A Proposal of a Multi-Objective Compact Particle Swarm Optimizer

Jorge Jiménez Montiel
CINVESTAV-IPN
Departamento de Computación
Av. IPN No. 2508
Mexico City, MEXICO 07360
jorgejimenezmontiel123@hotmail.com

Carlos A. Coello Coello
CINVESTAV-IPN
Departamento de Computación
Av. IPN No. 2508
Mexico City, MEXICO 07360
ccoello@cs.cinvestav.mx

Ma. Guadalupe Castillo Tapia
UAM Azcapotzalco
Departamento de Administración
Av. San Pablo No. 180
Mexico City, MEXICO 02200
mgct@azc.uam.mx

Abstract—**Throughout the years, several bio-inspired metaheuristics have been proposed to solve multi-objective problems. Nevertheless, most of the current metaheuristics are not suitable for applications having limited resources (e.g., limited available memory or computationally expensive objective function evaluations). In recent years, a wide variety of metaheuristics have been proposed that employ a statistical representation of the population through a probabilities vector. These are the so-called compact metaheuristics. Several metaheuristics of the state of the art have used a statistical representation to reduce the amount of memory required to be implemented in devices with limited computing resources. This paper presents a compact metaheuristic based on a particle swarm optimizer (PSO) for solving continuous and unconstrained multi-objective optimization problems. Our proposed approach is compared with respect to two multi-objective particle swarm optimizers (MOPSOs) and one compact multi-objective evolutionary algorithm (MOEA). The results indicate that our proposed approach is competitive with respect to the other MOPSOs and is able to outperform the compact MOEA used in our comparative study in most of the test problems adopted.**

Keywords—**Multi-objective optimization, particle swarm optimization, compact metaheuristics**

I. INTRODUCTION

Multi-objective problems occur naturally in a variety of areas. The use of bio-inspired metaheuristics to solve this type of problems is mainly due to their simplicity, generality and ease of use.

The success of particle swarm optimization (PSO) for solving single-objective optimization problems has motivated researchers to extend this metaheuristic for solving multi-objective optimization problems [1]. During the last few years, a wide variety of multi-objective particle swarm optimizers (MOPSOs) have been developed. For example, in 2011, Zapotecas and Coello [2] proposed dMOPSO which uses a decomposition approach similar to that employed by MOEA/D [3] which consists of transforming a multi-objective optimization problem into a set of single-objective problems. Each sub-problem is optimized using only information from neighboring sub-problems. This algorithm uses an archive of leaders that serve as guides during the search process and defines the concept of age, in which a particle is replaced after a certain number of iterations if it is not capable of improving its personal best. dMOPSO is a decomposition-based MOPSO representative of the state-of-the-art in the area. Nebro et al. [4] analyzed five MOPSOs and found out that they were unable to solve some multi-frontal problems satisfactorily because the velocity of the particles in those problems resulted in erratic movements towards the upper and lower limits of the decision space. This caused the particles to oscillate outside the region of interest. Taking as a starting point the MOPSO that obtained the best results in this study (OMOPSO [5]), they developed a new algorithm called SMPSO [6] (Speed-constrained Multi-objective PSO). In order to limit the step size of the velocity, SMOPSO adopts a constriction coefficient and introduces a mechanism to bound the accumulated velocity of each variable. Even today, SMPSO remains as a very competitive Pareto-based MOPSO.

After Harik et al. [7] introduced the compact genetic algorithm (CGA), a few other researchers have proposed variations of this algorithm, mainly oriented towards producing efficient hardware implementations. Gallagher et al. [8] proposed a variant of a compact genetic algorithm to optimize online a reconfigurable analog neural network to control physical processes. Since their control chips interact with physical processes which are orders of magnitude slower than digital signal processing speeds, decreasing the hardware size was crucial. Mininno et al. [9] extended the compact genetic algorithm to work directly with floating-point variables motivated by the fact that most micro-controller platforms are programmable with object-oriented software based on real-value floating-point variables. A compact Particle Swarm Optimizer (cPSO) was proposed by Neri et al. [10] to optimize in real-time an advanced control system of a model of an actual power plant. This cPSO follows the same prinicples of the original compact genetic algorithm: a tournament is held between two individuals to make a slight modification in the probabilities vector. However, unlike the CGA, the cPSO generates only one individual through the probability vector and the remaining one is generated by means of the formulas to update the position of a particle (see equation (1)).

Osorio et al. [11] proposed a compact version of a multi-objective differential evolution algorithm called Multi-Objective Compact Differential Evolution (mocDE). This approach is based on the Compact Differential Evolution proposed by Mininno et al. [12]. To solve multi-objective problems the authors adopted an aproach similar to that employed by MOEA/D, which, as indicated before, transforms a multi-objective optimization problem into a set of single-objective sub-problems. The solutions are stored into a positional external archive that assigns one position for each sub-problem.

In this paper, we introduce a decomposition-based multi-objective compact particle swarm optimizer (MOCPSO) for continuous and unconstrained MOPs. Our proposal adopts a statistical representation of the swarm by means of a real-valued probability vector and it constitutes (to the best of the authors' knowledge) the first proposal of a compact multi-objective particle swarm optimizer.

The remainder of this paper is organized as follows. Section II provides the basic background required to make of this a self-contained paper. Section III describes our proposed Multi-Objective Compact Particle Swarm Optimizer. Our experimental study and a discussion of results are provided in Section IV. Finally, our conclusions and some possible paths for future research are briefly discussed in Section V.

II. BASIC CONCEPTS

A. Particle Swarm Optimization

Particle swarm optimization (PSO), was proposed in 1995 by James Kennedy and Rusell Eberhart [13]. It is a metaheuristic inspired by the social behavior present in different species of animals. The general idea of PSO consists of adopting a population of particles (the *swarm*) that move over the entire search space obeying simple rules that alter both their position and their speed. PSO makes use of the so-called learning factors, which represent the tendency of a particle to follow the success of the best particle in the swarm (gbest) or to follow its own success during the search process (lbest), represented by c_1 and c_2 respectively. Generally, the values of these factors are established at the beginning and the sum of both must not be greater than four [14].

