J. Comp. Int. Sci. (2016) 7(1):13-20
http://epacis.net/jcis/PDF_JCIS/JCIS-0103.pdf
jcis@epacis.net

Pan-american Association of Computational Interdisciplinary Sciences

Experimental comparison of many-objective evolutionary preference-based methods in a parallel framework

Christian von Lücken[a1], Carlos Brizuela[b] and Benjamin Barán [a]
[a] Universidad Nacional de Asunción, Facultad Politécnica, Paraguay
[b] CICESE Research Center, Ensenada, México

Received on September 20, 2015 / accepted on January 10, 2016.

Abstract

Multi-objective Evolutionary Algorithms (MOEA) are used to solve complex multi-objective problems. As the number of objectives increases, Pareto-based MOEAs are unable to reproduce the same effectiveness showed for two or three objectives. Thus, several authors proposed preference-based methods as an alternative. On the other hand, parallelization has shown to be useful in evolutionary optimizations. This paper combines for the first time seven preference-based methods for many objective optimization in a multi-threading parallelization framework. Preference-based methods were used to replace the elitism procedure of the Non-dominated Sorting Genetic Algorithm II. Executions of each alternative were carried-out for the DTLZ-2 problem in a commodity multi-core platform. Obtained solutions were compared by different criteria, providing some insights into the improvements that the proposed combination may offer in many-objective optimization.

Keywords: *Multi-objective evolutionary algorithms, many-objective optimization, parallel evolutionary qlgorithms, computational mathematics.*

1. Introduction

Multi-objective optimization problems (MOPs) have several, possibly conflicting, objectives to optimize. Conflicts between objectives lead to a set of solutions called the Pareto set, denoted as $\mathcal{P}^\star$, while these solutions mapped to the objective space form the Pareto Front ($\mathcal{PF}^\star$). In most practical cases, it is difficult to obtain a complete and exact Pareto set of a MOP. Multi-objective Evolutionary Algorithms (MOEAs) have proven to be very effective to provide near optimal approximations in complex multi-objective problems having two or three optimization objectives [2].

A solution is said to dominate another one if it is not worse in any objective and it is strictly better in at least one objective, thus, Pareto set is formed with solutions that are non-dominated with respect to the whole solution space [4, 2]. As the number of objectives increases, so does the number of solutions that dominance operators classify as equivalent (or non-comparable) leading to a reduction of the search ability of Pareto-based MOEAs [9].

Improving MOEAs' performance to solve the so-called many-objective problems, i.e. problems having more than 4 objectives, is an important research area and different alternatives were proposed to achieve this improvement [18]. Among these alternatives are comparison relations that use preference information for solutions [9, 8, 9, 14, 7, 11].

[1]E-mail Corresponding Author: clucken@pol.una.py

On the other hand, parallel MOEAs (pMOEAs) are able to use larger populations and to reduce the execution time [2]. The literature reports various parallel MOEAs (pMOEAs) that have been used to solve diverse problems [15]. These pMOEAs were mainly developed and tested in distributed memory parallel systems; however, nowadays availability of lower cost shared memory multi-core platform requires a review of the parallelization methods in the new existing environment in order to leverage the computational power that these platforms may offer.

Parallel Pareto-based MOEAs has been extensively studied for a small number of objectives [2], and some works compare preference relations for many-objective problems [3, 11, 10]; however, to the best of our knowledge they were not tried in a parallel framework. Therefore, this paper presents for the first time a comparison of two parallelization approaches for the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [5], using seven ranking methods proposed for many-objective optimization problems [9, 8, 7, 14, 11].

This paper is organized as follows: Section 2 presents the methods and the parallel MOEA framework considered in this work. Section 3 presents the experimental setup and obtained results. Finally, Section 4 presents conclusions and future works.

2. Methods considered in this work

2.1. MOEAs considered for parallelization

Several modifications of the NSGA-II [5] ranking method were proposed to be used in many-objective optimization [18]. Also, in this paper the NSGA-II algorithm is used as test bed. With NSGA-II, the fitness of a solution is based on two values: its non-dominance ranking and its crowding distance. At each generation, an elitist procedure based on these values is used to ensure that the best elements survive from one generation to the next. Details of the NSGA-II can be found in [5].

