Technical Report on the solution of the CMAPSS-RUL dataset using Neural Networks

David Laredo*[1] and Jian-qiao Sun†[1]

[1]School of Mechanical Engineering, University of California, Merced

Abstract

In this report we present an a data-driven approach for estimating the Remaining Useful Life (RUL) of aero-engines. A "strided" time window approach is employed to generate training and test sets to be used with a conventional Multi-layer Percepton (MLP) which will serve as the main regressor for this application, no model for the engine is required . The proposed approach is evaluated on the publicly available C-MAPSS dataset. The accuracy of the proposed method is compared against other state-of-the art methods available in the literature.

Index terms— Artificial Neural Networks (ANN), Moving Time Window, RUL Estimation, C-MAPSS, Prognostics

1. Introduction

Traditionally, maintenance of mechanical systems has been carried out based on scheduling strategies, nevertheless strategies such as breakdown corrective maintenance and scheduled preventive maintenance are often costly and less capable of meeting the increasing demand of efficiency and reliability [1, 2]. Condition Based Maintenance (CBM) also known as intelligent Prognostics and Health Management (PMH) allows for maintenance based on the current health of the system, thus cutting costs and increasing the reliability of the system [3]. To avoid confusion, here we define prognostics as the estimation of remaining useful component life. The Remaining Useful Life (RUL) of a system can be estimated based on history trajectory data, this approach which we refer here as data-driven can help improve maintenance schedules to avoid engineering failures and save costs [4]. This paper proposes a Machine Learning (ML) approach for RUL estimation.

The existing PMH methods can be grouped into three different categories: model-based approaches [5] , data-driven approaches [6, 7] and hybrid approaches [8, 9].

*dlaredorazo@ucmerced.edu

†jqsun3@ucmerced.edu

Model-based approaches attempt to incorporate physical models of the system into the estimation of the RUL. If the system degradation is modeled precisely, model-based approaches usually exhibit better performance than data-driven approaches [10], nevertheless this comes at the expense of having extensive a prior knowledge of the underlying system and having a fine-grained model of such system (which usually involve expensive computations). On the other hand data-driven approaches tend to use pattern recognition to detect changes in system states. Data-driven approaches are appropriate when the understanding of first principles of system operation is not comprehensive or when the system is sufficiently complex (i.e. jet engines, car engines, complex machinery) such that developing an accurate model is prohibitively expensive. Common disadvantages for the data-driven approaches are that they usually exhibit wider confidence intervals than model-based approaches and that a fair amount of data is required for training.

2. Problem Statement and Background

A multi-objective optimization problem (MMOP) can be formally stated as:

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

where $F : \mathbb{R}^n \to \mathbb{R}^k$, $F(x) = (f_1(x), \dots, f_k(x))^T$ represents a vector of $k \geq 2$ *objective functions*. The *feasible decision vectors*, that form the set $\mathbb{X}$, are those $x \in \mathbb{R}^n$ that comply with the equality $h_i(x)$ and inequality $h_j(x)$ constraints.

The optimality of a MOP is defined by the concept of dominance [**?**].

Definition 2.1 (Pareto Dominance) *A point $y \in \mathbb{X}$ is dominated by a point $x \in \mathbb{X}$ ($x \prec y$) with respect to Eq.* (1) *if x is partially less than y, i.e., if $f_i(x) \leq f_i(y)$, for all $i \in 1, \dots, k$, and $f_j(x) < f_j(y)$ for some $j \in 1, \dots, k$. Otherwise it is non-dominated by x.*

Definition 2.2 (Pareto Optimality) *A decision vector $x^* \in \mathbb{X}$ is Pareto optimal with respect to Eq.* (1) *if there does not exist another decision vector $x \in \mathbb{X}$ such that $x \prec x^*$.*

Definition 2.3 (Pareto Weak Optimality) *A decision vector $x^* \in \mathbb{X}$ is* weakly *Pareto optimal with respect to Eq.* (1) *if there does not exist another decision vector $x \in \mathbb{X}$, such that $f_i(x) < f_i(x^*) \quad \forall i = 1, ..., k$.*

In general, the solution of a MOP consists not only of a single solution but of a set of solutions which have to be considered as optimal. The solution set is called the *Pareto set* and its corresponding image is called the *Pareto front* which typically forms a $(k-1)$-dimensional manifold [**?**], where k is the number of objectives involved in the problem. The concepts of Pareto set and Pareto front are formalized in the following definition:

Definition 2.4 (Pareto set and Pareto front) *The set of optimal points $\mathcal{P}$ for Eq.* (1)*,*

$$\mathcal{P} = \{x \in \mathbb{X} \mid \nexists y \in \mathbb{X} : y \prec x\}$$

is called the Pareto set. The image $F(\mathcal{P})$ is called the Pareto front.

The Jacobian of F at a point x is given by

$$J(x) = \begin{pmatrix} \nabla f_1 \\ \vdots \\ \nabla f_m \end{pmatrix} \in \mathbb{R}^k. \tag{2}$$

where $\nabla f_i(x)$ denotes the gradient of objective f_i. In case all the objectives of the MOP are differentiable the following famous theorem of Kuhn and Tucker **[?]** states a necessary condition for Pareto optimality for unconstrained MOPs.

