Research Article / Systems-Computer Engineering http://dx.doi.org/10.15446/ing.investig.v40n3.82906

EF1-NSGA-III: An evolutionary algorithm based on the first front to obtain non-negative and non-repeated extreme points

EF1-NSGA-III: Un algoritmo evolutivo basado en el primer frente para obtener puntos extremos no negativos y no repetidos

Luis Felipe Ariza Vesga[1], Johan Sebastián Eslava Garzón[2], and Rafael Puerta[3]

ABSTRACT

Multi-Objective and Many-objective Optimization problems have been extensively solved through evolutionary algorithms over a few decades. Despite the fact that NSGA-II and NSGA-III are frequently employed as a reference for a comparative evaluation of new evolutionary algorithms, the latter is proprietary. In this paper, we used the basic framework of the NSGA-II, which is very similar to the NSGA-III, with significant changes in its selection operator. We took the first front generated at the non-dominating sort procedure to obtain non-negative and non-repeated extreme points. This open-source version of the NSGA-III is called EF1-NSGA-III, and its implementation does not start from scratch; that would be reinventing the wheel. Instead, we took the NSGA-II code from the authors in the repository of the Kanpur Genetic Algorithms Laboratory to extend the EF1-NSGA-III. We then adjusted its selection operator from diversity, based on the crowding distance, to the one found on reference points and preserved its parameters. After that, we continued with the adaptive EF1-NSGA-III (A-EF1-NSGA-III), and the efficient adaptive EF1-NSGA-III (A2-EF1-NSGA-III), while also contributing to explain how to generate different types of reference points. The proposed algorithms resolve optimization problems with constraints of up to 10 objective functions. We tested them on a wide range of benchmark problems, and they showed notable improvements in terms of convergence and diversity by using the Inverted Generational Distance (IGD) and the HyperVolume (HV) performance metrics. The EF1-NSGA-III aims to resolve the power consumption for Centralized Radio Access Networks and the Bi-Objective Minimum Diameter-Cost Spanning Tree problems.

Keywords: evolutionary algorithm, many-objective optimization problem

RESUMEN

Los problemas de optimización de varios objetivos se han resuelto ampliamente usando algoritmos evolutivos durante algunas décadas. A pesar de que los algoritmos NSGA-II y NSGA-III se emplean con frecuencia como referencia para evaluar nuevos algoritmos evolutivos, este último es propietario. En este artículo, utilizamos el marco NSGA-II, similar al NSGA-III, con cambios en su operador de selección. Tomamos el primer frente generado por ordenamiento no dominante para obtener puntos extremos no negativos y no repetidos. Esta versión del NSGA-III se llama EF1-NSGA-III, y su implementación no comienza desde cero; eso sería reinventar la rueda. En lugar de eso, tomamos el código NSGA-II de los autores en el repositorio del Laboratorio de Algoritmos Genéticos Kanpur para extender el EF1-NSGA-III. Luego ajustamos su operador de selección de la diversidad en función de la distancia de hacinamiento al que se encuentra usando los puntos de referencia y preservamos sus parámetros. Después continuamos con el EF1-NSGA-III adaptativo (A-EF1-NSGA-III), y el eficiente adaptativo EF1-NSGA-III (A2-EF1-NSGA-III) contribuyendo en la explicación de cómo generar diferentes tipos de puntos de referencia. Los algoritmos propuestos resuelven problemas de optimización con restricciones de hasta 10 funciones objetivos. Los probamos en una amplia gama de problemas de referencia, y mostraron mejoras notables en términos de convergencia y diversidad utilizando las métricas de rendimiento de Distancia Generacional Invertida (IGD) e Hipervolumen (HV). El EF1-NSGA-III tiene como objetivo resolver el consumo de energía para las redes de acceso de radio centralizado y los problemas del árbol de expansión de diámetro mínimo bi-objetivo.

Palabras clave: algoritmo evolutivo, problema de optimización de muchos objetivos

Received: November 18th 2019
Accepted: September 2nd 2020

[1]Electrical Engineer, Universidad Nacional de Colombia, Colombia. M.Sc. Electronic and computers Engineering, Universidad de los Andes, Colombia. Affiliation: Ph.D. Student, Universidad Nacional de Colombia, Colombia. E-mail: lfarizav@unal.edu.co
[2]Electrical Engineer, Universidad Nacional de Colombia, Colombia. M.Sc. and Ph.D. Electrical Engineering, Universidade de Sao Paulo (USP), Brazil. Affiliation: Associate Professor, Universidad Nacional de Colombia, Colombia. E-mail: jseslavag@unal.edu.co
[3]M.Sc. Electronics Engineer, Pontificia Universidad Javeriana, Colombia. Ph.D. Photonics Engineering, Technical University of Denmark, Denmark. Affiliation: Senior Researcher, Ericsson, Sweden and Lecturer, Pontificia Universidad Javeriana, Colombia. E-mail: rafael.puerta@ericsson.com

Introduction

Genetic algorithms (GAs) are random-based evolutionary methods. They are preferred over classical optimization due to their versatility in solving complex problems and finding multiple solutions without **a priori** information

How to cite: Ariza-Vesga, L. F., Eslava-Garzón, J. S, and Puerta, R. (2020). A Non-Dominated Sorting Genetic Algorithm III using the first front of the non-dominated sort procedure to obtain extreme points. *Ingenierla e Investigación*, *40*(3). 10.15446/ing.investig.v40n3.82906

(Deb, 1999). They are inspired by fundamental genetic laws such as natural selection, first introduced by Fraser (1957), and further popularized by Holland (1975). When we study GAs, we contemplate an initial population of pseudo-random solutions that pass through genetic functions such as selection, crossover, and mutation to recombine and perturb solutions. Then, we evaluate these solutions with a fitness function in the hope of creating the fittest ones that will survive and evolve to the next generation. Finally, this process ends when we use a predefined termination criterion.

GAs then evolved to Multi-Objective and Many-objective Evolutionary Algorithms (MOEAs and MaOEAs) to optimize Multi-Objective or Many-Objective Optimization Problems (MOOPs and MaOPs) for fields such as engineering, business, mathematics, and physics (Li, Wang, Zhang, and Ishibuchi, 2018). They search for multiple solutions simultaneously on different non-convex and discontinuous regions next to the approximated Pareto front. Initially, research focused on solving MOOPs, but recently, there is an increasing interest in solving MaOPS. However, for MaOPs, we have a significant number of non-dominated solutions that exponentially increase with the number of objective functions. This is due to the selection operator and the dominant resistance caused by the dimensionality curse (Purshouse and Fleming, 2007).

The practical motivation of this paper is the implementation of an open-source version of the proprietary NSGA-III (Deb and Jain, 2014; Jain and Deb, 2014) called the EF1-NSGA-III (Ariza, 2019) that alleviates the above-mentioned issue of dimensionality. This algorithm solves problems with more than two objective functions, checks the feasibility of the population to fill the non-dominated sort procedure, and then uses the first front to generate non-negative and non-repeated extreme points during the normalization procedure. It has already been employed to reduce the power consumption for Cloud Radio Access Networks (Ariza, 2020), and resolve the Bi-Objective Minimum Diameter-Cost Spanning Tree problem (Prakash, Patvardhan, and Srivastav, 2020). The authors who solved the latter used the EF1-NSGA-III as a basis to generate a new algorithm called the Permutation-code NSGA-III (P-NSGA-III).

The EF1-NSGA-III uses a non-parametric method for diversity and executes faster when we take the renowned and efficient NSGA-II code, with prominent features such as simplicity, and an elitist approach. This algorithm, found at the repository of the Kanpur Genetic Algorithms Laboratory (KanGAL, 2011), is the core of the EF1-NSGA-III, but with significant changes to the selection operator. Our strategy for this paper is borrowed from professor Kalyanmoy Deb and researchers at Michigan State University. It helped them create and unify the NSGA-III to solve any mono-objective, multi-objective, and many-objective problem (Seada and Deb, 2015).

After we finished the extension of the EF1-NSGA-III, we continued with the adaptive EF1-NSGA-III (A-EF1-NSGA-III), and the efficient adaptive EF1-NSGA-III (A^2-EF1-NSGA-III), which use different schemes of adaptive reference points to increase their associated number of population members and accomplish a better distribution of solutions. The last two new algorithms are inspired by the A-NSGA-III and A^2-NSGA-III. These algorithms have already been implemented by referenced authors (Jain and Deb, 2013). Also, we contribute to explain how to generate reference points using the Das and Dennis (1998), two-layer, and adaptive methods.

The above contributions are the smoothest way we found to create a robust algorithm, rather than starting from scratch, as did Yarpiz (Matlab) (2018), jMetal (Java) (2018), nsga3cpp (C++) (Chiang, 2014), nsga3 (Python) (Marti, 2016), and PlatEMO (Matlab) (Tian, Cheng, Zhang, and Jin, 2017). Recently, a Multi-objective Optimization framework in python called pymoo was created, and its NSGA-III implementation is available (Blank and Deb, 2020). This implementation is employed to compare the NSGA-III and the EF1-NSGA-III in terms of some performance metrics.

In the remainder of this paper, we present a revision of related works. Then, we describe our extensions of the non-dominated sorting genetic algorithms EF1-NSGA-III, A-EF1-NSGA-III, and A^2-NSGA-III. After that, we present a detailed performance evaluation using statistical analysis. Finally, we draw conclusions.

Related work

Many real-life problems are MaOPs, and a whole army of evolutionary algorithms (Seada, Abouhawwash, and Deb, 2017, 2018) is waiting to be utilized to solve them. For example, some representative algorithms are the MOEA/D (Zhang and Li, 2007), the knee point-based algorithm (KnEA) (Zhang, Tian, and Jin, 2015), the HypE (Bader and Zitzler, 2011), and the NSGA-III. They are based on decomposition, convergence enhancements, indicators, and reference points, respectively. Any evolutionary algorithm mentioned before or even another extension of them has the chance to be improved considering different stages such as the generation of the parents and offspring populations, as well as the use of novel genetic functions that recombine and perturb solutions, the information feedback of individuals from previous iterations (Wang and Tan, 2019), or the evaluation of population members (the fitness allocation method) that allow solutions passing to the next generation.

Some of the already implemented learning methods to improve evolutionary algorithms are also inspired by nature, but, in this case, they mimic a herd or colony's behavior. We can mention some swarm intelligence methods as examples, such as chaotic krill herds (Wang, G., Guo, Gandomi, Hao, and Wang, H., 2014), elephant herds (Li,, Wang, and Alavi,, 2020), bee colonies (Wang and Ji, 2017), monarch butterflies (Wang et al., 2019), and moth colonies (Wang, 2016). These methods simulate the clustering behavior of chaotic krills, elephants, and also other insect behaviors such as migration and phototaxis. These algorithms imitate how groups of animals cooperate and learn by themselves or use information from other members to do a specific task such as finding food. Some NP-hard problems that benefited from those improvements are scheduling, image, feature selection, detection, path planning, cyber-physical social system, texture discrimination, saliency detection, classification, object extraction, economic load dispatch, global numerical optimization, multi-objective optimization, knapsack problem, and fault diagnosis problems.

As can be seen, there are many options to improve evolutionary algorithms including the NSGA-III algorithm. Some new extensions of the NSGA-III algorithm have been developed since 2013. These are the A-NSGA-III and A^2-NSGA-III, which allocate adaptive reference points to improve the distribution of solutions (Jain and Deb, 2013). Another example is the U-NSGA-III, which uses the NSGA-III as the basis for implementing a unified algorithm to solve problems with up to 15 objective functions (Seada and Deb, 2015). The θ-NSGA-III trades off the diversity and the convergence of problems with more than four objective functions focused on the improvement of the non-dominated sort procedure (Yuan, Xu, and Wang, 2014).