During the execution of the PSO algorithm, each particle affects its flight path by taking into account its own flight experience, and the flight experience of its neighbors. Let $x_i(t)$ denote the position of the i_{th} particle at the point in time t and let $v_i(t)$ denote its velocity. The position of the i_{th} particle at the time instant $t+1$, is defined by:

$$\begin{aligned} v_i(t+1) &:= v_i(t) + c_1 * rand_1 \times (gbest - x_i(t)) \\ &\quad + c_2 * rand_2 \times (lbest_i(t) - x_i(t)) \\ x_i(t+1) &:= x_i(t) + v_i(t+1) \end{aligned} \tag{1}$$

where $rand_1$ and $rand_2$ are uniformly distributed random numbers in the interval $[0,1]$.

B. Compact metaheuristics

In order to derive an equation for the optimal population size, Harik et al. [7] proposed a model in one dimension that relates a genetic algorithm with a random walk. Since there are no interactions between the building blocks for one dimension, the authors considered the possibility of solving each building block independently, giving rise to the compact genetic algorithm [15] (CGA). The CGA represents the population through a probability vector p of size n, starting with $p(i) := 0.5,\ \forall i,\ 1 \leq i \leq n$; Each dimension determines the probability that the i_{th} allele has one as its value. At each iteration, two individuals are generated and compete to determine a winner. Subsequently, the values of the probabilistic vector will be slightly modified so that they assimilate the values of the winner.

The CGA excels at solving continuos unimodal problems. However, it is not capable of producing acceptable solutions when faced with difficult problems (for example, deceptive or multimodal problems) since it does not have a memory to retain the required knowledge about the non-linearity of problems [16]. In order to solve difficult optimization problems without compromising the memory or the computational cost required, Ahn and Ramakrishna [17] proposed the compact genetic algorithm with persistent elitism (pe-cGa). Elitism may increase selection pressure by preventing the loss of genes with low prominence due to poor selection pressure [17]. The way in which elitism is incorporated into the compact genetic algorithm is by keeping the winner as the elite solution and replacing the loser with a new individual at each iteration.

The compact genetic algorithm as well as the other compact metaheuristics that have been proposed so far are mainly aimed to be used in hardware implementations due to their low memory and code requirements.

III. OUR PROPOSED MULTI-OBJECTIVE COMPACT PARTICLE SWARM OPTIMIZER

In this section we introduce our proposed Multi-Objective Compact Particle Swarm Optimizer. We start by describing our single-objective approach and subsequently its extension to solve multi-objective optimization problems. To solve MOPs we adopt a decomposition approach similar to the one proposed by Osorio et al. [11], which transforms a multi-objective optimization problem into a set of p weighted metric sub-problems and solves them simultaneously.

Let $\lambda^1, ..., \lambda^p$ be a set of even spread weight vectors and z^* be the reference point. The MOP can be decomposed into p scalar optimization sub-problems by using the Tchebycheff approach:

$$g^{te}(x|\lambda^*, z^*) = \max_{i=1,\dots,m} \lambda_i |f_i(x) - z_i^*| \tag{2}$$

The non-dominated solutions obtained during the search process are stored in an external archive, where each position of the archive is associated with one of the sub-problems. At the end of each iteration, the archive will be composed with the best solution found so far, for each sub-problem.

A. Compact Particle Swam Optimizer

Now that we have transformed a multi-objective optimization problem into p scalar problems, it is necessary to derive the compact metaheuristic that will be used to solve each one of them. In order to develop our proposal, we take as inspiration both the persitent elitist compact genetic algorithm [17] and the compact particle swarm optimizer [10].

In this case, instead of generating and storing the swarm of particles, we adopt a statistical representation consisting of two vectors μ and σ, both of length n, where the values μ_i and σ_i describe together the distribution of the particles over the search space in the i^{th} dimension, by means of a truncated normal probability distribution function. The observed advantage of using a truncated distribution is that there is greater control when generating solutions, since these will always be valid regardless of the standard deviation used. Burkardt [18] describes the formulas required to generate a truncated random number. Figure 1 shows an example of the distribution of the particles over a two-dimensional search space.

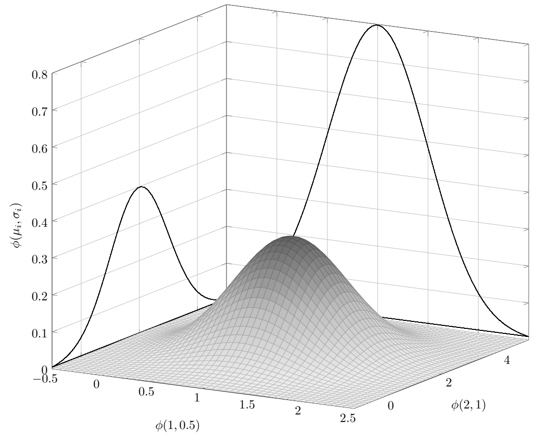


Fig. 1. This figure illustrates a hypothetical cluster that is represented by the probability vector $\mu = [1, 2]$, $\sigma = [0.5, 1]$, where $\phi(\mu_i, \sigma_i)$ is a normal probability distribution function with mean μ_i and standard deviation σ_i

Clearly, the formulas used to update the position and velocity of a particle only make sense when there are at least two particles (i.e., when there is a *swarm*). Having a single particle makes the definitions of $lbest$ and $gbest$ share the same meaning. Thus, it is enough to change the definition of either of these two concepts to solve this problem.

In the present proposal, $gbest$ will take the role of an elitist particle, that is, it will retain the best position found during the entire search process and $lbest$ will be used to move the particle towards new directions. At each iteration, a new $lbest$ is obtained by means of the probability vector, so that in each iteration, the particle will try to move both in the direction of the best solution found and in the direction indicated by the cluster. Note the similarity between our compact PSO and the compact genetic algorithm with persistent elitism. In both, through certain operators, a new solution is obtained that will try to replace the elitist solution.

At the end of each iteration, we move the probability vector towards the direction of the $gbest$, using a step size proportional to the effect that the movement of a particle would have on a swarm of size s. We define the standard deviation as the distance between the mean of the probability vector and $gbest$. This mechanism gives the particle the opportunity to correct its position in case of making a wrong decision. When the mean of the probability vector is relatively close to the $gbest$, the new individuals generated by means of the probability vector will also be close to the $gbest$ and when the distance between the average and the $gbest$ is high, the truncated normal distribution will tend to behave as if it was a uniform distribution.