Theorem 1 (KKT Conditions) *Let x^* be a Pareto point of problem 1, then there exists a vector $\alpha \in R^k$ with $\alpha_i \geq 0$, $i = 1, \ldots, k$, and $\sum_{i=1}^{k}$ such that*

$$\sum_{i=1}^{k} \alpha_i \nabla f_i(x^*) = J(x)^T \alpha = 0. \tag{3}$$

Points satisfying Eq. 3 are called *Karush-Kuhn-Tucker (KKT)* points. One important thing to outline is that given a KKT point x^* its associated weight vector α is normal to the linearization (tangent) of the Pareto front at $F(x^*)$. It can also be noted that $rank(J(x^*)^T) < k$ for any x that is a KKT point **[?]**.

2.1. Mixed-Integer Optimization

A mixed-integer multi-objective optimization problem (MMOP) can be formally stated as:

$$\begin{aligned} \min_{x \in \mathbb{Z}^{d_1} \times \mathbb{R}^{d_2}} \quad & F(x) \\ \text{s. t.} \quad & \\ & h_i(x) = 0,\ i = 1 \ldots m \\ & h_j(x) \leq 0,\ j = 1 \ldots p, \end{aligned} \tag{4}$$

where $F : \mathbb{Z}^{d_1} \times \mathbb{R}^{d_2} \to \mathbb{R}^k$, which means that the parameter vector can be formed either by real variables, discrete (or integer) variables or a mixture of both depending on the values of d_1 and d_2 respectively. For instance, Figure **??** displays a two dimensional integer space where $x_1, x_2 \in \mathbb{Z}$, Figure **??** represents a mixed-integer space where $x_1 \in \mathbb{Z}$ and $x_2 \in \mathbb{R}$, while in Figure **??** $x_1, x_2 \in \mathbb{R}$, that is, they belong to a real space.

The goal of Eq. (4), as in the continuous case, is to seek non-dominated (Pareto optimal) solutions of the objective function F on the feasible set $\mathbb{X}$.

In this work the theory developed for continuous MOPs will be used for the treatment of MMOPs since it can be shown, that under some assumptions, most of theory presented within this chapter holds for MMOPs.

2.2. Direct Zig Zag Method

The Direct Zig-Zag (DZZ) method [**?**] is a continuation method for solving discrete or mixed-integer bi-objective problems. The DZZ method searches Pareto optimal solutions along a zig-zag path close to the Pareto front. The local zig-zag path is identified based on a pattern search idea (e.g. Hooke-Jeeves method [**?**]) in which the search procedure only compares function values without computing the gradients of the objective functions. Thus, the DZZ method can, in general, be applied for black-box discrete or mixed-integer MOPs where the objective functions can be evaluated through numerical or simulation processes. The DZZ method guarantees local Pareto optimality of the solutions due to the neighborhood search inside the pattern search procedure. The method consists of two parts.

In the first part, a First Pareto Solution (FPS) is computed, from a starting point x_0, by means of a modified version of the well known pattern search method [**?**]. This FPS is the Pareto point which minimizes f_1 while attaining the smallest value of f_2.

The second part is performed in an iterative way. For each Pareto solution x_0^* the method looks for a neighboring solution x_1 that increases the value of f_1, i.e., $f_1(x_1) > f_1(x_0^*)$, this is called a *zig step*. Since it is assumed that the continuation will start at the Pareto point that minimizes f_1 (FPS) it is logical to think that the only direction to keep moving is the one that increases the values of f_1. From x_1 a pattern search like strategy is applied in order to find a non-dominated solution x_1^*, such that $x_1^* \neq x_0^*$, this is a *zag step*. Figure **??** displays a simple example of the application of the DZZ method. The first phase, namely the FPS phase goes from $f_(x_0)$ to $f(x_1)$. From $f(x_1)$ onwards, zig and zag steps are iteratively applied.

It is remarkable that the method does not use gradient information making it very efficient in terms of function evaluations. Nevertheless, it is only able to solve bi-objective problems so far, making this its greatest disadvantage.

2.3. The Directed Search Method

This method defines a way to steer the search for continuous MOPs by using a direction in objective space and mapping it into parameter space [**?**]. Some of the concepts used by this method are quite important for the development of the EDS, thus a brief explantion of the key concepts of the DS method are given next. The main idea is as follows.

Assume a point $x_0 \in \mathbb{R}^n$, in parameter space, with $rank(J(x_0)) = k$ and a vector $d \in \mathbb{R}^k$ representing a desired search direction in image space are given. Then, a search direction $\nu \in \mathbb{R}^n$ in decision space is sought such that for $y_0 := x_0 + t\nu$, where $t \in \mathbb{R}_+$ is the step size (i.e., y_0

represents a movement from x_0 in direction v), it holds:

$$\lim_{t \to 0} \frac{f_i(y_0) - f_i(x_0)}{t} = \langle \nabla f_i(x_0), v \rangle = d_i, \quad i = 1, ..., k. \tag{5}$$

Using the Jacobian of F, Eq. (5) can be stated in matrix vector notation as

$$J(x_0)v = d. \tag{6}$$

Hence, such a search direction v can be computed by solving a system of linear equations. Since typically the number of decision variables is (much) higher than the number of objectives for a given MOP, i.e., $n \gg k$, system (6) is (probably highly) underdetermined, which implies that its solution is not unique. One possible choice is to take

$$v_+ = J(x_0)^+ d, \tag{7}$$

where $J(x_0)^+ \in \mathbb{R}^{n \times k}$ denotes the pseudo inverse[1] of $J(x_0)$. A new iterate x_1 can be computed as the following discussion shows: given a candidate solution x_0, a new solution is obtained via $x_1 = x_0 + tv$, where $t > 0$ is a step size and $v \in \mathbb{R}^n$ is a vector that satisfies (6). Among the solutions of system (6), v_+ is the one with the smallest Euclidean norm. Hence, given t, one expects for a step in direction v_+ (decision space) the largest progress in d-direction (objective space).