Other works are the EliteNSGA-III, which increases the accuracy and diversity of the NSGA-III by maintaining an elite population archive to preserve previously generated elite solutions that would probably be eliminated by the original NSGA-III (Ibrahim et al., 2016). The E-NSGA-III utilizes extreme solutions in the population generation module to improve the overall quality of solutions (Wangsom, Bouvry, and Lavangnananda, 2018). The IFM-NSGAIII solves large-scale problems, instead of the small-scale problems, as the NSGA-III does, and introduces information feedback to influence the offspring population (Gu and Wang, 2020). The P-NSGA-III that modifies the elitist framework of the NSGA-III uses preferred vectors to improve local search (Shu, Wang, W., and Wang, R., 2018). Also, the Permutation-coded NSGA-III encodes chromosomes as permutations of graph vertices (Prakash et al., 2020).

Two more developments are the set of NSGA-III SBXAM, NSGA-III SIAM, and NSGA-III UCAM algorithms, which use an adaptive mutation operator and evaluate simulated binary crossover (SBX), uniform crossover (UC), and single-point crossover (SI) operators in large-scale and online problems (Yi, Deb, Dong, Alavi, and Wang, 2018; Yi et al., 2020). Finally, there is the B-NSGA-III, which converts continuous NSGA-III into binary NSGA-III modifying the initialization, crossover, and mutation operators for band selection in cloud contaminated hyper-spectral images (Gupta and Nanda, 2019).

The EF1-NSGA-III, A-EF1-NSGA-III, and A^2-EF1-NSGA-III algorithms

The basic framework of the proposed EF1-NSGA-III is similar to the KanGAL NSGA-II and inherits its parameters (Jain and Deb, 2013, 2014; Deb and Jain, 2014; Blank and Deb, 2020). It resolves MOOPs or MaOPs with conflicting objectives and inequality constraints focused on finding non-dominated solutions. Mathematically, the definition of the optimization problem is:

$$\text{Minimize} \quad f_i(x) \quad i = 1,2,\ldots,M \tag{1}$$

$$\text{Subject to} \quad g_j(x) \geq 0 \quad j = 1,2,\ldots,J \tag{2}$$

$$h_k(x) = 0 \quad k = 1,2,\ldots,K \tag{3}$$

$$x_l^{(L)} \leq x_l \leq x_l^{(H)}, \quad l = 1,2,\ldots,n \tag{4}$$

Algorithm 1 EF1-NSGA-III procedure

```
    Input: $H$ structured reference points $Z^s$, initial random parent
           population $P_t$, $| P_t |$= the population size $N$
  Output : $P_t + 1$
 1: if (The remainder when N is divided by 4) then
 2:     Start algorithm
 3: else
 4:     Abort algorithm
    end if
 5: for 1 to a number of generations defined by the user do
 6:     $S_t = \emptyset, i = 1$
 7:     $Q_t$ = Tournament selection+Mutation+crossover ($P_t$)
 8:     $R_t = P_t \cup Q_t$
 9:     $(F_1, F_2, \ldots)$ = Non-dominated-sort($R_t$)
10:     repeat
11:         $S_t = S_t \cup F_i$ and $i = i + 1$
12:     until $| S_t | \geq N$
13:     Last front to be included: $F_l = F_i$
14:     if ($| S_t |= N$) then
15:         $P_{t+1} = S_t$, continue
16:     else
17:         $P_{t+1} = \cup_{j=1}^{l-1} F_j$
18:         Points to be chosen from $F_l : K = N- | P_{t+1} |$
19:         Normalize objectives and create reference set $Z^r$ : Normalize
    $(f^n, S_t, F_1, Z^r, Z^s)$
20:         Associate each member $s$ of $S_t$ with a reference point:
    $[\pi(s, d(s)]$ = Associate $(S_t, Z^r)$: closest reference point, $d$: distance
    between $s$ and $\pi(s)$
21:         Compute niche counts of reference point $j \in Z^r$: $\rho_{j_{S_t/F_l}} =$
    $\sum_{s \in S_t/F_l}((\pi(s) = j)?1 : 0)$, and $\rho_{j_{F_l}} = \sum_{s \in F_l}((\pi(s) = j)?1 : 0)$
22:         Choose $K$ members one at a time from $F_l$ to construct $P_{t+1}$:
    Niching $(K, \rho_{j_{S_t/F_l}}, \rho_{j_{F_l}}, \pi, d, Z^r, F_l, P_{t+1})$
        end if
    end for
```

where x is a vector with p decision variables, M is the number of conflicting objectives, J is the number of inequality constraints, K is the number of equality constraints, and l is the number of bounds from low (L) to high (H).

We propose the EF1-NSGA-III, where individuals in the first front (F_1) are contemplated to obtain non-negative and non-repeated extreme points and guide the evolution of the algorithm. Algorithm 1 resumes the procedure.

The EF1-NSGA-III first checks if the remainder of N is divided by four to make systematic application pair-wise selection and pair-wise recombination operations (Seada and Deb, 2015). Then, it runs a finite number of generations defined in a loop by the user. In the beginning, the population P_t is generated randomly, and its size is N (afterwards, the parent population comes from the last generation). Its offspring Q_t is generated using tournament selection, mutation, and crossover operations, and its size is N. The combined parent and offspring population $R_t = P_t + Q_t$ has a size of $2N$. Population R_t is sorted according to different non-dominated sorting levels or fronts $(F_1, F_2, \ldots, F_l)$, where F_l is the last front. All members of each front are selected one at a time to build Population S_t. If front members reach the size of the population P_t, there is nothing more to do for that generation and $S_t = P_{t+1}$. If S_t population plus the last front exceeds the size of P_t, then the last front F_l is accepted partially, and $l + 1$ onward fronts are rejected. The last front F_l selects members for population S_t that maximize its diversity through the niching operator. This operator requires the determination of reference points on a hyper-plane and adaptive normalization of population members, association operator until reaching the niching preservation operation. Finally, the next generation parent population P_{t+1} is S_t, plus members from the niching operator, improve better the diversity until reaching the size N. There is no difference between the EF1-NSGA-III and the NSGA-III relay on normalization, niching procedures, and selection.

In the following section, we explain our implementations to generate reference points using simple algebraic principles. They are the well-known Das and Dennis, two-layer, k-layer, adaptive, and efficient adaptive methods (1998).

Reference Point Generators

Das and Dennis approach

We used the Das and Dennis method (1998) for EF1-NSGA-III, A-EF1-NSGA-III, and A^2-EF1-NSGA-III algorithms to generate well-distributed reference points and ensure diversity in obtained solutions. This method is the basis to generate different types of reference points such as two-layer, k-layer, adaptive, and efficient adaptive.

The predefined Das and Dennis approach places reference points on a normalized hyper-plane equally to all axis and has an intercept of one on each axis. If p divisions are considered along the M axis, the total number of reference points H is determined by

$$H = \binom{M + p - 1}{p} \tag{5}$$

This reference point generator is used for problems with up to five objective functions. An example can be seen in Figure 1a, for $M = 3$ and $p = 12$. High-dimensional problems with more objective functions require a different approach because the number of reference points increases exponentially with the number of objective functions.

Two-layer of Reference Points

This alternative reduces the number of reference points for high-dimensional problems compared to the Das and Dennis method. This approach is divided into the following steps:

1. *Generate reference points of the boundary layer:* Use the Das and Dennis approach. The boundary layer must satisfy the plane equation $x_1 + x_2 + \cdots + x_{nobj} = 1$, where $nobj = M$.

2. *Generate reference points of the inside layer:* Use the Das and Dennis approach again, but reference points of the inside layer should be on the plane $x_1 + x_2 + \cdots + x_{nobj} = 1 - \frac{1}{ndiv-1}$. The number of divisions of the inside layer is $ndiv - 1$, where $ndiv = p$.

3. *Move reference points from the inside layer to the boundary layer:* Reference points of the inside layer are moved to the boundary layer by adding the value $d = \frac{1}{(ndiv-1)*nobj}$ in all dimensions. Figure 1b, right side, depicts the distribution of reference points.

K-layer of Reference Points

It follows the two-layer reference points procedure, but with more than one inside layer. Additional information can be revised in Jiang and Yang (2017). Figure 1b, on the left side, shows the reference point distribution.

Adaptive Reference Points

First of all, this method requires the identification of ***useful*** or ***crowded*** reference points in cases where the niche count $\rho_{j_{S_t/F_l}} > 1 (j \in \{0, 1, \ldots, H-1\})$; and second, the generation of adaptive reference points around j_{th}. Before a new adaptive reference point is accepted, it must be non-repeated and lie on the positive orthant. After the adding task, the niche value of all reference points is updated, and adaptive reference points with niche count $\rho_{j_{S_t/F_l}} = 0$ are deleted. In the next generations, adaptive reference points are added and deleted as long as ***useful*** reference points exist. The procedure for each ***crowded*** reference point is described below:

1. *Generate initial adaptive reference points for one division:* Initial adaptive reference points $\boldsymbol{x_1} = (\frac{1}{ndiv}, 0, \ldots, 0)$, $\boldsymbol{x_2} = (0, \frac{1}{ndiv}, \ldots, 0)$ and $\boldsymbol{x_{nobj}} = (0, 0, \ldots, \frac{1}{ndiv})$ are generated using the Das and Dennis approach to satisfy the plane equation $x_1 + x_2 + \cdots + x_{nobj} = \frac{1}{ndiv}$.

2. *Move adaptive reference points to the plane $x_1 + x_2 + \cdots + x_{nobj} = 0$:* it is necessary to project the point $\boldsymbol{p} = (0, 0, \ldots, 0)$ onto the plane $x_1 + x_2 + \cdots + x_{nobj} = \frac{1}{ndiv}$. Any reference point of the plane can be chosen, for example, $\boldsymbol{r} = (\frac{1}{ndiv}, 0, \ldots, 0)$. With this vector in mind, vector $\boldsymbol{h} = \boldsymbol{p} - \boldsymbol{r} = (0, 0, \ldots, 0) - (\frac{1}{ndiv}, 0, \ldots, 0) = (-\frac{1}{ndiv}, 0, \ldots, 0)$ is obtained. Immediately, the unit normal vector is $\vec{\boldsymbol{u}} = \frac{1}{\sqrt[2]{nobj}}(1, 1, \ldots, 1)$. The vector to add initial adaptive reference points is:

$$\vec{\boldsymbol{d}} = |\, \boldsymbol{h}.\boldsymbol{u} \,| * \vec{\boldsymbol{u}} = \frac{1}{ndiv * nobj} * (1, 1, \ldots, 1) \tag{6}$$

3. *Move adaptive reference points to the **crowded** reference point jth*: In this step, after niching values $\rho_{j_{S_t/F_l}}$ are updated, we identify ***crowded*** reference points with niche values $\rho_{j_{S_t/F_l}} > 1$. After that, we add to the adaptive reference point near the ***crowded*** reference point. In the end, for a three-dimensional case, the last three reference points create the inner triangle around each crowded reference point, as shown in Figure 1c on the left side.

