ISSN 2007-9737

The Gradient Subspace Approximation as Local Search Engine within Evolutionary Multi-objective Optimization Algorithms

Sergio Alvarado[1], Carlos Segura[2], Oliver Schütze[1], Saúl Zapotecas[3]

[1] CINVESTAV-IPN, Computer Science Department, Mexico

[2] Center for Research in Mathematics (CIMAT), Guanajuato, Mexico

[3] Universidad Autónoma Metropolitana, Cuajimalpa Unit, Department of Applied Mathematics and Systems, Division of Natural Sciences and Engineering, Mexico

salvarado@computacion.cs.cinvestav.mx, carlos.segura@cimat.mx, schuetze@cs.cinvestav.mx, saul.zapotecas@gmail.com,

Abstract. In this paper, we argue that the gradient subspace approximation (GSA) is a powerful local search tool within memetic algorithms for the treatment of multi-objective optimization problems. The GSA utilizes the neighborhood information within the current population in order to compute the best approximation of the gradient at a given candidate solution. The computation of the search direction comes hence for free in terms of additional function evaluations within population based search algorithms such as evolutionary algorithms. Its benefits have recently been discussed in the context of scalar optimization. Here, we discuss and adapt the GSA for the case that multiple objectives have to be considered concurrently. We will further on hybridize line searchers that utilize GSA to obtain the search direction with two different multi-objective evolutionary algorithms. Numerical results on selected benchmark problems indicate the strength of the GSA-based local search within the evolutionary strategies.

Keywords. Multi-objective optimization, evolutionary computation, gradient subspace approximation (GSA), memetic algorithms, gradient-free local search, line search method.

1 Introduction

In many real-world applications it is necessary to solve optimization problems where several different objectives have to be considered at the same time. As an example, consider the design of a given product where two important objectives are certainly to maximize its performance while minimizing its cost. Such problems are known in literature as multi-objective optimization problems (MOPs). Solving MOPs is in general not an easy task. One important characteristic of MOPs is that their solution sets, the so-called Pareto sets, form objects of dimension $k-1$, where k is the number of objectives involved in the problem.

This is in contrast to "classical" scalar optimization problems (SOPs), where one typically obtains one single optimal solution (i.e., a zero dimensional set). Since the Pareto sets can apart from simple academic problems not be expressed analytically, one important task in multi-objective optimization is thus to find a suitable finite size representation of the solution set in order to provide the decision maker an overview of his/her possibilities.

Over the last two decades, specialized evolutionary algorithms, called multi-objective evolutionary

algorithms (MOEAs), have caught the interest of many researchers for the treatment of MOPs. Reasons for this include that MOEAs are applicable to a wide range of problems, are of global nature and hence in principle not too sensitive to the initial candidate set, and allow to compute a finite size representation of the Pareto set in one single run of the algorithm (see, e.g., [7, 1, 42, 5] and references therein).

However, one common drawback of all MOEAs is that they need quite a few function evaluations in order to obtain acceptable Pareto set approximations due to their relatively slow convergence rates. As a possible remedy, researchers have considered memetic algorithms, i.e., MOEAs hybridized with local search strategies that are mainly coming from mathematical programming techniques (e.g. [37, 21, 40, 15, 25, 3, 2, 13, 38]).

However, though memetic strategies increase in general the obtained approximation qualities which is caused by the local convergence of mathematical programming techniques, their overall cost is quite high due to the requirement of the gradient (and maybe even the Hessian), of the objectives. Note that not in all applications gradient information is at hand.

In that case, one can approximate this information e.g. via finite difference methods (e.g., [28]). However, the cost for the approximation of the gradient is linearly proportional to the dimension n of the decision space and even quadratic in case Hessian information is computed.

Even for other gradient free methods so-called exploration movements are required to gather the required information from the neighborhood to find a suitable search direction [18]. Such movements, that ideally generate the steepest descent direction, also come with in principle n additional function calls making the local search a quite costly procedure.

The Gradient Subspace Approximation (GSA), is a numerical technique that aims to compute the most greedy search direction at a given point x for an objective f out of a given subspace S ([33]). For unconstrained optimization problems this leads ideally to the normalized negative of the gradient $\nabla f(x)$.

One advantage of the GSA is that both inequality and equality constraints can be directly incorporated in the computation of the search direction. Another important aspect is that gradient information is not needed as the given neighborhood information can be exploited. Thus, the computation of the search direction comes in principle for free (in terms of additional function evaluations), within population based search techniques such as evolutionary algorithms. GSA has been investigated within memetic algorithms for the treatment of SOPs in [33].

In this paper, we apply GSA to the context of multi-objective optimization, i.e., we make a first effort to use and adapt the gradient estimator within the local search engines of memetic MOEAs. As it can be seen, GSA can in principle be coupled with any gradient based local search technique making it a universal and inexpensive tool for multi-objective memetic algorithms.

The remainder of this paper is organized as follows. Section 2, presents some basic concepts required for the understanding of this work. Section 3 presents some motivations and adaptations of GSA to the context of multi-objective optimization. In Section 4, we present two different memetic algorithms that use GSA for the local search engine. Section 5 presents numerical results on some selected benchmark functions on the two memetic algorithms including a comparison to their base algorithms. Finally, we will conclude and give possible paths for future work in Section 6.

2 Basic Concepts

2.1 Multi-Objective Optimization

A continuous MOP can be stated in mathematical terms as:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} F(x), \\ s.t.\ &g_i(x) \leq 0\ i = 1,\ldots,p, \\ &h_j(x) = 0\ j = 1,\ldots,m, \end{aligned} \tag{1}$$

where F is defined as the vector of the objective functions:

$$\begin{aligned} F :&\mathbb{R}^n \to \mathbb{R}^k, \\ F(x) &= (f_1(x), \ldots, f_k(x))^T, \end{aligned} \tag{2}$$

and where each objective $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, k$, is for convenience considered to be continuously differentiable. We stress, however, that this smoothness assumption is only needed for the theoretical considerations but not for the realization of the algorithms. The inequality constraints g_i, $i = 1, \ldots, p$, and the equality constraints h_j, $j = 1, \ldots, m$, form the feasible set Q, i.e.:

$$\begin{aligned} Q = \{x \in \mathbb{R}^n \ &: g_i(x) \leq 0, i = 1, \ldots, p, \\ &\text{and} \quad h_j(x) = 0, j = 1, \ldots, m\}. \end{aligned} \tag{3}$$

When defining the optimal solution we cannot proceed as for the classical scalar optimization case since F is vector-valued. For MOPs, optimality is defined based on the concept of dominance [29]:

Definition 1 *(a) Let $x, y \in Q$ and $f^x = f(x), f^y = f(y) \in \mathbb{R}^k$ be its respective images. We say that the vector x dominates y (denoted as $x \prec y$) iff $f_i^x \leq f_i^y, i = 1, \ldots, k$, and there exist an index j such that $f_j^x < f_j^y$.*

(b) A vector $x^* \in Q$ is called Pareto optimal *if there does not exist another point $y \in Q$ such that $y \prec x^*$.*

(c) The set of Pareto optimal solutions is called the Pareto set *(PS), i.e.,*

$$PS = \{x \in Q | \nexists y \in Q, y \prec x\}. \tag{4}$$

(d) Finally, the image of the Pareto set, $F(PS)$, is called the Pareto front *(PF).*

Typically, i.e., under certain mild assumptions on the MOP, both the Pareto set and the Pareto front form $(k-1)$-dimensional sets [14].

2.2 Solving a MOP

So far, many different classes of methods for the numerical solution of MOPs exist such as mathematical programming techniques [37, 6, 26], continuation methods [31, 14, 34, 32, 30, 23, 24, 22], or cell-mapping and subdivision techniques [9, 16, 12, 27, 11].

In this study we are particularly interested in multi-objective evolutionary algorithms [5, 7] and memetic algorithms [37, 21, 40, 15], as they have demonstrated to be efficient in many cases.

In the following we shortly present two different MOEAs that we use within this work as the base algorithms for our GSA-based local search: the decomposition based algorithm MOEA/D [42] and a memetic variant of the well-known algorithm NGSA-II that utilizes gradient information [19].

2.2.1 MOEA/D

The Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D, [42]), is a state-of-the-art evolutionary algorithm that decomposes the given MOP into several auxiliary SOPs. The main idea behind MOEA/D is to solve each of these SOPs simultaneously. At the end of the run of MOEA/D, it constructs an entire approximation of the Pareto front by combining the results obtained from each of the auxiliary SOPs.

There exist several scalarization methods coupled along with the MOEA/D. In this paper we are concerned in particular with the Tchebycheff decomposition [26]. This decomposition is based on the infinity norm. The Tchebycheff decomposition requires a reference point $z \in \mathbb{R}^k$ and a weight $w \in \mathbb{R}^k$ for each subproblem. The Tchebycheff decomposition can be defined as:

$$T(f^x, w, z) = \max_{i=1,\ldots,k} w_i |f_i^x - z_i|, \tag{5}$$

where f^x represents the image of x. MOEA/D establishes a way of udpating the reference point, meaning that no additional parameters are required.

Decomposition based methods have many advantages over "classical" elitist MOEAs that are entirely based on the dominance relation such as the potential for a faster convergence toward the

Pareto set [4, 36, 38]. In our context, GSA can be applied to all the auxiliary SOPs as done in [33].

In addition, there is another (yet unexplored), aspect of MOEA/D that makes it an interesting candidate for GSA: it already has implemented a certain neighborhood structure to decide which SOP is "near" to another one. Thus, one may use this structure to select the neighboring samples x_i for a given candidate solution x_0 which we, however, have to leave for future work.

The Algorithm 1, presents the pseudocode of MOEA/D using Tchebycheff decomposition. In the algorithm, N represents the number of subproblems and s_N represents the size of the neighborhood.

2.2.2 IG-NSGA-II

The improved gradient based NSGA-II (IG-NSGA-II) [19] is a modification of the work proposed in [41]. The IG-NSGA-II algorithm uses a local search strategy based on the gradient information of the objective functions to compute a descend direction for unconstrained MOPs. In particular, the descend direction for $k = 2$ is constructed according to the work proposed by Lara et al. in [20]:

$$\nu_L(x) = -\left(\frac{g_1(x)}{\|g_1(x)\|_2} + \frac{g_2(x)}{\|g_2(x)\|_2}\right), \qquad (6)$$

where $g_1 = \nabla f_1(x)$ and $g_2 = \nabla f_2(x)$.

The IG-NSGA-II implements two different mechanisms to preserve the diversity of the final solution set. Both mechanisms are based on a chaotic map that are used to generate new candidate solutions. The first mechanism is a new method to initialize the population of the algorithm. The second mechanism computes the spreading λ_s of the entire population at each generation. If this spreading value is below a certain threshold a new individual is created using a chaotic map. The new individual created by the chaotic approach is included in the current population as a mechanism to explore different regions.

Algorithm 2, presents the pseudocode for the IG-NSGA-II. Here, N again denotes the population size.