Given a direction $d \in \mathbb{R}^k \backslash \{0\}$ with $d_i \leq 0, i = 1, \ldots, k$, a point $x_0 \in \mathbb{R}^n$ with $\text{rank}(J(x_0)) = k$ and assuming that the image of F is bounded from below, a greedy search in direction d using Eq. (7) leads to the (numerical) solution of the following initial value problem:

$$\begin{aligned} x(0) &= x_0 \in \mathbb{R}^n \\ \dot{x}(t) &= J(x(t))^+ d, \quad t > 0. \end{aligned} \tag{8}$$

[1] If the rank of $J := J(x_0)$ is k (i.e., maximal) the pseudo inverse is given by $J^+ = J^T(JJ^T)^{-1}$.

Definition 2.5 (Critical Point) *Let $\gamma : [0, t_f] \to \mathbb{R}^n$ be a solution of Eq.* (8) *and let t_c be the smallest value of $t > 0$ such that*

$$\nexists v \in \mathbb{R}^n : \quad J(x(t))v = d. \tag{9}$$

.

Then t_c and $\gamma(t_c)$ are a critical value an critical point of (8) *respectively.*

By Definition 2.5, it is possible to divide the solution $\gamma : [0, t_f]$ of Eq. (8) into two phases, (see Figure **??**):

- $\gamma([0, t_c])$: the function $F(\gamma(t))$ gets the desired decay in d-direction.
- $\gamma((t_c, t_f])$: the function $F(\gamma(t))$ moves along the critical points of F. For the end point $\gamma(t_f)$, it holds $J(\gamma(t_f))^+ d = 0$.

The study made in [**?**] shows that *critical points* are not necessarily KKT points but are local solutions of the well known NBI [**?**] problem, therefore the DS descent method is only restricted to the detection of such critical points. To trace the solution curve of Eq. (8) numerically, specialized PC methods [**?**] can be used.

Using the ideas mentioned above, the authors of [**?**] developed a method to move towards and along the Pareto front of continuous MOPs, for a broader exposition of the DS please refer to [**?**].

3. The Enhanced Directed Search Method

3.1. Main idea

A major task in the development of the EDS continuation method is to "follow" a certain direction in objective space. For this, we will use the work developed in [**?**], which allows to map a direction from objective space $v \in \mathbb{R}^n$ to direction in parameter space $d \in \mathbb{R}^k$. This is achieved by solving the following single-objective optimization problem [**?**]:

$$\begin{aligned} \min_{v,\delta} \quad & \frac{1}{2} \|v\|_2^2 - \delta \\ \text{s. t.} \quad & J(x)v = \delta d, \end{aligned} \tag{10}$$

where $J(x) \in \mathbb{R}^{k \times n}$ is the Jacobian of F as defined in Eq. 1 in page 2 at the point x and $\delta \in \mathbb{R}$ is a scalar value. Let (v^*, δ^*) be a solution of Problem (10) where $d \neq 0$. Then for v^* and δ^* the following propositions [**?**] hold:

Proposition 3.1

a) $v^* \neq 0 \iff \delta^* > 0 \iff \exists v \in \mathbb{R}^n : Jv = d$. *In that case* $v^* = \delta^* J^+ d$.

b) $v^* = 0 \iff \delta^* = 0$. *In that case* x *is a critical point (see Definition 2.5).*

c) $v(x)$ *and* $\delta(x)$ *are continuous mappings.*

By Proposition 3.1 it follows that:

- δ^* can be used as a criterion for determining critical points.
- Problem (10) can be used for mapping a direction in objective space to a direction in parameter space.

Problem 10 along with the above propositions open the possibility for steering the search in a desired direction in objective space and mapping it to parameter space. As it can be seen, this is exactly the main idea underlying DS, nevertheless Proposition 3.1 provides a more stable criterion for determining critical points that the one provided in [**?**]. Hence, by solving Problem (10) for a point $x_0 \in \mathbb{R}^n$ and a direction in objective space $d \in \mathbb{R}^k$, one obtains a direction in parameter space $v \in \mathbb{R}^n$ such that $F(x_i)$, where $x_i = x_0 + tv$, moves along d for a suitable step size t, and a scalar $\delta \in \mathbb{R}$, where $\delta = 0$ when x_0 is a critical point. The aforementioned ideas serve as the basis for the predictor and corrector steps in the EDS method.

In the following we propose a new PC method for the continuation along (local) Pareto sets of a given MMOP. The Enhanced Directed Search (EDS) method is in principle capable of dealing with problems with $k \geq 2$, furthermore, neighborhood information is exploited in order to save function evaluations and hence improving the overall performance of the method.