4. *Repeat previous tasks for all crowded reference points.*

Efficient Adaptive Reference Points

This method has the following steps:

1. *Generate efficient adaptive reference points for one division*: Initial efficient adaptive reference points $\boldsymbol{x_1} = (\frac{1}{\lambda * ndiv}, 0, \ldots, 0)$, $\boldsymbol{x_2} = (0, \frac{1}{\lambda * ndiv}, \ldots, 0)$, and $\boldsymbol{x_{nobj}} = (0, 0, \ldots, \frac{1}{\lambda * ndiv})$ are generated using the Das and Dennis approach (λ is a positive number called the scaling factor; its value is 1 for this work). They satisfy the plane equation $x_1 + x_2 + \cdots + x_{nobj} = \frac{1}{\lambda * ndiv}$

2. *Move efficient adaptive reference points to the plane* $x_1 + x_2 + \cdots + x_{nobj} = 0$: Initial efficient adaptive reference points are moved a distance $d = \frac{1}{\lambda * ndiv * nobj}$ in all dimensions.

3. *Subtract efficient adaptive reference points*: One of the previous efficient adaptive reference points is subtracted. In a three-dimensional case, this task is executed three-fold to obtain nine points (three of them are the same vector (0,0,0)).

4. *Move efficient adaptive reference points to the* ***crowded*** *reference point jth*: The final structure of reference points is moved next to the ***crowded*** reference point at a distance $d = \frac{1}{\lambda * ndiv * nobj}$ in all axes. Considering a three-dimensional example, the efficient adaptive reference points create three triangles connected to the crowded reference point as depicted in Figure 1c on the right side.

5. *Repeat for all crowded reference points.*

In next subsections, we show the procedures for the EF1-NSGA-III, except the associate one that is the same as the one from the NSGA-III authors.

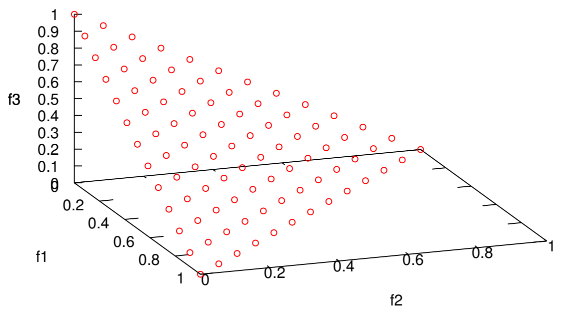
Normalization of Population Members

We normalized the objective values of population members to have the same range of the reference points as presented in Algorithm 2. The normalization procedure has the following steps:

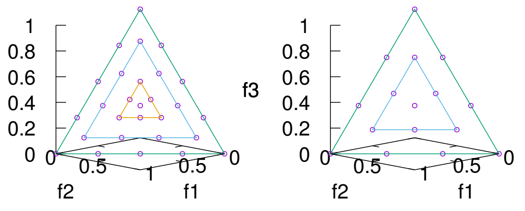
First, we construct the ideal vector $\bar{z} = \left\{z_1^{\min}, z_2^{\min}, z_j^{\min}, \ldots, z_M^{\min}\right\}$ of the population S_t which is determined by identifying the minimum value from each objective function. Second, we translate each objective $f_j(x)$ by subtracting $z_j^{\min}$, denoted as $f'_j(x) = f_j(x) - z_j^{\min}$. Third, we determine M extreme points to constitute a M-dimensional hyperplane that makes the Achievement Scalarization Function (***ASF***) minimum (Blank, Deb, and Roy, 2019), expressed as $z^{j,\max} = f'^{n}(x) \mid \min_{n \in F_1}\{\boldsymbol{ASF^j}\}$ where $j \in \{1, 2, \ldots, M\}$, and n is the population size of the first front F_1. The $\boldsymbol{ASF^j}$ function finds the maximum translated objective value divided by the scalarization value in all dimensions, denoted as $\boldsymbol{ASF^j} = \max_{\min M}\{f'_m(x)/s_m^j\}$. For example, in a three-objective problem, we have the scalarization vector $\boldsymbol{s^1} = (1, \epsilon, \epsilon)$ for $j = 1$, $\boldsymbol{s^2} = (\epsilon, 1, \epsilon)$ for $j = 2$, $\boldsymbol{s^3} = (\epsilon, \epsilon, 1)$ for $j = 3$, where ϵ is a small value such as 0,0000000001.

Fourth, we find the matrix $\boldsymbol{z^{\max -1}} = \left[[z^{1,\max}]; [z^{2,\max}]; [z^{3,\max}]; [\cdots]; [z^{M,\max}]\right]^{-1}$.

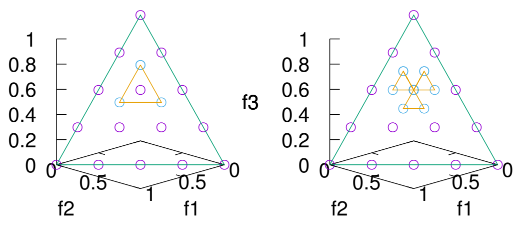
Finally, the objectives $f_j^n(x) = \frac{f'_j(x)}{a_j}$ are normalized using $a_j = \sum_{i=1}^{M}$ as the intercept (elements of $\boldsymbol{z^{\max -1}}$ j column). Eventually, repeated or negative intercepts can appear. Therefore, we use the $\boldsymbol{z^{\max -1}} = \left[[z_1^{\max}]; [z_2^{\max}]; [z_3^{\max}]; [\cdots]; [z_M^{\max}]\right]^{-1}$ vector composed of the maximum



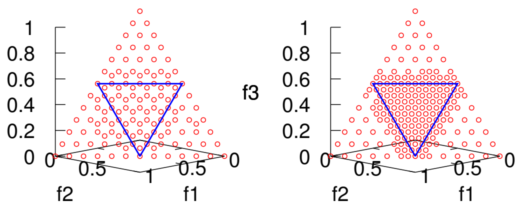
(a) Das and Dennis reference points distribution for $M = 3$ and $p = 12$.



(b) Left side, k-layer of reference points. Right side, two-layer of reference points.



(c) Left side, adaptive reference points. Right side, efficient adaptive reference points.



(d) Left-side, adaptive reference points. Right-side, efficient reference points.

Figure 1. Reference points approaches for NSGA-III, A-NSGA-III, and A^2-NSGA-III.
Source: Authors

value of each function to generate, intercept, and then normalize the functions (the last generation positive intercept can be utilized as well). Recently, authors have revealed how to handle degenerate cases and negative intercepts (Blank et al., 2019) to avoid the algorithm getting stuck, but that solution remains as future research.

Algorithm 2 Normalize ($\boldsymbol{f^n}$, S_t, F_1) procedure

```
    Input: $S_t$, $F_1$
    Output : $\boldsymbol{f^n}$, $Z^r$ (Das and Dennis reference points)
 1: for $j = 1$ to $M$ do
 2:     Compute ideal point: $Z_j^{min} = \min_{S \in S_t} f_j(\boldsymbol{s})$
 3:     Translate objectives: $f'_j(\boldsymbol{s}) = f_j(\boldsymbol{s}) - z_j^{min} \; \forall \boldsymbol{s} \in S_t$
 4:     Compute extreme points: $(\boldsymbol{z}^{j,max}, j = 1, \ldots, M)$ of $F_1$
    end for
 5: Compute intercepts $a_j$ for $j = 1, \ldots, M$
 6: Normalize objectives $\boldsymbol{f^n}$: $f_j^n(\boldsymbol{x}) = \frac{f'_i(\boldsymbol{x})}{a_i}$
```

Niche-preservation Operation

The niching procedure is presented in Algorithm 3. It identifies the reference point set $J_{\min_{S_t/F_l}} = j : argmin_{j \in Z^r} \rho_{S_t/F_l}$ having minimum ρ_{S_t/F_l} (Deb and Jain, 2014). If there are multiple associated reference points, one member of the set called $\bar{j} \in J_{\min_{S_t/F_l}}$ is chosen at random (the EF1-NSGA-III takes the first associated reference point, but any of them at random is valid).

For each generation, we neglect the reference point j which has population members associated with neither S_t/F_l nor F_l. Also, j is excluded if $\rho_{j_{F_l}} = 0$ and $\rho_{j_{S_t/F_l}} > 0$. The above conditions guarantee that the niching function runs K times to fill all P_{t+1} vacancies.

Algorithm 3 Niching ($K, \rho_{j_{S_t/F_l}}, \rho_{j_{F_l}}, \pi, d, Z^r, F_l$) procedure

```
    Input: $K, \rho_{j_{S_t/F_l}}, \rho_{j_{F_l}}, \pi(\boldsymbol{s} \in S_T), d(\boldsymbol{s} \in S_T), Z^r, F_l, P_{t+1}$
    Output : $P_{t+1}$
 1: $k = 1$
 2: while $k \le K$ do
 3:     $J_{\min_{\rho_{j_{S_t/F_l}}}} = \{j : argmin_{j \in Z^r \rho_{j_{S_t/F_l}}}\}$
 4:     $\bar{j}$=Select the first member of ($J_{\min_{\rho_{j_{S_t/F_l}}}}$)
 5:     $I_{\bar{j}} = \{\boldsymbol{s} : \pi = \bar{j}, \boldsymbol{s} \in F_l\}$
 6:     if $I_{\bar{j}} \neq 0$ then
 7:         if $\rho_{j_{S_t/F_l}} = 0 \&\& \rho_{j_{F_l}} = 0$ then
 8:             $P_{t+1} = P_{t+1} \cup (\boldsymbol{s} : argmin_{\boldsymbol{s} \in I_{\bar{j}}} d(\boldsymbol{s}))$
 9:         else
10:             $P_{t+1} = P_{t+1} \cup$ first member of $(I_{\bar{j}})$
            end if
11:         $\rho_{\bar{j}} = \rho_{\bar{j}} + 1, F_l = F_l \backslash \boldsymbol{s}$
12:         $k = k + 1$
13:     else
14:         $Z^r = Z^r \backslash \{\bar{j}\}$
        end if
    end while
```

After excluding some *useless* reference points, no matter how many S_t/F_l members are associated with the reference point $\bar{j}$, the one F_l member having the shortest perpendicular distance from the reference line $\bar{j}$ is added to P_{t+1}. Finally, the niche count $\rho_{\bar{j}_{S_t/F_l}}$ for reference point $\bar{j}$ is incremented by one and $\rho_{\bar{j}_{F_l}}$ is subtracted by one. Remember that, after niching, $\sum_{j=1}^{|H|} \rho_{j_{St/Fl}} = N$.

The next subsection explains the A-EF1-NSGA-III, and A^2-EF1-NSGA-III algorithms and their procedures to add and delete adaptive reference points.

Adaptive EF1-NSGA-III and Efficient Adaptive EF1-NSGA-III

Several MOOPs and MaOPs that are solved using the NSGA-III might have some reference points that will never be associated with the population. We address this fact through adaptive and efficient adaptive reference points that enhance the distribution of solutions.

We extended the A-EF1-NSGA-III and the A^2-EF1-NSGA-III - more details about their algorithms in Jain and Deb (2013). The last one has a higher adaptive reference point density, modulated by the scaling factor λ. It goes close to the Pareto-optimal front in smaller regions, allows adding reference points in the hyperplane corner, improves the deletion routine, and reduces the premature introduction of reference points in undesirable regions. The two adaptive EF1-NSGA-III algorithms define reference points with niche counts $\rho_{\bar{j}_{S_t/F_l}} > 0$ as ***useful***, and reference points with $\rho_{\bar{j}_{S_t/F_l}} = 0$ as ***useless***. The following procedures are added after the niche count of all reference points are updated:

Add adaptive or efficient adaptive reference points

This task generates new reference points based on the adaptive and efficient adaptive reference points studied before, which are plotted in Figure 1d at the end of the IDTLZ1 optimization process (the left-side corresponds to adaptive reference points and the right-side to efficient adaptive reference points). One of the NSGA-III termination criteria is to have all reference points associated with each population member ($\rho_{\bar{j}_{S_t/F_l}} = 1$), thus indicating a well-distributed Pareto front. Notwithstanding, sometimes that is impossible because the amount of reference points is higher than the population size, and some of them have niche counts $\rho_{\bar{j}_{S_t/F_l}} = 0$. They are relocated next to ***crowded*** reference points, and their capacity to capture a well-widespread Pareto front depends on the scaling factor λ and the settled down condition.