In this paper, the following operators and ranking methods are used to modify the fitness assignment procedure of the NSGA-II, for the sake of brevity the details are not explained here but we refer the interested reader to the corresponding references:

1. **Favour relation [8]:** this relation counts the the number of objectives in which a given solution outperforms another. Given $\mathbf{x}$ and $\mathbf{x}' \in \mathcal{X}_f$, where $\mathcal{X}_f$ represents the feasible solution space, it is said that $\mathbf{x}$ is favoured than $\mathbf{x}'$, denoted as $\mathbf{x} \prec_{favour} \mathbf{x}'$, if and only if

$$n_b(\mathbf{F}(\mathbf{x}), \mathbf{F}(\mathbf{x}')) > n_b(\mathbf{F}(\mathbf{x}'), \mathbf{F}(\mathbf{x})) \tag{1}$$

$$n_b(\mathbf{F}(\mathbf{x}), \mathbf{F}(\mathbf{x}')) = |\{f_i(\mathbf{x}) \text{ s.t. } f_i(\mathbf{x}) < f_i(\mathbf{x}')\}| \tag{2}$$

   In [8], the favour relation is proposed to be used with the Satisfiability Class Ordering classification (SCO) procedure [8] to sort solutions.

2. $\epsilon-$**Preferred Relation [14]:** the $\epsilon-$Preferred relation compares solutions by counting the number of times a solution exceeds user defined ϵ_i limits for each dimension and, in case of a tie, it uses the favour relation. Given two solutions $\mathbf{x}$ and $\mathbf{x}' \in \mathcal{X}_f$, it is said that $\mathbf{x}$ is $\epsilon-$preferred than $\mathbf{x}'$, denoted as $\mathbf{x} \prec_{\epsilon-preferred} \mathbf{x}'$, iff:

$$\mathbf{x} \prec_{\epsilon-exceed} \mathbf{x}' \vee (\mathbf{x}' \not\prec_{\epsilon-exceed} \mathbf{x} \wedge \mathbf{x} \prec_{favour} \mathbf{x}')$$

   where $\mathbf{x} \prec_{\epsilon-exceed} \mathbf{x}'$implies that:

$$|\{i : y_i < y'_i \wedge |y_i - y'_i| > \epsilon_i\}| > |\{i : y'_i < y_i \wedge |y'_i - y_i| > \epsilon_i\}|$$

   As in [8], in [14] the SCO algorithm is used to rank solutions.

3. **Preference Ordering based on order of efficiency (PO_k) [7]:** a solution $\mathbf{x}$ is considered to be efficient of order k if it is Pareto optimal in the $\binom{m}{k}$ subspaces of the objective space taking into account only k objectives at a time. The order of efficiency of a solution $\mathbf{x}$, denoted by $K(\mathbf{x})$ is the minimum k value for which $\mathbf{x}$ is efficient. Based on the order of efficiency a ranking procedure is proposed in [7].

4. **Preference Ordering based on efficiency degree ($PO_{k,z}$) [7]:** this method refines the ranking based on PO_k when several solutions share the same best order of efficiency. A solution $\mathbf{x}$ is efficient of order k with degree z if $\mathbf{F}(\mathbf{x})$ is not dominated by any member of the Pareto Front for exactly z out of the possible $\binom{m}{k}$ k-element subsets of the objectives. Details of the ranking method based on efficiency degree are in [7].

5. **$-\epsilon$-DOM ranking [11]:** the $-\epsilon$-DOM distance replaces the NSGA-II crowding distance. The $-\epsilon$-DOM distance of a solution $\mathbf{x}$ is:

$$max\{f_i(\mathbf{x}') - f_i(\mathbf{x}) : f_i(\mathbf{x}) < f_i(\mathbf{x}') \ , i \in \{1, \dots m\}\} \ \text{, or}$$
$$0 \ if \ \nexists i \ \text{ such that } f_i(\mathbf{x}) < f_i(\mathbf{x}'), i \in \{1, \dots m\}$$

   Therefore, given solutions $\mathbf{x}$ and $\mathbf{x}'$, *mepsd* reflects the smallest ϵ value such that if subtracted from all objectives of $\mathbf{F}(\mathbf{x}')$, $\mathbf{x}'$ dominates $\mathbf{x}$, while the $-\epsilon$-DOM rank value is the minimum of such values.