3.2. Predictor

For the predictor we make use of the ideas developed in [**?**]. Assume we are given a (local) Pareto point and its associated convex weight $\alpha \in \mathbb{R}^k$, further, we assume that $\operatorname{rank}(J(x)) = k - 1$. Then it is implied by Theorem 1 that α is orthogonal to the linearized Pareto front at $F(x)$. Thus, a search orthogonal to α could be promising to obtain new predictor points. To obtain such directions a QR-factorization of α can be computed, i.e.,

$$\alpha = QR, \tag{11}$$

where $Q = (q_1, \ldots, q_k) \in \mathbb{R}^{k \times k}$ is an orthogonal matrix and $R = (r_{11}, 0, \ldots, 0)^T \in R^{k \times 1}$ with $r_{11} \in \mathbb{R} \backslash \{0\}$. Since by Eq. (11) $\alpha = r_{11} q_1$, it follows that a well spread set of directions can be taken from any of the normalized search directions v_i such that:

$$J(x) v_i = q_{i+1}, \quad i = 1, \ldots, k-1. \tag{12}$$

It follows by the rank of $J(x)$ that the vectors $q_2, \ldots, q_k$, are in the image of $J(x)$ and hence Eq. (10) can be solved for each q_i where $i \in \{2, \ldots, k\}$. Please note that such predictor directions $v_i = \delta_i J^+ q_i$ do not have to be tangent to the *Pareto set*. Instead, $v = \delta J(x)^+ q$ points along the

linearized Pareto front. Since by Proposition 3.1 ν is the most greedy solution of Problem (10) with respect to q, we can expect that the image $F(p)$, where $p \in \mathbb{R}^n$ is the chosen predictor, is also close to the Pareto front, and thus, that only few iteration steps are required to correct back to the front. Hence, a set of predictors can be computed in the following way:

$$\begin{aligned} p_{i+} &= x + \hat{t}\frac{\nu_i}{||\nu_i||} \\ p_{i-} &= x - \hat{t}\frac{\nu_i}{||\nu_i||}, \end{aligned} \tag{13}$$

where $\hat{t}$ is the chosen step size. Note that both, the positive and the negative predictors have to be considered. This is done to ensure that the search is done in every possible direction and hence that a full covering of the Pareto front is obtained at the end of EDS execution.

Finally, to obtain the predictor p at the current point x, we need to select a suitable step size. In this study we are particularly interested in an evenly distributed set of solutions along the Pareto front. That is, at least for two consecutive solutions x_{i+1} and x_i we want that

$$||F(x_{i+1}) - F(x_i)|| \approx \tau, \tag{14}$$

where $\tau > 0$ is a user specified value. For this, we follow the suggestion made in **[?]** and take the step size

$$\hat{t} = \frac{\tau}{||J\nu||}. \tag{15}$$

The entire predictor phase is shown in Algorithm 1

Algorithm 1 Obtain predictors

Input: Initial point $x \in \mathbb{R}^n$, predictor direction in objective space $d \in \mathbb{R}^k$
Output: A set of predictors p_+ and p_-

1: Compute $J(x)$
2: Compute direction ν_p by solving Problem (10)
3: Compute predictor step size t_p by Eq. (15)
4: Set $p_+ = x + \hat{t}\nu$ and $p_- = x - \hat{t}\nu$
5: return $[p_+, p_-]$

Figure **??** exemplifies the predictor phase. First, for the local Pareto point x_i the α vector is computed, then a direction d_+ orthogonal to α is computed. Finally, after executing Algorithm 1, two predictors p_+ and p_- are obtained.

3.3. Corrector

For the computation of the corrector, a new directed search descent method was developed. Assume we are given a predictor p along with the weight vector α associated to the initial point

x from which p was computed. Furthermore, since by assumption $F(p)$ is *close* to $F(x)$ it makes sense to use the opposite direction of α, i.e., $-\alpha$, in order to move back to the Pareto front (see Figure **??**).

Thus, given $p \in \mathbb{R}^n$ and $-\alpha \in \mathbb{R}^k$, a direction $\nu_c \in \mathbb{R}^n$ that moves along $-\alpha$ can be computed by solving Eq. (10) for p and $-\alpha$. A movement along ν_c (and hence in $-\alpha$ direction in objective space) is then performed in the following way:

$$c_i = p_i + t\frac{\nu_c}{||\nu_c||}, \tag{16}$$

where t is the corrector step size. Contrary to the predictor phase, several corrector iterations may be computed until a certain stopping criterion (see Section 3.3.2) is met.

Two important issues arise now: first, how to compute a suitable step size for the corrector? And second, how to determine whenever the corrector is indeed a critical point? Both issues will be addressed in the following sections.

3.3.1 Corrector Step Size

As stated in Section 2.3 a movement along $d \in \mathbb{R}^k$ (in objective space) using the map (7), which maps to a movement ν in parameter space

$$x_{i+1} = x_i + t\nu, \tag{17}$$

is only possible for *sufficiently small* values of t. Let $\tilde{d} = F(x_{i+1}) - F(x_i) \in \mathbb{R}^k$, be the vector in objective space obtained after moving along ν direction for a given step size t, also let

$$\beta = cos^{-1}(\frac{\langle \tilde{d}, d \rangle}{||\tilde{d}||||d||}), \tag{18}$$

be the angle between $\tilde{d}$ and d. It was observed during our experiments that as t increases the deviation between $\tilde{d}$ and d, measured by the angle β, also increases. Therefore, the step size for the corrector plays an important role in the overall performance of the method.

