n global fronts,
- divides the i^{th} global front into NC clusters (i=1, …, n),
- distributes each cluster with equity between all the processors.

V. **Step 8. Termination:** if t=T, stop the algorithm, else go to Step 2.

4. Experimental results

In this section, we study the performance of the MFED and the following PMOEAs: DRMOGA, Cone-Separation and PSPEA which is a standard island (no migration). PSPEA is a parallel version with an independent SPEA running on each processor. The four parallel schemes studied are based on the Strength Pareto Evolutionary Algorithm (SPEA) [14] which is one of the most popular elitist MOEAs. It utilises an external population (the archive) in order to preserve diversity and prunes it when a predetermined size is exceeded. Four test problems [14, 15], which include: convex (ZDT1), non-convex (ZDT2), discontinuous (ZDT3) and non-uniform (ZDT6) Pareto Optimal Front (POF), are chosen for this comparative study (see appendix A). To solve the test functions, we have used bit coding for representing individuals. 20-bit length is used for each design variable of the problems studied. Since each sub-population evolves with its proper genetic operators, we have implemented three crossover operators (one point, two points and uniform crossover) and two mutation operators (one point and mapping mutation).

The parameters of the elitist MOEA are:

- Population size (N): 300 individuals distributed among the available processors.
- Archive size (N): 200 individuals.
- Maximum number of generations (T): 1000.
- Crossover probability (Pc): 0.86.
- Mutation probability (Pm): 0.1
- C: interval of the global clustering: 20 (distribution every 20 generations).
- Number of processors: 4.

All algorithms are implemented using C++ language on a local network with Pentium IV 2.4 GHz 80 Gb computers under Windows XP. The communication between the processors has been supported by the freely available MPICH (Message Passing Interface) parallelisation library.

To compare the results derived from each algorithm, we use the following metrics:

1. The spacing indicator gives a good indication of how evenly the solutions are distributed in the objective space. It is defined as:

$$S = \left[\frac{1}{n} \sum_{i=1}^{n} (d_i - \overline{d})^2 \right]^{1/2}$$

where:

$$d_i = \min_{(k \in PO) \wedge (k \neq i)} \left(\sum_{j=1}^{m} \left| f_j^i - f_j^k \right| \right)$$

$$\overline{d} = \frac{\sum_{i=1}^{n} d_i}{n}$$

where n is the number of solutions in the current front PO, d_i is the Euclidean distance in the objective space between the solution i in PO and its nearest solution in POF.

The smaller the spacing is, the more regularly distributed the solutions in PO are:

2. The S-metric (Hypervolume) [17] gives the size of the objective space enclosed by POF and a reference point Z^{ref}.
3. The generational distance [15] measures how far the elements in the set of non-dominated vectors found (PO) are from those in the Pareto optimal set (POF). For instance, it represents how far the current front PO is from the optimal front POF. Its value is defined as:

$$GD = \sqrt{\frac{\sum_{i=1}^{n} (d_i)^2}{n}}$$

where n is the number of individuals in the current front PO and d_i is the Euclidean distance between any of the individuals and the nearest individual from the Pareto-optimal front.

Ten trials are launched for each configuration. Then, the min, max and average for each performance metric are given.

ZDT1 problem

The results of the four algorithms are similar with respect to the S metric (Table 1). DRMOGA and Cone Separation result in a better distribution of solutions in the PO set found since they produce the smallest values for the spacing (Table 1). Indeed, the reproduction of neighbouring individuals in these two parallel models, which are based on the specialisation concept, generates close individuals in the objective space. The approximation of the true

Pareto front obtained by Cone Separation is the nearest one to POF since it obtains the least value for the generational distance (Table 2). MFED is less effective than the other models concerning the spacing and the generational distance. This stems from the fact that each sub-population in MFED performs computation on all promising regions of the search/objective space discovered and thus converges less rapidly than the other algorithms. Due to the easiness of this problem (the Pareto front is convex and uniform), the three algorithms PSPEA, DRMOGA and Cone Separation have produced results close to each other.