Delete adaptive or efficient adaptive reference points

This task deletes existing reference points. It is required to associate the added reference points to the population and update the niche value $\rho_{\bar{j}_{S_t/F_l}}$ (do not forget that $\sum_{j=1}^{|H|} \rho_{\bar{j}_{S_t/F_l}} = N$ for P_{t+1}). The deletion function is computationally more expensive than the adding one. In this case, before deleting *useless* reference points, we need to repeat the association and niching procedures and steady-state new reference points insertion routines.

The subsections below depict the constraint handling and the complexity discussion of EF1-NSGA-III algorithms.

Constraint Handling

We extended the constraint domination principle adopted in the NSGA-II algorithm for EF1-NSGA-III, A-EF1-NSGA-III, and A^2-EF1-NSGA-III. It is handled automatically by the initialization process, where all population members satisfy the bounds. Also, the creation of offspring solutions must be within lower and upper bounds. The evaluation of each individual considers inequality constraints, which are configured in the problem definition function at the beginning of the simulation. When the non-dominated sorting procedure starts, all individuals within the combined population $R_t = P_t \cup Q_t$ are categorized as feasible and infeasible. $\boldsymbol{x}^{(1)}$ is constraint-dominate $\boldsymbol{x}^{(2)}$ if one of the following conditions is true. First, if $\boldsymbol{x}^{(1)}$ is feasible and $\boldsymbol{x}^{(2)}$ is infeasible. Second, if $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ are infeasible and $\boldsymbol{x}^{(1)}$ has the smaller constraint violation. Third, if $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ are feasible and $\boldsymbol{x}^{(1)}$ dominates $\boldsymbol{x}^{(2)}$ with the usual domination principle.

There are four cases for the non-domination sort function in the tournament procedure to select individuals for the crossover function:

- First, all solutions are infeasible; solutions are sorted by minimum violation values and copied to the next generation.
- Second, the number of feasible individuals is lower than the population size; all feasible individuals are copied to the next generation.
- Third, the number of infeasible individuals equals twice the population size; the unconstrained non-domination sorting procedure is followed.
- Finally, the number of feasible individuals is higher or equal to the population size; the unconstrained non-domination sorting procedure is carried out with feasible individuals.

Complexity discussion

The NSGA-III and EF1-NSGA-III have a similar complexity because the EF1-NSGA-III inherits some of the NSGA-III procedures. The niche complexity requires $O(N)$ computations, and the rest is bounded by the maximum between $O(N^2 \log^{M-2} N)$ and $O(MN^2)$. However, if $M > N^2 \log^{M-2} N$, then he generation-wise complexity of the EF1-NSGA-III is $O(MN^2)$. The A-EF1-NSGA-II and A^2-EF1-NSGA-III have a higher complexity, thus requiring more computations for each generation. In those cases, the complexity of the niche function is $2 * O(N)$.

Performance evaluation

In this section, we provide the performance results of EF1-NSGA-III, A-EF1-NSGA-III, and A^2-EF1-NSGA-III to solve the DTLZ test (Deb, Thiele, and Laumanns, 2005) and real (water and car-side impact) problems handling constraints on three to ten objective functions. The employed hardware is a 3rd generation Intel core i7-3700x4 with 8 GBs of RAM, and the utilized software in Ubuntu 16.04, kernel 4.13.0-45-generic, and GCC version 5.4.0. We ran all the simulations using 20 realizations with the same random seeds for each problem.

Additionally, we took the population and the number of reference points for each DTLZ test problem reported in Table I of the work by Deb and Jain (2014) and in Table 1 of this work. For tuning purposes, we used the NSGA-III parameters employed by authors. They are shown in Table II of Deb and Jain (2014), as well as here in Table 2.

Table 1. Reference points and population sizes used for EF1-NSGA-III, A-EF1-NSGA-III, and A^2-EF1-NSGA-III algorithms. Nobj is the number of objective functions, and Boundary ndiv is the number of divisions of the boundary plane built with reference points. Within ndiv is the number of division of the inside plane built with reference points, and Popsize is the population size.

Nobj	Boundary ndiv	Inside ndiv	H	Popsize
3	12	0	91	92
5	6	0	210	212
8	3	2	156	156
10	3	2	275	276

Source: Authors

Table 2. Parameter values used in EF1-NSGA-III, A-EF1-NSGA-III, and A^2-EF1-NSGA-III algorithms.

Parameters	A/A^2/EF1-NSGA-III
Simulated binary crossover probability ρ_c	1
Polynomial mutation probability ρ_m	1/n
Distribution index for crossover η_c	30
Distribution index for mutation η_m	20

Source: Authors

Traditionally, the ZDT (Zitzler, Deb, and Thiele, 2000), DTLZ (Deb et al., 2005), WFG (Huband, Hingston, Barone, and While, 2006), and LZ/UF (Li and Zhang, 2009)(Zhang and Li, 2007) test suites are commonly-used benchmark multi-objective test problems. They are employed to validate MOEAs and MaOEAs under some difficulties, such as objective scalability, complicated Pareto sets, bias, disconnection, or degeneracy (Li et al., 2018). There are even more tests and real problems to evaluate evolutionary algorithms (Surjanovic and Bingham, 2013). However, we concentrated our effort to benchmark DTLZ test problems with different versions of the NSGA-III. Along with the proposed algorithms, we compared our results with those reported by authors and some publicly available open-source versions in terms of Inverted Generational Distance (IGD) and HiperVolume (HV) performance metrics (Chiang, 2014; Tian et al., 2017; Bi and Wang, 2017; Blank and Deb, 2020).

Normalized DTLZ1-4 Test Problems

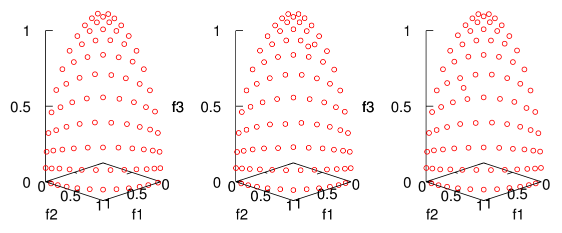
The original NSGA-III implementation (Deb and Jain, 2014; Jain and Deb, 2014), and others found in references of this work (Ariza, 2019; Chiang, 2014; Tian et al., 2017; Blank and Deb, 2020) find well-distributed solutions when solving MaOPs. We noticed that there is a small difference between them regarding the IGD performance metric on three-objective to ten-objective normalized DTLZ1-4 test problems, as summarized in Table 3. It shows that the EF1-NSGA-III outperforms most scenarios, followed by the proposals by Deb and Jain (2014), Blank and Deb (2020), Chiang (2014), and Tian et al. (2017). Nevertheless, it has poor performance when solving normalized eight-objective DTLZ1 test problems and above.

Table 3. Best, Worst and Median IGD values for NSGA-III

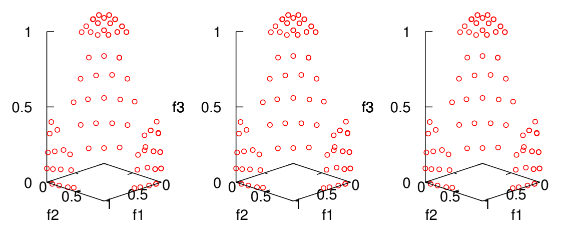
Problem	M	MaxGen	EF1-NSGA-III	Deb and J, 2014	Chiang, 2014	Tian, 2017	Blank, 2020
			1,419x10^{-4}	4,880x10^{-4}	7,593x10^{-4}	2,0567x10^{-1}	4,86x10^{-4}
DTLZ1	3	400	**1,527x10^{-3}**	4,880x10^{-3}	2,053x10^{-1}	2,1505x10^{-1}	5,519x10^{-3}
			3,686x10^{-4}	1,308x10^{-3}	2,845x10^{-3}	2,0643x10^{-1}	1,429x10^{-3}
			1,077x10^{-4}	5,116x10^{-4}	7,718x10^{-4}	5,27x10^{-1}	5,23x10^{-4}
DTLZ1	5	600	**3,437x10^{-4}**	1,979x10^{-3}	4,733x10^{-1}	5,35x10^{-1}	2,513x10^{-3}
			1,361x10^{-4}	9,799x10^{-4}	2,05x10^{-3}	5,27x10^{-1}	1,121x10^{-3}
			5,223x10^{1}	**2,044x10^{-3}**	3,454x10^{-3}	9,6773x10^{-1}	2,34x10^{-3}
DTLZ1	8	750	5,876x10^{1}	**8,721x10^{-3}**	4,290x10^{-1}	2,5829x10^{-1}	1,0268x10^{-1}
			5,554x10^{1}	**3,979x10^{-3}**	5,279x10^{-3}	9,7509x10^{-1}	3,804x10^{-3}
			8,033x10^{1}	**2,215x10^{-3}**	2,257x10^{-3}	0,998x10^{-1}	2,419x10^{-3}
DTLZ1	10	1 000	9,744x10^{1}	**6,869x10^{-3}**	2,744x10^{-1}	1,541x10^{-1}	5,443x10^{-3}
			8,256x10^{1}	**3,462x10^{-3}**	4,235x10^{-3}	1,101x10^{-1}	3,098x10^{-3}
			9,993x10^{-4}	1,262x10^{-3}	1,585x10^{-3}	5,4468x10^{-1}	1,137x10^{-3}
DTLZ2	3	250	3,451x10^{-3}	**2,114x10^{-3}**	6,911x10^{-3}	5,4524x10^{-1}	2,212x10^{-3}
			1,591x10^{-3}	**1,357x10^{-3}**	2,731x10^{-3}	5,4485x10^{-1}	1,385x10^{-3}
			2,610x10^{-3}	4,254x10^{-3}	4,701x10^{-3}	1,612x10^{-1}	4,372x10^{-3}
DTLZ2	5	350	6,582x10^{-3}	**5,862x10^{-3}**	1,547x10^{-1}	1,6523x10^{-1}	5,948x10^{-3}
			2,890x10^{-3}	4.982x10^{-3}	6,299x10^{-3}	1,6517x10^{-1}	4,897x10^{-3}
			5,582x10^{-3}	1,371x10^{-1}	1,54x10^{-1}	3,0123x10^{-1}	1,368x10^{-1}
DTLZ2	8	500	**9,013x10^{-3}**	1,811x10^{-1}	2,002x10^{-1}	5,9472x10^{-1}	1,754x10^{-1}
			7,001x10^{-3}	1,571x10^{-1}	1,740x10^{-1}	3,1599x10^{-1}	1,502x10^{-1}
			5,297x10^{-3}	1,350x10^{-1}	1,384x10^{-1}	2,123x10^{-1}	1,194x10^{-1}
DTLZ2	10	750	**6,560x10^{-3}**	1,697x10^{-1}	2,004x10^{-1}	7,011x10^{-1}	1,813x10^{-1}
			5,444x10^{-3}	1,528x10^{-1}	1,727x10^{-1}	5,319x10^{-1}	1,528x10^{-1}
			5,189x10^{-4}	9.751x10^{-4}	1,374x10^{-3}	5,4495x10^{-1}	3,295x10^{-3}
DTLZ3	3	1 000	**8,251x10^{-3}**	6,665x10^{-3}	1,129x10^{-1}	5,4955x10^{-1}	1,379x10^{-1}
			1,927x10^{-3}	4,007x10^{-3}	4,757x10^{-3}	5,4627x10^{-1}	5,647x10^{-3}
			5,342x10^{-4}	3,086x10^{-3}	3,462x10^{-3}	1,659x10^{-1}	1,839x10^{-3}
DTLZ3	5	1 000	**7,759x10^{-3}**	1,196x10^{-1}	1,04x10^{-1}	3,656x10^{-1}	9,898x10^{-3}
			1,103x10^{-3}	5,960x10^{-3}	9,447x10^{-3}	1,722x10^{-1}	6,487x10^{-3}
			2,200x10^{-3}	1,244x10^{-1}	1,564x10^{-1}	3,1355x10^{-1}	1,668x10^{-1}
DTLZ3	8	1 000	**1,044x10^{-1}**	9,649x10^{-1}	1,386x10^{-1}	3,346x10^{-1}	8,799x10^{-1}
			3,243x10^{-3}	2,375x10^{-1}	3,014x10^{-1}	3,1965x10^{-1}	3,429x10^{-1}
			1,383x10^{-3}	8,849x10^{-3}	2,014x10^{-1}	2,224x10^{-1}	9,186x10^{-3}
DTLZ3	10	1 500	**2,255x10^{-3}**	2,083x10^{-1}	4,817x10^{-1}	7,897x10^{-1}	6,517x10^{-1}
			1,764x10^{-3}	1,188x10^{-1}	3,092x10^{-1}	5,3427x10^{-1}	1,221x10^{-1}
			4,059x10^{-4}	2,915x10^{-4}	2,447x10^{-4}	5,3578x10^{-1}	**2,01x10^{-4}**
DTLZ4	3	600	5,319x10^{-1}	**4,286x10^{-1}**	5,33x10^{-1}	5,279x10^{-1}	9,50x10^{-1}
			5,317x10^{-4}	5,970x10^{-4}	5.778x10 $^{-4}$	5,452x10^{-1}	**3,30x10^{-4}**
			3,632x10^{-4}	9,849x10^{-4}	4,305x10^{-4}	1,64x10^{-1}	4,08x10^{-4}
DTLZ4	5	1 000	5,319x10^{-1}	1,721x10^{-3}	1,108x10^{-3}	1,659x10^{-1}	**1,595x10^{-3}**
			6,182x10^{-4}	1,255x10^{-3}	7,294x10^{-4}	1,651x10^{-1}	**5,65x10^{-4}**
			2,200x10^{-3}	5,079x10^{-3}	3,406x10^{-3}	3,151x10^{-1}	3,334x10^{-3}
DTLZ4	8	1 250	1,044x10^{-1}	6,051x10^{-1}	5,496x10^{-3}	6,5476x10^{-1}	**5,476x10^{-3}**
			3,243x10^{-3}	7,054x10^{-3}	4,345x10^{-3}	3,1532x10^{-1}	4,046x10^{-3}
			1,642x10^{-3}	5,694x10^{-3}	3,633x10^{-3}	4,19x10^{-1}	3,626x10^{-3}
DTLZ4	10	2 000	**2,254x10^{-3}**	1,076x10^{-1}	5,301x10^{-3}	4,761x10^{-1}	5.601x10^{-3}
			1,764x10^{-3}	6,337x10^{-3}	4,601x10^{-3}	4,401x10^{-1}	4,413x10^{-3}