6. **Crisp $(1-k)-$dominance relation [9]:** this relation counts the number of objectives in which a solution is better or equal than another. Let $\mathbf{x}$ and $\mathbf{x}' \in \mathcal{X}_f$, $\mathbf{F}(\mathbf{x}) = \mathbf{y}$, $\mathbf{F}(\mathbf{x}') = \mathbf{y}'$, it is said that $\mathbf{x}$ $(1-k)-$dominates $\mathbf{x}'$ iff :

$$n_e(\mathbf{y}, \mathbf{y}') < m \text{ and } n_b(\mathbf{y}, \mathbf{y}') \geq \frac{m - n_e}{k+1}$$

   where m is the number of objectives, $0 \leq k \leq 1$, n_b is as in Eq. (2), and $n_e(\mathbf{F}(\mathbf{x}), \mathbf{F}(\mathbf{x}'))$ is $|\{f_i(\mathbf{x}) \ \text{ s.t. } \ f_i(\mathbf{x}) = f_i(\mathbf{x}')\}|$. The $(1-k)-$dominance relation serves as a new definition of optima, called k-optimality, as well as the corresponding optimal set of solutions [9].

7. **Fuzzy $(1-k_F)-$dominance relation [9]:** the fuzzy extension of $(1-k)-$dominance is defined by determining membership functions μ_b^i , μ_e^i and μ_w^i for each objective function i. There are several membership functions that can be used. In case of a trapezoidal function four parameters (a, b, c, d) are required to represent a meaning of equality and difference.

2.2. Parallel Multi-objective Evolutionary Algorithms

This section introduces the island-based parallel framework considered in this paper. The island model is the most popular parallelization paradigm for MOEAs [2], it consists of a number of subpopulations or islands evolving independently which are provided with a mechanism to interchange individuals exploring for global optima. As the goal here is not to provide a full pMOEAs' survey, readers are kindly referred to [15, 16, 2] for further reading.

There are several alternatives to develop a pMOEA [16]. However, most pMOEAs are based on the parallelization of a current MOEA; particularly, the NSGA-II [5] is a good option for parallelization [17]. Therefore, this work considers NSGA-II as foundation for parallel implementations that differ in at least one of the following features: the way in which the ranking of individuals is produced, i.e. how fitness is assigned; the number of subpopulations and the process used to form subpopulations. Regarding the ranking of solutions, this work uses the original NSGA-II method and the seven alternatives presented in Subsection 2.1. While, considering how pMOEAs divide the main population, this work considers a random distributed population pMOEA (RND) and a K-means based distributed population pMOEA (KM) [13].

In the RND distribution, the population is randomly divided into a number of equal size partitions. In case of the KM distribution, it is based on the original proposal of Streichert et al. [13] which uses K-means to divide the search space of a given optimization problem in suitable partitions without a priori knowledge of its search space. In [13], zone constraints are implemented to limit subpopulations to their specific region. In this paper, K-means is used to divide the population; however, zone constraints are not considered.

An evolutionary iteration comprises the procedures of fitness assignment, selection, crossover, mutation and elitism that are carried out by a MOEA to produce a new offspring from the current population. As a mechanism to search for global optima, some methods, as in [12], combine subpopulation evolutions with iterations where the whole set of solutions is considered to be subject of the evolutionary process. In a parallel

Algorithm 1 Basic framework for parallel implementations used in this work

Read parameters: population size (N), selection, crossover and mutation probability. Set the ranking method RM and its parameters. Set the population partition method PM, the number of islands to be used τ. Read it_g, it_s, it_p: the maximum number of iterations, the number of single thread iterations and the number of parallel thread iterations.
Set $t = 0$
Create an initial random global evolutionary population P_t
while $t < it_g$ **do** ▷ it_g is the total number of evolutionary steps performed
  while $(t+1)$ *mod* $it_s \neq 0$ **do** ▷ it_s: iterations considering the evolutionary population as a whole
    Evolve P_t in P_{t+1} using the NSGA-II with a given ranking method
    $t = t + 1$
  end while
  Split P_t in $P_t^1, \ldots, P_t^\tau$ using a given population partition method PM
  In τ parallel threads
  for $t' = 0$ to it_p **do** ▷ it_p: iterations considering the population as subpopulations in parallel
    Evolve $P_{t+t'}^{Id}$ in $P_{t+t'+1}^{Id}$ using the NSGA-II with a given ranking method
    $t' = t' + 1$
  end for
  End parallel
  $t = t + it_p$
end while
Save non-dominated solutions from P_t

