SPIDER: decomposition and path-relinking based algorithm for bi-objective optimization problems

Nassime Aslimani, El-Ghazali Talbi, Rachid Ellaia

Journal of Global Optimization manuscript No.
(will be inserted by the editor)

SPIDER: decomposition and path-relinking based algorithm for bi-objective optimization problems

N. Aslimani*, R. Ellaia**, e-G. Talbi*

Received: date / Accepted: date

Abstract This paper proposes an original bi-objective optimization approach around the key feature of local conservation of the Pareto stationnarity along the gradient axes (LCPS).

The proposed algorithm consists of two steps. The decomposition step starts with the anchor points, generate N evenly points on the axes relating the anchor points to the utopia point. Then, the corresponding nearest reference points on the Pareto front are generated. In the path-relinking step, we carry out a path-relinking in the objective space, between each pair of Pareto solutions, following the best direction among the gradients axes.

The SPIDER algorithm largely outperforms state-of-the-art and popular evolutionary algorithms both in terms of the quality of the obtained Pareto fronts (convergence, cardinality, diversity) and the search time.

Keywords Pareto front · Pareto stationnarity · Bi-objective optimization · Gradient methods · Anchor points · Decomposition · Path-relinking

1 Introduction

Many real world problems require optimizing multiple conflicting objectives. Pareto optimality is generally used in the context of multi-objective optimization problems (MOPs). Indeed, while single objective optimization problems involves a unique optimal solution, MOPs present a set of compromised solutions, known as the Pareto optimal set [18]. These solutions are optimal in

* University of Lille, France. E-mail: nassime.aslimani@inria.fr, el-ghazali.talbi@univ-lille.fr
** EMI, Mohammed V University of Rabat, Morocco. E-mail: ellaia@emi.ac.ma

the sense that no single objective can be improved without decreasing at least one of the others. Without loss of generality, we assume that all objectives are to be minimized, then we consider a MOP of the form:

$$\min_{X \in S} \mathbf{F}(X) = (f_1(X), \cdots, f_m(X))^{\mathrm{T}} \tag{1}$$

where:

$f_k : \mathbb{R}^n \longrightarrow \mathbb{R}$, for $k \in \{1, \cdots, m\}$, denotes the objective functions, S is the design space: $S = \prod_{i=1}^{n} [l_i, u_i]$, X is the decision vector with n decision variables: $X = (x_1, \cdots, x_n) \in \mathbb{R}^n$,.

For the convenience of later discussion, we introduce some basic concepts:

Definition 1. *Pareto dominance*: let $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_n)$ be decision vectors (solutions). Solution X is said to dominate solution Y, denoted as $X \preceq Y$, if and only if :

$$\forall i \in [1, m] \ : \ f_i(X) \leq f_i(Y)) \wedge (\exists j \in [1, m] \ : \ f_j(X) < f_j(Y) \tag{2}$$

Definition 2. *Pareto optimal solution*: a solution X is Pareto optimal if it is not dominated by any other solution which means there is no other solution $Y \in S$ such that $Y \preceq X$.

Definition 3. *Pareto Set & Pareto front*: the set of all Pareto optimal solutions is called Pareto set (PS). The corresponding set of Pareto optimal objective vectors is called Pareto front (PF).

The Pareto front of continuous MOPs involving m objectives is, under mild smoothness conditions, an $(m-1)$-dimensional piece-wise-continuous manifold.

The main goal in solving MOPs is to find a "good" approximation of the Pareto front in terms of convergence, cardinality and diversity. In the last two decades many metaheuristics (e.g. evolutionary algorithms, swarm intelligence, local search) have been designed for solving MOPs [3][25][29]. One of the most frequently involved concept in continuous optimization is the gradient. The reason is that the gradient at any point of the design space indicate the direction along which the function decreases the most. This is the key principle underpinning a gradient based method. Indeed, in gradient based methods, the search operation is performed iteratively along the descent directions, that are built using local gradient information of the objective function with respect to changes in the design variables, so as to approximate the optima of the objective function.

Gradient based methods are suitable for solving single-objective optimization problems, with or without constraints. However, for multi-objective optimization, this is less well established. In order to solve MOPs using gradient based methods, a popular strategy is the weighted sum formulation (WSF), which consists in combining all the objectives into a single objective using a linear agregation method. Multiple solutions can be obtained by varying the weight coefficients among the objective functions to hopefully obtain different Pareto solutions. Many other scalarization functions have been used in the literature, such as the normal boundary intersection (NBI) [5], normal constraint method (NC) [17], physical programming method (PP) [16], goal programming (GP) [7], ϵ-constraint method [10], and the directed search domain (DSD) [22, 26]. Another gradient based strategies are used in memetic algorithms in which we hybridize multi-objective evolutionary algorithms (MOEAs) with local search strategies where gradient information is used to built a Pareto descent direction. For instance, Harada et al.[11] proposed a new gradient based local search method called the Pareto Descent Method, based on random selection of search directions among Pareto descents. Kim et al.[13] presented a directional operator to further enhance convergence of any MOEAs by introducing a local gradient search method to multi-objective global search algorithms. Recently, Lara et al.[14] compared hybrid methods with a new local search strategy without explicit gradient information and showed that using the gradient information was beneficial. Many other works propose pure gradient based methods for MOPs. In [8][9][21], a common descent direction is computed along all objectives. For instance, the multiple gradient descent algorithm was developed as an extension of the steepest descent method to MOPs [8]. Bosman have presented an analytical description of the entire set of descent directions, that he integrated in a new gradient based method for MOPs named CORL [1].

In this paper, we propose a new decomposition and path-relinking gradient based original approach (SPIDER) for solving bi-objective optimization problems. The proposed algorithm consists of two steps: decomposition and path-relinking. The decomposition step starts with the anchor points, generates N evenly reference points on the axes relating the anchor points to the utopia point. The reference points, representing the nearest points relating these points to the Pareto front, are then generated.

The path-relinking step is built around the feature of the local conservation of the Pareto stationary along the gradient axes (LCPS). This feature can be exploited in practice to generate a dynamic sliding on the Pareto front.

We carry out a path relinking between each pair of Pareto solutions in the objective space, following the best direction among the gradients axes. Two local searches operate in two opposite senses in a way that a first local search named D-Slide (Direct Slide) is designed to progress by increasing one chosen objective, whereas the other named R-Slide (Reverse Slide Process) is designed to return back. Besides, our SPIDER approach introduces some key ideas to overcome the relatively high cost generated by the gradient procedure computations.

Compared to existing multi-objective metaheuristics, the main characteristics of the proposed SPIDER algorithm are the following:

- **Fast and accurate convergence**: in most of multi-objective metaheuristics (e.g. evolutionary algorithms), the initial solutions are generated randomly [19]. Then, some stochastic operators are applied, (e.g. mutation, crossover) to generate new solutions. Based on the anchor points, our approach starts from a set of Pareto reference solutions. Once the decomposition of the objective space is done, the path-relinking step is *deterministic*.
- **Very low complexity of archiving**: archiving strategies in multi-objective metaheuristics are in general of high complexity (e.g. clustering procedure in SPEA, density-based procedure in NSGA-II and PESA-II). One has also to define the maximum size of the archive. Computational efficiency of non-dominance verification and sorting in Pareto fronts is in the order of $O(m.k.n^2)$, where n is the number of solutions in the Pareto front, k is the number of iterations, and m is the number of objectives [6]. Our approach is almost free-complexity in terms of archiving. The whole complexity of archiving is $O(m.n)$. Moreover, there is no need to limit the number of solutions in the archive. A dense and regular Pareto front is obtained.
- **Parallel independent decomposition of the objective space**: in most of the proposed decompositions strategies of the literature (e.g. MOEA/D [28]), the generated sub-problems are not independent, and correspond to single objective optimization problems using some scalarization strategies (e.g. weighted metrics, Tchebycheff) [18][20]. There is a need of cooperation in solving the sub-problems. In the SPIDER approach, all the generated sub-problems are independent. A parallel scalable implementation of the approach is straightforward. Moreover, our path-relinking procedure can be used as an intensification for any approximation found by a metaheuristic, with almost a free-complexity. The time consuming

part of SPIDER is the decomposition step in which we have to generate the reference points.

– **No need to manage diversity during search**: diversity management in multi-objective search is handled by complex procedures based on density estimation procedures (e.g. kernel methods such as fitness sharing, nearest-neighbor techniques such as crowding, and histograms) [23]. In the SPIDER approach, diversity is ensured by the decomposition step. There is no need to manage diversity in the path-relinking search procedure. We never generate solutions in the same region of the objective space.
– **Suitability for interactive optimization**: preferences in interactive optimization are generally defined in the objective space [2][15]. The SPIDER approach can handle efficiently those preferences to focus the search into a reduced part of the objective space. This process can be iterated to focus the search in more and more reduced sub-regions of the objective space.

The paper is organized as follows. Section 2 defines the local conservation of the Pareto stationarity along the gradient axes (LCPS). To the best of our knowledge, this is the first time the LCPS feature is introduced and proved. This feature will be used in the path-relinking step of the SPIDER algorithm. In section 3 we present the SPIDER algorithm by detailing the two main steps of the approach: the decomposition of the objective space, and the path-relinking in the objective space of the set of pairwise Pareto solutions. In section 4, the experimental settings and computational results against competing methods are detailed and analyzed. Finally, the section 5 enumerates some perspectives for the proposed SPIDER approach.