Assume we are given a predictor p along with a direction in parameter space ν and a direction d in objective space. Let $\tilde{d} = F(c) - F(p)$ where $c = p + t\nu$. A suitable step size t provides the largest progress in direction $\tilde{d}$ while keeping angle β bellow a user defined threshold. For our convenience we will define $\epsilon \in [0, 1]$ as the aforementioned threshold where $\epsilon = cos(\beta)$, therefore keeping the angle itself bellow a threshold $cos^{-1}(\epsilon)$. Furthermore we would like that the corrector c at least weekly dominates the previous point p. Hence, the optimal step size t can be computed by solving the following constrained single objective optmization problem (SOP):

$$\begin{aligned} \underset{t\in\mathbb{R}}{\text{minimize}} \quad & -||\tilde{d}|| \\ \text{s. t.} \quad & \\ & min(\tilde{d}) \qquad\quad \leq 0 \\ & \epsilon - \frac{\langle \tilde{d}, d\rangle}{||\tilde{d}||||d||} \quad \leq 0, \end{aligned} \tag{19}$$

where $\tilde{d} = F(p + tv) - F(p)$. The objective function of Problem (19) will try to maximize the movement along $\tilde{d}$. The first constraint ensures that the newly computed point $F(c)$ at least weakly dominates the previous one $F(p)$. We will now bring our attention to the second constraint.

Assume that a certain deviation from direction d, measured by $\epsilon \in [0, 1]$, is permitted. Then any direction $\tilde{d} = F(c) - F(p)$ that fulfills the constraint

$$\epsilon - \frac{\langle \tilde{d}, d\rangle}{||\tilde{d}||||d||} \leq 0 \tag{20}$$

points in direction d with a maximum deviation of $\beta = cos^{-1}(\epsilon)$ degrees. Thus, the smaller the value of ϵ, the greater the permitted angle between $\tilde{d}$ and d (and thus the deviation between them) is. Also note that by construction of constraint (20) angles $\beta \in (\frac{\pi}{2}, \frac{3\pi}{2})$, and thus solutions that are dominated by $F(p)$, are automatically discarded. Figure **??** depicts constraint (20), any point in between the dashed arrows is an acceptable corrector point c, recall that $\beta = cos^{-1}(\epsilon)$.

Therefore, by solving Problem (19) a suitable step size for the corrector can be obtained. Nevertheless, solving Problem (19) would directly impact on the efficiency of EDS method. Instead, a more efficient backtracking-like strategy has been adopted.

Assume we are given ϵ and an initial step size t_0 (usually the last suitable step size computed). We test whether t_0 fulfills the constraints of Problem (19), if it does we take t_0 as step size for the corrector, otherwise we shrink t_0 and try again until a suitable step size t_0 is found or until t_0 is bellow a user defined threshold. In case t_0 is suitable at the first try we set $t_1 = 2t_0$ and use it as initial guess for the next corrector step. Algorithm 2 shows how to compute a suitable step size for the corrector given a direction v in parameter space, a direction d in objective space, and an initial point x_0.

Algorithm 2 Corrector step size

Input: Current point $x \in \mathbb{R}^n$ along with its function value $F(x)$, step size guess $t \in \mathbb{R}^+$, $\nu \in \mathbb{R}^n$ direction and minimum step size threshold min_t.
Output: A step size t, a new point $x = x + t\nu$ along with its function value $F(x)$, and a flag b_first indicating whether step size h was suitable at first try.

Compute a random shrinking factor $\rho \in [0.1, 0.6]$
$b\_cont = 1$
$b\_first = 1$
$x\_init = x$
while $b\_cont \neq 0$ **do**
    $x = x\_init + t\nu$
    Evaluate $F(x)$
    if $F(x)$ satisfies the constraints of Problem (19) **then**
        $b\_cont = 0$
    else
        Shrink t by ρ factor, i.e., $t = \rho t$
        $b\_first = 0$
    end if
    if $t <$ min_t **then**
        $t = 0$
        $b\_cont = 0$
    end if
end while
return $[x, F(x), t, b\_first]$

Finally, it is worth to mention that although the value of ϵ is a user defined and problem dependent parameter, experimental results have shown that $\epsilon \in [0.6, 0.9]$ improve the overall performance of the EDS method.

3.3.2 Determining Critical Points

Now that we have a promising search direction $\nu \in \mathbb{R}^n$ and a suitable step size $t \in \mathbb{R}$ the remaining task of the corrector step is to determine whether a corrector point c is a critical point or not. Recall by Proposition 3.1 that, $\delta = 0$ when c is a critical point and that $\delta(x)$ is a continuous map. Therefore, the value of δ can be used for determining if a corrector c is indeed a critical point. Two thresholds for δ can be defined: max_$\delta > 0$ and $0 <$ min_$\delta <$ max_δ. The use of both thresholds and the entire corrector phase is described in Algorithm 3.

Algorithm 3 Compute corrector

Input: Direction $-\alpha \in \mathbb{R}^k$, predictor point $p \in \mathbb{R}^n$, thresholds min_$\delta \in \mathbb{R}^+$ and max_$\delta \in \mathbb{R}^+$
Output: A point $x \in \mathbb{R}^n$ along with its function value $F(x)$ and a flag $b \in \{0,1\}$ indicating whether x is a critical point or not.

$x = p$
while true **do**
  Compute $J(x) \in \mathbb{R}^{k \times n}$
  Compute ν, s.t. $J\nu = d$, by solving Problem (10)
  if $\delta <$ min_δ **then**
    return $[x, F(x), b = 1]$
  end if
  Obtain $[x, F(x), h, b\_first]$ by Algorithm 2
  if $h = 0$, i.e. no suitable step size could be computed **then**
    if $\delta <$ max_δ **then**
      return $[x, F(x), b = 1]$
    else
      return $[x, F(x), b = 0]$
    end if
  else
    if $b\_first = 1$ **then** $t = 2t$
    end if
  end if
end while
return $[x, F(x), b = 0]$

As it can be seen the thresholds min_δ and max_δ are critical for the robustness and performance of EDS method. Further investigation on this topic is left for future research.