Algorithm 1 Pseudocode of the MOEA/D

Require: vector function F, scalar function T, N weight vectors $w_1, \cdots, w_N$, the number of weight neighbors s_N
Ensure: Final population P

for $i = 1, \ldots, N$ **do**
  Create structure σ_i for the i-th subproblem
  Compute the indexes of the s_N 'closest' weights vectors from w_i and stores them in I_N
  Set $\sigma_i \to I := I_N$
  Generate a random vectors and store it in $x_a\mathbb{R}^n$.
  Set $\sigma_i \to x_\beta := x_a$
  Compute $f_a = F(x_a)$
  Set $\sigma_i \to f_\beta := f_a$
end for
Find the ideal point $z \in \mathbb{R}^k$
for $i = 1, \ldots, N$ **do**
  Compute $s_a = T(\sigma \to x, w_i, z)$
  Set $\sigma_i \to s_\beta := s_a$
end for
while Stopping criteria is not met **do**
  for $i = 1, \cdots, N$ **do**
    Set $I_a = \sigma_i \to I \cup i$
    Randomly select two indexes $j, k \in I_a$
    Set $x^m := \sigma_j \to x_\beta$
    Set $x^n := \sigma_k \to x_\beta$
    Set $s_{best} := \sigma_i \to s_\beta$
    Generate x_{new} using genetic operators on x^m and x^n.
    Compute $f_{new} = F(x_{xnew})$
    Compute $s_{new} = T(f_{new}, w_i, z)$
    if $s_{new} < s_{best}$ **then**
      Set $\sigma_i \to x_\beta := x_{new}$
      Set $\sigma_i \to f_\beta := f_{new}$
      Set $\sigma_i \to s_\beta := s_{new}$
    end if
    Update z
  end for
end while
Set $P := \emptyset$
for $i = 1, \ldots, N$ **do**
  Set $P := P \cup \sigma_i$
end for
return P

Algorithm 2 Pseudocode of the IG-NSGA-II

Require: frequency of local search k_l, threshold for chaotic operator ϵ_d
Ensure: final population P_G
1: Randomly generate the population P_0 using the chaotic initialization
2: Calculates $F(P_0)$
3: Apply genetic operators in P_0 to generate Q_0
4: $i = 0$
5: **while** Stopping criteria is not met **do**
6: Set $R_i = P_i \cup Q_i$
7: Calculate the rank value of R_i
8: Calculate the crowding distance of R_i
9: Select the N individuals with the lowest rank and highest crowding distance from R_i and store them in P_i.
10: Apply genetic operators in P_i to generate Q_i
11: **if** mod(i, k_l) == 0 **then**
12: Apply local search on Q_i
13: **end if**
14: Calculate s_d
15: **if** $s_d < \epsilon_d$ **then**
16: Apply chaotic operator on Q_i
17: **end if**
18: Set $i = i + 1$
19: **end while**
20: **return** $P_G := P_{i-1}$

2.3 Gradient Subspace Approximation

The Gradient Subspace Approximation (GSA) [33], is a method proposed to approximate the gradient of a function at a given candidate solution. GSA utilizes the information that is present for each generation of an evolutionary algorithm to compute such an approximation. Here, we present a brief description for the unconstrained case. For the treatment of constrained problems (inequality and equality constraints), the reader is referred to [33].

Consider the following single objective optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x), \tag{7}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is assumed to be differentiable. We say that $\nu \in \mathbb{R}^n$ is a descend direction of f at x if:

$$\langle \nu, \nabla f(x) \rangle < 0, \tag{8}$$

where $\nabla f(x)$ represents the gradient of f at x. It is known (e.g., [28]) that the vector with the maximal decay at x is given by the negative of the gradient. This (normalized) vector can be expressed also as the solution of the following auxiliary SOP:

$$\begin{aligned} &\min_{\nu \in \mathbb{R}^n} \langle \nu, \nabla f(x_0) \rangle, \\ &\text{s.t. } \|\nu\| = 1. \end{aligned} \tag{9}$$

To exploit the neighboring information of the candidate solution x_0, consider that r search directions ν_r are given along with the r directional derivatives of such search directions $\langle \nu_i, \nabla f(x_0) \rangle, i = 1, \ldots, r$. If we incorporate the neighborhood information such that the most greedy direction exist in $\nu \in \text{span}(\nu_1, \ldots, \nu_r)$, Problem (9) can be rewritten as follows:

$$\begin{aligned} &\min_{\lambda \in \mathbb{R}^r} \left(\langle \nabla f(x_0), \sum_{i=1}^{r} \lambda_i \nu_i \rangle = \sum_{i=1}^{r} \lambda_i \langle \nabla f(x_0), \nu_i \rangle \right), \\ &\text{s.t. } \left\| \sum_{i=1}^{r} \lambda_i \nu_i \right\|_2^2 - 1 = \lambda^T V^T V \lambda - 1 = 0, \end{aligned} \tag{10}$$

where

$$V = (\nu_1, \ldots, \nu_r) \in \mathbb{R}^{n \times r}. \tag{11}$$

The following result shows that the solution of (10) is unique under certain assumptions and that it can be expressed analytically.

Proposition 1 *Let the search directions $\nu_1, \ldots, \nu_r \in \mathbb{R}^n$, $r \leq n$, be linearly independent and:*

$$\tilde{\lambda}^* := -(V^T V)^{-1} V^T \nabla f(x_0). \tag{12}$$

Then, there exist a single solution of Equation (10) *given by:*

$$\lambda^* := \frac{\tilde{\lambda}^*}{\|V\lambda^*\|_2^2}, \tag{13}$$

and thus,

$$\nu^* = \frac{-1}{\|V\lambda^*\|_2^2} V (V^T V)^{-1} V^T \nabla f(x_0), \tag{14}$$

is the most greedy search direction in $span\{\nu_i, \ldots, \nu_r\}$.

Equation (14), requires that gradient information is given. To avoid such restriction we can approximate such information using the neighbors of the candidate solution. Given the candidate solution x_0, a set of neighbors $x_i, i = 1, \ldots, r$, and their respective function values $f(x_i), i = 1 \ldots, r$, the gradient information can be approximated such that:

$$\langle \nu, \nabla f(x_0) \rangle_i = \frac{f(x_i) - f(x_0)}{\|x_i - x_0\|_2} + O(\|x_i - x_0\|). \tag{15}$$

In the above equation, the search direction has apparently been chosen as:

$$\nu_i := \frac{x_i - x_0}{\|x_i - x_0\|_2}, \quad i = 1, \ldots, r. \tag{16}$$

For more details and for the extension to equality and inequality constrained problems we refer to [33].

3 Proposed Approach

In this section, we will adapt the local search tool GSA from [33] to the context of multi-objective optimization problems, in particular for the use within specialized evolutionary algorithms.

To this end, we will (a) investigate the applicability of GSA within MOEAs which includes a way to compute the neighborhood for the chosen samples, (b) discuss how to approximate the Jacobian matrix via GSA, (c) extend Laras descent direction for inequality constrained problems in order to increase its applicability, (d) discuss a particular choice of the step size control, and (e) discuss how to efficiently balance the resources for the GSA based local search.

Finally, we will (f) propose two different memetic algorithms, MOEA/D/GSA and IG-NSGA-II/GSA, that utilize GSA for their local search engines.

3.0.1 Applicability of GSA within MOEAs

In population based scalar optimization one typically observes that many individuals are located around the best found individual (denoted here by x_0), apart from very early stages of the search. Thus, for the usage of GSA one can expect that sufficient samples are at hand near x_0 to approximate $\nabla f(x_0)$. The situation, however, changes when considering multi-objective optimization problems as for such problems there does typically not exist one single solution, but an entire manifold of solutions. Hence, there is no single "best" solution any more where other solutions group around. Instead, the solutions are spread along the solution set.

Thus, the natural question arises if there exist enough samples within multi-objective evolutionary algorithms (or any other set or population based algorithms for the treatment of MOPs) such that GSA can be applied successfully. To answer this question empirically, we consider in the following the neighborhood sizes from populations obtained via MOEA/D for four different MOPs with different characteristics. In all cases, we use the 2-norm in decision to define the neighborhood. More precisely, we define the neighborhood for an individual p_0 of a given population P as:

$$N_\delta(p_0) := \{p \in P \backslash \{x_0\} \ : \ \|p - p_0\|_2 \leq \delta\}, \tag{17}$$

however, we stress that in principle any other norm can be chosen.
In our computations, we have used the values $\delta = 0.25$, 0.5, and 1. As MOPs we have chosen (see Table 8 for the formulations of all problems used in this work):

(a) CONV: unimodal, $n = 2$ decision variables, $k = 2$ objectives, linear and connected Pareto set, convex Pareto front,

(b) ZDT1: unimodal, $n = 30$, $k = 2$, linear and connected Pareto set, convex Pareto front,

(c) Kursawe: multi-modal, $n = 3$, $k = 2$, disconnected Pareto set and front, non-convex Pareto front, and

ISSN 2007-9737

(d) DTLZ3 (3): multi-modal, $n = 12$, $k = 3$, connected and linear Pareto set, concave Pareto front.

Figures 1-4 present the results of our experiments for population size $N = 100$ averaged over 30 independent runs. Shown are the number of neighbors (i.e., the magnitudes of $N_\delta(p_0)$), averaged over all runs and all elements of each population generated by MOEA/D. As it can be seen, the magnitudes of the neighborhoods quickly go over 5 which has been detected as an effective value for r in order to obtain a descent direction (and which has also been used for the computations we present in the sequel). These observations are in consensus with other experiments we have done.

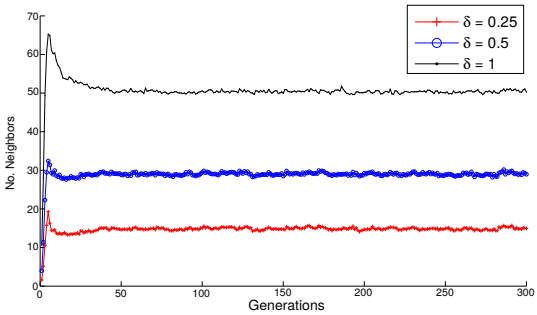


Fig. 1. Averaged number of neighbors for CONV

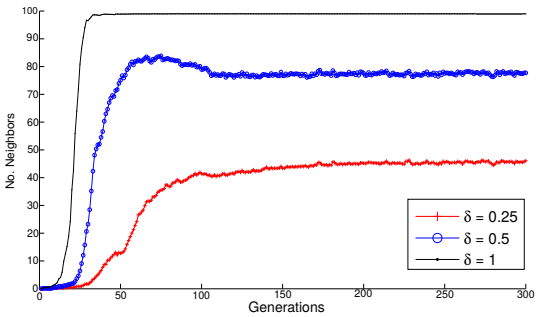


Fig. 2. Averaged number of neighbors for ZDT1

Based on this insight we choose our neighborhood $N(p_0)$ for a given candidate solution simply via selecting the r nearest elements (using the 2-norm) out of the population P. See Algorithm 3 for a pseudocode.