Table 1

Results for ZDT1 problem: S metric and Spacing

Algorithm	Smetric			Spacing		
	Min	Mean	Max	Min	Mean	Max
PSPEA	0,65201	0,654164	0,654164	0,0019373	0,00367523	0,00367523
DRMOGA	0,645368	0,6483987	0,650027	0,00365695	0,00512045	0,00627473
Cone separation	0,649734	0,6512663	0,652182	0,00204366	0,0031909	0,00410994
MFED	0,652839	0,6542778	0,655483	0,0104054	0,01245304	0,0167373

Table 2

Results for ZDT2 problem: Generational distance

Algorithm	Generational distance		
	Min	Mean	Max
PSPEA	5,23E-05	8,62E-05	0,00016884
DRMOGA	3,32E-05	4,63E-05	7,54E-05
Cone separation	1,28E-05	1,44E-05	1,65E-05
MFED	0,00014301	0,000646334	0,00232182

ZDT2 problem

The results for the S metric (Table 3) and the generational distance (Table 4) for the ZDT2 problem show that the MFED outperforms the other parallel algorithms according to these two metrics. MFED achieves a good approximation of the true Pareto optimal front POF. Its results for the spacing metric are the worst when compared to those of DRMOGA and Cone Separation. We can conclude that the non-dominated solutions found by MFED

are not uniformly distributed in the objective space. This may be due to the non--convexity of the POF so that some regions in the objective space are not as well explored as the others. On average, PSPEA, DRMOGA and Cone Separation provide similar results for the S metric (Table 3) but cone separation slightly improves the spacing.

Table 3

Results for ZDT2 problem: S metric and Spacing

Algorithm	Smetric			Spacing		
	Min	Mean	Max	Min	Mean	Max
PSPEA	0,4871	0,4892946	0,4871	0,0051399	0,0047197	0,0058635
DRMOGA	0,474847	0,4832458	0,49057	0,0037625	0,00682405	0,0104535
Cone separation	0,4889	0,4935705	0,496057	0,00232241	0,00360916	0,00574457
MFED	0,318422	0,3216687	0,325326	0,00932995	0,01202236	0,0147275

Table 4

Results for ZDT2 problem: Generational distance

Algorithm	Generational distance		
	Min	Mean	Max
PSPEA	0,00255696	0,003288901	0,00360972
DRMOGA	0,0049013	0,00587651	0,0069419
Cone separation	1,2775E-05	0,004449959	0,00515236
MFED	0,00022007	0,000747621	0,00158096

ZDT3 problem

Table 5

Results for ZDT3 problem: S metric and Spacing

Algorithm	Smetric			Spacing		
	Min	Mean	Max	Min	Mean	Max
PSPEA	0,740543	0,8400583	0,86043	0,00378297	0,0087262	0,0121672
DRMOGA	0,318422	0,4476161	0,863358	0,00232241	0,00679382	0,0209619
Cone separation	0,841538	0,8473562	0,85312	0,00486804	0,00789774	0,0167653
MFED	0,778603	0,7791379	0,779501	0,00366007	0,00673456	0,00869159

Table 6

Results for ZDT3 problem: Generational distance

Algorithm	Generational distance		
	Min	Mean	Max
PSPEA	0,00085425	0,001182975	0,00140321
DRMOGA	0,00221447	0,00309332	0,0044729
Cone separation	0.00153496	0.00179118	0.00201776
MFED	1,6994E-07	2,51092E-07	3,90953E-07

The problem ZDT3 is discontinuous, so the spacing is not significant. As shown in Table 5, the experimental results show that DRMOGA is capable of providing better quality of solutions with respect to the S metric than all the other algorithms. The results of PSPEA and Cone Separation on the three considered metrics are very close (Table 6 and Table 5). From Table 6, we conclude that MFED clearly performs better with respect to the generational distance. Since the Pareto front is discontinuous, the division of the objective space into cones in Cone-Separation may result in empty cones, so that it reduces the performance and the convergence of this algorithm. However, MFED is less affected by the discontinuity of the POF than the other methods.