Finally after a new critical point is computed, its associated weight vector α can be updated as follows ([**?**]):

$$\begin{aligned} \alpha = \arg\min_{\lambda \in \mathbb{R}^k} & \left\| \sum_{i=1}^{k} \lambda_i \nabla f_i(x) \right\|^2 \\ \text{s. t.} & \\ & \lambda_i \geq 0, \quad i = 1, \dots k \\ & \sum_{i=1}^{k} \lambda_i = 1, \quad i = 1, \dots k. \end{aligned} \tag{21}$$

We are now in the position to put predictor and corrector methods together into a continuation algorithm that we call Enhanced Directed Search (EDS). The EDS method is able to "follow" the curve (or manifold) of Pareto points of a MOP. This procedure perfectly fits into the category of PC methods. There are, nevertheless, still two issues that need to be solved before defining the algorithm of EDS: first, how to identify the parts of the manifold that have already been covered by EDS? And second, when and how should the glseds method be stopped? Answers to these questions are given in the following sections.

3.4. Determining Points Already Covered by EDS

The issue of determining points already computed by the EDS method arises already for BOPs. Recall that the predictor phase computes two predictors: one that maps to direction $d \in \mathbb{R}^k$ and another that maps to direction $-d \in \mathbb{R}^k$. This is done to ensure that by the end of EDS execution a well spread discretization of the Pareto front is obtained.

Let $x \in \mathbb{R}^n$ be a local Pareto point, also let $F(x)$. Now, assume that a set of predictors $p_+ = x + \hat{t}v$ and $p_- = x - \hat{t}v$ are computed for a fixed value of $\hat{t}$ and that from each of the predictors a new critical point can be reached by means of a corrector step. Let x_1 be one of such new critical points, shall we compute again a set of predictors and correctors with the same $\hat{t}$ from x_1 it is most likely that one of the new critical points, say x_2, will be within a "small" neighborhood of x and hence if the process is repeated with x_2 it is clear to see that a set of "repeated" points will be computed. Therefore a mechanism to avoid computing such "repeated" points should be developed.

To achieve this, it is crucial to provide an efficient method to keep track of the computed solutions. To this end, the method proposed in [**?**] is used. According to [**?**] the objective space can be divided into a set of *small* boxes where each box $\mathcal{B} \subset \mathbb{R}^k$ is represented by a center $\hat{c} \in \mathbb{R}^k$ and a radius $\hat{r} \in \mathbb{R}^k$. Therefore, the covering box $\mathcal{B}(F(x))$ of a given solution $F(x)$ can be unequivocally determined. For a complete description of the strategy used please refer to [**?**].

Figure **??** depicts the box covering algorithm for three phases of the eds method. It can be seen how as the EDS method advances more boxes $\mathcal{B}$ are added to the objective space.

3.5. Handling Box Constrained Problems

The strategy for handling box constrained problems in the EDS method is straightforward. This strategy is widely used and consists on the following: let $x \in \mathbb{R}^n$ be a box constrained vector, i.e., $lb \leq x \leq ub$, where $ub \in \mathbb{R}^n$ and $lb \in \mathbb{R}^n$ are the upper and lower boundaries vectors for x. If any of the components of x is bigger than its corresponding component in ub, i.e., $x_i > ub_i$, we make $x_i = ub_i$. The same applies if $x_i < lb_i$. In short, this strategy projects the value of the components that violate the constrains x_i to the n-dimensional box formed by ub and lb.

3.6. Stopping Criteria for the EDS Method

Our last task is to define the stopping criteria for the EDS method, which is in fact straightforward. Assume that each newly computed critical point (x) is stored in a queue $\mathcal{M}$. The EDS method takes a point x_i from $\mathcal{M}$ at each iteration and tries to compute new predictors (and hence a correctors) from this point and add the newly computed critical points to $\mathcal{M}$. It is then obvious to stop EDS whenever the queue $\mathcal{M}$ is empty. Nevertheless, if we keep adding points to $\mathcal{M}$ the queue will never be empty. Thus, criteria for adding new points to $\mathcal{M}$ should be defined.

It is indeed simple to define which points should be added to $\mathcal{M}$. Here is a list of all the conditions that will avoid a point x from being added to $\mathcal{M}$:

1. Any point whose δ value is above the user defined thresholds min_δ and max_δ
2. Any point x, whose function value $F(x)$ lies within a box that already contains a point (see Section 3.4).
3. Any point x, whose associated weight vector $\alpha \in \mathbb{R}^k$ does not comply with the condition: $max(\alpha) \leq 1 - tol$, where $0 < tol < 0.1$ is a user defined, problem dependent tolerance. This condition excludes points that lie on the boundary of the Pareto front and thus are likely to be dominated ones.

Now we have all the necessary elements to introduce EDS PC method. Its pseudo-code is presented in Algorithm 4.