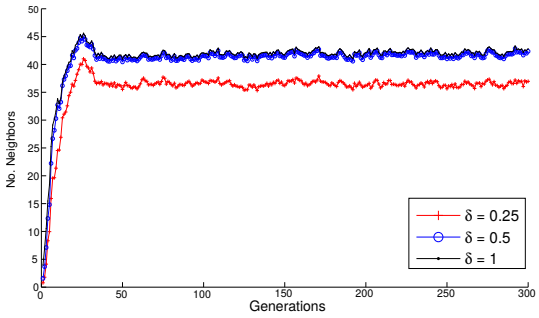


Fig. 3. Averaged number of neighbors for Kursawe

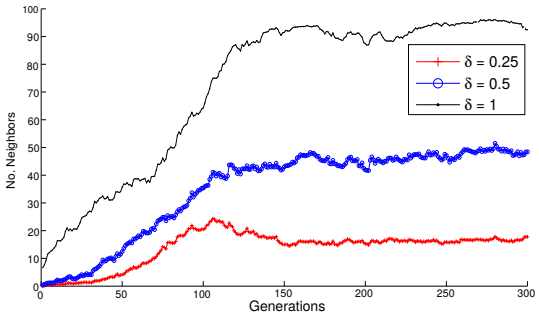


Fig. 4. Averaged number of neighbors for the three-objective DTLZ3

Hereby we assume that the population is given by μ elements, i.e.:

$$P = \{p_1, \ldots, p_\mu\}. \tag{18}$$

We stress that one potential problem when using MOEA/D is that populations can have duplicate individuals (as a point can be the best found solution for several scalarization problems at the same time). Hence, we have to avoid such solutions as else the distance between the candidate solution and the sample is zero, and GSA is not applicable. More sophisticated neighborhood selection strategies that e.g. take into account the condition number of V or the one MOEA/D is using are subject of ongoing research.

3.1 Approximating the Jacobian

Assume we are given a MOP of the form (1), and that we are given a candidate solution x_0 together with the r search directions $\nu_1, \ldots, \nu_r \in \mathbb{R}^n$ that form the subspace.

Algorithm 3 Selection of neighborhood $N(p_0)$ with r nearest elements from population P

Require: population P, candidate solution p_0, neighboring size r
Ensure: neighborhood $N(p_0)$ consisting of the r nearest elements to p_0
Set $N(p_0) := \emptyset$
for $i = 1, \ldots, \mu$ **do**
    $\phi_i = \infty$
end for
for $i = 1, \ldots, \mu$ **do**
    if $\|p_i - p_0\|_2 > 0$ **then**
        Set $\phi_i := \|p_i - p_0\|_2$
    end if
end for
Sort ϕ in ascending order, denote by ω the set of resulting indexes
for $i = 1, \ldots, r$ **do**
    $N(p_0) := N(p_0) \cup P_{\omega_i}$
end for
return $N(p_0)$

$$S := \text{span}\{\nu_1, \ldots, \nu_r\}. \tag{19}$$

For every objective f_i, the best approximation $\widetilde{g_i}(x_0)$ of the gradient $\nabla f_i(x_0)$ at a given point x_0 within S can according to the above discussion be computed via:

$$\widetilde{g_i} := V(V^T V)^{-1} V^T \nabla f_i(x_0) \in \mathbb{R}^n. \tag{20}$$

The Jacobian matrix $J(x_0)$ of F at x_0 is defined by:

$$J(x_0) = \begin{pmatrix} \nabla f_1(x_0)^T \\ \vdots \\ \nabla f_k(x_0)^T \end{pmatrix} \in \mathbb{R}^{k \times n}. \tag{21}$$

Thus, using (20), a best approximation $\widetilde{J}(x_0)$ of $J(x_0)$—in the sense that each row vector of the matrix is the best approximation of the transposed gradient of f_i at x_0—can be obtained via:

$$\widetilde{J}(x_0) := \begin{pmatrix} \widetilde{g_1}^T \\ \vdots \\ \widetilde{g_k}^T \end{pmatrix} = J(x_0)^T V (V^T V)^{-1} V^T. \tag{22}$$

For the gradient free realization, consider the product of the first two matrices on the right hand side of (22) it is:

$$\mathcal{F} := J(x_0)^T V = \begin{pmatrix} \langle \nabla f_1(x_0), \nu_1 \rangle & \ldots & \langle \nabla f_1(x_0), \nu_r \rangle \\ \vdots & & \vdots \\ \langle \nabla f_k(x_0), \nu_1 \rangle & \ldots & \langle \nabla f_k(x_0), \nu_r \rangle \end{pmatrix}. \tag{23}$$

That is, every entry:

$$m_{ij} = \langle \nabla f_i(x_0), \nu_j \rangle, \tag{24}$$

of the matrix $\mathcal{F}$ is a directional derivative and can be approximated via the finite difference method described above, and this approximation comes for free in terms of additional function evaluations as the values $f_i(x_j)$ are already known. Thus, using (22), where $\mathcal{F}$ is computed as described, yields a Jacobian approximation without explicitly computing the objectives' gradients. The algorithm can hence be seen as gradient free.

3.1.1 Laras Descent Direction for Inequality Constrained MOPs

Once the values of $\tilde{J}(x_0)$ are computed, we can compute the approximation of the descent direction (6) via:

$$\widetilde{\nu}_L = \left(\frac{\widetilde{g}_1}{\|\widetilde{g}_1\|_2} + \frac{\widetilde{g}_2}{\|\widetilde{g}_2\|_2} \right), \tag{25}$$

where $\widetilde{g}_i, i = 1, 2$, represent the row vectors of the matrix $\widetilde{J}(x_0)$. One interesting aspect would be to know under which conditions (25) actually is a descent direction for both objectives which we have to leave for future work.

One main restriction of descent direction (6) is that it is only applicable for unconstrained MOPs. Here, we make a first attempt to extend the descent direction also for inequality contrained problems where we can make use of the gradient projection method of the GSA. In particular, we propose to first compute the direction $\widetilde{\nu}$ and then to project the vector back to the feasible region if needed (refer to

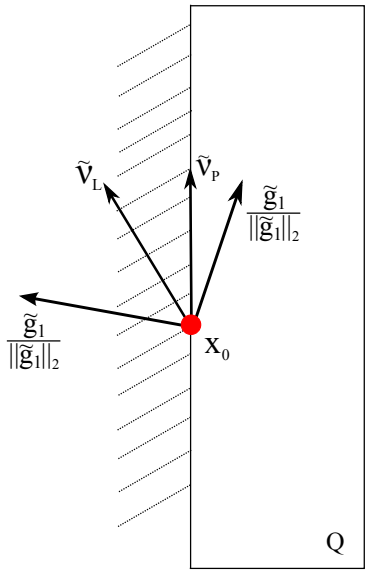


Fig. 5. Correction of the obtained descent direction ν_L to the feasible region Q

Figure 5 for a graphical illustration of the correction where $\widetilde{\nu}_L$ points outside the feasible region).

Consider that at the candidate solution x_0 we have p active inequality constraints and that $\widetilde{\nu}_L$ points into an infeasible region (i.e., formally we have $\nabla g_i(x_0)^T \widetilde{\nu}_L < 0$ for at least one of the active constraints g_i). To guarantee that for the projection $\widetilde{\nu}_L$ it holds $\nabla g_i(x_0)^T \widetilde{\nu}_L \geq 0$ for all active inequality constraints, define $\widetilde{M} \in \mathbb{R}^{p \times r}$ as follows:

$$\widetilde{M} = G(x_0)^T V = \langle \nabla(g_i(x)), \nu_j \rangle, i = 1, \ldots, p, \quad j = 1, \ldots, r, \tag{26}$$

where $G(x_0)$ is defined as the Jacobian of the active constraints:

$$G(x_0) = \begin{pmatrix} \nabla g_1(x_0)^T \\ \vdots \\ \nabla g_p(x_0)^T \end{pmatrix} \in \mathbb{R}^{p \times n}. \tag{27}$$

Then, compute the matrix $K \in \mathbb{R}^{r \times (r-1)}$ those column matrix build an orthonormal basis of $\widetilde{M}$ (which can be done e.g. via a QR-factorization of $\widetilde{M}$). The desired projection is thus given by:

$$\tilde{\nu}_P := VK(VK)^T \nu_L. \tag{28}$$

The gradient free realization is similar as above: note that every entry of $\widetilde{M}$ is again a directional derivative (now for the constrained functions). Thus, we can compute the entries $\hat{m}_{ij}$ of $\widetilde{M}$ via:

$$\begin{aligned} \hat{m}_{ij} &= \langle \nabla g_i(x_0), \nu_j \rangle \\ &= \frac{g_i(x_j) - g_i(x_0)}{\|x_j - x_0\|_2} + O(\|x_j - x_0\|), \\ & i = 1, \ldots, p, \; j = 1, \ldots, r. \end{aligned} \tag{29}$$

The use of the projected direction has led to satisfying results in our computations. However, also here it is interesting to know under which condition $\tilde{\nu}_P$ actually is a descent direction for all objectives which we have to leave as well for future work.

3.1.2 Computing the Step Size

The GSA is a tool to obtain search directions that are used within line search methods. That is, given a current iterate x_i and a search direction ν_i, the new iterate is computed via:

$$x_{i+1} = x_i + t_i \nu_i, \tag{30}$$

where $t_i > 0$ is the chosen step size. The proper choice of t_i is a delicate problem: too small step sizes lead on the one hand to a high probability of getting a better (i.e., dominating) solution, but on the other hand the gain may be too small. In particular, this gain is likely smaller than obtained via the evolutionary strategy (and hence, the local search has no effect on the quality but comes with an additional cost and is thus decreasing the overall performance of the algorithm). In turn, if iterates x_i are already near to optimal solutions, large step sizes lead to waste of resources as x_{i+1} will be dominated by x_i.

In our computations we choose an initial step size t_0 together with Armijo-like backtracking methods (e.g., [10, 28]). For the selection of the initial step size t_0 we have observed that it makes sense to use the "size" of the neighborhood $N(x_i)$ as described above.

In early stages of the search, it is likely that this size is much larger compared to the sizes in later stages. More precisely, given x_0, and a neighborhood $N(x_0)$ with r elements we compute t_0 via:

$$t_0 := \frac{1}{n \cdot r} \sum_{x_i \in N(x_0)} \sum_{j=1}^{n} |x_{ij} - x_{0j}|,\ i = 1, \ldots, r. \tag{31}$$

3.1.3 Balancing the Local Search Resources

One of the challenges that the memetic strategies have is to decide how and when to apply the local search technique. In particular, for some MOPs this task can become very difficult to solve. One such problem, next to the cost caused by the local sarch, is that if the local searcher is applied too often it is possible that the solution set loses diversity [17]. In MOEA/D one can relatively easy measure the percentage of improvement achived by the local search strategy using the scalar functions of each subproblem.