multicore platform, the aforementioned approach can be efficiently applied to interchange information among subpopulations; therefore, it is considered in this work.

Algorithm 1 presents the basic framework for the parallel implementation of the NSGA-II and its variants used in this work. First, the algorithm reads and set its parameters; besides the ussual MOEA parameters, also, the total number of evolutionary steps (it_g), the number of single thread iterations (it_s), and the number of parallel iterations (it_p) must be given. Then, the global number of iterations t is set to 0, and the global population P_t is created at random. Thereafter, it_s evolutionary iterations are executed in a single thread considering the evolutionary population as a whole. The next step is to split $P(t)$ into τ subpopulations $(P_t^1 \ldots, P_t^\tau)$ using the procedure PM. Once the global population was partitioned, τ threads are created, one for each island. Each thread has an identifier Id, thus at each thread Id, evolution of P_t^{Id} occurs during it_p iterations. When the execution of the parallel iterations in all the threads ends, the global count of iterations t is updated to $t + it_p$. The procedure continues until t reaches the maximum number of iterations. Finally, the final set of solutions is saved.

3. Experimental comparison

3.1. Experimental setup and metrics

In order to determine if, in a many-objective optimization problem, the original NSGA-II and its seven variants (presented in Subsection 2.1) can improve their results by parallelization, this paper compares the results obtained by their parallel and sequential implementations. The parallel versions are based on the framework presented in Subsection 2.2 using the RND and the KM distributions. The test problem used in this work is the DTLZ-2 with 8 objectives [6].

Parallel methods were executed using 2 and 4 threads. Thus, a total of 40 different combinations of number of threads, ranking and distribution methods were executed. For each combination, 10 runs were performed using the same set of 10 initial populations. Executions were performed using the following common parameters: population size: 400, stopping criterion 400 iteration, binary coding of 32 bits per variable, one point crossover probability: 0.8, mutation probability: 0.002. For the ϵ-Preferred relation, the ϵ value is 0.0001; for the $(1-k)-$dominance relation, k is 0.5; and, for the $(1-k_F)-$dominance relation, k is also 0.5 and a fuzzy trapezoidal rule is used ($a = -0.001$, $b = 0$, $c = 0$, $d = 0.001$). No fine tuning of the above parameters was considered. For parallel methods, subpopulation iteration is set to 5, followed by another 5 iterations of the whole population until the stop criterion is reached.

Sequential and parallel programs were implemented in the C language. Parallel methods were implemented for execution in Sharing Memory Platforms using the OpenMP library [1]. The experimental com-

putational platform was a machine provided with two Intel Xeon quad-core Processors E5640 (12M Cache, 2.66 GHz, 5.86 GT/s) and 16 GB of main memory running the GNU/Linux operating system.

To evaluate a given set of non-dominated solutions $\mathcal{S}_t$ in a population P_t, different aspects of it may be measured as the proximity to the true Pareto Front and the distribution or the extend of solutions. Therefore, the following metrics are considered in this work [4]:

1. Number of solutions in $\mathcal{PF}^\star$ (N): this metric counts the obtained solutions that are in the true Pareto set. In DTLZ-2 the Pareto Front is composed by those solutions having $\sum_{i=1}^{m} f_i^2(\mathbf{x}) = 1$ [6]. The N metric is expected to be maximized.
2. Generational Distance (GD): this metric measures the average distance between obtained solutions in objective space and the true Pareto Front of the problem. Thus, this metric should be minimized. Since GD requires a reference $\mathcal{PF}^\star$ to be computed, and equations to produce $\mathcal{PF}^\star$ are known, a set of a set of 2000 optimal solutions was determined analytically.
3. Spread (Δ): this metric measures the extent of the Pareto Front that the obtained set of solutions covers. The spread metric is expected to be minimized.
4. Spacing (S): this metric measures if the obtained non-dominated solutions are uniformly distributed. The metric is the average distance of each point from its nearest neighbour, thus, if solutions are uniformly distributed the value of this metric approaches zero. Then, the value of this metric is expected to be minimized.