Source: Authors

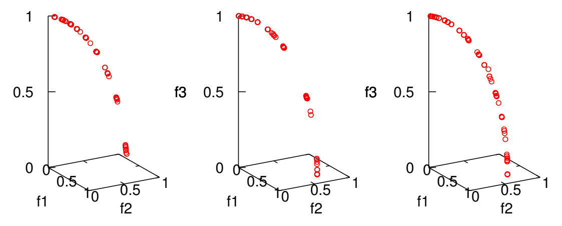
We visualized some three-objective problems solved with EF1-NSGA-III, A-EF1-NSGA-III, and A^2-EF1-NSGA-III in Figure 2. They are DTLZ2 (Figure 2a), C2-DTLZ2 (Figure 2b), DTLZ5 (Figure 2c), DTLZ7 (Figure 2d), DTLZ1 (Figure 2e), DTLZ4 (Figure 2f), IDTLZ1 (Figure 2g), and Car-Side Impact (Figure 2h) problems, respectively. At first glance, The EF1-NSGA-III seems to achieve well-distributed and well-spread solutions for those problems. Also, the A^2-EF1-NSGA-III performs better than the A-EF1-NSGA-III, and the latter is better than the EF1-NSGA-III for some DTLZ test problems with difficulties, for instance, in normalized DTLZ5, DTLZ7, and IDTLZ1 test problems.



(a) 3D DTLZ2 test problem.

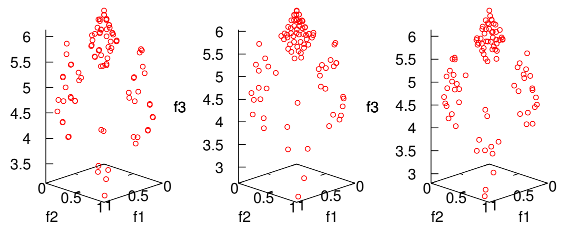


(b) 3D C2-DTLZ2 test problem.

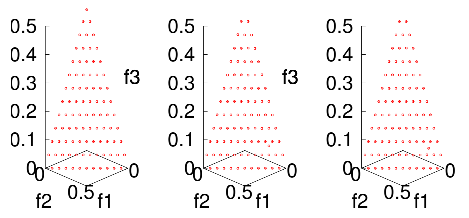


(c) 3D DTLZ5 test problem.

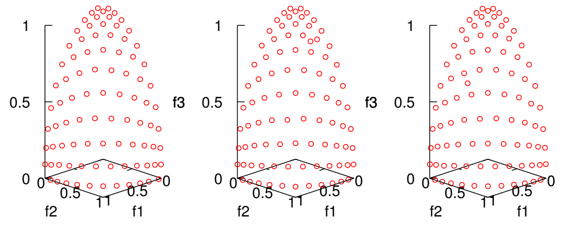
Figure 2. From left to right: Using the EF1-NSGA-III, EF1-A-NSGA-III and EF1-A^2-NSGA-III to solve three dimensional problems.
Source: Authors



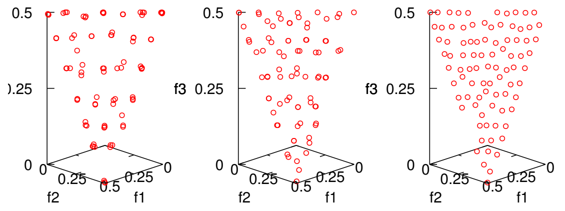
(d) 3D DTLZ7 test problem.



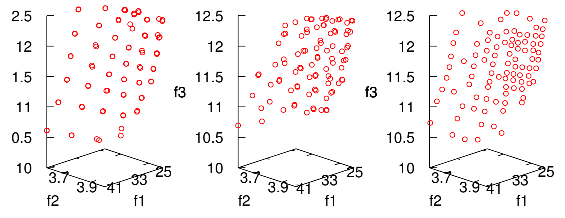
(e) 3D DTLZ1 test problem.



(f) 3D DTLZ4 test problem.



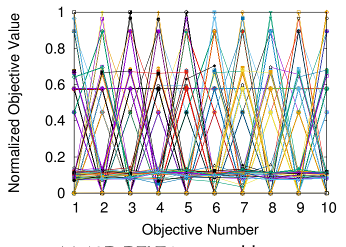
(g) 3D IDTLZ1 test problem.



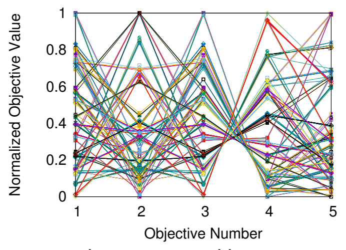
(h) 3D Car-Side Impact problem.

Figure 2. Continuation

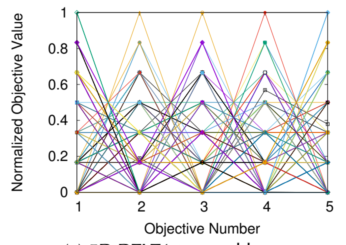
High dimensional problems are depicted in Figure 3 as well. In this case, we employed parallel coordinates plots to visualize the solutions for ten-objective DTLZ4 (Figure 3a), five-objective Water (Figure 3b), five-objective DTLZ1 (Figure 3c), and five-objective DTLZ2 (Figure 3d) problems, respectively.



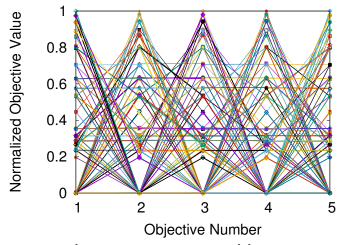
(a) 10D DTLZ4 test problem.



(b) 5D Water problem.



(c) 5D DTLZ1 test problem.



(d) 5D DTLZ2 test problem.

Figure 3. EF1-NSGA-III solving High dimensional problems.
Source: Authors

Normalized DTLZ5 and DTLZ7 Test Problems

DTLZ5 and DTLZ7 test problems have irregular Pareto fronts that reveal the diversity maintenance of reference point adaption.

The DTLZ5 is a degenerate test problem whose Pareto Front lies on a dimension curve independent of the number of objectives. On the contrary, the DTLZ7 problem examines the algorithm's ability to sustain sub-population in different Pareto-optimal regions. Figures 2c and 2d depict the Pareto fronts of three-objective DTLZ5 and DTLZ7 problems. Other algorithms, such as Adaw (Li and Yao, 2020), have a consistently better distribution of solutions when solving both problems mentioned before. We observed that the EF1-NSGA-II is slightly better than other proposals in terms of the IGD metric, as shown in Table 4.