This section describes a mechanism that takes into consideration the improvement obtained by the GSA and how to measure it. Next, it compares the techniques and redistributes the resources between the base evolutionary algorithm and the GSA-based local search technique in each generation. The mechanism discussed in this section is based on the the scheme proposed in [17] for the treatment of SOPs.

At this point we consider the evolutionary operators and the GSA as two standalone techniques. On each generation the balancing mechanism measures the quality of each technique. To do so, the improvement must be computed for each individual in the population. Lets consider a given individual x_i^j in the population at the j-th generation.

After its offspring x_i^{j+1} is calculated, it is possible to compute the quality of the technique that generates the individual. Lets assume that the points x_i^j, x_i^{j+1} and its respective images $f_i^j = F(x_i^j)$ and $f_i^{j+1} = F(x_i^{j+1})$ are given. Using the Tchebycheff decomposition described in Equation (5), the quality term can be described as:

$$q(f_i^j, f_i^{j+1}, w, z) = T(f_i^j, w, z) - T(f_i^{j+1}, w, z), \tag{32}$$

where w represent the weight of the i-th subproblem where x_i^j is the best solution found so far and z represents the ideal point.

Once the qualities of all subproblems in MOEA/D are computed, the method uses such values to compute an average quality for each technique. To compare the quality at generation j, we assume two different populations created according to the creating method. P_{T1}^j represents the subpopulation generated by evolutionary techniques and P_{T2}^j represents the subpopulation generated with the GSA method. For each one of the techniques the average quality $Q(P_{Ti}^j), i = 1, 2$, is given by:

$$Q(P_{Ti}^j) = \frac{1}{|P_{Ti}^j|} \sum_{x_i^j \in P_{Ti}^j} q_i(x_i^j). \tag{33}$$

Now, the balance mechanism uses the quality of each technique to compute the number of function calls that each technique is going to use on the next generation. The participation ratios are computed as shown in Algorithm 4. Participation ratios are used to distribute the number of function calls to each technique. In order to ensure that both techniques are applied at least on a percentage of the population, the parameters r_{min} and r_{max} are introduced. Such parameters establish a lower and an upper bound for the techniques.

3.1.4 Memetic Algorithms

Based on the above discussions, we will propose in the sequel two different memetic algorithms whose local searchers utilize GSA. The first one, MOEA/D/GSA, is based on MOEA/D and is designed for the treatment of unconstrained MOPs. Algorithm 5 presents the pseudo code of this algorithm. Here, it is important to mention that the initial participation ratio of the GSA is set on a lower value compared with the one of the evolutionary algorithm.

The reason for this is that in principle, when the algorithm starts the neighborhood size of

Algorithm 4 Computation of participation ratios

Require: Original participation ratios $R(P_{Ti}^{j})$, Average techniques quality $Q(P_{Ti}^{j})$, Decrement ratio r^{-}
Ensure: New participation ratios $R(P_{Ti}^{j+1})$

Find the best quality and stored as q_{best}
$a^{+} := 0$
$n^{+} := 0$
for $i = 1, 2$ **do**
  Set $d^{-} := 0$
  if $Q(P_{Ti}^{j})$ is different to q_{best} **then**
    $d^{-} := r^{-}\left(q_{best} - \frac{Q(P_{Ti}^{j})}{q_{best}}\right)$
    if $R(P_{Ti}^{j}) - d^{-} < r_{min}$ **then**
      $d^{-} := R(P_{Ti}^{j}) - r_{min}$
    end if
    Set $a^{+} := a^{+} + d^{-}$
  else
    $n^{+} := n^{+} + 1$
  end if
  $R(P_{Ti}^{j+1}) := R(P_{Ti}^{j+1}) - d^{-}$
end for
Set $a^{+} := \frac{a^{+}}{n^{+}}$
for $i = 1, 2$ **do**
  if $Q(P_{Ti}^{j})$ is equal to q_{best} **then**
    if $R(P_{Ti}^{j}) + a^{+} > r_{max}$ **then**
      Set $R(P_{Ti}^{j+1}) := r_{max}$
    else
      Set $R(P_{Ti}^{j+1}) := R(P_{Ti}^{j+1}) + a^{+}$
    end if
  end if
end for

any candidate solution of the population may be very large. Because of this size it is not quite recommendable to apply GSA since it is quite possible that the offspring do not present an improvement (note that GSA is based on forward difference methods which only work properly if the samples x_i are near to the candidate solution x_0).

The next memetic strategy we propose here is IG-NSGA-II/GSA whose base algorithm is IG-NSGA-II (Algorithm 2) and whose offspring are created as shown in Algorithm 6. The main difference resides in the creation of the offspring individuals. As we will use the projected search direction $\tilde{\nu}_P$ from (28), IG-NSGA-II/GSA is capable of handling inequality constrained MOPs.

Algorithm 5 Pseudocode of the MOEA/D/GSA

Randomly create initial population P_0.
Set $R(P_{T1})^{0} := 0.8, R(P_{T2})^{0} := 0.2$.
Set $k := 1$.
while Stopping criteria is not met **do**
  Select P_{T2}^{k} using random individuals from P_{k-1}.
  Create offspring $O(P_{T2}^{k})$ by using GSA.
  Calculate quality $Q(P_{T2}^{k})$.
  Select P_{T1}^{k} using random individuals from P_{k-1}.
  Create offspring $O(P_{T1}^{k})$ by using MOEA/D.
  Calculate quality $Q(P_{T1}^{k})$.
  Update the participation ratios $R(P_{T1})^{k}$, $R(P_{T2})^{k}$ using the qualities of each technique.
  Select the best individuals from $P_k \cup O(P_{T2}^{k}) \cup O(P_{T1}^{k})$ and save it into P_{k+1}
  Set $k := k + 1$
end while

Algorithm 6 Creation of an offspring using IG-NSGA-II/GSA

Require: Candidate solution x_i, neighborhood $N(x_0)$
Ensure: New offspring x_{i+1}

Calculate ν_L using Equation (25)
Construct direction ν_P from Equation (28)
Calculate initial step size t using $N(x_0)$
if x_{i+1} improves x_i **then**
  Set $x_{i+1} = x_i + t\nu_P$
  return x_{i+1}
else
  return x_i
end if

4 Experimental Results

In this section, we present different experiments dedicated to show the strength of GSA as an effective tool within multi-objective memetic algorithms. For this, we will first consider shortly the GSA as standalone algorithm.

Further on, we will consider and compare the two newly proposed memetic algorithms MOEA/D/GSA and IG-NSGA-II/GSA on several bechmark functions, where we will consider (a) unconstained MOPs, (b) MOPs with relatively easy constraints, and (c) MOPs with complex constraints.

4.1 GSA within Laras Method

First we investigate one line search method, namely the iteration:

$$x_{i+1} = x_i + t_i \nu_{L,i}, \tag{34}$$

where $\nu_{L,i}$ is the GSA-approximation of the descent direction (25) at the current iterate x_i and where t_i is the step size control as described above. As example we consider the unconstrained convex bi-objective problem:

$$f_j(x) = \sum_{i=1}^{n} \|x_i - a_{ji}\|_2^2, \quad j = 1, 2, \tag{35}$$

where $a_1 = (1, \ldots, 1)^T \in \mathbb{R}^n$ and $a_2 = (-1, \ldots, -1)^T \in \mathbb{R}^n$. For the value of n we have chosen 10.

First we investigate the influence of the number r of sampling points. Figure 6 presents the result of the following experiment: we have taken $x_0 \in \mathbb{R}^{10}$ at random and have generated a set of r neighorhood samples from $N_\delta(x_0)$ for $\delta = 0.15$ for $r = 3, 5$, and 7.

The figure shows the points $y_0 := F(x_0)$, the iterate $y_1 := F(x_0 + \nu_L)$ (i.e., the image of the iterate $x_1 := x_0 + \nu_L$ for the step size $t = 1$) and further the approximations of y_1 when using GSA and r samples. As it can be seen, already for $r = 5$ a significant move toward the Pareto front can be obtained, and for $r = 7$ the performance comes quite close to the usage of the exact gradient.

Since the consideration of *one* step has no significance, we consider an entire trajectory of solutions starting with x_0 using $r = 5$. In Figure 7 a sequence of 15 iterations is shown. As it can be seen, the iterations come close to the Pareto front.

As already mentioned above, more theoretical investigations are required to fully understand GSA based local search which we, however have to leave to future research. In the following we focus on the effect of the GSA based local search engines within memetic multi-objective evolutionary algorithms.

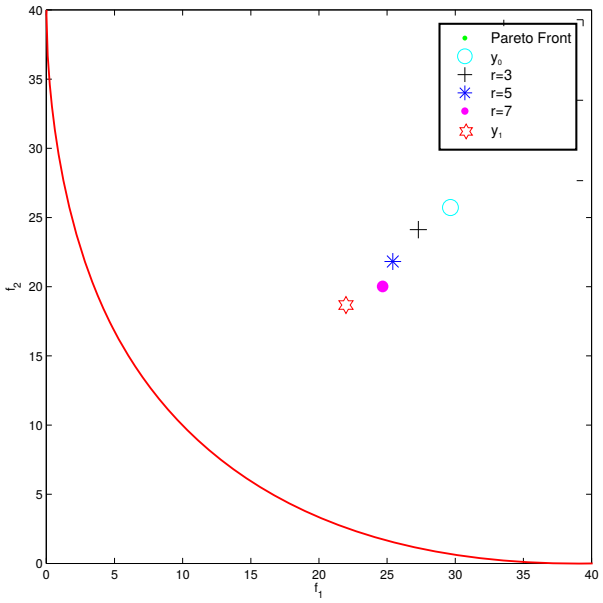


Fig. 6. Results of the approximation using several values of r

4.2 MOEA/D/GSA

For this experiment we use the original version of the MOEA/D algorithm as it was presented in [42]. A comparison using two different state-of-the-art indicators is performed. The Δ_2 indicator [35] and the hypervolume indicator [44] are selected for the performance assessment. Both indicators measure spread and convergence along the Pareto front to a certain extent. We propose to compare the algorithms (the standalone and the memetic version), after they have spent a certain number of function calls. We perform our comparison when the algorithms reached $30,000$ function evaluations for the functions with two objectives and $50,000$ for $k = 3$. We stress that MOEA/D is not explicitly aiming at one of any existing performance indicators. Instead, it

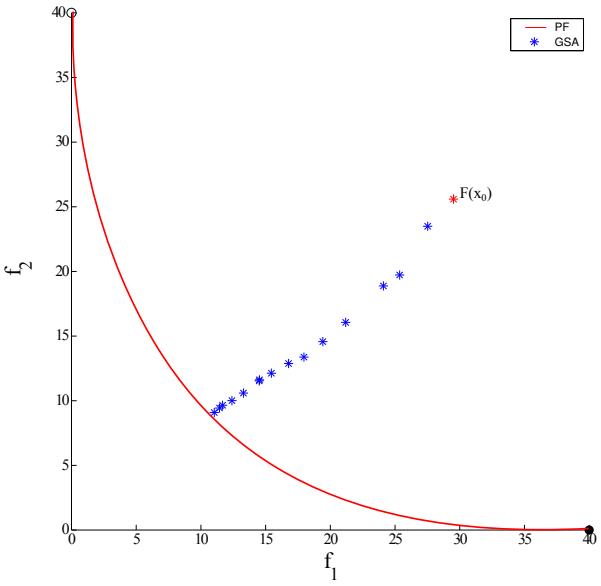


Fig. 7. Standalone GSA applied on quadratic function

measures the success via improvements in the scalarization functions.