Algorithm 1 Compact PSO

```
Require: $F : \mathbb{R}^n \to \mathbb{R}$,
  $n$: number of decision variables of the problem,
  $T$: number of iterations,
  $mp$: perturbation probability,
  $s$: hypothetical swarm size,
  $lo_i$: lower bound for the decision variable $i \in n$,
  $up_i$: upper bound for the decision variable $i \in n$,
  $rand$: is a uniformly distributed random number generator
  $randn$: is a normally distributed random number generator
Ensure: $gbest$: the best result found during the search process.
  $c_1 := 2,\ c_2 := 2$
  for $i := 1$ to $n$ do
    $x_i := \text{rand}(lo_i, up_i)$
    $\mu_i := \text{rand}(lo_i, up_i)$
    $\sigma_i := 10 \cdot (up_i - lo_i)$
    $gbest_i := \text{rand}(lo_i, up_i)$
  end for
  if $f(x) < f(gbest)$ then
    swap(x, gbest)
  end if
  while $t < T$ do
    for $i := 1$ to $n$ do
      $lbest_i := \text{randn}(\mu_i, \sigma_i, lo_i, up_i)$
      $r_1 := \text{rand}(0, 1),\ r_2 := \text{rand}(0, 1)$
      $v_i := c_1 * r_1 * (gbest_i - x_i) + c_2 * r_2 * (lbest_i - x_i)$
      {Correct $v_i$ to ensure is between $lo_i, up_i$}
      $x_i := x_i + v_i$
      {Correct $x_i$ to ensure is between $lo_i, up_i$}
    end for
    if $f(x) < f(gbest)$ then
      $gbest := x$
    end if
    for $i := 1$ to $n$ do
      $\mu_i := \mu_i + \frac{1}{s}(gbest_i - \mu_i)$
      $\sigma_i := |gbest_i - \mu_i|$
      if $\text{rand}(0, 1) \leq mp$ then
        $\sigma_i := 10 \cdot (up_i - lo_i)$
      end if
    end for
    $t := t + 1$
  end while
```

1) The Main Structure of Our Proposed Algorithm: At the beginning (Algorithm 1), the single particle of the swarm and

$gbest$ are randomly generated. The probability vector is also initialized so that it represents a uniform distribution. At each iteration, a new individual $lbest$ will be obtained by means of the probability vector. This new individual, along with $gbest$, are used to calculate the new velocity of the particle and, subsequently, affect its position. The new solution is compared with respect to $gbest$ and will replace it in case its evaluation represents an improvement. Next, we move the probability vector towards the $gbest$ position in a step proportional to the size of the swarm. Finally, we apply a turbulence mechanism, which works in a similar way to the uniform mutation in a genetic algorithm. If a random uniform number between zero and one is less than or equal to the perturbation probability, the standard deviation of the i^{th} particle is restored. In each iteration, only one evaluation of the objective function is performed. Therefore, the total number of evaluations of the objective function of our proposal is T + 2, since two initial evaluations are required to determine the gbest at the beginning of the algorithm.

B. Multi-Objective Compact Particle Swam Optimizer

Since the compact PSO uses a statistical representation of the swarm, it is not possible to directly assign a sub-problem to each particle and it is not possible to store the non-dominated solutions obtained during the search process.

1) Archive of non-dominated solutions: In order to store the non-dominated solutions, we use an external archive similar to the one proposed by Osorio et al. [11] that assigns to each position of the archive a weighting vector which represents a Tchebycheff weighted sub-problem.

At the end of the iterative process, the solutions stored in the external archive will end up conforming our Aproximation of the Pareto Optimal Set. Algorithm 2 shows how the solutions found during the search process are stored in the archive. The weighting vectors are generated using the algorithm described in [3].

When the external archive is empty, the candidate solution becomes the best solution for all the Tchebycheff weighted sub-problems. Otherwise, the candidate solution will be stored at position i in the external archive if it solves the Tchebycheff weighted sub-problem i better than the stored solution, and is not dominated by it. If the candidate solution was stored in the archive, the algorithm will return the index of the sub-problem stored in the archive with the minimum value (Algorithm 3) to be used as the next swarm leader. In other words, it will return the position of the solution that is closest to the ideal objective vector. If not, then a random index is returned, implying that any solution stored could be the next leader of the swarm.

C. Main Framework

Like its single-objective counterpart, the compactMOPSO (see Algorithm 4) starts by generating two initial solutions through a uniform random distribution. The solutions will compete to determine the initial leader of the swarm by means of Pareto dominace. Because the external archive is initially empty, the swarm leader will be stored in all the positions, since it represents the best solution found so far for each sub-problem.

At each iteration, a new solution is obtained by means of the probability vector ($lbest$) and together with $gbest$ they will determine the new velocity of the particle. Subsequently, the particle will be moved using the new calculated speed. This particle will be a potential solution to be stored in the external archive if it dominates $gbest$ or if both are incomparable to each other.

For the multi-objective approach, we set the size of the hypothetical swarm to one. This is due to the fact that we observed a significant reduction in the number of function evaluations and a remarkable improvement in the speed of convergence as the size of the hypothetical swarm was reduced. Equation (3) shows the updated formula for calculating the mean of the probability vector. While we were carrying out the performance tests of our proposal with the test problems, we observed that the values from 1.21 to 1.29 for both c_1 and c_2 presented the best convergence rates, especially in multifrontal problems (ZDT4). For our tests we opted for 1.29, but any value in that interval presents similar results. And finally, we set the standard deviation of the probability vector as the distance between the position of the particle and the leader of the swarm.

$$\begin{aligned} \mu_i &:= \mu_i + \frac{1}{s}(gbest_i - \mu_i) \\ &:= \mu_i + \frac{1}{1}(gbest_i - \mu_i) \\ &:= gbest_i \end{aligned} \tag{3}$$

IV. EXPERIMENTAL RESULTS

In order to assess the performance of our proposal, we compared its results with respect to those obtained by two state-of-the-art MOPSOs: SMPSO [6] and dMOPSO [2], and with respect to the only compact multi-objective evolutionary algorithm that we are aware of: mocDE [11].