Table 4. Average and Standard Deviation IGD values for NSGA-III

Problem	M	MaxGen	EF1-NSGA-III	Tian, 2017	Bi, 2017	Blank, 2020
DTLZ5	3	1 000	6,6588x10^{-2}	1,195x10^{-2}	1,438x10^{-2}	**1,468x10^{-2}**
			4,114x10^{-2}	3,07x10^{-3}	2,19x10^{-3}	**1,589x10^{-3}**
DTLZ5	5	1 000	1,585x10^{-1}	1,515x10^{-1}	**9,654x10^{-2}**	9,912x10^{-2}
			4,67x10^{-2}	6,582x10^{-2}	3,60x10^{-2}	**2,35x10^{-2}**
DTLZ5	8	1 000	**3,007x10^{-1}**	4,1928x10^{-1}	3,473x10^{-1}	3,712x10^{-1}
			4,376x10^{-2}	5,07x10^{-1}	9,93x10^{-2}	9,042x10^{-2}
DTLZ5	10	1 500	**1,902x10^{-1}**	2,487x10^{-1}	4,548x10^{-1}	5,114x10^{-1}
			3,932x10^{-2}	5,96x10^{-2}	8,49x10^{-2}	7,712x10^{-2}
DTLZ7	3	1 000	1,108x10^{-1}	9,063x10^{-2}	**7,794x10^{-2}**	1,308x10^{-1}
			6,186x10^{-2}	6,64x10^{-2}	**1,59x10^{-3}**	1,151x10^{-1}
DTLZ7	5	1 000	2,955x10^{-1}	**2,782x10^{-1}**	2,853x10^{-1}	2,955x10^{-1}
			1,227x10^{-2}	1,64x10^{-2}	5,83x10^{-3}	**5,642x10^{-3}**
DTLZ7	8	1 000	**5,679x10^{-1}**	7,746x10^{-1}	7,441x10^{-1}	6,242x10^{-1}
			3,182x10^{-2}	5,36x10^{-2}	**1,68x10^{-2}**	2,994x10^{-2}
DTLZ7	10	1 500	**6,525x10^{-1}**	9,250x10^{-1}	9,850x10^{-1}	7,579x10^{-1}
			3,218x10^{-2}	**2,40x10^{-2}**	1,00x10^{-1}	4,293x10^{-2}

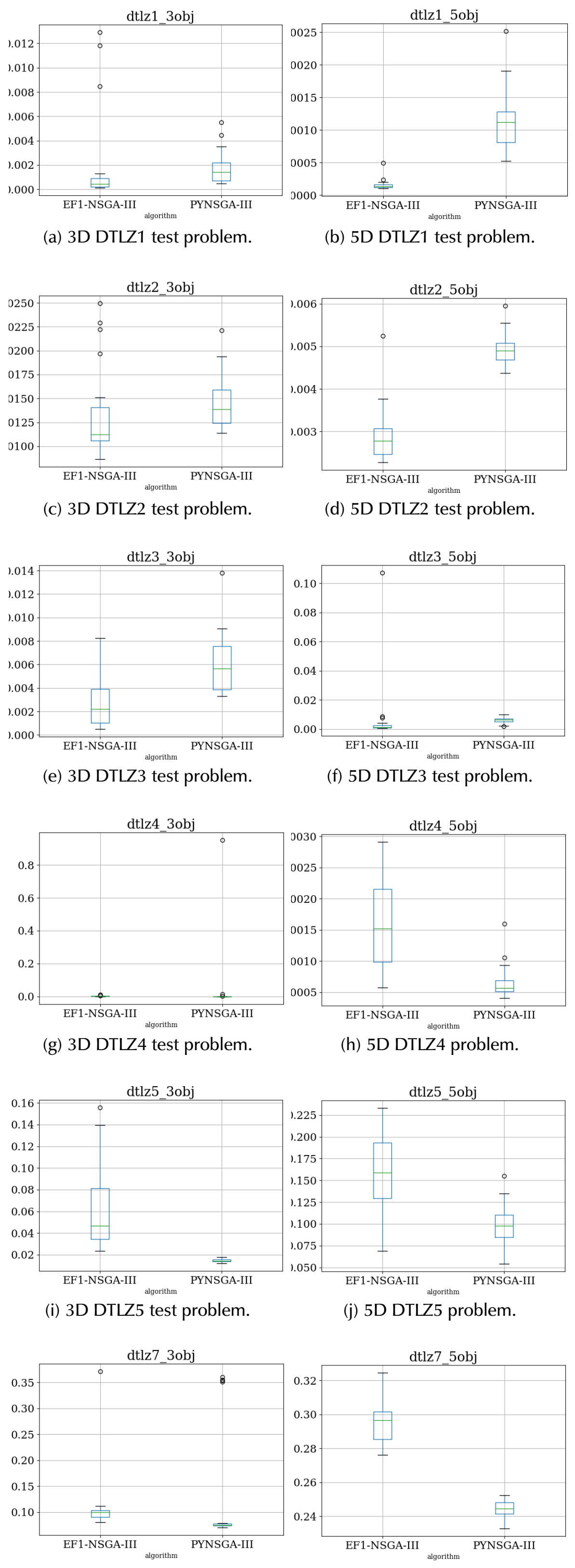
Source: Authors

DTLZ1-5 and DTLZ7 Statistical benchmark

We made a statistical benchmark between EF1-NSGA-III and PYNSGA-III (Pymoo NSGA-III algorithm) in terms of IGD and HyperVolume (HV) performance metrics for DTLZ1-5 and DTLZ7 problems. Their statistical boxplots are shown in Figures 4 and 5 for three-objective and five-objective DTLZ problems running 20 realizations with the same random seeds. First, from an IGD perspective, we see in Figures 4a, 4b, 4c, 4d, 4e, and 4f, that the EF1-NSGA-III performs better than the PYNSGA-III and provides lower IGD values and dispersion. Notwithstanding, Figures 4g, 4i, 4j, 4k, 4l, and 4h depict a different scenario, where the PYNSGA-III performs better than the EF1-NSGA-III.

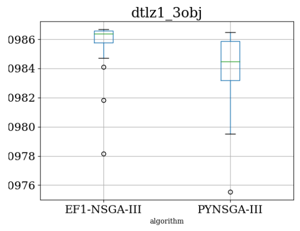
Then, when we analyzed the boxplots of Figure 5 related to the HV performance metric, we notice the same behavior obtained concerning the IGD performance metric. In this scenario, high HV values mean a better performance of convergence and distribution of solutions. That validation helps us be confident of the performance of the EF1-NSGA-III with respect to the PYNSGA-III. For example, Figures 5a, 5b, 5c, 5d, 5e, and 5f, accomplish a better performance in terms of HV for the EF1-NSGA-III rather than the PYNSGA-III. Hovewer, the PYNSGA-III performs better than the EF1-NSGA-III for DTLZ4, DTLZ5 and DTLZ7, as seen in Figures 5g, 5h, 5i, 5j, 5k and 5l.

Exceptionally, the EF1-NSGA-III strives to solve the DTLZ4 test problem for five objectives and above, and those drawbacks are represented with a significant data dispersion, as shown in Figures 4h and 5h. This kind of convergence difficulty occurs for some seeds, so the idea is to increase the number of realizations to reduce this effect.

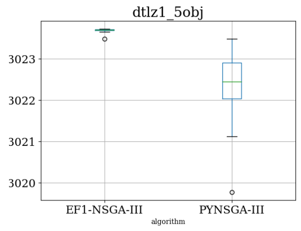


(a) 3D DTLZ1 test problem. (b) 5D DTLZ1 test problem.
(c) 3D DTLZ2 test problem. (d) 5D DTLZ2 test problem.
(e) 3D DTLZ3 test problem. (f) 5D DTLZ3 test problem.
(g) 3D DTLZ4 test problem. (h) 5D DTLZ4 problem.
(i) 3D DTLZ5 test problem. (j) 5D DTLZ5 problem.
(k) 3D DTLZ7 test problem. (k) 5D DTLZ7 problem.

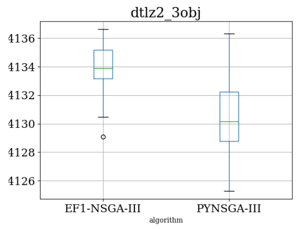
Figure 4. Boxplots in terms of IGD.
Source: Authors



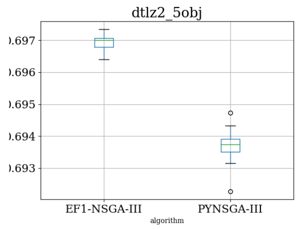
(a) 3D DTLZ1 test problem.



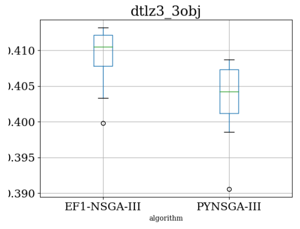
(b) 5D DTLZ1 test problem.



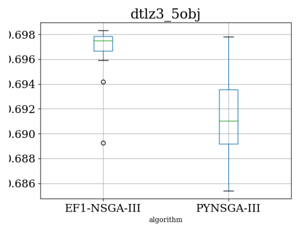
(c) 3D DTLZ2 test problem.



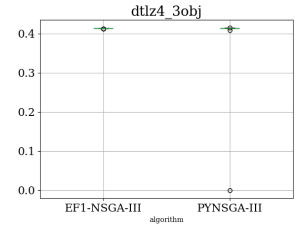
(d) 5D DTLZ2 test problem.



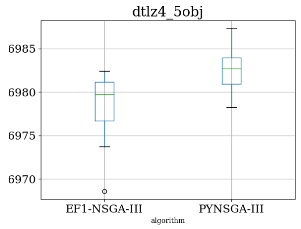
(e) 3D DTLZ3 test problem.



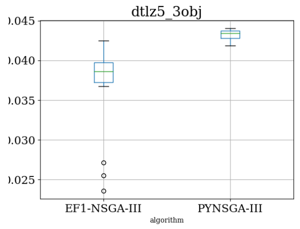
(f) 5D DTLZ3 test problem.



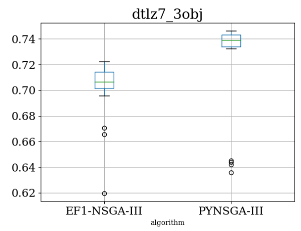
(g) 3D DTLZ4 test problem.



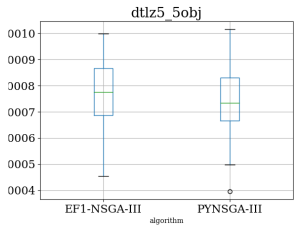
(h) 5D DTLZ4 problem.



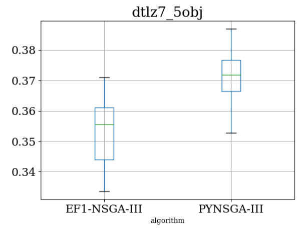
(i) 3D DTLZ5 test problem.



(j) 3D DTLZ7 test problem.



(k) 5D DTLZ5 problem.



(l) 5D DTLZ7 problem.

Figure 5. Boxplots in terms of HV.
Source: Authors

Inverted-DTLZ1 Test Problem

The inverted-DTLZ1 problem is an unconstrained test problem that modifies the DTLZ1 problem. The DTLZ1 transformation requires $f_i(x) = 0{,}5(1 + g(x)) - f_i(x)$, for $i = 1, \ldots, M$ objectives. We can see that the A^2-EF1-NSGA-III distributes the population better than the A-EF1-NSGA-III and the EF1-NSGA-III. The Hyper-volume values are shown in Table 5 and visually confirmed in Figure 2g. We employed the same algorithm from reference (Blank and Deb, 2020) to evaluate the HV. It takes the obtained Pareto front from EF1-NSGA-III and PYNSGA-III and a reference point. We used the point [0,6, 0,6, 0,6] for the three-objective IDTLZ1 problem, the point [39,4, 12,5] for the Car-side impact problem, and the point [75000, 1400, 2900000, 6700000, 26000] for the water problem to obtain HV values. Sometimes, for instance, the Car-side impact problem solved with the EF1-NSGA-III has better HV values than the A^2-EF1-NSGA-III and the A-EF1-NSGA-III. It remains as future work where the number of realizations should be increased.