Hence, along with the two state-of-the-art indicators, we propose a specialized indicator that measuares the averaged scalar value of the whole population (and thus, in a sense the overall success of the MOEA/D variants). In particular, we propose to use the indicator W that is defined as follows:

$$W(P) = \sum_{i}^{N} T(F(x_i), w_i, z), \tag{36}$$

where $x_i \in P$ is the best found individual for the i-th subproblem. We introduce this new indicator, as we have observed that if W decreases (and thus, in average also the best found solutions for all sub-problems of MOEA/D), this does not necessarily yield better hypervolume nor Δ_p values. In other words, the sub-problems are not matched to these (or any other), performance indicators.

To confirm that the GSA method can really improve a memetic algorithm it becomes necessary to make a comparison on different test functions. In particular, for this memetic strategy we selected two of the state-of-the-art benchmark suites. The first selected set of functions is the modified version of the ZDT functions proposed in [39]. For the second part of the experiments we use the DTLZ benchmark for $k = 3$ [8].

Table 1 presents the parameters used for each one of these benchmarks.

We performed an experiment in order to compare the performance of the memetic algorithm based on the GSA as a local searcher. For each comparison we observed the results over 30 independent runs. The experiments were performed over the ZDT test functions and the DTLZ test functions. The effectiveness of the MOEA/D/GSA was measured comparing the memetic algorithm along with two other methods: the base variant of MOEA/D and a memetic strategy based on the Nelder-Mead method. In these experiments we use a modified version of the Nelder-Mead method as follows: we modified the algorithm to incorporate neighboring information on it (as the GSA does). In particular, we incorporate r individuals from the population into the construction of the simplex. In case that $r < (n + 1)$, we computed the remaining $n - r + 1$ using the mechanisms proposed in the original Nelder-Mead algorithm. The resulting algorithm is termed here MOEA/D/NM.

Tables 2 to 6 present the results obtained by the three algorithms on the three different performance indicators. The nadir points for the hypervolume indicator are set as follows: $(5, 5)^T$ for the ZDT test functions, $(11, 11)^T$ for DTLZ 1-4 and $(5, 5, 5)^T$ for DTLZ 5-7. The tables present the results at a certain stage of the algorithm. That is, we measured the indicators after a certain number of function calls. For the ZDT test function we measure the results after $15,000$ function evaluations. Meanwhile, for the DTLZ functions we obtained the results after $30,000$ function evaluations. Bold numbers indicate that the algorithm is outperforming the other ones significantly. As it can be seen, MOEA/D/GSA wins in most cases. For instance, for the W indicator, MOEA/D/GSA significantly wins in 8 out of 12 cases, and is only outperformed on ZDT3 by MOEA/D/NM.

Figure 8 presents the computed Pareto fronts for the best individual on each of the ZDT problems.

Table 1. Parameters for the MOEA/D/GSA algorithm

Parameter	Description	Value	
		ZDT	DTLZ
η_c	Distribution index for crossover	20	
η_m	Distribution index for mutation	20	
p_c	Crossover probability	0.95	
p_m	Mutation probability	$1/n$	
N	Number of subproblems	100	300
k	Number of objective	2	3
r	Number of neighbors for GSA	5	
G_{max}	Maximum participation GSA	0.2	

Table 2. Δ_2 indicator results on the ZDT and DTLZ test function

	MOEA/D	MOEA/D/GSA	MOEA/D/NM
ZDT1	0.28312	**0.20459**	0.54943
(st. dev.)	(0.19010)	(0.02726)	(0.14881)
ZDT2	1.93103	**0.21283**	5.25555
(st. dev.)	(2.55223)	(0.02728)	(2.54624)
ZDT3	0.19964	**0.16347**	0.27417
(st. dev.)	(0.12773)	(0.04867)	(0.18484)
ZDT4	0.77793	**0.30239**	1.08488
(st. dev.)	(0.50872)	(0.28498)	(0.60226)
ZDT6	**0.27876**	0.61617	0.32326
(st. dev.)	(0.04682)	(0.27990)	(0.06259)
DTLZ1	0.30190	**0.28239**	0.36506
(st. dev.)	(0.71422)	(0.77269)	(0.80286)
DTLZ2	0.06878	0.06859	0.06884
(st. dev.)	(0.00065)	(0.00087)	(0.00085)
DTLZ3	3.50345	**1.43777**	4.07758
(st. dev.)	(4.28843)	(3.22033)	(4.36678)
DTLZ4	0.44036	**0.43869**	0.44319
(st. dev.)	(0.00871)	(0.01101)	(0.01194)
DTLZ5	73.02189	60.40117	**53.95727**
(st. dev.)	(7.48926)	(7.51920)	(10.55201)
DTLZ6	73.02189	60.40117	**53.95727**
(st. dev.)	(7.48926)	(7.51920)	(10.55201)
DTLZ7	**11.86712**	19.64119	11.97166
(st. dev.)	(0.47831)	(0.61246)	(0.61289)

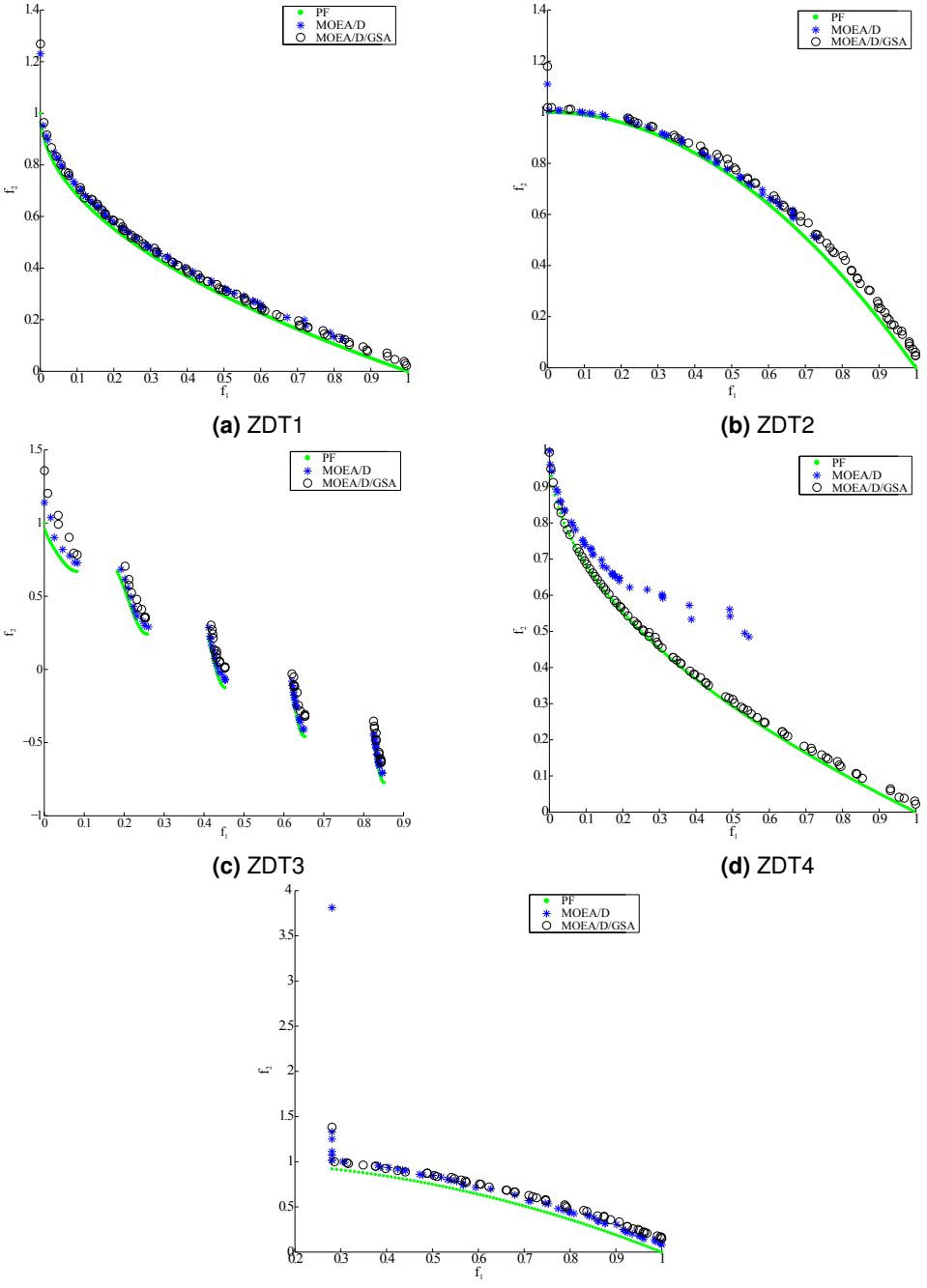
Table 3. Hypervolume indicator results on the ZDT and DTLZ test functions

	MOEA/D	MOEA/D/GSA	MOEA/D/NM
ZDT1	24.50301	**24.61340**	24.31119
(st. dev.)	(0.15101)	(0.01050)	(0.12874)
ZDT2	23.02988	**24.24493**	21.23483
(st. dev.)	(1.45125)	(0.01949)	(0.96625)
ZDT3	27.94969	**28.01895**	27.61912
(st. dev.)	(0.34197)	(0.04197)	(0.63348)
ZDT4	24.02487	24.48137	**23.82835**
(st. dev.)	(0.32408)	(0.17950)	(0.38275)
ZDT6	22.92276	**23.07460**	22.89177
(st. dev.)	(0.05367)	(0.05973)	(0.07056)
DTLZ1	120.84596	**120.85190**	120.84041
(st. dev.)	(0.07311)	(0.07338)	(0.08505)
DTLZ2	120.20962	120.20952	120.20952
(st. dev.)	(0.00030)	(0.00064)	(0.00064)
DTLZ3	116.12579	**118.57561**	115.44508
(st. dev.)	(5.08719)	(3.81988)	(5.18048)
DTLZ4	119.74700	119.74231	119.74801
(st. dev.)	(0.00817)	(0.02676)	(0.00491)
DTLZ5	114.48096	115.48724	**115.89698**
(st. dev.)	(0.58667)	(0.57578)	(0.82226)
DTLZ6	114.48096	115.48724	**115.89698**
(st. dev.)	(0.58667)	(0.57578)	(0.82226)
DTLZ7	**49.86711**	49.68336	49.84884
(st. dev.)	(0.05572)	(0.04674)	(0.03746)

Here, it is possible to observe that in most of the cases the GSA helps the algorithm to achieve a "better" approximation along the entire Pareto front.