A. Test problems

We adopted several benchmarks. The first is the so-called Ziztler-Deb-Thiele (ZDT) [19] test suite. The remaining ones are some of the standard test problems described in [20]. We adopted $n = 30$ for solving the ZDT problems and $n = 2$ for the rest of the problems, were n indicates the number of decision variables. Specifically, the following test problems were used:

- ZDT1, ZDT2, ZDT3, ZDT4, ZDT6
- Bihn
- Deb1, Deb2, Deb3
- Fonseca1, Fonseca2
- Laumanns
- Lis
- Murata

Algorithm 2 External Archive

```
Require: $F : \mathbb{R}^n \to \mathbb{R}^m$;
    $x \in \mathbb{R}^n$: candidate solution to be stored,
    $z$: ideal objective vector,
    $p$: number of sub-problems,
    $m$: number of objectives,
    $n$: number of decision variables,
    $\lambda \in \mathbb{R}^{p \times m}$: weighting vectors set,
    $A \in \mathbb{R}^{p \times m}$: external archive,
Ensure: positive integer that indicates the position of the next
    leader to be used in A
  for $i := 1$ to $m$ do
    if $f_i(x) < z_i$ then
      {Update the ideal objective vector}
      $z_i := f_i(x)$
    end if
  end for
  {Add the candidate solution to the archive}
  if $A = \emptyset$ then
    for $i := 1$ to $p$ do
      $A_i := x$
    end for
    return 1
  else
    for $i := 1$ to $p$ do
      $f_1 := \max_{1 \leq j \leq m} \lambda_{i,j} |f_j(x) - z_j|$
      $f_2 := \max_{1 \leq j \leq m} \lambda_{i,j} |f_j(A_i) - z_j|$
      if $f_1 < f_2 \wedge f(A_i) \npreceq f(x)$ then
        $A_i := x$
      end if
    end for
  end if
  if the candidate solution was added at least in one position
  of A then
    return index of the next leader, Algorithm 3
  else
    return a random index from A
  end if
```

Parameters	MOCPSO	SMPSO	dMOPSO	mocDE
N_{pop}	1	100	100	1
N_{gen}	$20,000$	200	200	$20,000$
$Size\ A$	100	100	100	100
W	-	0.1	$U(0.1, 0.5)$	-
c_1	1.29	$U(1.5, 2.5)$	$U(1.2, 2.0)$	-
c_2	1.29	$U(1.5, 2.5)$	$U(1.2, 2.0)$	-
T_A	-	-	2	-
F	-	-	-	1
P_c	-	-	-	0.1
mp	0.01	$1/n$	-	-

TABLE I. PARAMETERS USED IN THE EXPERIMENTS FOR EACH ALGORITHM

Algorithm 3 Get the index of the next swarm leader

```
Require: $F : \mathbb{R}^n \to \mathbb{R}^m$;
    $z$: ideal objective vector,
    $m$: number of objectives
    $p$: number of sub-problems,
    $\lambda \in \mathbb{R}^{p \times m}$: weighting vectors set,
    $A \in \mathbb{R}^{p \times m}$: external archive,
Ensure: index of the next swarm leader
  $leader := 0$
  for $i := 1$ to $p$ do
    $f_1 := \max_{1 \leq j \leq m} \lambda_{i,j} |f_j(x) - z_j|$
    $f_2 := \max_{1 \leq j \leq m} \lambda_{i,j} |f_j(A_{leader}) - z_j|$
    if $f_1 < f_2$ then
      $leader := i$
    end if
  end for
  return $leader$
```

Test problem	Reference point
Binh	50.1 50.1
Deb1	1.1 1.1
Deb2	0.9 1.1
Deb3	1.1 1.1
Fonseca1	1.1 1.1
Fonseca2	1.1 1.1
Laumanns	4.1 4.1
Lis	0.92 0.84
Murata	4.1 4.1
ZDT1	1.1 1.1
ZDT2	1.1 1.1
ZDT3	0.9 1.1
ZDT4	1.1 1.1
ZDT6	1.1 1.1

TABLE II. REFERENCE POINTS USED TO CALCULATE THE HYPERVOLUME

B. Experimental settings

The results reported here were obtained by performing 30 independent executions of each of the algorithms, for each test problem. The number of function evaluations was limited to 20,000. The parameters used in each algorithm are summarized in Table I, where N_{pop} represents the population size. N_{gen} represents the number of generations. $Size\ A$ is the size of the external archive used, W represents the inertia weigth used in SMPSO and dMOPSO, c_1, c_2 are the learning factors from PSO, T_a represents the age threshold from dMOPSO, F is the differential variation and p_c is the crossover probability from mocDE. Finally, m_p is the mutation probability used by MOCPSO and SMPSO.

In order to compare our results, three performance indicators were adopted: the hypervolume indicator (HV) [21], the spacing indicator (S) [22], and the inverted generational distance plus (IGD+) [23]. The results are presented in Tables III and IV. The first table presents the results when comparing our proposal with respect to state-of-the-art MOPSOs, and the second table presents the results when compared with respect to the Multi-Objective Compact Differential Evolution algorithm. These tables show the mean and standard deviation (in parentheses) of the metaheuristics in each of the test

Algorithm 4 Compact MOPSO

```
Require: $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $\lambda \in \mathbb{R}^{p\times m}$, $A \in \mathbb{R}^{p\times m}$,$n$, $m$, $z$, $T$, $mp$, $lo_i$, $up_i$
Ensure: $A$: Pareto aproximated optimal set
  $z_i := \infty,\ \forall i \in [1, m]$ {Initialize the ideal objective vector}
  $c_1 := 1.29,\ c_2 := 1.29$
  for $i := 1$ to $n$ do
    $x_i :=$ rand$(lo_i, up_i)$,
    $\mu_i :=$ rand$(lo_i, up_i)$,
    $\sigma_i := 10 \cdot (up_i - lo_i)$,
    $gbest_i :=$ rand$(lo_i, up_i)$
  end for
  if $f(x) \prec f(gbest)$ then
    swap(x, gbest)
  end if
  store $gbest$ into $A$
  while $t < T$ do
    for $i := 1$ to $n$ do
      $lbest_i :=$ randn$(\mu_i, \sigma_i, lo_i, up_i)$
      $r_1 :=$ rand$(0, 1)$, $r_2 :=$ rand$(0, 1)$
      $v_i := c_1 * r_1 * (gbest_i - x_i) + c_2 * r_2 * (lbest_i - x_i)$
      {Correct $v_i$ to ensure is between $lo_i, up_i$}
      $x_i := x_i + v_i$
      {Correct $x_i$ to ensure is between $lo_i, up_i$}
    end for
    if $f(x) \prec f(gbest)$ then
      Store $x$ into $A$
      if $rand() < 0.5$ then
        $gbest :=$ new leader from $A$ {Algorithm 3}
      end if
    else if $f(gbest) \not\prec f(x)$ and $f(x) \not\prec f(gbest)$ then
      Store $x$ into $A$
      $gbest :=$ new leader from $A$ {Algorithm 3}
    else if $f(gbest) \prec f(x)$ then
      $gbest :=$ new leader from $A$ {Algorithm 3}
    end if
    for $i := 1$ to $n$ do
      $\mu_i := gbest_i,\ \sigma_i := |gbest_i - x_i|$
      if rand$(0, 1) \leq mp$ then
        $\sigma_i := 10 \cdot (up_i - lo_i)$
      end if
    end for
    $t := t + 1$
  end while
```