Table 5. Best, Worst and Median Hyper-volume values obtained for A^2-NSGA-III, A-NSGA-III and NSGA-III

Problem	M	MaxGen	A^2-NSGA-III	A-NSGA-III	NSGA-III
			3,7771	2,6168	**4,4349**
Car-Side Impact (EF1-NSGA-III)	3	500	**4,8367**	4,6992	4,6179
			4,1767	3,9892	**4,5316**
			2,9195x10^{24}	2,7388x10^{24}	2,7656x10^{24}
Water (EF1-NSGA-III)	5	500	**3,0128x10^{24}**	2,895x10^{24}	2,901x10^{24}
			2,967x10^{24}	2,847x10^{24}	2,8444x10^{24}
			5,905x10^{-2}	5,799x10^{-2}	5,782x10^{-2}
IDTLZ1 (EF1-NSGA-III)	3	400	**6,191x10^{-2}**	6,182x10^{-2}	6,189x10^{-2}
			6,032x10^{-2}	6,037x10^{-2}	5,987x10^{-2}
			6,374x10^{-2}	6,094x10^{-2}	6,111x10^{-2}
IDTLZ1(Deb and Jain, 2014)	3	400	**6,621x10^{-2}**	6,54x10^{-2}	6,305x10^{-2}
			6,569x10^{-2}	6,540x10^{-2}	6,229x10^{-2}
			N/A	2,191x10^{-1}	2,063x10^{-1}
IDTLZ1 (Tian et al., 2017)	3	400	N/A	2,127x10^{-1}	1,978x10^{-1}
			N/A	2,153x10^{-1}	2,018x10^{-1}
			0,545	0,540	0,534
Water (Jain and Deb, 2013)	5	500	**0,540**	0,5349	0,528
			0,543	0,5365	0,531

Source: Authors

Normalized constrained DTLZ test problems

We considered three cases of constrained DTLZ test problems. The first is the C1-DTLZ1 problem that uses the constraint $c(x) = 1 - \frac{f_M(x)}{0{,}6} - \sum_{i=1}^{M-1} \frac{f_i(x)}{0{,}5} \geq 0$. The second is the C1-DTLZ3 problem that employs the constraint $c(x) = (\sum_{i=1}^{M} f_i(x)^2 - 16)(\sum_{i=1}^{M} f_i(x)^2 - r^2)) \geq 0$, where $r = \{9, 12{,}5, 12{,}5, 15, 15\}$ is the radius of the hyper-sphere for $M = \{3, 5, 8, 10, 15\}$. Finally, the C2-DTLZ2 problem that utilizes the constraint $c(x) = -\min\{\min_{i=1}^{M}[(f_i(x) - 1)^2 + \sum_{j=1, j\neq i}^{M} f_j^2 - r^2], [\sum_{i=1}^{M}(f_i(x) - \frac{1}{\sqrt{M}})^2 - r^2)]\} \geq 0$, where $r = 0{,}4$ for $M = 3$ and $0{,}5$ otherwise.

Table 6 shows a consistently better performance of the EF1-NSGA-III in terms of IGD for the studied constrained problem, except for C1-DTLZ1 on eight-objective to ten-objective problems, where the EF1-NSGA-III gets stuck. The approximated Pareto-front of C2-DTLZ2 is depicted in Figure 2b that shows a good distribution of solutions.

Table 6. Best, Worst and Median IGD values obtained for NSGA-III. Constrained DTLZ problems

Problem	M	MaxGen	U/H	EF1-NSGA-III	Deb and J, 2014	Tian, 2017
				2,996x10^{-4}	1,229x10^{-3}	2,002x10^{-2}
C1-DTLZ1	3	500	91/91	**1,389x10^{-2}**	2,256x10^{-2}	2,139x10^{-2}
				2,948x10^{-3}	4,932x10^{-3}	2,026x10^{-2}
				1,393x10^{-4}	2,38x10^{-3}	5,092x10^{-2}
C1-DTLZ1	5	600	210/210	**2,215x10^{-3}**	1,024x10^{-2}	5,259x10^{-2}
				1,165x10^{-4}	4,347x10^{-3}	5,203x10^{-2}
				7,010x10^{-1}	**4,843x10^{-3}**	9,006x10^{-2}
C1-DTLZ1	8	800	156/156	7,072x10^{-1}	**4,140x10^{-2}**	9,684x10^{-2}
				7,037x10^{-1}	**1,361x10^{-2}**	9,548x10^{-2}
				7,300x10^{-1}	**3,042x10^{-3}**	1,078x10^{-1}
C1-DTLZ1	10	1 000	275/275	7,3323x10^{-1}	**2,762x10^{-2}**	1,095x10^{-1}
				7,315x10^{-1}	**6,358x10^{-3}**	1,086x10^{-1}
				4,158x10^{-4}	1,581x10^{-3}	4,703x10^{-2}
C2-DTLZ2	3	250	58/91	**1,780x10^{-3}**	6,733x10^{-3}	4,866x10^{-2}
				6,631x10^{-4}	2,578x10^{-3}	4,823x10^{-2}
				6,955x10^{-4}	2,762x10^{-3}	1,382x10^{-1}
C2-DTLZ2	5	350	80/210	**1,0801x10^{-3}**	7,596x10^{-3}	1,397x10^{-1}
				9,173x10^{-4}	3,873x10^{-3}	1,386x10^{-1}
				1,066x10^{-3}	1,404x10^{-2}	2,363x10^{-1}
C2-DTLZ2	8	500	72/156	**2,591x10^{-3}**	8,662x10^{-1}	8,596x10^{-1}
				1,410x10^{-3}	2,352x10^{-2}	2,406x10^{-1}
				4,929x10^{-4}	1,978x10^{-2}	2,648x10^{-1}
C2-DTLZ2	10	750	110/275	**8,234x10^{-4}**	3,491x10^{-2}	5,118x10^{-1}
				5,894x10^{-4}	2,694x10^{-2}	2,716x10^{-1}
				5,189x10^{-4}	8,649x10^{-4}	N/A
C1-DTLZ3	3	1 000	91/91	**8,251x10^{-3}**	(13)	N/A
				2,184x10^{-3}	8,139x10^{-3}	N/A
				2,189x10^{-4}	1,028x10^{-3}	N/A
C1-DTLZ3	5	1 500	210/210	**2,384x10^{-3}**	(15)	N/A
				5,260x10^{-4}	5,101x10^{-2}	N/A
				3,002x10^{-4}	1,656x10^{-3}	N/A
C1-DTLZ3	8	2 500	156/156	**2,257x10^{-4}**	(14)	N/A
				4,538x10^{-4}	0,196x10^{-2}	N/A
				3,347x10^{-4}	2,437x10^{-3}	N/A
C1-DTLZ3	10	3 500	275/275	**7,117x10^{-4}**	(18)	N/A
				3,747x10^{-4}	1,445x10^{-2}	N/A

Source: Authors

Car-Side Impact and Water Problems

The Car-Side Impact problem optimizes the frontal structure of the vehicle for crash-worthiness. It minimizes the weight, the pubic force experienced by a passenger, and the V-Pillar's average velocity to resist the impact load. Its mathematical formulation can be consulted in Jain and Deb (2014). We can see, in Figure 2h, that the A^2-EF1-NSGA-III has a better distribution of solutions compared to the A-EF1-NSGA-III and EF1-NSGA-III. Nevertheless, in Table 5, we notice that the EF1-NSGA-III achieves better HV values than the A^2-EF1-NSGA-III or the A-EF1-NSGA-III. It is necessary to use statistical analysis with more realizations to better understand the distribution of solutions.

Conversely, as shown in Table 5 and Figure 3b, the Water problem has better performance when it is solved with the A^2-EF1-NSGA-III, A-EF1-NSGA-III, and the EF1-NSGA-III, which is expected. This problem has five objective functions, three design variables, and seven constraints. Its mathematical formulation is found in the work by (Jain and Deb, 2014).

Variation of convergence metrics

We used IGD and Υ values in each generation to visualize the convergence. IGD convergence is depicted in Figure 6 and the Υ convergence in Figure 7. The IGD convergence plot is divided into two cases. The first (Figure 6 on the left side) contains better IGD decreasing performance for DTLZ1, DTLZ3, DTLZ4, DTLZ5, and DTLZ7 problems. However, the DTLZ1 is the fastest. In contrast, the second (Figure 6, on the right side) shows bad decreasing IGD performance for IDTLZ1, DTLZ5, and DTLZ7 problems. They will not achieve good performance even though we increase the number of generations. High dimensional DTLZ3 and DTLZ4 problems probably will achieve a good IGD value, but they require significantly more generations. We also divided the Υ convergence plot in two cases. The first (Figure 7, top side) for low dimensional cases, and the second, for high dimensional cases (Figure 7, bottom side). We noticed that this metric is representative for problems up to eight objectives. We also saw that the DTLZ1, DTLZ2, DTLZ3, and DTLZ4 problems exhibit better Υ convergence, but DTLZ3 and DTLZ4 problems require more generations to accomplish similar Υ values. Finally, high dimensional test problems did not provide good Υ values, as shown in Figure 7, bottom part.

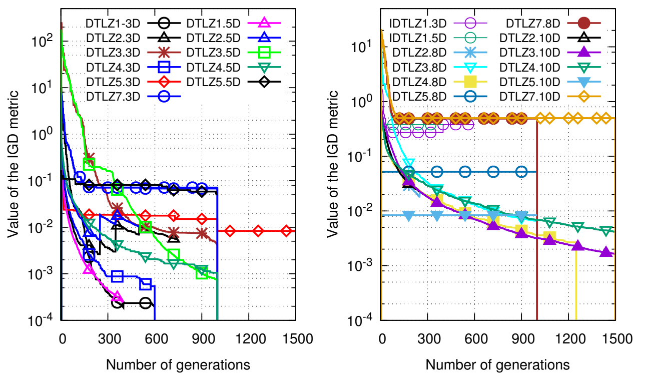


Figure 6. IGD convergence. Seed = 0,5.
Source: Authors

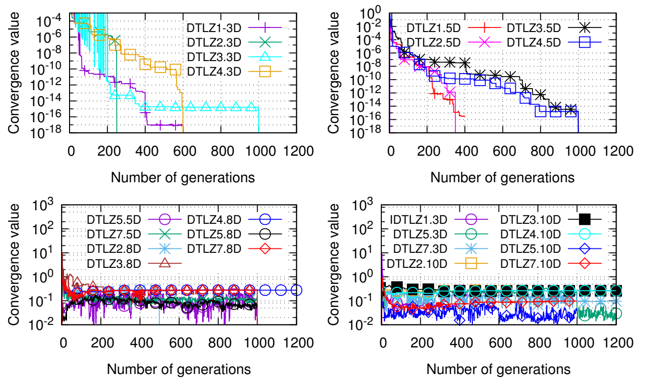


Figure 7. Υ convergence. Seed = 0,5. Source: Authors.
Source: Authors

Conclusions

We successfully extended the NSGA-II algorithm from the KanGAL's Code to the EF1-NSGA-III, the A-EF1-NSGA-III, and the A^2-EF1-NSGA-III. These algorithms have been applied to solve constrained and unconstrained Many-objective optimization problems (up to ten-objective problems). Our proposals have shown their niche in finding a set of well-converged and well-diversified solutions repeatedly over multiple runs. They are susceptible to be employed as a reference for a comparative evaluation of new evolutionary algorithms because they yield consistently better results than those reported in references (Deb and Jain, 2014; Jain and Deb, 2014; Bi and Wang, 2017; Tian et al., 2017; Chiang, 2014; Blank and Deb, 2020).