2 Local conservation of the Pareto stationarity along the gradient axes

In this section we describe some original and important properties of the gradient in the objective space. These properties will be fundamental in the development of the SPIDER path relinking step. First we recall an important theorem giving a necessary condition for optimality:

Theorem 1. *Karush-Kuhn-Tucker (KKT) condition* [12][27]: consider X a Pareto optimal solution. Then X is Pareto stationary, which means:

$$\exists \alpha_1, \alpha_2, \cdots, \alpha_m \geqslant 0 \text{ such that, } \sum_{i=1}^{m} \alpha_i = 1 \text{ and } \sum_{i=1}^{m} \alpha_i \nabla f_i(X) = 0.$$

The key idea of our SPIDER approach relies on the local conservation of the Pareto stationarity along the gradients axes of the different objectives (LCPS), feature that we have been formulated and successfully proved within the following theorem:

Theorem 2. *Local conservation of the Pareto stationarity (LCPS)*: let F be a smooth function. If X is a Pareto stationary point, for λ small enough, then the points $X + \lambda\nabla f_j(X)$, $j = 1, \cdots, m$, are also Pareto stationary. In other words:

$$\exists \alpha_1, \alpha_2, \cdots, \alpha_m \geqslant 0 \text{ such that, } \sum_{i=1}^{m} \alpha_i = 1 \text{ and } \sum_{i=1}^{m} \alpha_i \nabla f_i(X) = 0,$$

$$\Longrightarrow \forall j \in [1, m] \quad , \sum_{i=1}^{m} \alpha_i \nabla f_i(X + \lambda \nabla f_j(X)) = o(\lambda).$$

Proof:

Let X a Pareto stationary point and let $\alpha_1, \alpha_2, \cdots, \alpha_m \geqslant 0$ such that:

$$\sum_{i=1}^{m} \alpha_i = 1 \text{ and } \sum_{i=1}^{m} \alpha_i \nabla f_i(X) = 0. \tag{3}$$

Let $(i, j) \in [\![1, m]\!]^2$. By applying the Taylor formula to the function ∇f_i

$$\nabla f_i(X + \lambda \nabla f_j(X)) = \nabla f_i(X) + \lambda \nabla^2 f_i(X)(\nabla f_j(X))^T + o(\lambda) \tag{4}$$

By multiplying by α_i, equation (4) becomes:

$$\alpha_i \nabla f_i(X + \lambda \nabla f_j(X)) = \alpha_i \nabla f_i(X) + \lambda \alpha_i \nabla^2 f_i(X)(\nabla f_j(X))^T + o(\lambda). \tag{5}$$

On other hand equations (3) give:

$$\alpha_i \nabla f_i(X) = -\sum_{k \neq i} \alpha_k \nabla f_k(X), \text{ and then, } \alpha_i \nabla^2 f_i(X) = -\sum_{k \neq i} \alpha_k \nabla^2 f_k(X).$$

Thus, equation (5) becomes:

$$\begin{aligned} \alpha_i \nabla f_i(X + \lambda \nabla f_j(X)) &= -\sum_{k \neq i} \alpha_k \nabla f_k(X) - \lambda \Big(\sum_{k \neq i} \alpha_k \nabla^2 f_k(X) \Big) (\nabla f_j(X))^T + o(\lambda) \\ &= -\sum_{k \neq i} \Big(\alpha_k \nabla f_k(X) + \lambda \alpha_k \nabla^2 f_k(X)(\nabla f_j(X))^T \Big) + o(\lambda) \\ \alpha_i \nabla f_i(X + \lambda \nabla f_j(X)) &= -\sum_{k \neq i} \alpha_k \nabla f_k(X + \lambda \nabla f_j(X)) + o(\lambda). \end{aligned} \tag{6}$$

Thus we get:

$$\sum_{k=1}^{m} \alpha_k \nabla f_k(X + \lambda \nabla f_j(X)) = o(\lambda). \tag{7}$$

Remark 1. (Geometric illustration)
In the case of two objectives, we have a geometric illustration of the local Pareto stationarity feature. Indeed, if we move from X in the direction of $\nabla f_1(X)$ with a small enough step λ:

$$X_d^+ = X + \lambda \nabla f_1(X), \qquad X_d^- = X - \lambda \nabla f_1(X).$$

With the linear approximation formula, we get:

$$F(X_d^+) - F(X) \simeq \lambda \nabla F(X)(\nabla f_1(X))^T \quad \text{and} \quad F(X_d^-) - F(X) \simeq -\lambda \nabla F(X)(\nabla f_1(X))^T.$$

Thus, for λ small enough:

$$F(X_d^-) - F(X) \simeq -(F(X_d^+) - F(X)), \quad \text{that } is, \quad \overrightarrow{AP} \simeq -\overrightarrow{AQ},$$

where A, P, Q represent X, X_d^+, X_d^- in the objective space.

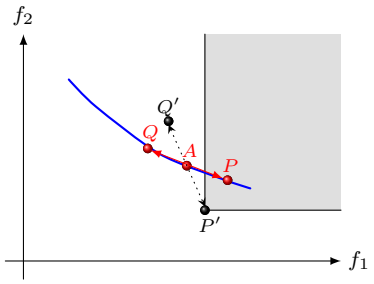


***Fig.* 1** Illustration of a small displacement along the gradient from a Pareto point.

We observe that for small moves along $\nabla f_1(X)$, the two corresponding vectors of displacement $\overrightarrow{AP}$ and $\overrightarrow{AQ}$ must be tangent to the Pareto set in A, as illustrated in Fig.1. In fact, otherwise the alternatives are $\overrightarrow{AP'}$ and $\overrightarrow{AQ'}$ and one of the two generated points (P' in this case) improve the Pareto set, and that is in contradiction with the nature of the Pareto set.

This feature can be exploited to slide on the Pareto front by moving from a Pareto solution along the two opposite gradients of the involved objectives. But this requires some practical precautions. In fact, in practice, there is no guarantee to keep moving on the Pareto front even with a suitable sequence of step-size for the conservation of the Pareto stationarity starting with a Pareto solution. In fact dynamic of the sliding dynamic may take off from the PF particularly in a singularity point. Also, the LCPS feature involves

eventually local Pareto optimality rather than global Pareto optimality. An other difficulty is the breaking of the Pareto stationarity in case of a discontinue Pareto set.

All these considerations bring us to devise some efficient strategies besides this feature in order to make it operational into our SPIDER approach.

3 The SPIDER algorithm

The SPIDER algorithm consists of two steps:

- **Regular decomposition of the objective space:** this step allows to sample the Pareto front in a regular way. This process generates a set of N_p reference points $X_k, k = 1, \cdots, N_p$.
- **Adaptive path relinking:** this step represents the central idea of the proposed approach which is built on the local conservation feature of the Pareto stationarity. This feature indicates that following the direction of the gradient of the objectives on the Pareto set, produce normally a slide on the Pareto front.

3.1 Regular decomposition of the objective space

The goal of this decomposition step is sampling an approximation of the Pareto front involving N_p reference points, by performing the following steps:

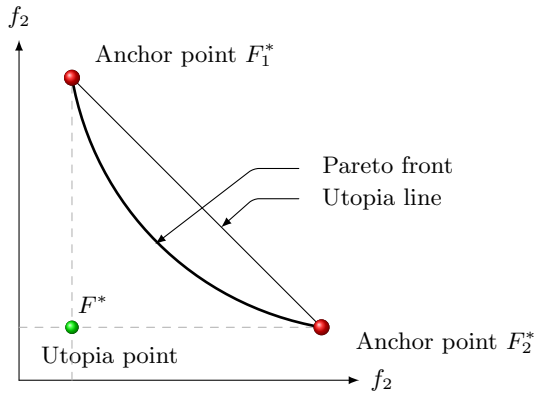
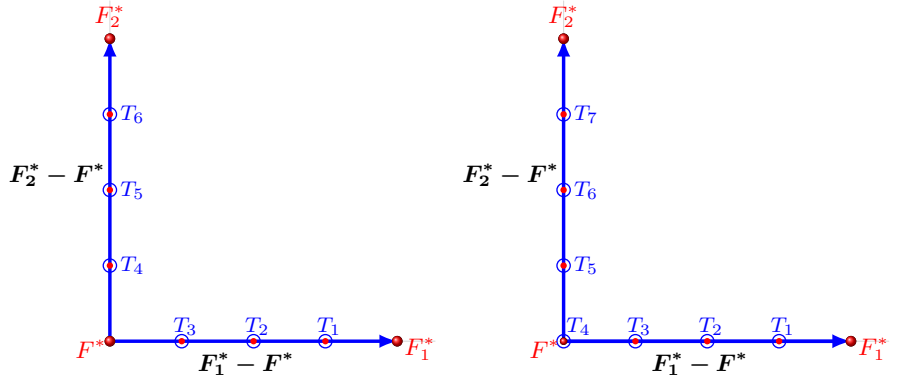


***Fig.* 2** Illustration of anchor points, utopia point and utopia Line.

– **Determination of the anchor points:** first, we determine the anchor points X_1^*, X_2^*, which represent the solutions of single-objective problems $X_i^* = \underset{X \in S,\ l_i \leq x_i \leq u_i}{Argmin} f_i(X)$ (see Fig. 2).
– **Generation of target points:** Let $X_1 = X_1^*$ and $X_{N_p} = X_2^*$, $F_1^* = F(X_1^*)$, $F_2^* = F(X_2^*)$, $F^* = (f_1(X_1^*), f_2(X_2^*)$; This step generates in a uniform way $N_t = N_p - 2$ targets points $(T_i)_{1 \leqslant i \leqslant N_t}$, (See Fig. 3) defined as:

$$T_i = F^* + \alpha_i(F_1^* - F^*) + \beta_i(F_1^* - F^*), \tag{8}$$

$$\text{with } \alpha_i = \frac{\max(1+n-i,0)}{1+n}, \quad \beta_i = \frac{\max(i+n-N_t,0)}{1+n}, \quad n = \lfloor N_t/2 \rfloor$$



(a) Selection of even target points: $N_t = 6$ *(b) Selection of odd target points: $N_t = 7$*

***Fig.* 3** Illustration of target points selection with $N_p = 8$ and $N_p = 9$.

– **Generation of the reference solutions:** this step will generates the corresponding regular set of reference solutions in the objective space $(X_i)_{1<j<N_p}$ by solving for each $i \in \{2, \cdots, N-1\}$, the single objective problem (Fig.4):

$$X_j = \underset{X \in S,\ l_i \leq x_i \leq u_i}{Argmin} \| F(X) - T_{j-1} \|^2 \tag{9}$$

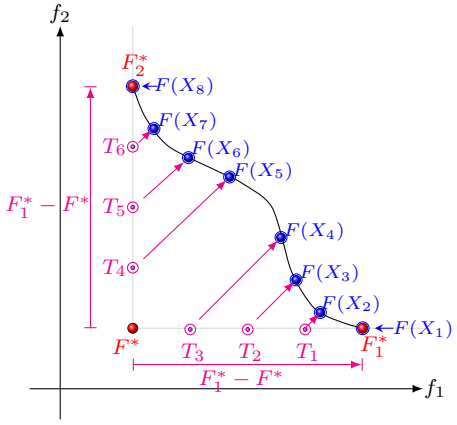


Fig. 4 Illustration of the decomposition procedure for $N_p = 8$.

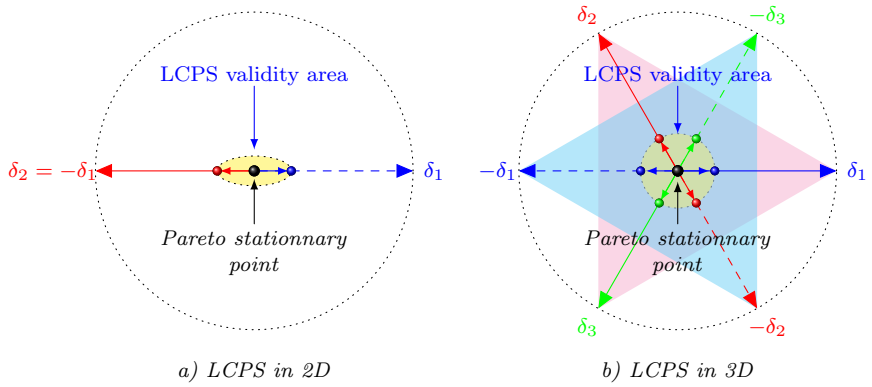
Furthermore, since the target points T_i belong to the axes connecting the utopia point F^* to the anchor points F_1^*, F_2^*, then the corresponding reference solutions $F(X_i)$ must theoretically be on the Pareto front (Fig.4).