problems adopted for each performance indicator. Each row of the table represents a test problem and each column represents a metaheuristic. A grayscale was used to indicate the best results in such a way that the best result is highlighted with the darkest tone. The absence of color indicates that the comparison of results was not statistically significant using the Wilconxon rank sum test and therefore it was not possible to establish a conclusion in that case. The reference points used to calculate the hypervolume indicator are shown in Table II.

C. Discussion of Results

1) Hypervolume indicator results: The hypervolume indicator provides the volume of the portion of the objective space that is dominated by an approximation of the Pareto front, so it can be assumed that the larger the value obtained, the better the performance of a metaheuristic. With respect to HV, the best performer was SMPSO, which obtained the best result in eight of the test problems adopted, followed by our proposal that obtained the best results in five problems, four of which come from the ZDT set (ZDT1-ZDT4), so it can be said that our proposal had a very good performance in this set of problems. With respect to mocDE, our proposal obtained the best results on nine occasions as shown in Table IVa, while mocDE obtained the best results in four occasions. The fact that our proposal obtained the best result in the ZDT4 problem indicates that it is suitable for multi-frontal problems. This is something worth noting, considering that our proposal is a MOPSO that has no actual population and highlighting the fact that mocDE did not achieve convergence in this problem after performing 20,000 evaluations.

2) Spacing indicator results: Table IIIb shows the results obtained with respect to the spacing indicator. The spacing indicator allows knowing the degree of uniformity in the distribution of the solutions, so that the lower the value, the better the solutions are distributed. This indicator makes clear the advantage offered by the use of the crowding distance for the selection of leaders adopted by SMPSO, which obtains the first place in thirteen of the fourteen test problems, being surpassed only in the problem ZDT6 by our proposal. In the second position, it is our proposal, which, in spite of using the same decomposition approach, outperformed the results obtained by dMOPSO and mocDE in most of the problems. It is worth noting that both mocDE and dMOPSO as well as MOCPSO had poor results for the Binh test problem with respect to the spacing indicator.

3) IGD+ indicator results: The IGD+ indicator lets us know how close an approximate Pareto set is to a reference set. By using this indicator, a closer competition was observed. Although SMPSO obtained the best results in eight problems, it obtained the worst results for almost all the ZDT problems (taking into account only the results of the MOPSOs). It is with respect to this indicator that our proposal obtained its best results, since it obtained on five occasions the best results and on six the second best results.

4) Summary: From this study, we conclude that our proposal presents a competitive performance for solving optimization problems with two objectives, obtaining the second place in the category of MOPSOs being outperformed only by SMPSO and generating better results than mocDE for the test problems adopted. It was observed that our proposal has a good performance in Pareto fronts with both concave and convex geometry, since it satisfactorily solved the ZDT and Deb test suites. However, it was observed that in problems presenting a disconnected front (Laumanns, ZDT3 and Deb2), our proposal had a poor distribution of solutions, since these are concentrated on a specific region. This could also be

HV	MOCPSO	SMPSO	dMOPSO
binh	2080.345660 (0.086067)	2084.745125 (0.089551)	2080.262541 (0.081066)
deb1	0.538489 (0.000001)	0.538768 (0.000017)	0.538445 (0.000184)
deb2	0.984969 (0.000006)	0.985986 (0.000020)	0.984292 (0.000008)
deb3	0.280226 (0.000001)	0.280884 (0.000046)	0.280243 (0.000001)
fonseca1	0.272069 (0.000013)	0.272958 (0.000027)	0.272100 (0.000010)
fonseca2	0.547391 (0.000020)	0.547541 (0.000035)	0.547425 (0.000013)
laumanns	14.061445 (0.000203)	14.086216 (0.000943)	14.059843 (0.000217)
lis	0.523729 (0.002113)	0.483804 (0.005022)	0.527092 (0.001687)
murata	3.146369 (0.000045)	3.146362 (0.000069)	3.144908 (0.000024)
zdt1	0.871574 (0.000005)	0.871630 (0.000173)	0.871275 (0.000144)
zdt2	0.538489 (0.000001)	0.538454 (0.000174)	0.538294 (0.000047)
zdt3	1.327975 (0.000007)	1.326108 (0.068996)	1.324509 (0.000546)
zdt4	0.870843 (0.001259)	0.870945 (0.000393)	0.870069 (0.000851)
zdt6	0.504572 (0.000001)	0.515904 (0.139041)	0.504174 (0.000673)

(a)

SP	MOCPSO	SMPSO	dMOPSO
binh	1.255077 (0.008260)	0.068229 (0.011203)	1.289514 (0.013618)
deb1	0.004213 (0.000010)	0.001194 (0.000205)	0.004782 (0.002989)
deb2	0.015639 (0.000092)	0.003179 (0.000351)	0.017584 (0.000058)
deb3	0.006879 (0.000016)	0.001706 (0.000689)	0.006890 (0.000011)
fonseca1	0.007712 (0.000034)	0.001337 (0.000293)	0.007833 (0.000073)
fonseca2	0.003950 (0.000041)	0.001329 (0.000195)	0.004002 (0.000033)
laumanns	0.100494 (0.000275)	0.007582 (0.001094)	0.103513 (0.000415)
lis	0.007025 (0.000721)	0.002778 (0.000945)	0.009053 (0.001273)
murata	0.005204 (0.000408)	0.003094 (0.000520)	0.011381 (0.000099)
zdt1	0.010045 (0.000040)	0.001634 (0.000266)	0.011060 (0.001288)
zdt2	0.004221 (0.000022)	0.001485 (0.000253)	0.004313 (0.000258)
zdt3	0.016925 (0.000057)	0.006011 (0.003096)	0.023636 (0.000868)
zdt4	0.010106 (0.000567)	0.002238 (0.000365)	0.009938 (0.000181)
zdt6	0.002057 (0.000020)	0.021369 (0.015597)	0.053270 (0.142027)