References

Ariza Vesga, L. F. (2019). *A fast non-dominated sorting genetic algorithm extension to solve many-objective problems.* https://github.com/lfarizav/NSGA-III

Ariza Vesga, L. F. (2020). *Many-objective problems op-timization focused on energy e?ciency applied to 5G heterogeneous cellular networks using the small cell switch-o? framework* (Doctoral thesis, Univer-sidad Nacional de Colombia, Bogotá, Colombia). https://repositorio.unal.edu.co/handle/unal/77582

Bader, J. and Zitzler, E. (2011). HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. *Evolutionary Computation, 19*(1), 45-76. 10.1162/evco_a_00009

Bi, X. and Wang, C. (2017). An improved NSGA-III algorithm based on elimination operator for many-objective optimization. *Memetic Computing, 9*(4), 361-383. 10.1007/s12293-017-0240-7

Blank, J. and Deb, K. (2020). Pymoo: Multi-Objective Optimization in Python. *IEEE Access, 8*, 89497-89509. 10.1109/access.2020.2990567

Blank, J., Deb, K., and Roy, P. C. (2019). Investigating the Normalization Procedure of NSGA-III. In Deb, K., Goodman, E., Coello-Coello, C. A., Klamroth, K., Miettinen, K., Mostaghim, S., and Reed, P. (Eds.) *Evolutionary Multi-Criterion Optimization* (vol. 11411, pp. 229-240). New York, NY: Springer, Cham 10.1007/978-3-030-12598-1_19

Chiang, T.-C. (2014). *nsga3cpp: A c++ implementation of nsga-iii*. https://web.ntnu.edu.tw/~tcchiang/publications/nsga3cpp/nsga3cpp.htm

Das, I., and Dennis, J. E. (1998). Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. *SIAM Journal on Optimization, 8*(3), 631-657. 10.1137/s1052623496307510

Deb, K. (1999). An introduction to genetic algorithms. *Sadhana, 24*, 293315. 10.1007/BF02823145

Deb, K. and Jain, H. (2014). An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation, 18*(4), 577-601. 10.1109/tevc.2013.2281535

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182-197. 10.1109/4235.996017

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2005). Scalable Test Problems for Evolutionary Multiobjective Optimization. In Abraham, A., Jain, L., and Goldberg, R. (Eds.) *Evolutionary Multiobjective Optimization* (pp. 105-145) New York, NY: Springer, Cham. 10.1007/1-84628-137-7_6

Fonseca, C., Paquete, L., and Lopez-Ibanez, M. (2006). *An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator*. Paper presented at the IEEE International Conference on Evolutionary Computation, Vancouver, BC, Canada. 10.1109/cec.2006.1688440

Fraser, A. (1957). Simulation of Genetic Systems by Automatic Digital Computers II. Effects of Linkage on Rates of Advance Under Selection. *Australian Journal of Biological Sciences, 10*(4), 492. 10.1071/bi9570492

Gu, Z. and Wang, G. (2020). Improving NSGA-III algorithms with information feedback models for large-scale manyobjective optimization. *Future Generation Computer Systems, 107*, 49-69. 10.1016/j.future.2020.01.048

Gupta, R., Nanda, S. J. (2019). *Many-Objective B/NSGA-III for Band Selection in Cloud Contaminated Hyper-Spectral Images*. Paper presented at the 2019 International Conference on Information Technology (ICIT), Bhubaneswar, India. 10.1109/icit48102.2019.00068

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. 10.7551/mitpress/1090.001.0001

Huband, S., Hingston, P., Barone, L., and While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation, 10*(5), 477-506. 10.1109/tevc.2005.861417

Ibrahim, A., Rahnamayan, S., Martin, M. V., and Deb, K. (2016). *EliteNSGA-III: An improved evolutionary many-objective optimization algorithm*. Paeper presented at the 2016 IEEE Congress on Evolutionary Computation (CEC). 10.1109/cec.2016.7743895

Ishibuchi, H., Imada, R., Setoguchi, Y., and Nojima, Y. (2016). *Performance comparison of NSGA-II and NSGA- III on various many-objective test problems*. Paper presented at 2016 IEEE Congress on Evolutionary Computation (CEC). 10.1109/cec.2016.7744174

Jain, H. and Deb, K. (2013). An Improved Adaptive Approach for Elitist Nondominated Sorting Genetic Algorithm for Many-Objective Optimization. In Purshouse, R. C., Fleming, P. J., Fonseca, C. M., Greco S., and Shaw, J. (Eds.) *Evolutionary Multi-Criterion Optimization* (vol. 7811, pp. 307-321). New York, NY: Springer, Cham. 10.1007/978-3-642-37140-0_25

Jain, H. and Deb, K. (2014). An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach. *IEEE Transactions on Evolutionary Computation, 18*(4), 602-622. 10.1109/tevc.2013.2281534

Jiang, S. and Yang, S. (2017). A Strength Pareto Evolutionary Algorithm Based on Reference Direction for Multiobjective and Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation, 21*(3), 329-346. 10.1109/tevc.2016.2592479

jMetal (2015). *Metaheuristic algorithms in java*. http://jmetal.sou rceforge.net

KanGAL (2011). *Software Developed at KanGAL*. http://www.iitk .ac.in/kangal/codes.shtml

Li, K., Wang, R., Zhang, T., and Ishibuchi, H. (2018). Evolutionary Many-Objective Optimization: A Comparative Study of the State-of-the-Art. *IEEE Access, 6*, 26194-26214. 10.1109/access.2018.2832181

Li, H. and Zhang, Q. (2009). Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation, 13*(2), 284-302. 10.1109/tevc.2008.925798

Li, M. and Yao, X. (2020). What Weights Work for You? Adapting Weights for Any Pareto Front Shape in Decomposition-Based Evolutionary Multiobjective Optimisation. *Evolutionary Computation, 28*(2), 227-253. 10.1162/evco_a_00269

Li, W., Wang, G., and Alavi, A. H. (2020). Learning-based elephant herding optimization algorithm for solving numerical optimization problems. *Knowledge-Based Systems, 195*, 105675. 10.1016/j.knosys.2020.105675

Li, H., Deb, K., Zhang, Q., Suganthan, P., and Chen, L. (2019). Comparison between MOEA/D and NSGA-III on a set of novel many and multi-objective benchmark problems with challenging di?culties. *Swarm and Evolutionary Computation, 46*, 104-117. 10.1016/j.swevo.2019.02.003

Marti, L. (2016). *An implementation of nsga-iii in python*. https://github.com/lmarti/nsgaiii

Prakash, V. P., Patvardhan, C., and Srivastav, A. (2020). A novel Hybrid Multi-objective Evolutionary Algorithm for the bi-Objective Minimum Diameter-Cost Spanning Tree (bi-MDCST) problem. *Engineering Applications of Artificial Intelligence, 87*, 103237. 10.1016/j.engappai.2019.103237

Purshouse, R. C. and Fleming, P. J. (2007). On the Evolutionary Optimization of Many Conflicting Objectives. *IEEE Transactions on Evolutionary Computation, 11*(6), 770-784. 10.1109/tevc.2007.910138

Redcedartech (2015). *Heeds smashes barriers on multi-objective design studies*. https://redcedartech.com/newsletters/HEE DSNews-Mar15.htm

Seada, H. and Deb, K. (2015). U-NSGA-III: A Unified Evolutionary Optimization Procedure for Single, Multiple, and Many Objectives: Proof-of-Principle Results. In António Gaspar-Cunha, A., Henggeler-Antunes, C., and Coello-Coello, C. (Eds.) *Evolutionary Multi-Criterion Optimization* (vol. 9019, pp. 34-49). New York, NY: Springer, Cham. 10.1007/978-3-319-15892-1_3

Seada, H., Abouhawwash, M., Deb, K. (2017). Towards a Better Balance of Diversity and Convergence in NSGA-III First Results. In Trautmann, H., Rudolph, G., Klamroth, K., Schütze, O., Wiecek, M., Jin, Y., and Grimme C. (Eds.) *Evolutionary Multi-Criterion Optimization* (vol. 10173, pp. 545-559). New York, NY: Springer, Cham. 10.1007/978-3-319-54157-0_37

Seada, H., Abouhawwash, M., and Deb, K. (2018). Multiphase Balance of Diversity and Convergence in Multiobjective Optimization. IEEE *Transactions on Evolutionary Computation, 23*(3), 503-513. 10.1109/tevc.2018.2871362

Shu, Z., Wang, W., and Wang, R. (2018). Design of an Optimized Architecture for Manned and Unmanned Combat System-of-Systems: Formulation and Coevolutionary Optimization. *IEEE Access, 6*, 52725-52740. 10.1109/access.2018.2870969

Surjanovic, S. and Bingham, D. (2013). *Virtual Library of Simulation Experiments: Test Functions and Datasets*. http://www.sfu.ca/ ssurjano

Tian, Y., Cheng, R., Zhang, X., and Jin, Y. (2017). PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization [Educational Forum]. *IEEE Computational Intelligence Magazine, 12*(4), 73-87. 10.1109/mci.2017.2742868

Wangsom, P., Bouvry, P., and Lavangnananda, K. (2018). *Extreme Solutions NSGA-III* (*E-NSGA-III*) *for Scien-tific Workflow Scheduling on Cloud*. Paper presented at the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA. 10.1109/icmla.2018.00184

Wang, G., Guo, L., Gandomi, A. H., Hao, G., and Wang, H. (2014). Chaotic Krill Herd algorithm. Information Sciences, 274, 17-34. 10.1016/j.ins.2014.02.123

Wang, G. and Tan, Y. (2019). Improving Metaheuristic Algorithms with Information Feedback Models. *IEEE Transactions on Cybernetics, 49*(2), 542-555. 10.1109/tcyb.2017.2780274

Wang, H. and Yi, J. (2017). An improved optimization method based on krill herd and artificial bee colony with information exchange. *Memetic Computing, 10*(2), 177-198. 10.1007/s12293-017-0241-6

Wang, G., Deb, S., and Cui, Z. (2019). Monarch butterfly optimization. *Neural Computing and Applications, 31*(7), 1995-2014. 10.1007/s00521-015-1923-y

Wang, G. (2016). Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Computing, 10*(2), 151-164. 10.1007/s12293-016-0212-3

Yarpiz (2018). *Nsga-iii: Non-dominated sorting genetic algorithm, the third version*. http://yarpiz.com/456/ypea126-nsga3

Yi, J., Deb, S., Dong, J., Alavi, A. H., and Wang, G. (2018). An improved NSGA-III algorithm with adaptive mutation operator for Big Data optimization problems. *Future Generation Computer Systems, 88*, 571-585. 10.1016/j.future.2018.06.008

Yi, J., Xing, L., Wang, G., Dong, J., Vasilakos, A. V., Alavi, A. H., and Wang, L. (2020). Behavior of crossover operators in NSGA-III for large-scale optimization problems. *Information Sciences, 509*, 470-487. 10.1016/j.ins.2018.10.005

Yuan, Y., Xu, H., and Wang, B. (2014). An improved NSGA-III procedure for evolutionary many-objective optimization. In Igel, C. (Ed.) *GECCO '14: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation* (pp. 661-668). New York, NY: ACM. 10.1145/2576768.2598342

Zhang, Q., Liu, W., and Li, H. (2009). *The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances*. Paper presented at the 2009 IEEE Congress on Evolutionary Computation. 10.1109/cec.2009.4982949

Zhang, Y., Wang, G., Li, K., Yeh, W., Jian, M., and Dong, J. (2020). Enhancing MOEA/D with information feedback models for large-scale many-objective optimization. *Information Sciences, 522*, 1-16. 10.1016/j.ins.2020.02.066

Zhang, Q. and Li, H. (2007). MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation, 11*(6), 712-731. 10.1109/tevc.2007.892759

Zhang, X., Tian, Y., and Jin, Y. (2015). A Knee Point-Driven Evolutionary Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation, 19*(6), 761-776. 10.1109/tevc.2014.2378512

Zhang, Q., Zhou, A., Zhao S., Suganthan, P. N., Liu, W., and Tiwari S. (2009). *Multiobjective optimization test instances for the cec-2009 special session and competition*. https://www.al-roomi.org/multimedia/CEC_Database/CEC2009/MultiObjectiveEA/CEC2009_MultiObjectiveEA_TechnicalReport.pdf

Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation, 8*(2), 173-195. 10.1162/106365600568202