3.2. Experimental results

Table 1 shows average values and standard deviation of the considered metrics evaluated over the final result of the 10 runs performed for each implemented combination. In this table, each row corresponds to a given ranking method, while columns are for the execution type. Besides the metrics values of the obtained solutions, Table 1 also shows the average and standard deviation of the execution time in seconds of each implementation as an additional metric. The column label Sequential shows the values for sequential executions; the columns 2-RND and 2-KM are for the values for parallel executions using the random and K-means population distribution methods with two threads, respectively. Similarly, 4-RND and 4-KM are for parallel executions using using the random and K-means population distribution methods with threads.

Number of solutions in the Pareto Set (N)

Part	Sequential	2-RND	2-KM	4-RND	4-KM
NSGAII	0.0(0.0)	0.0(0.0)	**0.1(0.3)**	**0.1(0.3)**	0.0(0.0)
ϵ-DOM	342.5(6.576)	337.9(5.262)	346.4(11.706)	340.8(8.328)	**358.0(9.529)**
ϵ-Preferred	72.6(83.493)	120.7(57.331)	67.5(38.792)	**151.0(77.062)**	100.9(44.7)
$(1-k)$-dom.	264.2(104.945)	**349.4(86.165)**	233.2(88.933)	339.7(60.402)	235.3(76.013)
$(1-k_F)$-dom.	388.3(35.1)	**399.5(0.671)**	399.3(0.9)	398.8(0.980)	399.2(0.980)
Favour	143.0(101.444)	155.2(59.499)	167.1(52.922)	**229.4(105.428)**	132.4(60.622)
PO_k	385.8(5.192)	384.2(3.572)	390.7(6.357)	381.3(4.124)	**393.4(2.107)**
$PO_{k,z}$	390.3(3.226)	383.7(2.648)	391.0(1.897)	385.9(3.961)	**393.0(2.608)**

Generational Distance (GD)

Part	Sequential	2-RND	2-KM	4-RND	4-KM
NSGAII	2.69E-2(1.40E-3)	2.74E-2(6.50E-4)	2.59E-2(8.64E-4)	2.69E-2(1.11E-3)	**2.50E-2(9.60E-4)**
ϵ-DOM	7.06E-3(5.26E-4)	7.02E-3(7.44E-4)	6.70E-3(9.12E-4)	6.37E-3(6.31E-4)	**5.20E-3(5.76E-4)**
ϵ-Preferred	2.04E-5(9.98E-6)	1.11E-5(4.14E-6)	1.51E-5(5.24E-6)	**9.46E-6(2.35E-6)**	1.15E-5(3.11E-6)
$(1-k)$-dom.	7.07E-6(1.90E-6)	5.96E-6(1.55E-6)	7.47E-6(1.99E-6)	**5.81E-6(5.65E-7)**	7.18E-6(1.21E-6)
$(1-k_F)$-dom.	1.39E-3(8.97E-4)	1.10E-3(6.46E-5)	**1.09E-3(6.74E-5)**	1.12E-3(9.24E-5)	1.12E-3(8.13E-5)
Favour	1.14E-5(5.98E-6)	9.49E-6(3.57E-6)	8.52E-6(1.45E-6)	**7.58E-6(1.79E-6)**	1.15E-5(6.60E-6)
PO_k	6.00E-3(1.36E-3)	5.02E-3(5.19E-4)	4.64E-3(1.70E-3)	4.17E-3(9.55E-4)	**1.90E-3(8.24E-4)**
$PO_{k,z}$	5.39E-3(1.11E-3)	5.17E-3(8.94E-4)	5.49E-3(1.81E-3)	4.29E-3(8.19E-4)	**2.22E-3(7.52E-4)**

Spread (Δ)