(b)

IGD+	MOCPSO	SMPSO	dMOPSO
binh	0.132765 (0.001583)	0.082647 (0.015560)	0.133024 (0.001364)
deb1	0.002141 (0.000008)	0.002275 (0.000266)	0.002151 (0.000020)
deb2	0.001350 (0.000017)	0.000666 (0.000100)	0.001580 (0.000013)
deb3	0.001381 (0.000005)	0.001099 (0.000123)	0.001518 (0.000004)
fonseca1	0.001413 (0.000014)	0.000998 (0.000213)	0.001401 (0.000009)
fonseca2	0.002388 (0.000017)	0.002180 (0.000297)	0.002379 (0.000012)
laumanns	0.010461 (0.000033)	0.007024 (0.000789)	0.010488 (0.000053)
lis	0.002238 (0.000091)	0.001408 (0.000102)	0.002213 (0.000087)
murata	0.006220 (0.000086)	0.005706 (0.001028)	0.005985 (0.000046)
zdt1	0.002350 (0.000008)	0.002539 (0.000261)	0.002482 (0.000050)
zdt2	0.000057 (0.000007)	0.002344 (0.000181)	0.000241 (0.000112)
zdt3	0.001770 (0.000014)	0.005537 (0.015443)	0.003261 (0.000107)
zdt4	0.002681 (0.000663)	0.002827 (0.000333)	0.003049 (0.000435)
zdt6	0.001961 (0.000010)	0.010784 (0.035322)	0.001921 (0.000037)

(c)

TABLE III. COMPARISON OF RESULTS WITH RESPECT TO OTHER MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZERS

HV	MOCPSO	MOCDE
binh	2080.345660 (0.086067)	2070.566014 (1.545972)
deb1	0.538489 (0.000001)	0.538050 (0.000039)
deb2	0.984969 (0.000006)	1.035871 (0.160952)
deb3	0.280226 (0.000001)	0.282617 (0.014601)
fonseca1	0.272069 (0.000013)	0.278440 (0.007371)
fonseca2	0.547391 (0.000020)	0.541182 (0.003453)
laumanns	14.061445 (0.000203)	12.939475 (0.482121)
lis	0.523729 (0.002113)	0.419784 (0.059171)
murata	3.146369 (0.000045)	3.140816 (0.000251)
zdt1	0.871574 (0.000005)	0.867511 (0.001148)
zdt2	0.538489 (0.000001)	0.538207 (0.002395)
zdt3	1.327975 (0.000007)	2.005953 (0.050731)
zdt4	0.870843 (0.001259)	0.000000 (0.000000)
zdt6	0.504572 (0.000001)	0.464850 (0.017585)

(a)

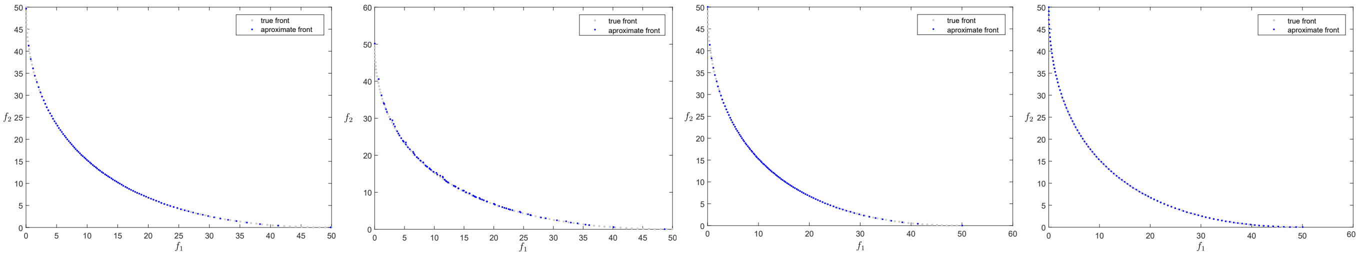
SP	MOCPSO	MOCDE
binh	1.255077 (0.008260)	1.552369 (0.119910)
deb1	0.004213 (0.000010)	0.005329 (0.000218)
deb2	0.015639 (0.000092)	0.025052 (0.017504)
deb3	0.006879 (0.000016)	0.008425 (0.003782)
fonseca1	0.007712 (0.000034)	0.012917 (0.002963)
fonseca2	0.003950 (0.000041)	0.010921 (0.001545)
laumanns	0.100494 (0.000275)	0.070851 (0.079104)
lis	0.007025 (0.000721)	0.007644 (0.009618)
murata	0.005204 (0.000408)	0.015474 (0.000545)
zdt1	0.010045 (0.000040)	0.014918 (0.001834)
zdt2	0.004221 (0.000022)	0.011008 (0.001633)
zdt3	0.016925 (0.000057)	0.025262 (0.001973)
zdt4	0.010106 (0.000567)	0.396595 (0.740801)
zdt6	0.002057 (0.000020)	0.060257 (0.101237)

(b)

IGD+	MOCPSO	MOCDE
binh	0.132765 (0.001583)	0.233960 (0.014480)
deb1	0.002141 (0.000008)	0.002421 (0.000054)
deb2	0.001350 (0.000017)	0.002057 (0.000618)
deb3	0.001381 (0.000005)	0.001598 (0.000216)
fonseca1	0.001413 (0.000014)	0.002112 (0.000514)
fonseca2	0.002388 (0.000017)	0.006499 (0.001147)
laumanns	0.010461 (0.000033)	0.153165 (0.064276)
lis	0.002238 (0.000091)	0.021887 (0.008995)
murata	0.006220 (0.000086)	0.006702 (0.000113)
zdt1	0.002350 (0.000008)	0.005026 (0.000500)
zdt2	0.000057 (0.000007)	0.005149 (0.000737)
zdt3	0.001770 (0.000014)	0.001050 (0.000268)
zdt4	0.002681 (0.000663)	89.278937 (11.044041)
zdt6	0.001961 (0.000010)	0.026168 (0.011180)