3.2 Adaptive path relinking

On a Pareto point, due to the KKT conditions, the directions of the two gradients are opposites to each other, in a way that if we move locally from this point in the direction of one of the involved gradients, the corresponding objective improves while the remaining objective regresses since the move is realized in the opposite direction of its gradient (Fig.5). Thus, the resulting point is not dominated by the considered Pareto point. In other words, moving locally from a Pareto point along the gradient direction generates a non dominated point, which offers a manner to move along the Pareto front. We slide between the reference points by considering displacement along the best directions given by the two gradients such as to correct potential "take-off" moves.

The adaptive path relinking strategy is composed of three procedures:

- **Best Direction Move (BDM):** this procedure generates a displacement by following the best direction given by the available gradient information.
- **Direct SLide (D-Slide):** this procedure is designed to realize a direct slide on the PF. It operates on a pair of reference points (X_k, X_{k+1}), and generates a trajectory by following the best direction to move given by the



a) LCPS in 2D *b) LCPS in 3D*

***Fig.* 5** Illustration of the LCPS feature for 2 and 3 objective functions.

BDM starting with X_k and keep moving until reaching the next reference point X_{k+1}.

– **Reverse SLide (R-Slide):** this procedure is the opposite version of the D-Slide which means that it operates in the opposite direction. The R-Slide is carried out systematically after each D-Slide in order to see if there is a way to improve/correct its trajectory realized between X_k and X_{k+1}. Thus, the R-Slide operates on a pair of reference points (X_k, X_{k+1}) and tries to correct the trajectory by restarting from X_{k+1} until reaching X_k.

3.3 Best direction to move (BDM)

To define the best direction to move (BDM), we first need to highlight some terminologies. Let $C = F(X)$ be a point in the objective space, and $D = F(X + \lambda d)$ a point of the trajectory resulting by a move from X. We say that the trajectory progresses from X if it improves the first objective which translates to $Y_d > Y_c$. Otherwise, we say that the trajectory regresses from X.

The SPIDER approach considers the reference points $(X_k)_{1 \leqslant k \leqslant N_p}$ to be on the PF, and carry out a trajectory starting with the first anchor point X_1 (i.e. first reference point) and progressing until the second anchor point X_{N_p} (i.e. last reference point) and going through the transition points $(X_k)_{1<k<N_p}$. To achieve this goal, the SPIDER uses two procedures realizing a round trip: D-Slide and R-Slide. Both procedures rely on the BDM procedure that determines the best direction to realize the right move.

In this regard, the D-Slide is designed to keep moving on the left in the objective space. Thus, any move to the right has to be banished for the slide process. Likewise, the R-Slide progresses exclusively from the left to the right. In order to operate correctly and further optimally, we have developed a procedure named BDM that determines for a given solution point X, the best direction to move for both cases. More precisely, the BDM procedure selects the best direction δ among the four available gradients directions $\pm\nabla f_i(X)$ by considering the following steps:

– For a move on the right of $F(X)$ (D-Slide): let $\lambda > 0$ be small enough. A move on the right of $F(X)$ means that $f(X + \lambda\delta) > f(X)$. Therefore, a first choice to consider is given by the direction $\nabla f_1(X)$, since we have $f(X + \lambda\nabla f_1(X)) > f(X)$. Let then $d_1 = \nabla f_1(X)$. The second gradient direction to consider is $\pm\nabla f_2(X)$ depending on whether $f(X+\lambda\delta) > f(X)$ or not. More precisely, the second direction d_2 is given by:

$$d_2 = \begin{cases} \nabla f_2(X), & if \quad f(X + \lambda\nabla f_2(X)) > f(X) \\ -\nabla f_2(X), & otherwise \end{cases} \tag{10}$$

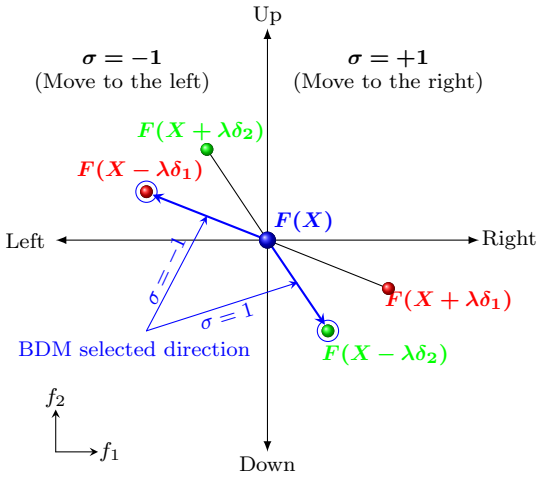


***Fig.* 6** Illustration of BDM selection in case of $\delta_1 \neq 0$ and $\delta_2 \neq 0$.

For the two considered directions, d_1 and d_2 satisfy the criteria of a correct D-Slide move. Finally, we select among them the biggest descent. First,we consider the main case in which Let $d_i \neq 0, i = 1, 2$, then $\delta_i, i = 1, 2$, represent the normalized gradient for both directions: $\delta_i = \frac{d_i}{\|d_i\|}$, $i = 1, 2$. Thus, the best direction δ is defined as follow:

- Case $\nabla f_1(X) \neq 0$ and $\nabla f_2(X) \neq 0$:
$$\delta = \begin{cases} \delta_1, & if \quad g(X+\lambda\delta_1) < g(X+\lambda\delta_2) \\ \delta_2, & otherwise \end{cases},$$
- Case $\nabla f_1(X) \neq 0$ and $\nabla f_2(X) = 0$: $\delta = \delta_1$.
- Case $\nabla f_1(X) = 0$ and $\nabla f_2(X) \neq 0$: $\delta = \delta_2$.

– Likewise, for a move on the left of $F(X)$ (R-Slide process), the best direction will be chosen among the direction $-\nabla f_1(X)$ and $\pm\nabla f_2(X)$, by following the same steps (Fig.6).

Algorithm 1 (resp. Algorithm 2) details the BDM procedure (resp. projection procedure used by BDM).

Algorithm 1 : **BDM**

Input : F, X, λ, σ ▷σ indicates the move direction: to the left: $\sigma = -1$, to the right: $\sigma = +1$
Output: δ : The best direction to move according to σ
$C = F(X)$;
$G = \nabla F(X); G = \mathbf{Proj}(G, L, U)$;
if $\|G_1\| > 0$ **and** $\|G_2\| > 0$ **then**
  $\delta_1 = \sigma G_1$;
  $\delta_2 = \begin{cases} G_2, & \textbf{if } \sigma(f(X+\lambda G_2) - f(X)) > 0 \\ -G_2, & otherwise \end{cases}$;
  $\delta_1 = \delta_1/\|\delta_1\|$; $\delta_2 = \delta_2/\|\delta_2\|$;
  $X_1 = \mathbf{Proj}(X + \lambda \cdot \delta_1, L, U)$; $X_2 = \mathbf{Proj}(X + \lambda \cdot \delta_2, L, U)$;
  $\delta = \begin{cases} -\delta_1, & \textbf{if } g(X_1) < g(X_2) \\ \delta_2, & otherwise \end{cases}$
else if $\|G_1\| > 0$ and $\|G_2\| = 0$ **then**
  $\delta_1 = \sigma G_1$; $\delta = \delta_1/\|\delta_1\|$;
else
  $\delta_2 = \begin{cases} G_2, & \textbf{if } \sigma(f(X+\lambda G_2) - f(X)) > 0 \\ -G_2, & otherwise \end{cases}$
  $\delta = \delta_2/\|\delta_2\|$
end if

Algorithm 2 : **Proj**

Input : X, L, U
Output: Y : Projection of X on $[L,U]$: $l_k \leqslant y_k \leqslant u_k$
$Y = X$;
$Y = max(Y, L)$;
$Y = min(Y, U)$;

For the rest of the paper, we adopt the following notations:

◦ In the objective space, we refers to the objectives f_1 and f_2, by the coordinates Y and Z respectively.

◦ For sake of simplicity, in the feasible objective space we identify each point $P = [Y; Z]$, with its associate point $\overline{P} = [X; Y; Z]$ such as $F(X) = [Y; Z]$. Thus: $F(X_p) = (f_1(X_p); f_2(X_p)) = (Y_p; Z_p)$.

◦ Moreover, we adopt the following notation:

$$P \lhd Q \Leftrightarrow (Y_p \leqslant Y_q \quad \text{and} \quad Z_p < Z_q)$$

3.4 Direct Slide (D-Slide)

As already mentioned, the D-Slide procedure generates non dominated points for all pairwise of reference points (X_k, X_{k+1}) existing between the anchor points. Thus D-Slide starts with X_k and releases small displacements in the direction of X_{k+1} using the procedure BDM. D-Slide generates an archive of non dominated points, noted RND. Each displacement point dominated by none of the two reference points and its previous displacement points is archived.

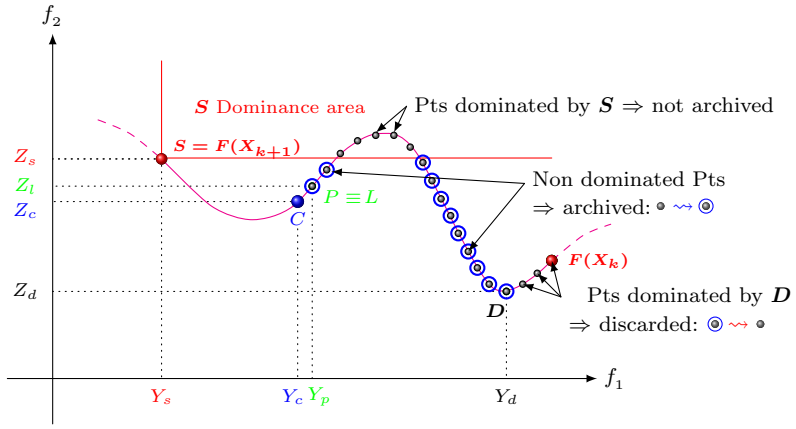


Fig. 7 Illustration of the D-Slide procedure.

This trajectory is generated by applying the BDM procedure recursively starting with X_k. For sake of efficiency, we have improved this approach by proceeding by cycles. This approach uses fixed step-size λ available for all the cycles. Using a discretization approach, each cycle generates N_d points by considering $(n/N_d)\lambda, n = 1..N_d$ instead of λ (Fig.8).