4.3 IG-NSGA-II/GSA

The next set of experiments is prepared in order to demonstrate that the GSA can be used to handle constrained problems. Since we are replacing the local search method of the IG-NSGA-II we adopt all the parameters from such algorithm. First, we consider some of the function presented in [5]. The definitions of such functions can be found in Table 9. The maximization problems presented on the definitions were transformed into minimization problems. The constraints of the problems were



(a) ZDT1

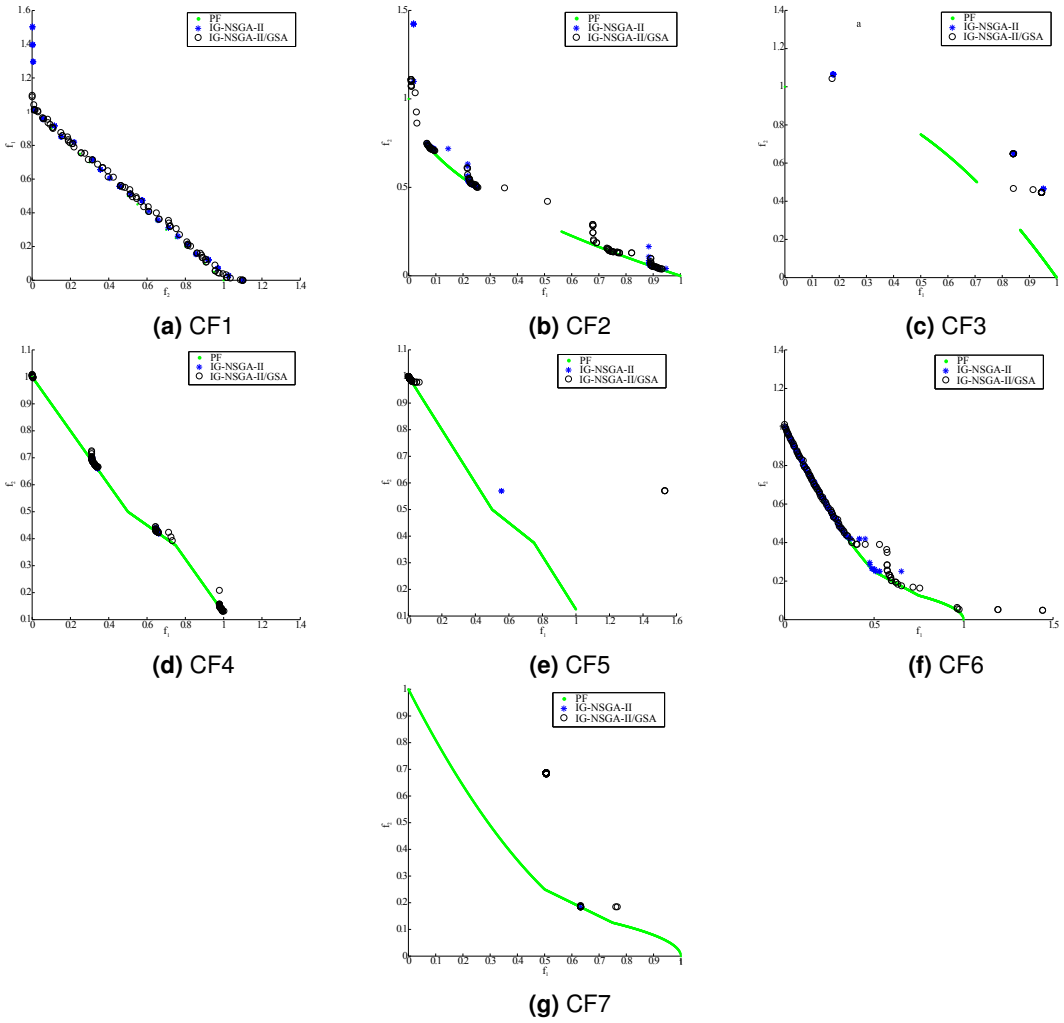
(b) ZDT2

(c) ZDT3

(d) ZDT4

(e) ZDT6

Fig. 8. Pareto fronts of ZDT problems



(a) CF1 **(b)** CF2 **(c)** CF3

(d) CF4 **(e)** CF5 **(f)** CF6

(g) CF7

Fig. 9. Pareto fronts of CF problems

also transformed into the form $a(x) \leq 0$. We stress that all constraints are relatively easy in the sense that the feasible set is not of a highly complex structure. Table 4 presents the values for the parameters for the memetic algorithm.

The results obtained in Table 5 show that the GSA can improve the results in most of the functions according to the Δ_2 indicator. Table 5 also presents the function evaluation used as stopping criteria. Moreover, the table also presents the nadir point used to compute the hypervolume indicator. It is important to mention that some indicator values obtained by the GSA outperformed significantly the standalone algorithm. Taking these results into consideration, now we are in position to confirm the efficiency of the GSA when it is used within a memetic strategy.

As a last experiment we use the constrained CF functions proposed in [43] those constraints can considered to be complex. We perform a similar comparison as in the previous method. We measure the Δ_2 and the hypervolume indicators at certain stage of the evolution (i.e. at $30,000$ and $50,000$ function evaluations). For the experiments we set the nadir point for hypervolume as $(5,5)^T \in \mathbb{R}^2$ and $(5,5,5)^T \in \mathbb{R}^3$. Table 7 presents

Table 4. Parameters for the IG-NSGA-II/GSA algorithm

Parameter	Description	Value
η_c	Distribution index for crossover	20
η_m	Distribution index for mutation	20
γ_1	Crossover probability	0.9
γ_2	Mutation probability	$1/n$
N	Number of individuals	100
r	Number of neighbors for GSA	5
γ_3	Frequency of the local search	3

the averaged results measured by the proposed indicators. The results are obtained taking in consideration only the feasible solutions obtained by the algorithms. By such reason the function CF10 is not in the statistical results since neither of the algorithms obtained feasible solutions.

Figure 9 presents some of the Pareto fronts obtained by the algorithms on the CF functions.

5 Conclusions and Future Work

In this paper, we have argued that the gradient subspace approximation (GSA) is a powerful tool for local search within memetic algorithms for the numerical treatment of multi-objective optimization problems (MOPs). The GSA utilizes existing neighborhood information I from a given candidate solution and is able to approximate the best approximation of the gradient within the subspace of the decision variable space that is defined by I. Thus, the computation of the search direction via GSA comes ideally for free for population based search algorithms such as evolutionary algorithms. The strengths of GSA within memetic algorithms for the treatment of scalar optimization problems have recently been reported in [33]. Here, we have discussed the GSA for the case that multiple objectives have to be considered concurrently. To this end, we have

— empirically shown that the existing neighborhood information within populations of multi-objective evolutionary algorithms is sufficient for the application of GSA,

— discussed how to approximate the Jacobian matrix at a given candidate solution x_0 via GSA,

— discussed how to adapt Laras search direction in case inequality constraints are at hand and how to choose the step size control,

— proposed two particular GSA-based memetic algorithms, and have presented numerical results on some selected benchmark problems. In both cases the application of the local search–that comes basically for free in terms of additional function calls–significantly improved the performance of the base algorithm. More precisely, the increase of the performance was significant for unconstrained problems and for the MOPs with relatively easy constraints. For complex constraints, no such significant performance improvements could be obtained so far.

In conclusion, this first study is very promising and will encourage us for future research in this direction. This will include the development of more sophisticated constraint handling techniques which are mandatory to extend the applicability of the algorithms to more complex problems. Further, a fine tuning and a more advanced interplay of local and global search elements will be helpful to increase the overall performance.

Finally, one interesting next step would be to use GSA-based local search within indicator based evolutionary algorithms as such methods naturally define a (population based) scalar optimization problem that is induced by the indicator.

Acknowledgements

The authors acknowledge funding from Conacyt project no. 285599 ”Toma de decisiones multiobjetivo para sistemas altamente complejos”.

References

1. **Beume, N., Naujoks, B., & Emmerich, M. (2007).** SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, Vol. 181, No. 3, pp. 1653 – 1669.

Table 5. Averaged indicator results for constrained problems

Function	Hypervolume		Δ_2		Max Eval.
	IG-NSGA-II	IG-NSGA-II/GSA	IG-NSGA-II	IG-NSGA-II/GSA	(Nadir point)
Belegundu	212.8721	**213.1354**	1.6844	**1.5900**	3,000
(std. dev.)	(0.3205)	(0.2261)	(0.1587)	(0.1030)	(12,12)
Binh(2)	10,294.0037	**10,300.6309**	0.7047	**0.6172**	3,000
(std. dev.)	(17.2207)	(9.8055)	(0.2812)	(0.1667)	(250,50)
Binh(4)	705.4772	**728.7873**	0.8863	**0.5061**	30,000
(std. dev.)	(12.2397)	(8.0167)	(0.1915)	(0.1239)	(5,7,5)
Obayashi	**22.0321**	21.9269	0.8084	**0.7792**	20,000
(std. dev.)	(0.7574)	(0.7103)	(0.2869)	(0.2772)	(5,5)
Osyczka	**59.8672**	59.5651	3.2946	**2.3881**	20,000
(std. dev.)	(1.4790)	(1.7873)	(1.9568)	(1.4040)	(30,30)
Osyczka(2)	12,780.7978	**13,701.1984**	47.8491	**30.8044**	30,000
(std. dev.)	(42.6364)	(11.0854)	(4.3380)	(1.0707)	(0,85)
Srinivas	212.8191	**213.1927**	1.4871	**1.3925**	3000
(std. dev.)	(0.3723)	(0.0714)	(0.2364)	(0.1769)	(250,50)
Tamaki	**124.3239**	124.3054	**0.1353**	0.1515	10,000
(std. dev.)	(0.0363)	(0.0252)	(0.0252)	(0.0366)	(5,5,5)
Tanaka	22.9022	**24.9672**	0.0672	**0.0468**	10,000
(std. dev.)	(0.7516)	(0.0249)	(0.0472)	(0.0100)	(5,5)
Viennet(4)	190.2843	**191.6098**	0.1015	**0.0978**	10,000
(std. dev.)	(0.6212)	(0.4552)	(0.0048)	(0.0060)	(8,-10,30)

2. **Bhuvana, J. & Aravindan, C. (2016).** Memetic algorithm with preferential local search using adaptive weights for multi-objective optimization problems. *Soft Computing*, Vol. 20, No. 4, pp. 1365–1388.

3. **Bosman, P. A. N. (2012).** On gradients and hybrid evolutionary algorithms for real-valued multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, Vol. 16, No. 1, pp. 51–69.

4. **Brown, M. & Smith, R. E. (2005).** Directed multi-objective optimization. *International Journal of Computers, Systems, and Signals*, Vol. 6, No. 1, pp. 3–17.

5. **Coello, C. A. C., Lamont, G. B., & Veldhuizen, D. A. V. (2007).** *Evolutionary Algorithms for Solving Multi-objective Problems*, volume 5. Springer.