Part	Sequential	2-RND	2-KM	4-RND	4-KM
NSGAII	3.19E-1(1.77E-2)	3.09E-1(1.55E-2)	2.98E-1(1.25E-2)	2.96E-1(1.73E-2)	**2.73E-1(1.05E-2)**
ϵ-DOM	5.21E-1(4.00E-2)	5.45E-1(3.10E-2)	**5.03E-1(3.15E-2)**	5.84E-1(3.69E-2)	5.15E-1(4.51E-2)
ϵ-Preferred	1.00E+00(1.15E-6)	1.00E+00(9.41E-9)	1.00E+00(2.21E-7)	**1.00E+00(4.90E-12)**	1.00E+00(1.80E-8)
$(1-k)$-dom.	1.00E+00(2.84E-11)	**1.00E+00(0.00E+00)**	1.00E+00(1.61E-13)	**1.00E+00(0.00E+00)**	1.00E+00(6.49E-13)
$(1-k_F)$-dom.	8.03E-1(1.23E-1)	7.43E-1(3.78E-2)	7.72E-1(4.02E-2)	**7.42E-1(3.00E-2)**	7.83E-1(6.25E-2)
Favour	1.00E+00(5.52E-11)	1.00E+00(1.62E-13)	1.00E+00(5.16E-11)	**1.00E+00(7.28E-14)**	1.00E+00(2.12E-9)
PO_k	**7.10E-1(3.24E-2)**	7.36E-1(5.28E-2)	8.45E-1(7.34E-2)	9.32E-1(1.18E-1)	9.66E-1(1.25E-1)
$PO_{k,z}$	**7.14E-1(3.95E-2)**	8.10E-1(1.60E-1)	8.28E-1(4.37E-2)	8.68E-1(7.02E-2)	9.43E-1(6.97E-2)

Spacing (S)

Part	Sequential	2-RND	2-KM	4-RND	4-KM
NSGAII	3.41E-1(1.45E-2)	3.24E-1(1.20E-2)	3.16E-1(1.04E-2)	3.05E-1(1.36E-2)	**2.97E-1(1.66E-2)**
ϵ-DOM	3.29E-1(2.80E-2)	3.36E-1(2.58E-2)	3.09E-1(3.25E-2)	3.27E-1(2.55E-2)	**2.48E-1(1.72E-2)**
ϵ-Preferred	2.04E-7(5.17E-7)	1.45E-9(3.02E-9)	2.56E-8(7.58E-8)	**4.57E-13(1.11E-12)**	4.92E-10(1.35E-9)
$(1-k)$-dom.	1.83E-14(5.46E-14)	**4.30E-18(9.51E-18)**	1.40E-14(3.70E-14)	5.55E-18(1.31E-17)	3.28E-14(9.83E-14)
$(1-k_F)$-dom.	8.32E-3(1.64E-3)	8.14E-3(1.83E-3)	**7.67E-3(1.57E-3)**	8.17E-3(2.09E-3)	8.04E-3(1.40E-3)
Favour	2.03E-12(5.46E-12)	6.64E-15(1.23E-14)	2.69E-13(7.10E-13)	**1.51E-15(4.24E-15)**	7.73E-11(2.31E-10)
PO_k	1.39E-1(2.24E-2)	1.24E-1(2.01E-2)	9.75E-2(2.71E-2)	1.00E-1(2.70E-2)	**4.14E-2(1.34E-2)**
$PO_{k,z}$	1.33E-1(3.53E-2)	1.29E-1(2.74E-2)	9.88E-2(2.69E-2)	1.05E-1(2.69E-2)	**5.16E-2(2.72E-2)**

Execution Time (seconds)