Algorithm 4 EDS Predictor-Corrector method

Input: Starting local Pareto point $x_0 \in \mathbb{R}^n$ along with is associated weight vector $\alpha \in \mathbb{R}^k$, $\tau \in \mathbb{R}$ value defining the spreadness of the solutions, min_$\delta \in \mathbb{R}^+$ and max_$\delta \in \mathbb{R}^+$ thresholds and minimum step size threshold min_$h \in \mathbb{R}^+$
Output: Finite size approximation of $\mathcal{P}$ and $\mathcal{F}^*$

Set $\mathcal{P} = \mathcal{P} \cup x_0$ and $\mathcal{F} = \mathcal{F} \cup F(x_0)$
Compute $J(x)$
Enqueue($\mathcal{M}$, x_0)
while $\mathcal{M}$ not empty **do**
    $x = \text{Dequeue}(\mathcal{M})$
    Compute a set of promising directions in objective space by Eq. (11)
    for each $q_i \in i = 2, k$ **do**
        Set $d = q_i$
        Compute a set of predictors p_+ and p_- using Algorithm 1
        for each predictor **do**
            Compute a corrector x_1 using Algorithm 3
            if x_1 does not meet any of the criteria in Section 3.6 **then**
                Enqueue($\mathcal{M}$, x_1)
                $\mathcal{P} = \mathcal{P} \cup x_1$
                $\mathcal{F} = \mathcal{F} \cup F(x_1)$
            end if
        end for
    end for
end while
return $[\mathcal{P}, \mathcal{F}]$

Finally, one important aspect related to the performance of EDS method has to do with the information we keep or recompute regarding the current set of solutions. For each point in the queue $\mathcal{M}$, the corresponding function and α values are stored respectively. Furthermore, the Jacobian matrix is also stored. Storing all this information has a direct impact on the memory requirements of the method. Nevertheless, since one of our goals is to make the method efficient in terms of functions evaluations we have taken this approach to avoid recomputing information.

3.7. Using Neighborhood Information

So far, the EDS method requires Jacobian information for the computation of ν via Eq. (10). In this section, we present an alternative way to obtain such search directions ν without explicitly computing or approximating the Jacobian. Instead, the neighborhood information is exploited and used for the approximation of ν. This method can be viewed as a particular finite difference method [**?**], however, it has the advantage that the information of points already computed can be exploited in order to approximate the Jacobian matrix and hence some function evaluations can be saved. This is particularly interesting within the context of set-based optimization strategies such as MOEAs. The approach taken here was inspired by the usage of neighborhood information in the DS method [**?**] and the Gradient Subspace Approximation (GSA) method [**?**].

The general idea behind the method is as follows: given a point x_0 that is designated for local search as well as another point x_i whose function value is known that is in the vicinity of x_0, then the given information can be used to approximate the directional derivative in direction

$$\nu_i := \frac{(x_i - x_0)}{||x_i - x_0||}, \tag{22}$$

without any additional cost (in terms of function evaluations). That is, it holds

$$f_{\nu_i}(x_0) = \langle \nabla f(x_0), v_i \rangle = \frac{f(x_i) - f(x_0)}{||x_i - x_0||} + \mathcal{O}(||x_i - x_0||). \tag{23}$$

This can be seen by considering the forward difference quotient on the line search function $f_{\nu_i}(t) = f(x_0 + t\nu_i)$.

Now assume a candidate solution $x_0 \in \mathbb{R}^n$ is designated for local search and further r search directions $\nu_i \in \mathbb{R}^n$, $i = 1, \ldots, n$, are given. Then, the matrix $\mathcal{A} := JV \in \mathbb{R}^{k \times r}$, where $V = (\nu_1, \ldots, \nu_r) \in \mathbb{R}^{n \times r}$, is as follows:

$$\mathcal{A} = JV = (\langle \nabla f_i(x), \nu_j \rangle)_{i=1,\ldots,k. \quad j=1,\ldots,n}. \tag{24}$$

Hence, every element m_{ij} of JV is defined by the value of the directional derivative of objective f_i in direction ν_j, an can be approximated as in Eq. (23). An approximation to the Jacobian matrix can thus be obtained by computing

$$J \approx \mathcal{A}(V^T V)^{-1} V. \tag{25}$$

Crucial for the approximation of $\mathcal{A}$ is the choice of tests point x_i. If the function values of points in a neighborhood $N(x_0)$ are already known, it seems to be wise to include them to build the matrix V (see Figure **??**).

Nevertheless, it might be that further test points have to be sampled to obtain the n neighbors. More precisely, assume that we are given $x_0 \in \mathbb{R}^n$ as well as l neighboring solutions $x_1, \ldots, x_l \in N(x_0)$. One desirable property of all the remaining search directions is that they

are both orthogonal to each other and orthogonal to the previous ones. In order to compute the new search directions $v_{l+1}, \ldots, v_r$, $r > 1$, one can proceed as follows: compute $V = QR = (q_1, \ldots, q_l, q_{l+1}, \ldots, q_n)R$. Then it is by construction $v_i \in span\{q_1, \ldots, q_i\}$ for $i = 1, \ldots, l$, and hence

$$\langle v_i, q_j \rangle = 0, \qquad \forall i \in \{i, \ldots, l\}, j \in \{l+1, \ldots, r\}. \tag{26}$$

One can thus e.g. set

$$v_{l+i} = q_{l+1}, \qquad i = 1, \ldots, r-1 \tag{27}$$
$$x_{l+i} = x_0 + v_{l+i}, \qquad i = 1, \ldots, r-l \tag{28}$$

This way, neighborhood information can be used to approximate the value of the Jacobian. By doing this, some function evaluations can be saved by using the information computed in previously iterations. It is also important to note that the choice of r and the size of the neighborhood $N(x_0)$ has a direct impact on the quality of the method, it can be seen that for $r = k$ search directions ν_i, $i = 1, \ldots, r$, one can find a descent direction ν by solving Eq. (10) and using J as in Eq. (25) regardless of n. However, by construction it is $\nu \in span\{\nu_1, \ldots, \nu_r\}$, which means that only a r-dimensional subspace of the $\mathbb{R}^n$ space is explored in each step. One would expect that the more search directions ν_i are taken into account, the better the choice of ν is. In fact, for the development of this work $r = n$ is used. As for the size of neighborhood, we recommend $0.01 \leq N(x_0) \leq 1$, nevertheless, it must be taken into account that this is a problem dependent parameter.