6. **Das, I. & Dennis, J. E. (1998).** Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. *SIAM Journal on Optimization*, Vol. 8, No. 3, pp. 631–657.

7. **Deb, K. (2001).** *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley Interscience Series in S. Wiley.

8. **Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2002).** Scalable multi-objective optimization test problems. *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, IEEE, pp. 825–830.

9. **Dellnitz, M., Schütze, O., & Hestermeyer, T. (2005).** Covering Pareto sets by multilevel subdivision techniques. *Journal of Optimization Theory and Applications*, Vol. 124, No. 1, pp. 113–155.

10. **Drummond, L. M. G. & Svaiter, B. F. (2005).** A steepest descent method for vector optimization. *Journal of Computational and Applied Mathematics*, Vol. 175, pp. 395–414.

11. **Fernández, J., Schütze, O., Hernández, C., Sun, J.-Q., & Xiong, F.-R. (2016).** Parallel simple cell mapping for multi-objective optimization. *Engineering Optimization*, Vol. 48, No. 11, pp. 1845–1868.

12. **Hernández, C., Naranjani, Y., Sardahi, Y., Liang, W., Schütze, O., & Sun, J.-Q. (2013).** Simple cell mapping method for multi-objective optimal

Table 6. $W(x)$ indicator results on the ZDT and DTLZ test functions

	MOEA/D	MOEA/D/GSA	MOEA/D/NM
ZDT1	20.0472	**19.5837**	22.0835
(st. dev.)	(1.3782)	(0.0074)	(4.1524)
ZDT2	20.0472	**19.5837**	22.0835
(st. dev.)	(1.3782)	(0.0074)	(4.1524)
ZDT3	31.4010	31.3602	**31.0517**
(st. dev.)	(0.0518)	(0.0558)	(1.3529)
ZDT4	13.8381	**13.1125**	13.7468
(st. dev.)	(1.0074)	(0.0282)	(0.7020)
ZDT6	28.8086	**22.9416**	28.8137
(st. dev.)	(0.0168)	(0.0608)	(0.0143)
DTLZ1	7.9511	**7.9359**	8.28029
(st. dev.)	(0.0105)	(0.0259)	(0.0200)
DTLZ2	23.7744	23.7826	21.5239
(st. dev.)	(0.0040)	(0.0055)	(0.0002)
DTLZ3	24.6343	**24.2808**	24.6249
(st. dev.)	(0.5718)	(0.2980)	(6.4775)
DTLZ4	23.7954	23.8013	21.5240
(st. dev.)	(0.0054)	(0.0049)	(0.0006)
DTLZ5	47.3060	**47.3059**	47.3190
(st. dev.)	(0.0003)	(0.0002)	(0.0068)
DTLZ6	51.6431	**50.9752**	62.51037
(st. dev.)	(0.9658)	(1.0471)	(2.6750)
DTLZ7	268.9945	268.9987	325.7539
(st. dev.)	(0.0001)	(0.0001)	(0.1023)

feedback control design. *International Journal of Dynamics and Control*, Vol. 1, No. 3, pp. 231–238.

13. **Hernández, V. A. S., Schütze, O., Rudolph, G., & Trautmann, H. (2016).** The hypervolume based directed search method for multi-objective optimization problems. *Journal of Heuristics*, Vol. 22, No. 3, pp. 273–300.
14. **Hillermeier, C. (2001).** *Nonlinear Multiobjective Pptimization: a Generalized Homotopy Approach*, volume 135. Springer Science & Business Media.
15. **Ishibuchi, H., Yoshida, T., & Murata, T. (2003).** Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 2, pp. 204–223.
16. **Jahn, J. (2006).** Multiobjective search algorithm with subdivision technique. *Computational Optimization and Applications*, Vol. 35, No. 2, pp. 161–175.
17. **LaTorre, A., Muelas, S., & Peña, J.-M. (2011).** A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: A scalability test. *Soft Computing*, Vol. 15, No. 11, pp. 2187–2199.
18. **Lewis, R. M., Torczon, V., & Trosset, M. W. (2000).** Direct search methods: Then and now. *Journal of Computational and Applied Mathematics*, Vol. 124, No. 1, pp. 191–207.
19. **Liu, T., Gao, X., & Yuan, Q. (2016).** An improved gradient-based NSGA-II algorithm by a new chaotic map model. *Soft Computing*.
20. **López, A. L., Coello, C. A. C., & Schütze, O. (2010).** A painless gradient-assisted multi-objective memetic mechanism for solving continuous bi-objective optimization problems. *IEEE Congress on Evolutionary Computation*, IEEE, pp. 1–8.
21. **Lopez, E. M. & Coello, C. A. C. (2016).** A Parallel Multi-objective Memetic Algorithm Based on the IGD+ Indicator. *International Conference on Parallel Problem Solving from Nature*, Springer, pp. 473–482.
22. **Martín, A. & Schütze, O. (2017).** Pareto tracer: a predictor-corrector method for multi-objective optimization problems. Engineering Optimization (to appear).
23. **Martin, B., Goldsztejn, A., Granvilliers, L., & Jermann, C. (2014).** On continuation methods for non-linear bi-objective optimization: towards a certified interval-based approach. *Journal of Global Optimization*, Vol. 64, No. 1, pp. 1–14.
24. **Martin, B., Goldsztejn, A., Granvilliers, L., & Jermann, C. (2016).** On continuation methods for non-linear bi-objective optimization: towards a certified interval-based approach. *Journal of Global Optimization*, Vol. 64, No. 1, pp. 3–16.
25. **Martínez, S. Z. & Coello, C. A. C. (2012).** A direct local search mechanism for decomposition-based multi-objective evolutionary algorithms. *2012 IEEE Congress on Evolutionary Computation*, pp. 1–8.
26. **Miettinen, K. (2012).** *Nonlinear Multiobjective Optimization*, volume 12. Springer Science & Business Media.
27. **Naranjani, Y., Hernández, C., Xiong, F.-R., Schütze, O., & Sun, J.-Q. (2016).** A hybrid method of evolutionary algorithm and simple cell mapping for multi-objective optimization problems. *International Journal of Dynamics and Control*, pp. 1–13.

Table 7. Averaged indicator results for CF problems

Function	Hypervolume		Δ_2	
	IG-NSGA-II	**IG-NSGA-II/GSA**	**IG-NSGA-II**	**IG-NSGA-II/GSA**
CF1	22.6991	**22.7063**	0.3488	**0.2809**
(St. Dev.)	(0.0108)	(0.0160)	(0.1134)	(0.0457)
CF2	23.9966	**24.1846**	3.7094	**2.6267**
(St. Dev.)	(0.2304)	(0.1409)	(0.8217)	(1.1457)
CF3	21.7779	**21.9743**	8.8626	**8.6404**
(St. Dev.)	(0.7876)	(0.8831)	(1.3550)	(1.7076)
CF4	22.5691	**22.9710**	4.0301	**3.4467**
(St. Dev.)	(0.6465)	(0.5158)	(0.8088)	(0.7596)
CF5	**20.8112**	20.6415	**10.0656**	11.6749
(St. Dev.)	(1.0878)	(1.1346)	(2.1827)	(3.3711)
CF6	23.6467	**24.0427**	3.6043	**1.7815**
(St. Dev.)	(0.4994)	(0.2493)	(1.8984)	(0.5572)
CF7	20.9140	**21.3019**	16.1164	**13.6514**
(St. Dev.)	(0.6613)	(0.9006)	(5.2138)	(6.0834)
CF8	153.2641	**154.5096**	55.8651	**47.8166**
(St. Dev.)	(3.3233)	(3.2633)	(7.8279)	(10.3452)
CF9	123.3425	**123.6872**	15.1201	**14.6615**
(St. Dev.)	(0.7494)	(0.3098)	(2.8301)	(1.9893)

28. Nocedal, J. & Wright, S. (2006). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer.

29. Pareto, V. (1896). *Cours D'Economie Politique*. F. Rouge, Switzerland.

30. Pereyra, V., Saunders, M., & Castillo, J. (2013). Equispaced Pareto front construction for constrained bi-objective optimization. *Math Comput Model*, Vol. 57, No. 9-10, pp. 2122–2131.

31. Rakowska, J., Haftka, R. T., & Watson, L. T. (1993). Multi-objective control-structure optimization via homotopy methods. *SIAM Journal on Optimization*, Vol. 3, No. 3, pp. 654–667.

32. Ringkamp, M., Ober-Blöbaum, S., Dellnitz, M., & Schütze, O. (2012). Handling high dimensional problems with multi-objective continuation methods via successive approximation of the tangent space. *Engineering Optimization*, Vol. 44, No. 6, pp. 1117–1146.

33. Schütze, O., Alvarado, S., Segura, C., & Landa, R. (2016). Gradient subspace approximation: A direct search method for memetic computing. *Soft Computing*, pp. 1–20.

34. Schütze, O., Dell'Aere, A., & Dellnitz, M. (2005). On continuation methods for the numerical treatment of multi-objective optimization problems. *Dagstuhl Seminar Proceedings*, Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

35. Schütze, O., Esquivel, X., Lara, A., & Coello, C. A. C. (2012). Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, Vol. 16, No. 4, pp. 504–522.

36. Schütze, O., Lara, A., & Coello, C. A. (2011). On the influence of the number of objectives on the hardness of a multiobjective optimization problem. *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 4, pp. 444–455.

37. Schütze, O., Lara, A., Sanchez, G., & Coello, C. A. (2010). HCS: A new local search strategy for memetic multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, Vol. 14, No. 1, pp. 112–132.

38. Schütze, O., Martín, A., Lara, A., Alvaradoa, S., Salinas, E., & Coello, C. A. C. (2016). The directed search method for multiobjective memetic algorithms. *Journal of Computational Optimization and Applications*, Vol. 63, pp. 305–332.