ZDT6 problem

Table 7

Results for ZDT6 problem: S metric and Spacing

Algorithm	Smetric			Spacing		
	Min	Mean	Max	Min	Mean	Max
PSPEA	0,702722	1,4089039	2,25275	0,0438337	0,19234355	0,548809
DRMOGA	1,00986	1,611229	2,07032	0,0344967	0,22391992	0,834136
Cone separation	1,72002	1,722573	1,72408	0,00286583	0,01388675	0,0669603
MFED	4,48067	4,482684	4,48369	0,00898641	0,11755475	0,608804

Table 8

Results for ZDT6 problem: Generational distance

Algorithm	Generational distance		
	Min	Mean	Max
PSPEA	0,599989	1,2432687	2,09514
DRMOGA	1,04722	1,520106	1,92963
Cone separation	0,0119125	0,01499197	0,0173161
MFED	5,9604E-08	0,012355877	0,0707847

Table 7 shows that cone separation, DRMOGA and PSPEA largely outperform MFED with respect to the S metric. Since ZDT6 has a non-uniform POF, the spacing metric is not very significant. Nevertheless, the spacing value obtained by Cone Separation is remarkably lower than the others (Table 7). Cone separation generates a set of non-dominated solutions which are near to the true Pareto front (Table 8) and fairly dispersed in the objective space. For ZDT6, the only difficulty is that the POF has a non-uniform distribution of its points. The performance of MFED is very much affected by the equitable distribution of the fronts. So, the distribution decreases the convergence of each sub-population.

Conclusions

This paper proposes a new parallel model of genetic algorithms for multi--objective optimisation problems which is based on the clustering technique. In MFED, at regular intervals, each island receives an initial sub-population that is different from those received by the others. Then it performs a search over all the promising regions found in the search space. The idea of the equitable distribution insures the reception of a set of representative individuals of these promising regions, by each processor. At the same time, MFED increases the convergence to the true Pareto front. Thanks to the use of different genetic operators in each island, the diversity all over the population is maintained. Experiments have been carried on four PMOEAs (MFED, DRMOGA, PSPEA and Cone Separation) while using a well known multi-objective benchmark function set that covers a wide range of difficulties (discontinuity, non-convexity, non-uniformity) in finding the Pareto front. Experimental results have shown that the performance of each model depends on the shape

of the search space. The MFED outperforms the other algorithms on the ZDT3 problem which has a discontinuous POF, while DRMOGA and Cone Separation perform better on the ZDT1 problem with convex POF. But, MFED converges less rapidly than the others.

On the one hand, the PMOEAs that are based on the specialisation concept ensure a good performance vis à vis some difficulties or vis à vis some performance criteria. On the other hand, MFED that doesn't limit each island to a specific region of the search space, (each island perform a global exploration), succeeds where the other algorithms fail. Thus, the two division/redistribution mechanisms are complementary and essential for improvement of results in different types of problems.

Appendix

The benchmark functions set used in this work addresses the following minimisation problem:

1. ZDT2.

$$f_1(\vec{x}) = x_1$$
$$f_2(\vec{x}) = g(x)[1-(f_1(x)/g(x))^2]$$
$$g(x) = 1+\frac{9}{n-1}\sum_{i=2}^{n} x_i$$
with
$$x_i \in [0,1],\ n = 30$$

2. ZDT6.

$$f_1(\vec{x}) = 1-\exp(-4x_1)\sin^6(6\Pi x_1)$$
$$f_2(\vec{x}) = g(x)[1-(f_1(x)/g(x))^2]$$
$$g(x) = 1+9((\sum_{i=2}^{n} x_i)/9)^{0.25}$$
with
$$x_i \in [0,1],\ n = 10$$

3. ZDT3.

$$f_1(\overrightarrow{x}) = x_1$$
$$f_2(\vec{x}) = g(x)[1-\sqrt{f_1(x)/g(x)} - ((f_1(x)/g(x))\sin(10\Pi f_1)]$$
$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$

with

$$x_i \in [0,1],\ n = 30$$

4. ZDT1.

$$\begin{aligned} \min \quad & f_1(x) = x_1 \\ \min \quad & f_2(x) = 1 - \sqrt{x_1 / g(x)} \\ & g(x) = 1 + 9\left(\frac{\sum_{i=2}^{n} x_i}{n-1}\right) \\ & x_i \in [0,1] \quad \forall i \in \{1,2,.....,30\} \end{aligned}$$

References

[1] Branke J., Schmeck H., Deb K., Reddy M.: *Parallelizing Multi-Objective Evolutionary Algorithms: Cone Separation. Congress on Evolutionary Computation.* IEEE, Piscataway 2004, pp. 1952-1957.