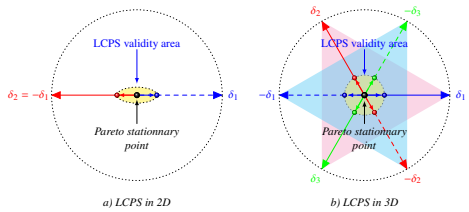


***Fig.* 8** Illustration of the points generation by the SPIDER cycle approach.

First we introduce the D-Slide and R-Slide parameters:

F: multi-objective function,

X_k: first reference point,

X_{k+1}: second reference point,

C: current point,

D: last archived non dominated point,

λ: step-size of a gradient move,

N_d: maxinun number of BDM calls per cycle (discretization approach),

RND: archive of non dominated design points,

RT: temporary archive used to store points on the reverse way (R-Slide),

ε_r: tolerance level for the accepted non dominated points.

The pseudo-code of D-Slide is shown in the algorithm 3.

Algorithm 3 : **D-Slide** pseudo code

Input : F, X_k , X_{k+1} λ, D
Output: RND, C
Set the current point: $X_c = X_k; C = F(X_k)$;
Set the border point: $S = F(X_{k+1})$;
while $Y_s < Y_c$ **do**
  $n = 1$, $X_e = X_c$, $P = C$;
  while $n < N_d$ **do**
    $\delta = \mathbf{BDM}(F, X_e, (n/N_d)\lambda, -1)$; ▷ Here $\sigma = -1 \Rightarrow$ move to the left
    $X_c = \mathbf{Proj}(X_e + (n/N_d)\lambda \cdot \delta, L, U)$; $[Y_c, Z_c] = F(X_c)$
    if $d(P, C) < \varepsilon$ **or** $Y_c < Y_s$ **or** $Z_c > Y_s$ **or** $Y_c < Y_p$ **then**
      $n = n + 1$
      continue
    end if
    if $C \triangleleft D$ **and** $Z_c < Z_s$ **and** $d(C, D) < \varepsilon$ **then**
      Delete all the elements of RND dominated by C;
      Add X_c to the top of RND: $RND = [RND, X_c]$;
    else if $Y_c < Y_d$ **and** $Z_c > Z_d$ **and** $Y_c < Y_s$ **then**
      Add X_c to the top of RND: $RND = [RND, X_c]$;
    end if
    Update D with the last element of RND;
    $n = n + 1$
  end while
end while

In the following, the D-Slide algorithm is detailed. Indeed, here are enlightenments for some technical considerations:

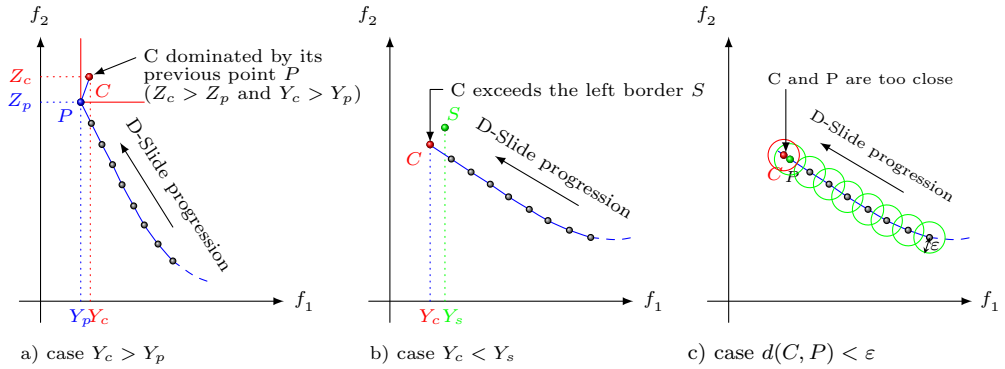


***Fig.* 9** Illustration of degenerating cases in D-Slide process.

i) **The exit process conditions** (Code lines $10-12$): indeed, to prevent degenerating displacements, the D-Slide process must reset its cycles in the following cases (Fig.9):

- The objective point Y_c is generated in the wrong direction, meaning that C is placed before the previous point P: $Y_c < Y_p$.
- The current objective point C is too near the previous one: $d(C, P) < \varepsilon$.
- The current objective point Y_c exceeds the left border S: $Y_c > Y_s$.

ii) **Store/Discard on archive *RND*** (code lines $13-18$): the mechanism of Store/Discard used by the D-Slide process is a dynamic mechanism in a way that it can discard the previous stored non dominated points provided by the RND archive. The principle of this mechanism is shown in Fig.10.

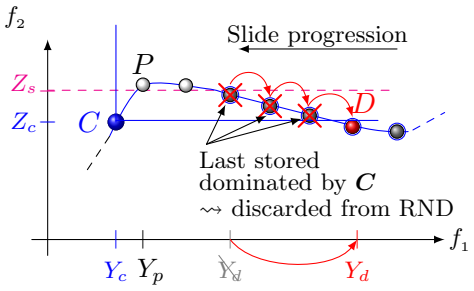


***Fig.* 10** Illustration of discarding mechanism in D-Slide procedure.

3.5 Reverse Slide (R-SLide

The structure of Reverse Slide is very similar to the Direct Slide one. However it involves the following considerations:

- The Reverse Slide realizes the reverses trajectory. It is executed systematically after each D-Slide procedure to correct the trajectory realized by D-Slide (see Fig.11).
- Due to the feedback effect produced by D-Slide process, the Reverse process involves additionally a task of correcting and/or improving the path. Moreover, the Reverse procedure updates the archive RND by erasing the points dominated on the return way (see Fig.11). This update is realized under some conditions (involving especially the correction tolerance ε_r) that ensure a real improvement to the path already realized with the D-Slide process. Thus the correction process is broken once this tolerance condition is not satisfied anymore (see Fig.12).
- Moreover, in order to realize the correction process, we introduce an auxiliary procedure named $ErasProc$ (Erasing procedure). For instance, it introduce the penultimate point of RND which is unavoidable in the correction process as illustrated in Fig.11.

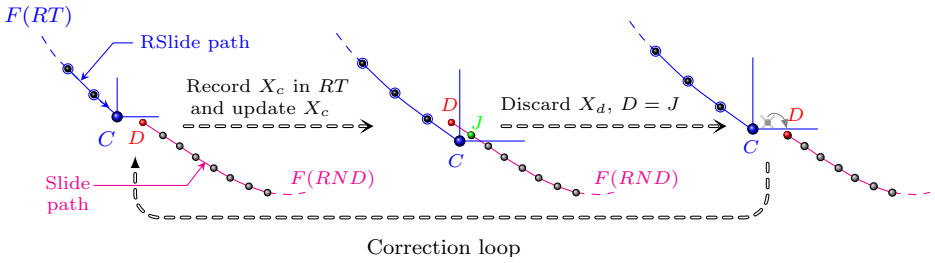


***Fig.* 11** R-Slide correction mechanism.

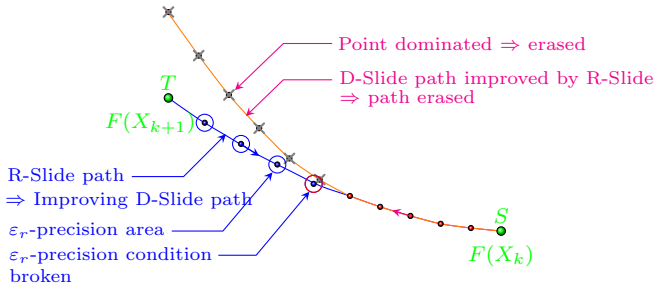


***Fig.* 12** Illustration of improving D-Slide path with R-Slide procedure.

The detailed steps of both R-Slide algorithm and *ErasProc* are shown respectively in Algorithm 4 and 5.

Algorithm 4 : **R-Slide** pseudo code

Input : F, X_k , X_{k+1} λ, D
Output: RND : archive
Set the border points: $S = F(X_{k+1})$; $T = F(X_{k+1})$;
Set the current design point: $X_c = X_k$; $[Y_c, Z_c] = F(X_k)$;
$RT = \emptyset$;
while $Y_c < Y_t$ **or** $Z_c < Z_t$ **do**
  $n = 1$, $X_e = X_c$, $P = C$,
  while $n < N_d$ **do**
    $\delta = \mathbf{BDM}(F, X_e, (n/N_d)\lambda, +1)$; ▷ Here $\sigma = +1 \Rightarrow$ move to the right
    $X_c = \mathbf{Proj}(X_e + (n/N_d)\lambda \cdot \delta, L, U)$; $[Y_c, Z_c] = F(X_c)$;
    if $d(P, C) < \varepsilon$ **or** $Y_c < Y_s$ **or** $Z_c > Y_s$ **or** $Y_c < Y_p$ **then**
      $n = n + 1$; **continue**;
    end if
    Set $Cond =$ **false** ▷ Boolean condition for recording X_c
    if $RND = \emptyset$ **then**
      $Cond =$ **true**
    else
      $[RND, Cond] = \mathbf{ErasProc}(RND, C)$
    end if
    if $Z_c < Y_s$ **and** $Z_c > Z_p$ **or** $Cond$ **then**
      Add X_c to bottom of RT: $RT = [X_c, RT]$;
      Update D with the last element of RND;
    end if
    Set d as the distance between the two last recorded points in RT;
    if $d < \varepsilon$ **then**
      return
    end if
    $n = n + 1$
  end while
end while
$RND = RND \cup RT$

In fact, the Erasing procedure involves the three following configurations depending on the dominance relations between C and D:

- case $D \triangleleft C$: in this case there is no possibility of correction, and thus D remains in RND.
- case $D \not\triangleleft C$: we need to see beyond D, i.e. the penultimate point in a way that D is erased from RND when $C \triangleleft J$ as shown in figure 11.
- case $C \triangleleft D$: here we erase all the points of RND dominated by C which must be archived (Cd =true).

Algorithm 5 : **ErasProc** pseudo code

Input : RND, C
Output: RND, Cd
if $C \triangleleft D$ **then**
  Delete all the elements of RND dominated by C
  Update D
  Set $Cd =$ **true** ▷ C must integrate RT
else if $D \triangleleft C$ **then**
  return ▷ here D must remain in RND
else if $\#RND > 1$ **then**
  Set J as the penultimate point of RND ▷ see Fig. 11
  if $C \triangleleft J$ **then**
    Delete the last element of RND, and set $Cd =$ **true**
  end if
end if

After getting the reference points, the D-Slide and R-Slide are supposed to generate an approximation of the Pareto front such as all the solution points in the objective space are localized between $Y_1 = f_1(X_1^*)$ and $Y_2 = f_1(X_2^*)$. Therefore, we must consider the eventuality of solutions existing beyond these two borders which are approximate anchor solutions, meaning they are eventually far enough from the real anchor solutions. The SPIDER algorithm is detailed in Algorithm 6.