NSGAII	76.1(0.294)	62.6(1.42)	67.5(0.866)	**50.4(0.685)**	56.0(0.588)
ϵ-DOM	108.8(0.902)	89.7(0.389)	90.5(1.62)	**73.6(0.568)**	78.6(1.14)
ϵ-Preferred	134.4(2.13)	106.0(0.624)	118.4(2.37)	**87.4(0.676)**	102.5(2.69)
$(1-k)$-dom.	92.3(0.451)	78.2(0.676)	86.3(2.85)	**62.8(0.782)**	75.7(2.18)
FAFuzzy	232.0(8.18)	165.6(3.19)	182.8(3.59)	**142.7(1.06)**	156.1(6.11)
Favour	116.5(2.71)	92.8(1.23)	104.4(2.27)	**78.2(1.88)**	91.1(1.86)
PO_k	81.0(0.696)	70.9(0.536)	74.2(1.52)	**58.0(0.709)**	67.0(1.60)
$PO_{k,z}$	81.3(0.788)	71.0(0.738)	74.1(0.99)	**57.7(0.898)**	66.3(2.30)

Table 1: Average values of metrics N, GD, Spread, Spacing, and execution time for the methods considered in this work solving the DTLZ2 problem with 8 objectives

To analyse the results in Table 1, for each metric, the best metric value for each row (ranking method) is boldfaced, while, the best value for each column (execution method) is in gray. Thus, a cell that is in boldface and gray is for the solution set with best metric value. In this way, it is easier to note that for most of the considered metrics, at least one parallel implementation obtains a solution set that is better than its corresponding sequential counterparts in convergence or diversity. The result is remarkable since parallel and sequential implementations were executed using the same number of iterations, and, therefore, the same number of objective function evaluations. Moreover, parallel implementations obtain better results with improved execution times.

As it is expected, regarding the metrics related to convergence (GD and N) executions of the original NSGA-II with the considered parameters obtain worse values than the alternatives that were proposed for many-objective problems. However, considering the Spread of solutions NSGA-II obtains the best values.

The implementations of $(1-k_F)$-dominance relation obtain the best convergence values, but they cover a small portion of the Pareto Front as their Spread metric indicate. In fact, with the exception of ϵ-DOM, the alternatives to the original NSGA-II ranking procedures obtain solutions concentrated into a region with Spread values greater than 0.7. Spacing values shows that in spite that solutions provided by $(1-k)$-dominance and Favour concentrates in a small area their solutions are better distributed than in the case of the other alternatives.

Regarding the number of solutions in the Pareto Front, results of the ϵ-DOM sequential and parallel implementations ranks 4 or 5 to 8, with an average value of more than 340; while, regarding the Spread metric these implementations receive the second position. Thus, ϵ-DOM implementations obtain many solutions covering a larger portion of the Pareto Front than the other alternatives. In spite of an analysis of the ϵ-DOM spacing metric results shows that solutions in the covered region are not well distributed, we consider that this method offers the best trade-off between convergence and diversity among the considered

ranking methods.

Between the population distribution methods considered in this work, the results show that the best parallelization choice (and the number of subpopulations considered) depends on the ranking method to be used. As an example, for the N metric and the ϵ-DOM ranking, the KM with 4 subpopulations obtains the best value; for the ϵ-Preferred the best value is for RND using 4 partitions, while, in case of the $(1-k)$-dominance the best result also is for the Random partitioning, but with two subpopulations. Other experiments are needed to determine the relation that exist between the metric results and the ranking and partition methods with varying population size and number of subpopulations.

4. Conclusions and future work

The increasing availability of multi-core platforms made necessary to adapt existing serial and parallel algorithms, as well as to assess them in this new computing platform. Also, development of newer algorithms for many-objective problems is needed. A first step for this development is to analyse the current existing alternatives.

In this work, sequential and parallel versions of the NSGA-II using different ranking methods that have been developed for many objective problems were evaluated over a set of performance metrics. Comparison between the various considered methods was executed using a single parallel framework in order to combine the different methods that have been considered.

The obtained results have shown that in most of the studied cases parallel MOEAs outperform their sequential counterparts. Comparison results have also shown that, for the considered experimental setting and metrics, the best serial ranking method is, in general, the best in parallel executions. Among the considered ranking methods, the ϵ-DOM ranking appears to provide the best trade-off between convergence and diversity. The obtained results, also indicates that different methods may be useful at different moments of the search of solutions. Thus, the algorithm choice must consider if the decision maker is interested in obtaining a solution set with a large diversity or if it is interested in emphasizing convergence to a given region.