[2] Cantú-Paz E.: *A Survey of Parallel Genetic Algorithms*. Technical Report. Illlinois Genetic Algorithms Laboratory, 1997.

[3] Coello C.A., van Veldhuizen D.A., Lamont G.B.: *Evolutionary Algorithms for Solving Multi-Objective Problems*. 2nd Edition, Springer Science + Business Media, New York 2007.

[4] Deb K., Pratap A., Agarwal S., Meyarivan T.: *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II*. "IEEE Transactions on Evolutionary Computation" 2002, Vol. 6, pp. 182-196.

[5] de Toro F., Ortega J., Fernandez J., Diaz A.: *PSFGA: A Parallel Genetic Algorithm for Multiobjective Optimization*. Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-Based Processing, IEEE, Gran--Canaria 2002, pp. 384-391.

[6] Erickson M., Mayer A., Horn J.: *The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems*. In: *First International Conference on Evolutionary Multi-Criterion Optimization*. E. Zitzler, K. Deb, L. Thiele, C.A. Coello Coello, D. Corne (eds). Lecture Notes in Computer Science, No. 1993, Springer-Verlag, Berlin-Heidelberg 2001, pp. 681-695.

[7] Essabri A., Gzara M., Loukil T.: *Parallel Multi-Objective Evolutionary Algorithm with Multi-Front Equitable Distribution*. The Fifth International Conference on Grid and Cooperative Computing (GCC 2006), Hunan 2006, pp. 241-244.

[8] Hiroyasu T., Kaneko M., Miki M.: *A Parallel Genetic Algorithm with Distributed Environment Scheme.* Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, No. 2, 2000, pp. 619-625.

[9] Hiroyasu T., Miki M., Watanabe S.: *Distributed Genetic Algorithms with a New Sharing Approach in Multiobjective Optimization Problems*. Congress on Evolutionary Computation, Washington, D.C. July 1999, pp. 69-76.

[10] Horn J., Nafpliotis N., Goldberg D.E.: *A Niched Pareto Genetic Algorithm for Multiobjective Optimization*. Procedings 1st IEEE Conference Evolutionary Computation. IEEE World Congress Computational Computation, 1, Orlando, Florida 1994, pp. 82-87.

[11] Kamiura J., Hiroyasu T., Miki M.: *MOGADES: Multi-Objective Genetic Algorithm with Distributed Environment Scheme*. Computational Intelligence and Applications (Proceedings of the 2nd International Workshop on Intelligent Systems Design and Applications: ISDA-02), Atlanta 2002, pp. 143-148.

[12] Van Veldhuizen D.A., Lamont G.B.: *Multiobjective Evolutionary Algorithm Test Suites*. In: *Proceedings of the 1999 ACM Symposium on Applied Computing*. J. Carroll et al. (eds). ACM, San Antonio, Texas 1999, pp. 351-357.

[13] Xu K., Louis S.J., Mancini R.C.: *A Scalable Parallel Genetic Algorithm for X-ray Spectroscopic Analy*sis. Genetic and Evolutionary Computation Conference (GECCO-2005)-Proceedings-ToC, Washington D.C. 2005, pp. 811-816.

[14] Zitzler E., Thiele L.: *Multi-Objective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach*. "IEEE Transactions on Evolutionary Computation" 1999, 3, pp. 257-271.

[15] Zitzler E., Laumanns M., Thiele L.: SPEA2: *Improving the Strength Pareto Evolutionary Algorithm for Multi-Objective Optimization*. "Evolutionary Methods for Design, Optimisation, and Control" 2002, pp. 19-26.

[16] Srinivas N., Deb K.: *Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms.* "Evolutionary Computation" 1995, 2, pp. 221-248.