Algorithm 6 : **SPIDER** pseudo code

1: **Input** : F, N_p, N_d, λ, ε, ε_r
2: **Output**: RND : archive of non dominated points
3: Get the N_p reference points $(X_k)_{1\leqslant k\leqslant N_p}$ by applying the sampling Pareto front decompositon procedure;
4: Set $RND = \emptyset$;
5: $X_c = X_1; C = F(X_1); D = F(X_1)$;
6: $cont$ = **true**; ▷ loop looking for solutions X beyond X_1^*: $X > X_1^*$(code lines $7-17$)
7: **while** $cont$ **do**
8: $Ys = Yc$;
9: $P = C; \delta = \mathbf{BDM}(F, X_c, \lambda, 1)$;
10: $X_c = \mathbf{Proj}(X_c + \lambda \cdot \delta, L, U)$; $C = F(X_c)$
11: **if** $d(P, C) < \varepsilon$ **or** $Y_c < Y_s$ **or** $Z_c > Y_s$ **or** $Y_c < Y_p$ **then**
12: $cont$ = **false**;
13: **else if** $Y_c < Y_d$ **then**
14: $RND = X_c \cup RND$;
15: **end if**
16: **end while**
17: $k = 1$; ▷ loop looking for solutions X such as : $X_2^* < X < X_1^*$ (code lines $18-33$)
18: **while** $k < N_p$ **do**
19: $[C, RND] = \mathbf{D\text{-}Slide}(F, X_k, X_{k+1}, RND, \lambda)$;
20: $S = F(X_{k+1})$;
21: **if** $C \lhd S$ **then**
22: $X_{k+1} = X_c$;
23: **else if** $Y_c < Y_s$ **and** $Z_c > Z_s$ **then**
24: $X_c = X_p$;
25: **end if**
26: **if** $(Y_d > Y_s + \varepsilon_r)$ **or** $(Z_d < Z_s)$ **then**
27: $[C, RND] = \mathbf{R\text{-}Slide}(F, X_k, X_{k+1}, RND, \lambda)$;
28: **end if**
29: $k = k + 1$;
30: **end while**
31: $X_c = X_{N_p}; D = F(X_{N_p}); C = F(X_c)$;
32: cont=**true**; ▷ loop looking for solutions X beyond $X_2^* : X < X_2^*$ (lines $34-44$)
33: **while** $cont$ **do**
34: $Ys = Yc$;
35: $X_p = X_c; P = C; \delta = \mathbf{BDM}(F, X_c, \lambda, -1)$;
36: $X_c = \mathbf{Proj}(X_c - \lambda \cdot \delta, L, U)$; $C = F(X_c)$
37: **if** $d(P, C) < \varepsilon$ **or** $Y_c < Y_s$ **or** $Z_c > Y_s$ **or** $Y_c > Y_p$ **then**
38: $cont$ = **false**;
39: **else if** $Y_c < Y_d$ **then**
40: $RND = RND \cup X_c$;
41: **end if**
42: **end while**

4 Computational experiments

The proposed algorithm SPIDER is implemented on Matlab. The computing platform used consists of an Intel(R)Core(TM) i3 4005U CPU 1:70 GHz with 4 GB RAM.

4.1 Test Problems

In order to evaluate the performance of the proposed SPIDER algorithm, 16 test problems are selected from the literature. These problems are covering different type of difficulties and are selected to illustrate the capacity of the algorithm to handle diverse type of Pareto fronts. In fact, all these test problems have different levels of complexity in terms of convexity and continuity. For instance, the test problems KUR and ZDT3 have disconnected Pareto fronts ; ZDT4 has too many local optimal Pareto solutions, whereas ZDT6 has non convex Pareto optimal front with low density of solutions near Pareto front. The test problems and their properties are shown in tables 1-2.

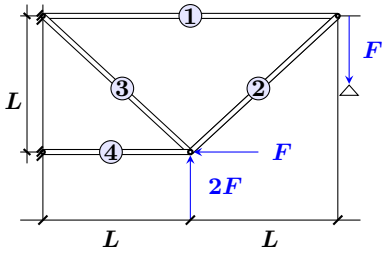
Table 1 Benchmark Problems used in our experiments.

Problem	Name	n	Bounds	Objective functions	Comments
F_1	***ZDT1***	30	$[0,1]^n$	$f_1(x) = x_1;\ f_2(x) = g(x)(1 - \sqrt{x_1/g(x)})$	Convex
				$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	Pareto front
F_2	***ZDT2***	30	$[0,1]^n$	$f_1(x) = x_1;\ f_2(x) = g(x)(1 - (x_1/g(x))^2)$	Non Convex
				$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	Pareto front
F_3	***ZDT3***	30	$[0,1]^n$	$f_1(x) = x_1;\ f_2(x) = g(x)(1 - \sqrt{x_1/g(x)} - (x_1/g(x))\sin(10\pi x_1)$	Convex
				$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	Pareto front
F_4	***ZDT4***	30	$[0,1]^n$	$f_1(x) = x_1;\ f_2(x) = g(x)(1 - \sqrt{x_1/g(x)})$	Convex
				$g(x) = 1 + 10(n-1)\sum_{i=2}^{n} x_i^2 - 10\cos(4\pi x_i)$	Pareto front
F_5	***POL***	2	$[-\pi,\pi]^2$	$f_1(x) = 1 + (A_1 - B_1)^2 + (A_2 - B_2)^2;\ f_2(x) = (x_1+3)^2 + (x_2+1)^2$ $A_1 = 0.5\sin(1) - 2\cos(1) + \sin(2) - 1.5\cos(2)$ $A_2 = 1.5\sin(1) - \cos(1) + 2\sin(2) - 0.5\cos(2)$ $B_1 = 0.5\sin(x_1) - 2\cos(x_1) + \sin(x_2) - 1.5\cos(x_2)$ $B_2 = 1.5\sin(x_1) - \cos(x_1) + 2\sin(x_2) - 0.5\cos(x_2)$	Convex
F_6	***ZDT6***	30	$[0,1]^n$	$f_1(x) = 1 - \exp(4x_1)\sin^6(6\pi x_1));\ f_2(x) = g(x)(1 - \sqrt{x_1/g(x)})$	Convex
				$g(x) = 1 + 9\left[\frac{1}{n-1}\sum_{i=2}^{n} x_i\right]^{0.25}$	Pareto front
F_7	***KUR***	3	$[-5,5]^3$	$f_1(x) = \sum_{i=2}^{n}(-10\exp(-0.2\sqrt{x_i^1 + x_i^2}));\ f_2(x) = \sum_{i=1}^{n}\lvert x_i\rvert^{0.8} + 5\sin(x_i^3)$	Convex

Table 2 Benchmark problems used in our experiments.

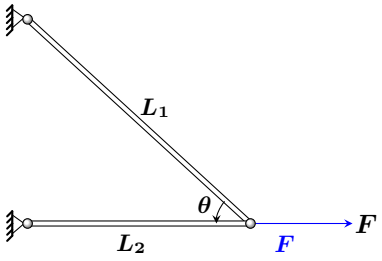
Problem	Name	n	Bounds	Objective functions	Comments
F_8	***Deb***	2	$[0.1, 1]^2$	$f_1(x) = x_1$; $f_2(x) = \frac{1}{x}(2 - \exp[-(\frac{y-0.2}{0.004})^2] - 0.8\exp[-(\frac{y-0.6}{0.4})^2])$	NonConvex
F_9	***SCH***	1	$[-10^3, 10^3]$	$f_1(x) = x^2$; $f_2(x) = (x-2)^2$	Convex
F_{10}	***FON***	3	$[-4, 4]^3$	$f_1(x) = 1 - \exp\big(-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})\big)$; $f_2(x) = 1 - \exp\big(-\sum_{i=1}^{3}(x_i + \frac{1}{\sqrt{3}})\big)$	Convex
F_{11}	***MUR***	2	$[0, 10]\times$ [-10, 10]	$f_1(x) = 2\sqrt{x_1}$; $f_2(x) = x_1(1 - x_2) + 5$	NonConvex
F_{12}	***MSC***	1	$[-2, 2]$	$f_1(x) = \exp(-x) + 1.4\exp(-x^2)$; $f_2(x) = \exp(x) + 1.4\exp(-x^2)$	NonConvex
F_{13}	***MOP1***	1	$[-2, 2]$	$f_1(x) = -x\mathbf{1}_{[-2,1]}(x) + (x-2)\mathbf{1}_{]1,3]}(x) + (4-x)\mathbf{1}_{]3,4]}(x)$ $+(x-4)\mathbf{1}_{]4,5]}(x)$; $f_2(x) = (x-5)^2$	NonConvex
F_{14}	***No-Hole***	2	$[-1, 1]^2$	$f_1(x) = (t+1)^2 + a$; $f_2(x) = (t-1)^2 + a$	Convex
F_{15}	***Hole***	2	$[-1, 1]^2$	$f_1(x) = (t+1)^2 + a + b\exp[-c(t-d)^2]$; $f_2(x) = (t-1)^2 + a + b\exp[-c(t+d)^2]$	NonConvex

Moreover, we have considered two real applications in truss structure optimization (Fig.13 and Fig.14). The detailed mathematical formulation of the problems can be found respectively in [30] and [24].



$$
\begin{aligned}
min\ \ & f_1(x) = L(2x_1 + \sqrt{2}x_2 + \sqrt{x_3} + x_4) \\
& f_2(x) = \tfrac{FL}{E}(\tfrac{2}{x_1} + \tfrac{2\sqrt{2}}{x_2} - \tfrac{2\sqrt{2}}{x_3} + \tfrac{2}{x_4}) \\
\textbf{s.t.}\ \ & (F/\sigma) \le x_1, x_4 \le 3(F/\sigma) \\
& \sqrt{2}(F/\sigma) \le x_2, x_3 \le 3(F/\sigma) \\
& \sqrt{2} \le x_3 \le 3 \\
& 1 \le x_4 \le 3
\end{aligned}
$$

***Fig.* 13** *Four-bar Truss Problem (TR4).*



$$\min F(S_1, S_2) = (\mu_V(S_1, S_2), \sigma_V(S_1, S_2))^T$$
$$\textbf{s.t.} \quad 22 < S_1 < 200$$
$$50 < S_2 < 100$$

***Fig.* 14** *2-bar lattice problem (TR2).*

4.2 Parameters setting

The SPIDER approach can use any single objective algorithm to approximate the anchors points and the reference points (i.e. decomposition procedure). In our experiments, we use a genetic algorithm (GA)[1] to handle this issue. The parameter setting in GA was as follows: the population size is 50, the crossover rate is $c = 0.9$, the mutation factor is $m = 0.1$, and the number of generation is 500 for each anchor point and 250 for each reference point. The other algorithm parameters were set as follows: number of reference points $N_p = 4$, maximum number of BDM call per discretization cycle $N_d = 100$, step size of the gradient move $\lambda = 0.1$, tolerance level for the accepted solutions $\varepsilon = 10^{-6}$, tolerance level for R-Slide improvement $\varepsilon_r = 10^{-4}$. This parameter configuration was adopted for all the experiments. The algorithm have been run on each test problem for 10 times.

4.3 Performances measures

In multi-objective optimization, there are three main criteria for evaluating the performance of an algorithm: convergence to the Pareto front, cardinality and diversity of the Pareto front. Three performance measures were adopted in this study: the generational distance (GD) to evaluate the convergence, the Spacing (S) and the Spread (Δ) to evaluate the diversity and cardinality.