39. Shukla, P. K. (2007). *On Gradient Based Local Search Methods in Unconstrained Evolutionary*

Table 8. Unconstrained Problems Definition

Function Name	Definition	
ZDT1	$f_1(x) = x_1$ $f_2(x) = g(x)\left(1-\sqrt{\frac{f_1(x)}{g(x)}}\right)$ $g(x) = 1+\frac{9}{n-1}\sum_{i=2}^{n} x_i$	$0 \leq x_i \leq 1,\ i=1,\ldots,n$ $n=30$ $k=2$
ZDT2	$f_1(x) = x_1$ $f_2(x) = g(x)\left(1-\left(\frac{f_1(x)}{g(x)}\right)^2\right)$ $g(x) = 1+\frac{9}{n-1}\sum_{i=2}^{n} x_i$	$0 \leq x_i \leq 1,\ i=1,\ldots,n$ $n=30$ $k=2$
ZDT3	$f_1(x) = x_1$ $f_2(x) = 1-\sqrt{\frac{f_1(x)}{g(x)}}-\left(\frac{f_1(x)}{g(x)}\right) sin(10\pi f_1(x))$ $g(x) = 1+\frac{9}{n-1}\sum_{i=2}^{n} x_i$	$0 \leq x_i \leq 1,\ i=1,\ldots,n$ $n=30$ $k=2$
ZDT4	$f_1(x) = x_1$ $f_2(x) = 1-g(x)\left(\frac{f_1(x)}{g(x)}\right)^2$ $g(x) = 1+10(n-1)+\sum_{i=2}^{n}(x_i^2-10cos(4\pi x_i))$	$-5 \leq x_i \leq 5,\ i=1,\ldots,n$ $n=10$ $k=2$
ZDT6	$f_1(x) = 1-\left(e^{-4x_1}\right) sin(6\pi x_1)$ $f_2(x) = 1-g(x)\left(\frac{f_1(x)}{g(x)}\right)^2$ $g(x) = 1+9\left(\frac{\sum_{i=2}^{n} x_i}{n-1}\right)^{0.25}$	$0 \leq x_i \leq 1,\ i=1,\ldots,n$ $n=10$ $k=2$
DTLZ1	$f_1(x) = \frac{1}{2}x_1(1+g(x))$ $f_2(x) = \frac{1}{2}(1-x_1)(1+g(x))$ $g(x) = 100\left(5+\sum_{i=2}^{n}\left((x_i-0.5)^2-cos(20\pi(x_i-0.5))\right)\right)$	$0 \leq x_i \leq 1,\ i=1,\ldots,n$ $n=7$ $k=2$
DTLZ2	$f_1(x) = (1+g(x))cos\left(\frac{x_1\pi}{2}\right)$ $f_2(x) = (1+g(x))sin\left(\frac{x_1\pi}{2}\right)$ $g(x) = \sum_{i=2}^{n}(x_i-0.5)$	$0 \leq x_i \leq 1,\ i=1,\ldots,n$ $n=11$ $k=2$
DTLZ3	$f_1(x) = (1+g(x))cos\left(\frac{x_1\pi}{2}\right)$ $f_2(x) = (1+g(x))sin\left(\frac{x_1\pi}{2}\right)$ $g(x) = 100\left(10+\sum_{i=2}^{n}\left((x_i-0.5)^2-cos(20\pi(x_i-0.5))\right)\right)$	$0 \leq x_i \leq 1,\ i=1,\ldots,n$ $n=11$ $k=2$
DTLZ4	$f_1(x) = (1+g(x))cos\left(\frac{x_1^{100}\pi}{2}\right)$ $f_2(x) = (1+g(x))sin\left(\frac{x_1^{100}\pi}{2}\right)$ $g(x) = \sum_{i=2}^{n}(x_i-0.5)$	$0 \leq x_i \leq 1,\ i=1,\ldots,n$ $n=11$ $k=2$
DTLZ5	$f_1(x) = (1+g(x))cos\left(\frac{x_1\pi}{2}\right) cos\left(\left(\frac{\pi}{4(1+g(x))}\right)(1+2x_2g(x))\right)$ $f_2(x) = (1+g(x))cos\left(\frac{x_1\pi}{2}\right) sin\left(\left(\frac{\pi}{4(1+g(x))}\right)(1+2x_2g(x))\right)$ $f_3(x) = (1+g(x))sin\left(\frac{x_1\pi}{2}\right)$ $g(x) = \sum_{i=3}^{n}(x_i-0.5)^2$	$0 \leq x_i \leq 1,\ i=1,\ldots,n$ $n=10$ $k=3$
DTLZ6	$f_1(x) = (1+g(x))cos\left(\frac{x_1\pi}{2}\right) cos\left(\left(\frac{\pi}{4(1+g(x))}\right)(1+2x_2g(x))\right)$ $f_2(x) = (1+g(x))cos\left(\frac{x_1\pi}{2}\right) sin\left(\left(\frac{\pi}{4(1+g(x))}\right)(1+2x_2g(x))\right)$ $f_3(x) = (1+g(x))sin\left(\frac{x_1\pi}{2}\right)$ $g(x) = \sum_{i=3}^{n}(x_i-0.5)^{0.1}$	$0 \leq x_i \leq 1,\ i=1,\ldots,n$ $n=12$ $k=3$
DTLZ7	$f_1(x) = x_1$ $f_2(x) = x_2$ $f_3(x) = (1+g(x))h(x)$ $g(x) = 1+\frac{9}{8}\sum_{i=3}^{n} x_i \qquad h(x) = k-\frac{\sum_{i=1}^{k-1}(x_i sin(1+3\pi x_i))}{1+g(x)}$	$0 \leq x_i \leq 1,\ i=1,\ldots,n$ $n=10$ $k=3$
CONV	$f_1(x) = (x_1-1)^4+(x_2-1)^2$ $f_2(x) = (x_1-1)^2+(x_2-1)^2$	$-5 \leq x_1 \leq 5,\ -5 \leq x_2 \leq 5$ $n=2,\ k=2$
KURSAWE	$f_1(x) = \sum_{i=1}^{2}\left(-10e^{-0.2\sqrt{x_i^2+x_{i+1}^2}}\right)$ $f_2(x) = \sum_{i=1}^{3}\left(\lvert x_i\rvert^{0.8}+5sin\left(x_i^3\right)\right)$	$-5 \leq x_i \leq 5,\ i=1,\ldots,n$ $n=3,\ k=2$
DTLZ3 (3)	$f_1(x) = (1+g(x))cos\left(\frac{x_1\pi}{2}\right) cos\left(\frac{x_2\pi}{2}\right)$ $f_2(x) = (1+g(x))cos\left(\frac{x_1\pi}{2}\right) sin\left(\frac{x_2\pi}{2}\right)$ $f_3(x) = (1+g(x))sin\left(\frac{x_1\pi}{2}\right)$ $g(x) = 100\left(10+\sum_{i=2}^{n}\left((x_i-0.5)^2-cos(20\pi(x_i-0.5))\right)\right)$	$0 \leq x_i \leq 1,\ i=1,\ldots,n$ $n=12$ $k=3$

Table 9. Constrained Problems Definition

Function n	Definition	Constraints
Belegundu $n = 2$	$f_1(x) = -2x_1 + x_2$ $f_2(x) = 2x_1 + x_2$	$-x_1 + x_2 - 1 \leq 0$ $x_1 + x_2 - 7 \leq 0$ $0 \leq x_1 \leq 5$ $0 \leq x_2 \leq 3$
Binh(2) $n = 2$	$f_1(x) = 4x_1^2 + 4x_2^2$ $f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2$	$(x_1 - 5)^2 + x_2^2 - 25 \leq 0$ $-(x_1 - 8)^2 - (x_2 + 3)^3 + 7.7 \leq 0$ $0 \leq x_1 \leq 5$ $0 \leq x_2 \leq 3$
Binh(4) $n = 2$	$f_1(x) = 1.5 - x_1(1 - x_2)$ $f_2(x) = 2.25 - x_1(1 - x_2^2)$ $f_3(x) = 2.625 - x_1(1 - x_2^3)$	$-x_1^2 - (x_2 - 0.5)^2 + 9 \leq 0$ $(x_1 - 1)^2 + (x_2 - 0.5)^2 - 6.25 \leq 0$ $-10 \leq x_1 \leq 10$ $-10 \leq x_2 \leq 10$
Obayashi $n = 2$	Maximize $f_1(x) = x_1$ $f_2(x) = x_2$	$x_1^2 + x_2^2 \leq 1$ $0 \leq x_1 \leq 1$ $0 \leq x_2 \leq 1$
Osyczka $n = 2$	$f_1(x) = x_1 + x_2^2$ $f_2(x) = x_1^2 + x_2$	$12 - x_1 - x_2 \geq 0$ $x_1^2 + 10x_1 - x_2^2 + 16x_2 - 80 \geq 0$ $2 \leq x_1 \leq 7$ $5 \leq x_2 \leq 10$
Osyczka 2 $n = 6$	$f_1(x) = -(25(x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 1)^2 + (x_4 - 4)^2 + (x_5 - 1)^2)$ $f_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2$	$x_1 + x_2 - 2 \geq 0$ $6 - x_1 - x_2 \geq 0$ $2 - x_2 + x_1 \geq 0$ $2 - x_1 + 3x_2 \geq 0$ $4 - (x_3 - 3)^2 - x_4 \geq 0$ $(x_5 - 3)^3 + x_6 - 4 \geq 0$ $0 \leq x_1, x_2, x_6 \leq 10$ $1 \leq x_3, x_5 \leq 5$ $0 \leq x_4 \leq 6$
Srinivas $n = 2$	$f_1(x) = (x_1 - 2)^2 + (x_2 - 1)^2 + 2$ $f_2(x) = 9x_1 - (x_2 - 1)^2$	$x_1^2 + x_2^2 - 225 \leq 0$ $x_1 - 3x_2 + 10 \leq 0$ $-20 \leq x_1, x_2 \leq 20$
Tamaki $n = 3$	Maximize $f_1(x) = x_1$ $f_2(x) = x_2$ $f_3(x) = x_3$	$x_1^2 + x_2^2 + x_3 \leq 1$ $x_1, x_2, x_3 \geq 0$
Tanaka $n = 2$	$f_1(x) = x_1$ $f_2(x) = x_2$	$-x_1^2 - x_2^2 + 1 + 0.1cos\left(16 \arctan(\frac{x_1}{x_2})\right) \leq 0$ $(x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5$ $0 < x_1, x_2 \leq \pi$
Viennet (4) $n = 2$	$f_1(x) = \frac{(x_1-2)^2}{2} + \frac{(x_2+1)^2}{13} + 3$ $f_2(x) = \frac{(x_1+x_2-3)^2}{175} + \frac{(2x_2-x_1)^2}{17} - 13$ $f_3(x) = \frac{(3x_1-2x_2+4)^2}{8} + \frac{(x_1-x_2+1)^2}{27} + 15$	$-4x_1 - x_2 + 4 \geq 0$ $x_1 + 1 \geq 0$ $x_2 - x_1 + 2 \geq 0$ $-4 \leq x_1, x_2 \leq 4$

Multi-objective Optimization. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 96–110.

40. **Sindhya, K., Miettinen, K., & Deb, K. (2013).** A hybrid framework for evolutionary multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, Vol. 17, No. 4, pp. 495–511.
41. **Yu, G., Chai, T., & Luo, X. (2011).** Multiobjective production planning optimization using hybrid evolutionary algorithms for mineral processing. *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 4, pp. 487–514.
42. **Zhang, Q. & Li, H. (2007).** MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, Vol. 11, No. 6, pp. 712–731.
43. **Zhang, Q., Zhou, A., Zhao, S., Suganthan, P. N., Liu, W., & Tiwari, S. (2008).** Multiobjective optimization test instances for the CEC 2009 special session and competition. *University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report*, Vol. 264.
44. **Zitzler, E. & Thiele, L. (1999).** Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 4, pp. 257–271.

Article received on 01/02/2017; accepted on 06/09/2017.
Corresponding author is Sergio Alvarado.