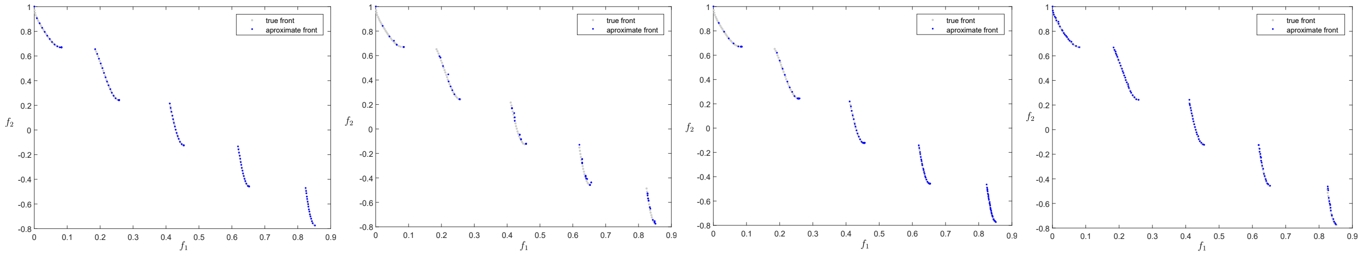
(c)

TABLE IV. COMPARISON OF RESULTS WITH RESPECT TO ANOTHER COMPACT METAHEURISTIC



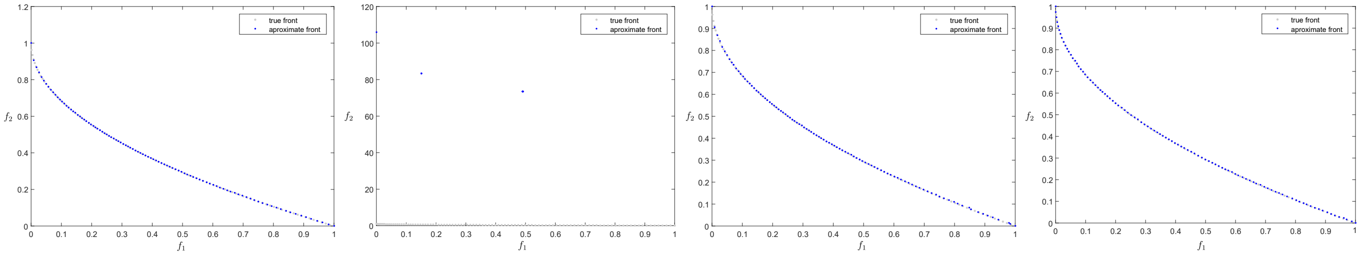
(a) MOCPSO (b) MOCDE (c) DMOPSO (d) SMPSO

Fig. 2. Aproximated Pareto Front for the Binh test problem



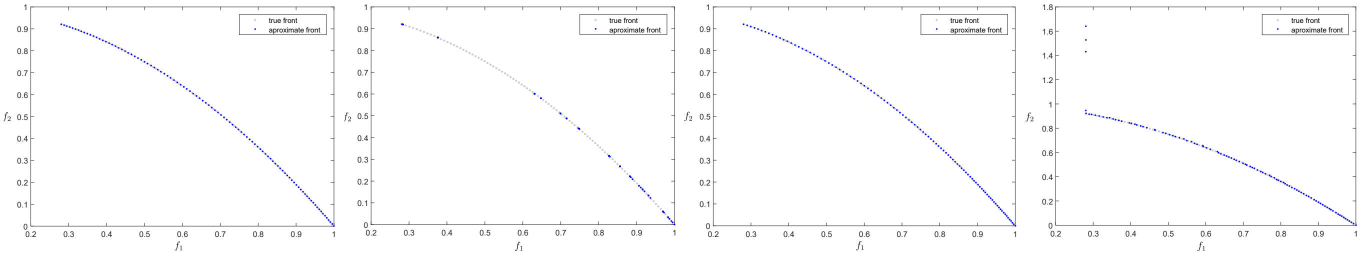
(a) MOCPSO (b) MOCDE (c) DMOPSO (d) SMPSO

Fig. 3. Aproximated Pareto Front for ZDT3



(a) MOCPSO (b) MOCDE (c) DMOPSO (d) SMPSO

Fig. 4. Aproximated Pareto Front for ZDT4



(a) MOCPSO (b) MOCDE (c) DMOPSO (d) SMPSO

Fig. 5. Approximated Pareto Front for ZDT6 problem

observed in dMOPSO and mocDE, so it can be inferred that the decomposition approach does not have a good performance in this sort of problems. Our proposal obtained very good results in the ZDT4 and ZDT6 problems, which suggests that it may be suitable for multifrontal problems.

Although the decomposition approach is able to solve multi-objective optimization problems with more than two objectives [3], our proposal was not able to satisfactorily solve the DTLZ problem set [24] using more than three decision variables. With the exception of problem DTLZ6 (where our proposal was able to converge using eleven decision variables in twenty thousand evaluations), we observed that both diversity and convergence of the solutions decreased as the number of decision variables increased, being impossible for our proposal to fully cover the surfaces that represent the Pareto front for such problems. This is the reason why it was decided to use problems having only two objectives in our experimental study.

V. CONCLUSIONS AND FUTURE WORK

In this work, a compact PSO was proposed for multi-objective optimization problems, which we called MOCPSO. A decomposition approach was adopted whereby a multi-objective problem is transformed into a set of weighted metric sub-problems. To solve these sub-problems, a new compact PSO was proposed whose design is inspired by the compact genetic algorithm with persistent elitism.

By using a statistical representation of the population, we eliminate the need to store the swarm of particles, so unlike other MOEAs, our proposal only requires an external archive to store the non-dominated solutions found during the search process. The experimental results showed that MOCPSO is able to solve complex problems with moderate dimensionality considering that it satisfactorily solved the set of ZDT test problems using thirty decision variables. Our proposal obtained the best results in the set of ZDT problems. Although SMPSO obtained the best results from the experimental study, it was observed that compactMOPSO is very competitive with respect to it, and it was observed that in several occasions it was superior to mocDE and dMOPSO for the set of test problems used.