– The convergence metric (GD) measure the extent of convergence to the true Pareto front. It is defined as:

$$GD = \frac{1}{N} \sum_{i=1}^{N} d_i, \tag{11}$$

[1] Available in the yarpiz library www.yarpiz.com.

where N is the number of solutions found and d_i is the Euclidean distance between each solution and its nearest point in the true Pareto front. The lower value of GD, the better convergence of the Pareto front to the real one.

– The Spread Δ, beside measuring the regularity of the obtained solutions, also quantifies the extent of spread in relation to the true Pareto front. The Spread is defined as:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N} |d_i - \overline{d}|}{d_f + d_l + d_i + (N-1)\overline{d}}. \tag{12}$$

where d_i is the Euclidean distance between two consecutive solutions in the obtained set, d_f and d_l denotes the distance between the boundary solutions of the true Pareto front and the extreme solutions in the set of obtained solutions, $\overline{d}$ denotes the average of all distances d_i, $i = 1, 2, \cdots, N-1$ under assumption of N obtained non-dominated solutions.

– The Spacing metric S indicates how the solutions of an obtained Pareto front are spaced with respect to each other. It is defined as:

$$S = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - \overline{d})^2} \tag{13}$$

4.4 Performance analysis

The obtained computational results are summarized in Table 3 in term of the mean and the standard deviation (Std) of the used metrics (GD, S, Δ) for 10 independent experiments, the average number of Pareto solutions found (NS), the average number of function evaluations (FEs), the average execution time in seconds (Time). Moreover the obtained Pareto fronts and the true fronts for all considered problems are shown in Figures 15-18. The left figures show all the generated points and the dynamics of the algorithm (i.e. trajectories). Different colors of the left figures show the obtained Pareto fronts for every pair of the reference points. The right figures show the final obtained Pareto front corresponding to the archive. By analyzing the GD metric statistics, we can see that the proposed SPIDER is well converging to the true Pareto front for all tested problems. Furthermore, the Spacing and Spread measures indicate that the SPIDER has the ability of generating uniform and diverse solutions. Also we observe that the SPIDER algorithm obtains a high number

of non-dominated solutions (NS) which can reach thousands for some problems. Indeed the complexity of archiving Pareto solutions is very low, with a worst case complexity of the order of $O(n)$.

Based on all these considerations, we can assess the efficiency of the proposed approach, in terms of convergence, diversity and cardinality, for a large panel of problems from the standard ones to the most challenging ones such as the hole problem test.

	GD		S		Δ				
Problem	Mean	Std	Mean	Std	Mean	Std	NS	FEs	Time
Zdt1	2,84e-05	1,32e-07	1,00e-03	2,68e-01	2,74e-01	3,68e-03	1003	87560	23,71
Zdt2	2,16e-05	4,54e-07	3,84e-04	2,25e-01	2,26e-01	7,02e-04	1002	91136	24,30
Zdt3	6,87e-05	1,06e-06	1,78e-02	7,09e-01	7,11e-01	1,39e-03	274	95043	24,57
Zdt4	9,40e-04	8,33e-04	1,30e-03	2,79e-01	3,75e-01	8,75e-02	1004	69986	28,47
Zdt6	7,52e-05	9,05e-06	1,24e-02	4,05e-01	6,70e-01	1,71e-01	102	61286	22,46
Pol	3,57e-06	4,41e-07	3,11e-01	9,69e-01	9,69e-01	2,43e-05	3310	70488	21,91
Kur	1,44e-04	9,78e-08	3,64e-02	4,78e-01	4,79e-01	4,16e-04	1541	81305	27,82
Deb	1,91e-04	1,35e-07	1,15e-02	9,94e-01	9,95e-01	7,63e-04	904	55708	20,49
SCH	1,30e-04	5,10e-15	3,63e-04	9,62e-02	9,62e-02	9,21e-11	2002	60228	20,55
FON	1,48e-05	1,71e-08	3,05e-04	7,80e-01	7,80e-01	6,12e-06	2002	64307	24,07
Mur	1,11e-05	3,17e-09	8,16e-05	5,29e-02	5,31e-02	1,85e-04	3004	68242	19,48
MSC	6,21e-05	6,17e-15	1,66e-03	4,62e-01	4,62e-01	1,21e-09	4002	70156	20,82
MOP1	9,94e-06	4,62e-15	1,79e-01	9,66e-01	9,66e-01	2,47e-10	2004	60499	19,90
NoHole	3,58e-05	6,49e-06	3,44e-02	3,56e-01	5,84e-01	1,93e-01	1346	58576	19,17
Hole	1,21e-02	1,44e-03	2,80e-02	7,09e-01	7,69e-01	4,87e-02	1216	59222	18,16

Table 3 Simulation results for the problem test $F_1 - F_{15}$.