Future works include, among others: to extend the comparison to other problems; to test the proposed parallel approach with different thread numbers and parameters; to analyse the relations among population sizes, number of subpopulations, and partition and ranking methods; to improve the partition techniques; and, to analyse optimized parallelization schemes. Additionally, the development of new metrics for many-objective problems is needed to evaluate the convergence and diversity properties of the obtained solution sets.

References

[1] Chapman, B., Jost G., Van Der Pas R., Kuck, D.J. Using OpenMP: portable shared memory parallel programming. The MIT Press, 2007.

[2] Coello Coello, C., Lamont, G., Van Veldhuizen, D. Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, New York, second edition, 2007. ISBN 978-0-387-33254-3.

[3] Corne, D. and Knowles, J. Techniques for highly multiobjective optimisation: some nondominated points are better than others. In Proc. of the 9th Ann. Conf. on Genetic and Evol. Comput., GECCO '07, pages 773–780. ACM Press, 2007.

[4] Deb, K. Multi-objective optimization using evolutionary algorithms. Wiley, 2001.

[5] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evol. Comput., 2002.
doi:10.1109/4235.996017

[6] Deb, K., Thiele, L., Laumanns, M., Zitzler, E. Scalable multi-objective optimization test problems. In Proc. of the 2002 Congr. on Evol. Comput., 2002.
doi:10.1109/CEC.2002.1007032

[7] di Pierro, F., Khu, S., Savić, D. An investigation on Preference Order ranking scheme for multiobjective evolutionary optimization. IEEE Trans. on Evol. Comput., 2007.
doi:10.1109/TEVC.2006.876362

[8] Drechsler, N. , Drechsler, R., Becker. B. Multi-objective optimisation based on relation *favour*. In First Int. Conf. on Evol. Multi-Criterion Optim. 2001.
doi:10.1007/3-540-44719-9_11

[9] Farina, M., Amato, P. On the Optimal Solution Definition for Many-criteria Optimization Problems. In Proceedings of the NAFIPS-FLINT International Conference'2002, pages 233–238, Piscataway, New Jersey, June 2002. IEEE Service Center.

[10] Ishibuchi, H., N. Tsukamoto, N., Nojima, Y. Evolutionary many-objective optimization: A short review. In 2008 IEEE Congr. on Evol. Comput., 2008.
doi:10.1109/CEC.2008.4631121

[11] Köppen, M., Yoshida, K. Substitute distance assignments in NSGA-II for handling many-objective optimization problems. In Proc. of the 4th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2007, 2007.
doi:10.1007/978-3-540-70928-2_55

[12] de Toro Negro, F., Ortega, J., Ros, E., Mota, S., Paechter, B., Martín, J.M. PSFGA: parallel processing and evolutionary comput. for multiobjective optimization Parallel Comput., 2004.
doi:10.1016/j.parco.2003.12.012

[13] Streichert, F., Ulmer, H., Zell, A. Parallelization of Multi-objective Evolutionary Algorithms Using Clustering Algorithms. In Proc. of the 3th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2005, 2005.
doi:10.1007/978-3-540-31880-4_7

[14] Sülflow, A., Drechsler, N., Drechsler, R. Robust Multi-objective Optimization in High Dimensional Spaces. In Proc. of the 4th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2007, 2007.
doi:10.1007/978-3-540-70928-2_54

[15] Talbi, E., Mostaghim, S., Okabe, T., Ishibuchi, H., Rudolph, G., Coello-Coello, C. Parallel approaches for multiobjective optimization. In *Multiobjective Optimization*, 2008.
doi:10.1007/978-3-540-88908-3_13

[16] van Veldhuizen, D., Zydallis, J., and Lamont, G. Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms. IEEE Trans. on Evol. Comput., 7(2):144–173, April 2003.

[17] von Lücken, C. Algoritmos evolutivos para optimización multiobjetivo: Un estudio comparativo en un ambiente paralelo asíncrono. Master's thesis, Universidad Nacional de Asunción, Paraguay, 2003. (In Spanish).

[18] von Lücken, C., Barán, B., Brizuela, C. A survey on multi-objective evolutionary algorithms for many-objective problems. Comput. Optimization and Applications, 1(1):1–50, 2014.
doi:10.1007/s10589-014-9644-1