There are several aspects of our proposal that can be improved:

- **Study of mechanisms to improve the exploration of the search space:** It was observed that the use of persistent elitism compromised the search process, since it always tries to generate solutions close to the leader and, therefore, prevents the exploration of other regions of the search space.
- **Change the selection of the leader by more sophisticated mechanisms:** As observed in the results, the decomposition approach did not obtain a good distribution in disconnected fronts. This point is of great relevance, given that the leader is the main responsible for guiding the only particle of the swarm towards the best solutions.

A final point to note, and that was not mentioned throughout the article, is that although we used 20,000 function evaluations for each problem adopted in the expertimetal results, we noted that our proposal had a very high convergence speed, obtaining very good results from 2,000 function evaluations on most of the test problems adopted (with the exception of ZDT4 that requires at least 15,000 evaluations). A study on the effects of the parameters of our proposal and a comparison of the results obtained with respect to different multi-objective metaheuristics by reducing the number of required function evaluations will be part of our future work.

ACKNOWLEDGMENTS

The first author acknowledges support fom CONACyT through a scholarship to pursue graduate studies in computer science at CINVESTAV-IPN. The second author gratefully acknowledges financial support from the 2018 SEP-Cinvestav Fund (application no. 4) and from CONACYT grant no. 2016-01-1920 (*Investigación en Fronteras de la Ciencias 2016*). He is currently on Sabbatical leave at the Departamento de Sistemas of the Universidad Autónoma Metropolitana Unidad Azcapotzalco (UAM-Azcapotzalco).

REFERENCES

[1] M. Reyes-Sierra and C. A. Coello Coello, “Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art,” *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 287–308, 2006.

[2] S. Zapotecas Martínez and C. A. Coello Coello, “A Multi-objective Particle Swarm Optimizer Based on Decomposition,” in *2011 Genetic and Evolutionary Computation Conference (GECCO’2011)*. Dublin, Ireland: ACM Press, July 12-16 2011, pp. 69–76.

[3] Q. Zhang and H. Li, “MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, December 2007.

[4] J. J. Durillo, J. García-Nieto, A. J. Nebro, C. A. Coello Coello, F. Luna, and E. Alba, “Multi-Objective Particle Swarm Optimizers: An Experimental Comparison,” in *Evolutionary Multi-Criterion Optimization. 5th International Conference, EMO 2009*. Nantes, France: Springer. Lecture Notes in Computer Science Vol. 5467, April 2009, pp. 495–509.

[5] M. Reyes Sierra and C. A. Coello Coello, “Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance,” in *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*. Guanajuato, México: Springer. Lecture Notes in Computer Science Vol. 3410, March 2005, pp. 505–519.

[6] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. A. Coello Coello, F. Luna, and E. Alba, “SMPSO: A New PSO-based Metaheuristic for Multi-objective Optimization,” in *2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM’2009)*. Nashville, TN, USA: IEEE Press, March 30 - April 2 2009, pp. 66–73, ISBN 978-1-4244-2764-2.

[7] G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller, “The gambler’s ruin problem, genetic algorithms, and the sizing of populations,” *Evolutionary Computation*, vol. 7, no. 3, pp. 231–253, Sept 1999.

[8] J. C. Gallagher, S. Vigraham, and G. Kramer, “A family of compact genetic algorithms for intrinsic evolvable hardware,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 111–126, April 2004.

[9] E. Mininno, F. Cupertino, and D. Naso, “Real-valued compact genetic algorithms for embedded microcontroller optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 203–219, April 2008.

[10] F. Neri, E. Mininno, and G. Iacca, “Compact particle swarm optimization,” *Information Sciences*, vol. 239, pp. 96–121, August 2013.

[11] J. M. Osorio Velázquez, C. A. Coello Coello, and A. Arias-Montaño, “Multi-objective compact differential evolution,” in *2014 IEEE Symposium on Differential Evolution (SDE’2014)*, Orlando, Florida, USA, 9-12 December 2014, pp. 49–56, ISBN 978-1-4799-4462-0.

[12] E. Mininno, F. Neri, F. Cupertino, and D. Naso, “Compact differential evolution,” *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 32–54, Feb 2011.

[13] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95 - International Conference on Neural Networks*, vol. 4. IEEE, November 1995, pp. 1942–1948.

[14] J. Kennedy, “Bare bones particle swarms,” in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium (SIS’2003)*. IEEE, April 2003, pp. 80–87.

[15] G. R. Harik, F. G. Lobo, and D. E. Goldberg, “The compact genetic algorithm,” *IEEE Transactions On Evolutionary Computation*, vol. 3, no. 4, pp. 287–297, November 1999.

[16] C. W. Ahn, “Advances in evolutionary algorithms: Theory, design and practice,” in *Studies in Computational Intelligence*. Springer-Verlag Berlin Heidelberg, 2006, pp. 45–83.

[17] Chang Wook Ahn and R. S. Ramakrishna, “Elitism-based compact genetic algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 367–385, August 2003.

[18] J. Burkardt, “The truncated normal distribution,” https://people.sc.fsu.edu/~jburkardt/presentations/truncated_normal.pdf, pp. 21–24, October 2014, online; accessed 29 September 2019.

[19] E. Zitzler, K. Deb, and L. Thiele, “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results,” *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, Summer 2000.

[20] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed. New York: Springer, September 2007, ISBN 978-0-387-33254-3.

[21] E. Zitzler and L. Thiele, “Multiobjective Optimization Using Evolutionary AlgorithmsA Comparative Study,” in *Parallel Problem Solving from Nature–PPSN V*. Amsterdam, The Netherlands: Springer Verlag. Lecture Notes in Computer Science Vol. 1498, September 27-30 1998, pp. 292–301.

[22] J. R. Schott, “Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization,” Master’s thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.

[23] H. Ishibuchi, H. Masuda, and Y. Nojima, “A Study on Performance Evaluation Ability of a Modified Inverted Generational Distance Indicator,” in *2015 Genetic and Evolutionary Computation Conference (GECCO 2015)*. Madrid, Spain: ACM Press, July 11-15 2015, pp. 695–702, ISBN 978-1-4503-3472-3.

[24] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, “Scalable Test Problems for Evolutionary Multiobjective Optimization,” in *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*. USA: Springer, 2005, pp. 105–145.