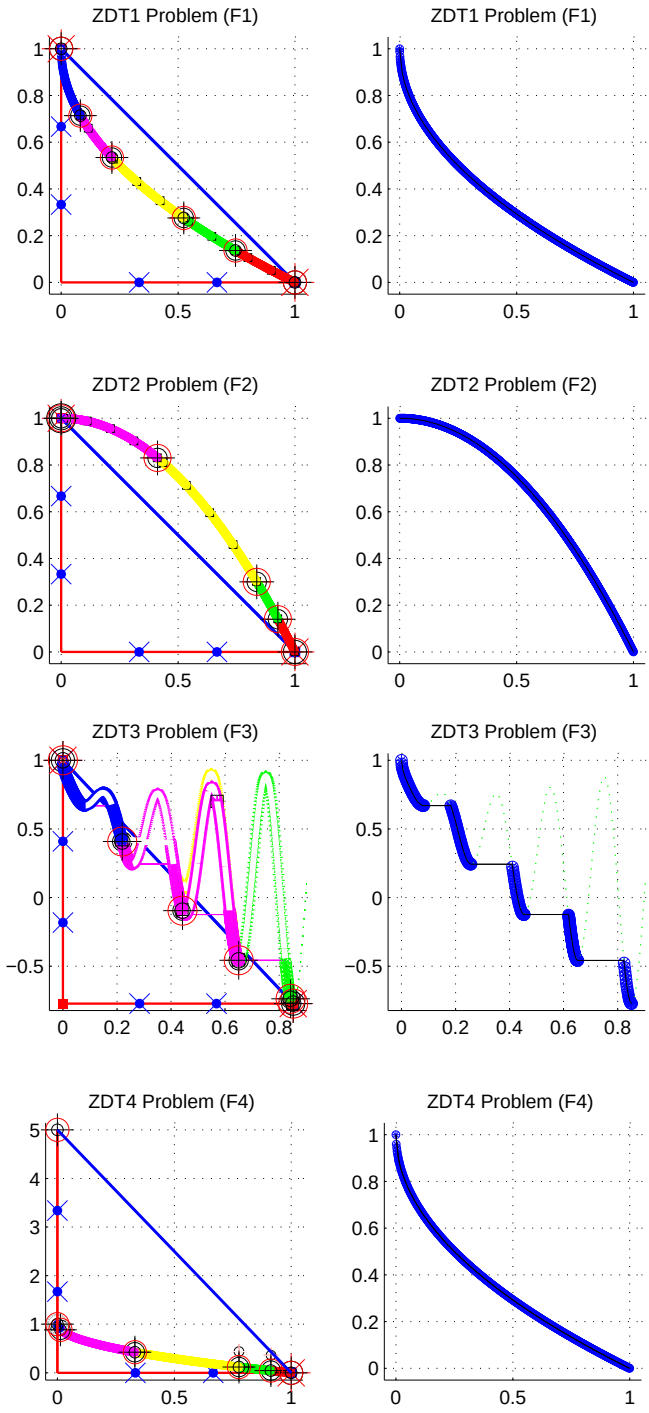


***Fig.* 15** *Obtained Pareto front for problems* $F_1 - F_4$.

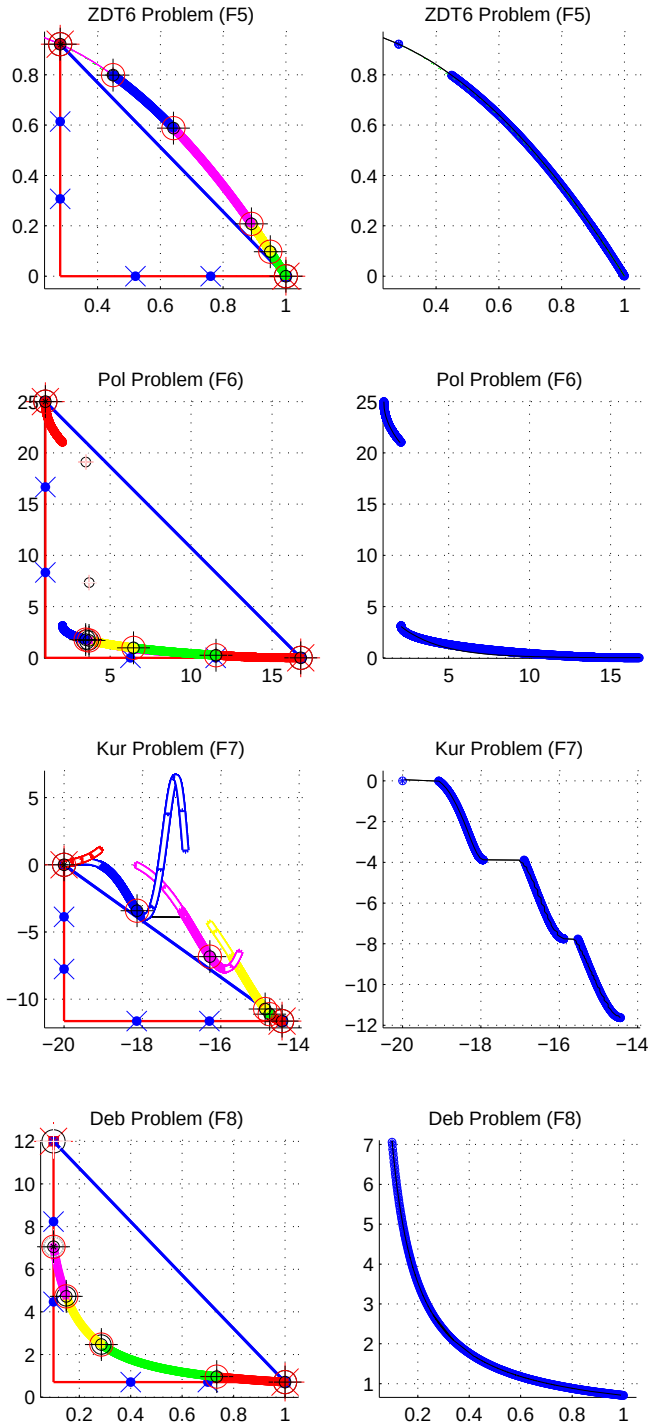


***Fig.* 16** *Obtained Pareto fronts for problems* $F_5 - F_8$.

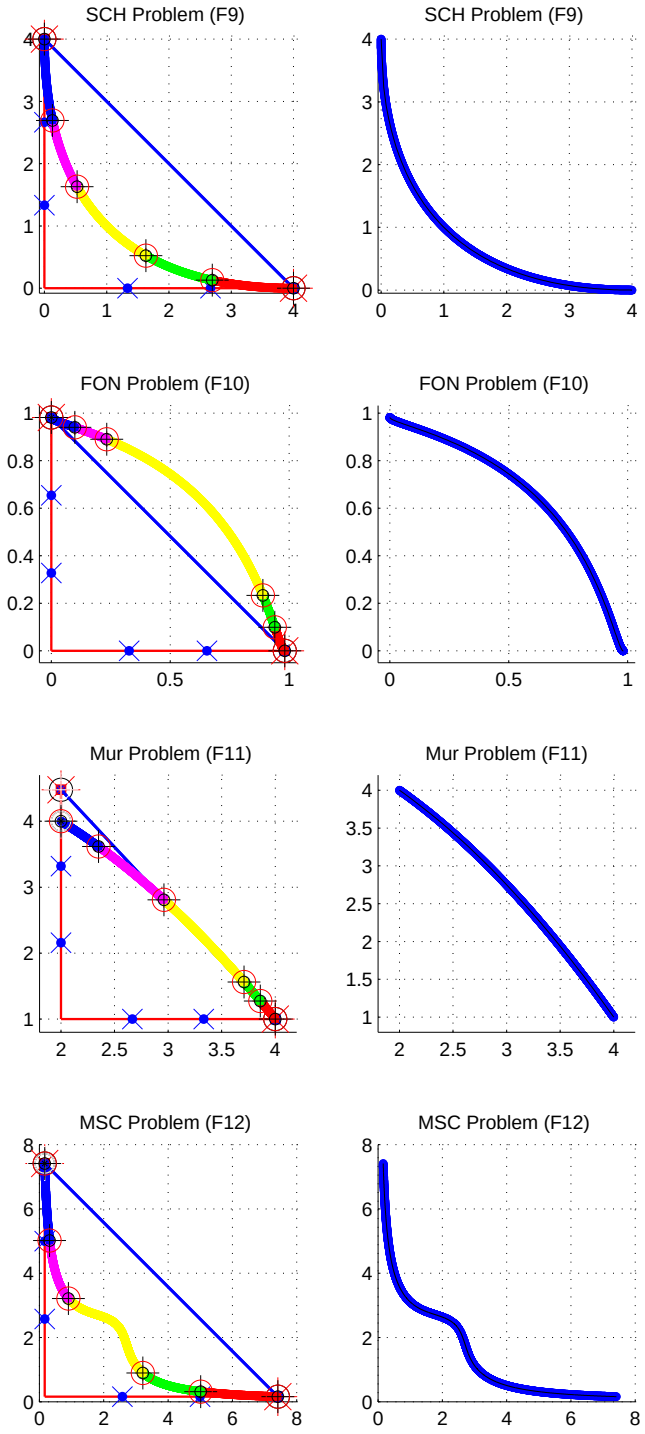


***Fig.* 17** *Obtained Pareto front for problems* $F_9 - F_{12}$.

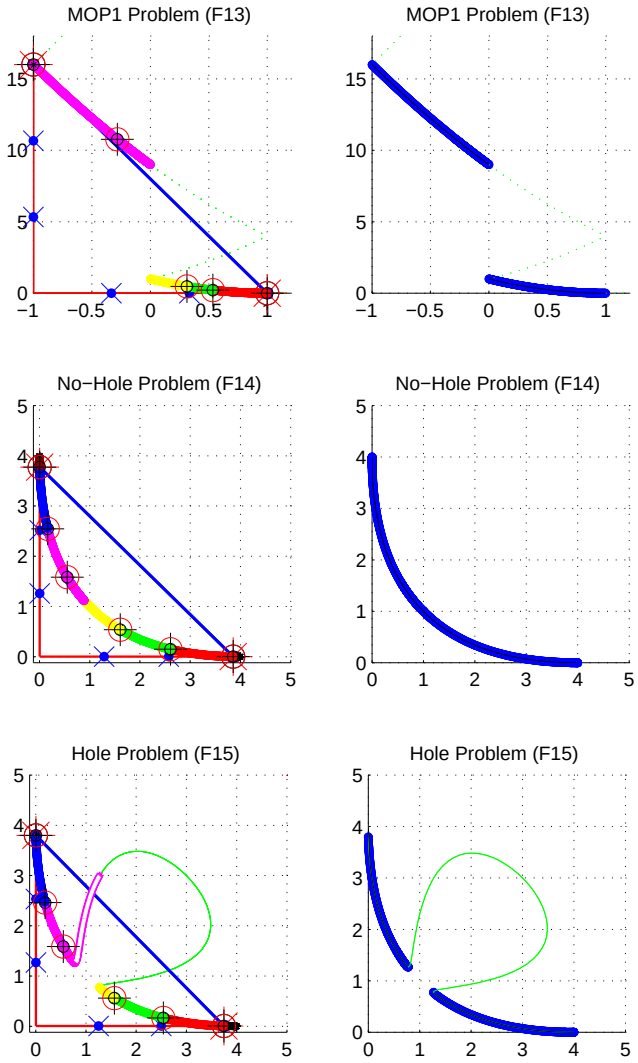


***Fig.* 18** *Obtained Pareto fronts for problems* $F_{13} - F_{15}$.

4.5 Comparison with some state-of the-art evolutionary algorithms

The proposed algorithm SPIDER is compared with three popular evolutionary algorithms: MOEA/D [28], NSGA-II [6] and PESA-II [4] [2]. The computational results using the performance indicators (GD, S and Δ) for 100000 function evaluations are shown in table 4.

[2] MATLAB implementation obtained for the yarpiz library available at www.yarpiz.com.

Table 4 Comparison of MOEA/D, NSGA-II, PESA-II and SPIDER for some considered test problems.

		GD		S		Δ			
Problem	Method	Mean	Std	Mean	Std	Mean	Std	NS	CPU(s)
	MOEA/D	2,81e-02	2,92e-02	2,42e-02	6,83e-03	9,95e-01	7,98e-02	100	86,67
Zdt1	NSGA-II	9,17e-02	1,03e-02	2,17e-02	1,56e-03	7,95e-01	2,86e-02	100	105,08
	PESA-II	5,38e-02	5,31e-03	3,73e-01	2,68e-02	8,82e-01	6,01e-02	100	40,97
	SPIDER	**2,84e-05**	**1,32e-07**	**1,00e-03**	**2,68e-01**	**2,74e-01**	**3,68e-03**	**1003**	**23,71**
	MOEA/D	1,32e-01	4,55e-02	2,66e-02	3,87e-03	1,13e+00	8,99e-03	100	61,61
Zdt2	NSGA-II	1,49e-01	1,74e-02	1,65e-02	2,31e-03	9,10e-01	1,69e-02	100	113,18
	PESA-II	9,04e-02	3,42e-03	5,40e-01	4,61e-02	8,32e-01	2,73e-02	100	32,82
	SPIDER	**2,16e-05**	**4,54e-07**	**3,84e-04**	**2,25e-01**	**2,26e-01**	**7,02e-04**	**1002**	**24,30**
	MOEA/D	2,08e-02	8,00e-03	6,01e-02	1,52e-02	1,19e+00	3,43e-02	100	82,50
Zdt3	NSGA-II	6,27e-02	4,74e-03	3,75e-02	1,23e-02	8,25e-01	2,12e-02	100	112,91
	PESA-II	4,45e-02	3,42e-03	3,54e-01	3,22e-02	8,48e-01	1,04e-01	100	33,93
	SPIDER	**6,34e-05**	**3,43e-06**	**1,78e-02**	**9,98e-05**	**7,14e-01**	**7,13e-03**	**343**	**10,66**
	MOEA/D	8,28e-01	5,75e-01	1,70e-01	1,45e-01	1,09e+00	5,77e-02	100	62,65
Zdt4	NSGA-II	4,46e-01	1,33e-01	3,03e-01	1,79e-01	8,85e-01	9,56e-02	100	129,75
	PESA-II	1,12e+01	5,16e-01	2,72e+01	4,86e+00	1,11e+00	5,16e-02	100	18,00
	SPIDER	**3,12e-05**	**3,48e-06**	**1,10e-03**	**6,33e-05**	**2,78e-01**	**6,77e-04**	**1004**	**20,28**
	MOEA/D	5,40e-01	1,08e-01	1,35e+00	1,01e+00	1,25e+00	1,54e-01	100	91,11
Pol	NSGA-II	2,48e+00	5,84e-02	1,75e+00	2,09e-03	9,72e-01	3,55e-03	100	123,86
	PESA-II	1,52e+01	6,02e+00	1,01e+01	1,16e+00	9,77e-01	9,85e-02	100	45,64
	SPIDER	**3,57e-06**	**4,41e-07**	**3,11e-01**	**9,69e-01**	**9,69e-01**	**2,43e-05**	**3310**	**21,91**
	MOEA/D	4,22e-01	1,18e-01	4,63e-02	2,02e-02	1,09e+00	5,98e-02	100	57,86
Zdt6	NSGA-II	3,08e-01	2,38e-02	1,57e-01	4,15e-02	8,73e-01	2,93e-02	100	121,89
	PESA-II	4,42e-01	4,69e-03	2,00e+00	6,31e-01	1,03e+00	4,97e-02	100	29,17
	SPIDER	**4,29e-01**	**6,21e-17**	**3,11e-01**	**0,00e+00**	**9,81e-01**	**0,00e+00**	**3309**	**10,40**
	MOEA/D	5,51e-03	5,51e-03	6,76e-01	4,35e-01	1,42e+00	1,80e-01	100	80,83
Kur	NSGA-II	5,31e-04	2,64e-05	1,22e-01	4,15e-03	4,10e-01	3,10e-02	100	117,86
	PESA-II	1,78e-01	1,28e-02	4,15e+00	4,86e-01	9,04e-01	5,75e-02	100	33,76
	SPIDER	**1,02e-04**	**8,37e-06**	**2,58e-02**	**3,41e-06**	**4,79e-01**	**5,12e-05**	**1483**	**21,23**
	MOEA/D	2,95e-02	1,99e-03	8,13e-02	5,63e-03	8,18e-01	1,83e-02	100	125,97
NoHole	NSGA-II	2,95e-02	1,99e-03	8,13e-02	5,63e-03	8,18e-01	1,83e-02	100	125,97
	PESA-II	3,80e-02	4,42e-03	1,43e+00	1,22e-01	8,04e-01	3,90e-02	100	28,33
	SPIDER	**3,58e-05**	**6,49e-06**	**3,44e-02**	**3,56e-01**	**5,84e-01**	**1,93e-01**	**1346**	**19,17**

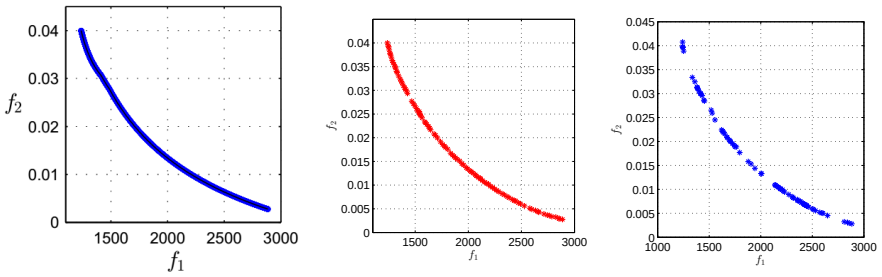
The table 5 illustrates a comparison of the results for the two real applications problems obtained on the one hand by our SPIDER approach and on the other hand by the NSGA-II and PESA-II reference methods. By analyzing the obtained results, it is clear that the SPIDER approach has the best performance in terms of convergence to the front as well as a better distribution

of solutions. Moreover, we observe that for the same number of evaluation functions our approach is able to generate a much larger number of solutions compared to the other approaches. The reason is that evolutionary algorithms imposes in general a restriction on the size of the archive of non-dominated points (here 100 points) since the procedure of archiving is very expensive at each iteration of the algorithm. This is not the case of our SPIDER approach which has a dynamic archiving mechanism allowing it to store non-dominated points by an incremental procedure that does not require to completely upset the archive of non-dominated points but which updates it rather locally.

Moreover, thanks to the mechanism of discretization by cycles of the gradient displacements, the distribution of the non dominated points reflect automatically the regularity due to the steps of displacements and a filtering mechanism of the type "crowding" is no longer required in our SPIDER approach. This leads to a gradual "accumulation" of the solutions in the archive has almost no cost and thus makes it operates with no restrictions on the size of the archive. The consequence of this important advantage is that unlike the other methods in comparison, the capture of the front by the SPIDER method is more continuous and therefore more precise, as illustrated by the figures 19 and 21. In addition, the SPIDER method is much faster (less expensive in CPU time) compared to other methods.

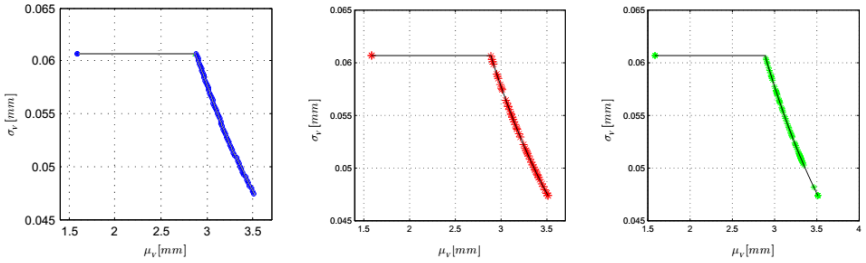
		GD		S		Δ			
Problem	Method	Mean	Std	Mean	Std	Mean	Std	NS	CPU(s)
	NSGA-II	3.61e-01	4.70e-02	2.36e+00	2.55e-01	6.23e-01	2.84e-02	100	251.57
TR4	PESA-II	9.73e-01	1.13e+01	4.36e+00	5.25e-01	7.31e-01	1.84e-01	100	120.75
	SPIDER	**2,31e-02**	**5,32e-03**	**2,18e-01**	**9,14e-01**	**9,14e-01**	**2,38e-05**	**5362**	**42,75**
	NSGA-II	8,26e-03	8,19e-03	1,30e-01	1,40e-03	1,11e+00	1,26e-02	100	319.67
TR2	PESA-II	1,62e-01	1,37e-02	9,09e-02	3,63e-02	1,00e+00	2,36e-02	100	154.55
	SPIDER	**3,28e-03**	**5,33e-03**	**4,26e-02**	**9,64e-02**	**8,76e-01**	**3,08e-02**	**819**	**71.86**

Table 5 Comparisons results for the real problems $TR4$ and $TR2$.



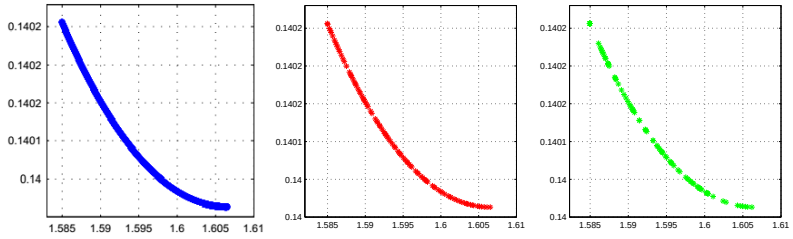
(a) *PF captured by SPIDER* (b) *PF captured by NSGA-II* (c) *PF captured by PESA-II*

***Fig.* 19** *Obtained Pareto fronts by SPIDER, NSGA-II and PESA-II for the problem TR4.*



(a) *PF captured by SPIDER* (b) *PF captured by NSGA-II* (c) *PF captured by PESA-II*

***Fig.* 20** *PF captured by SPIDER, NSGA-II and PESA-II for TR2 problem with frequency band (600,800)Hz.*



(a) *PF captured by SPIDER* (b) *PF captured by NSGA-II* (c) *PF captured by PESA-II*

***Fig.* 21** *Obtained Pareto fronts by SPIDER, NSGA-II and PESA-II for the problem TR2 with frequency band (100,300).*

5 Conclusion and future work

In this paper, we have successfully developed the SPIDER algorithm which is based on the decomposition of the objective space, and path-relinking of a set of pairwise Pareto solutions. Path relinking is based on the key idea of the local conservation of the Pareto stationarity feature on the local Pareto front (LCPS). The LCPS feature was introduced and proved for the first time to the best of our knowledge.

The proposed SPIDER algorithm was tested on various benchmark problems with different features and complexity levels. The results obtained amply demonstrate that the approach is efficient in converging to the true Pareto fronts and finding a diverse set of solutions along the Pareto front. Our approach largely outperforms some popular evolutionary algorithms such as MOEA/D, NSGA-II, and PESA-II in terms of the convergence, cardinality and diversity of the obtained Pareto fronts. The SPIDER algorithm is characterized by its fast and accurate convergence, very low complexity of archiving, parallel independent decomposition of the objective space, suitability for interactive optimization, and the no need for diversity management.

The path-relinking step of SPIDER can be used from any Pareto front approximation found by a multi-objective metaheuristic. This intensification mechanism is carried out in almost a free cost. Indeed, the most important cost of the SPIDER algorithm is due to the decomposition step. We are extending the SPIDER approach for constrained MOPs based on the projected Gradient. Moreover, different path-relinking approaches are investigated for many-objective optimization problems.

References

1. P. Bosman. On gradients and hybrid evolutionary algorithms for real-valued multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 16(1):51–69, 2011.
2. Jürgen Branke, Jurgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowiński. *Multiobjective optimization: Interactive and evolutionary approaches*, volume 5252. Springer Science & Business Media, 2008.
3. Carlos A Coello Coello. Multi-objective optimization. *Handbook of Heuristics*, pages 1–28, 2018.
4. David W. Corne, Nick R. Jerram, Joshua D. Knowles, and Martin J. Oates. Pesa-ii: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, pages 283–290, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

5. Indraneel Das and J. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8, 07 2000.
6. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.
7. Kalyanmoy Deb. Nonlinear goal programming using multi-objective genetic algorithms. *Journal of the Operational Research Society*, 52(3):291–302, 2001.
8. J-A. Désidéri. Mutiple-gradient descent algorithm for multiobjective optimization. 2012.
9. J. Fliege and B. F. Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494, 2000.
10. Haimes Y.V, L.S Lasdon, and Wismer D.A. On a bicriterion formation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-1(3):296–297, 1971.
11. K. Harada, J. Sakuma, and S. Kobayashi. Local search for multiobjective function optimization: pareto descent method. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 659–666. ACM, 2006.
12. E. Hosseinzade and H. Hassanpour. The karush-kuhn-tucker optimality conditions in interval-valued multiobjective programming problems. *Journal of applied mathematics & informatics*, 29(5_6):1157–1165, 2011.
13. K. Hyoung and M-S. Liou. New multi-objective genetic algorithms for diversity and convergence enhancement. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, page 1168, 2009.
14. A. Lara, G. Sanchez, C. A. C. Coello, and O. Schutze. Hcs: A new local search strategy for memetic multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 14(1):112–132, 2009.
15. Mariano Luque, Francisco Ruiz, and Kaisa Miettinen. Global formulation for interactive multiobjective optimization. *OR Spectrum*, 33(1):27–48, 2011.
16. Achille Messac. Physical programming-effective optimization for computational design. *AIAA journal*, 34(1):149–158, 1996.
17. Achille Messac and Christopher Mattson. Normal constraint method with guarantee of even representation of complete pareto frontier. *Aiaa Journal - AIAA J*, 42:2101–2111, 04 2004.
18. Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.
19. Silvia Poles, Yan Fu, and Enrico Rigoni. The effect of initial population sampling on the convergence of multi-objective genetic algorithms. In *Multiobjective programming and goal programming*, pages 123–133. Springer, 2009.
20. Alejandro Santiago, Héctor Joaquín Fraire Huacuja, Bernabé Dorronsoro, Johnatan E Pecero, Claudia Gómez Santillan, Juan Javier González Barbosa, and José Carlos Soto Monterrubio. A survey of decomposition methods for multi-objective optimization. In *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*, pages 453–465. Springer, 2014.
21. S. Schäffler, R. Schultz, and K. Weinzierl. Stochastic method for the solution of unconstrained vector optimization problems. *Journal of Optimization Theory and Applications*, 114(1):209–222, 2002.
22. R. Schilling, W. Haase, J. Periaux, H Baier, G Bugeda, et al. Numerical method for generating the entire pareto frontier in multobjective optimization. 2005.

23. Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.
24. W. Stadler and J. Dauer. 'multicriteria optimization in engineering: A tutorial and survey'. *Structural optimization: Status and promise*, 150, 1993.
25. E-G. Talbi. *Metaheuristics: from design to implementation*. Wiley, 2009.
26. S. Utyuzhnikov, P. Fantini, and M. Guenov. A method for generating a well-distributed pareto set in nonlinear multiobjective optimization. *Journal of Computational and Applied Mathematics*, 223(2):820–841, 2009.
27. O. L. De Weck. Multiobjective optimization: History and promise. In *Invited Keynote Paper, GL2-2, The Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems, Kanazawa, Japan*, volume 2, page 34, 2004.
28. Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.
29. Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32 – 49, 2011.
30. H. Zidani, E. Pagnacco, R. Sampaio, R. Ellaia, and JE. Souza de Cursi. Multi-objective optimization by a new hybridized method: applications to random mechanical systems. *Engineering Optimization*, 45(8):917–939, 2013.