3.8. Handling Mixed-Integer Problems

The EDS method is now capable of computing the connected components of MOPs with $k \geq 2$ in an efficient way. Nevertheless, our main objective in this work is the development of a continuation method for MMOPs. A simple modification can be done to the EDS method in order to make it capable, under a certain condition on the structure of the problem, of solving MMOPs.

The aforementioned condition has to do with the search space of the problem. Recall that MMOPs are defined almost exactly as MOPs except that the parameter space is defined by a mixture of real and discrete variables (see Section 2.1). The key for the mechanism that allows the EDS method to solve MMOPs is the following.

Definition 3.1 (Domain restriction) *Let $f : \mathbf{E} \mapsto \mathbf{F}$ be a function from a set $\mathbf{E}$ to a set $\mathbf{F}$, so that the domain of f is in $\mathbf{E}$ ($\operatorname{dom} f \subseteq \mathbf{E}$). If a set $\mathbf{A}$ is a subset of $\mathbf{E}$, then the restriction of f to $\mathbf{A}$ is the function*

$$f|_{\mathbf{A}} : \mathbf{A} \mapsto \mathbf{E}$$

Given two functions $f : \mathbf{A} \mapsto \mathbf{B}$ and $g : \mathbf{D} \mapsto \mathbf{B}$ such that f is a restriction of g, that is, $\mathbf{A} \subseteq \mathbf{D}$ and $f = g|_{\mathbf{A}}$, then g is an extension of f.

In this work we will only consider MMOPs whose parameter space can be *extended* to the real domain. In other words, the parameter space $\mathbf{E} = \mathbb{R}^{d_1} \times \mathbb{Z}^{d_2}$ must be a *restriction* of $\mathbb{R}^n$ for $n = d_1 + d_2$. This condition is necessary for the computation of a direction ν such that $J(x)\nu = \delta d$, since the approximation of $J(x)$ requires that samples in a small neighborhood of x can be taken (see Section 3.7). In case this sampling process is not possible, ν direction can not be computed.

Thus, the EDS method can only be used to solve MMOPs that comply with the aforementioned condition. The treatment of mixed-integer variables is achieved by the use of the following rounding operator:

Definition 3.2 (Rounding operator) *Let $d \in \mathbb{R}^k$ be a direction in objective space and let $x \in \mathbb{R}^{d_1} \times \mathbb{Z}^{d_2}$ be a point from which we would like to move in a direction $\nu \in \mathbb{R}^n$ such that $J(x)\nu = \delta d$ for a given step size $t \in \mathbb{R}$. Let also be $\mathcal{R} \subseteq \mathbb{Z}^+$ and $\mathcal{I} \subseteq \mathbb{Z}^+$ two sets of indexes, where $\mathcal{R}$ is the set of indexes that denote the real components of the vector x and $\mathcal{I}$ is the set of indexes denote the integer components of the vector x. Finally, let $\lambda \in (0, 1]$. Then, a movement from x in direction ν can be performed by*

$$round(x_j, \nu_j) = \begin{cases} x_j + t\nu_j, & if \quad j \in \mathcal{R} \\ x_j + ceil(\nu_j), & if \quad j \in \mathcal{I} \text{ and } \nu_j \geq 0 \text{ and } abs(\nu_j) \geq \lambda \\ x_j + floor(\nu_j), & if \quad j \in \mathcal{I} \text{ and } \nu_j < 0 \text{ and } abs(\nu_j) \geq \lambda \end{cases}$$

Hence, performing a movement from a point $x \in \mathbb{R}^{d_1} \times \mathbb{Z}^{d_2}$ in direction $\nu \in \mathbb{R}^n$, such that the outcome of such movement is a new point $\hat{x} \in \mathbb{R}^{d_1} \times \mathbb{Z}^{d_2}$ can be achieved by applying the *rounding* operator on each of the components of x.

The main task of the *rounding* operator, is to ensure that the point $\hat{x}$, which results from performing the line search along $\nu \in \mathbb{R}^n$, belongs to the appropriate space, that is, to the mixed-integer space in case the problem is mixed-integer or to the real space in case of dealing with a problem defined within the real space.

Recall that $\nu \in \mathbb{R}^n$ is the greediest direction in parameter space such that $J(x)\nu = \delta d$, that is, for a step size $t \in \mathbb{R}$, a movement along ν, i.e. $\hat{x} = x + t\nu$, maps to a movement $d \in \mathbb{R}^k$ in objective space. Let us assume that $x \in \mathbb{R}^{d_1} \times \mathbb{Z}^{d_2}$ (that is x is a mixed-integer variable), since $\nu \in \mathbb{R}^n$ belongs to the real space if we perform the line search as stated before, $\hat{x}$ will belong to the real space instead of belonging to a mixed-integer one. Thus, in order to keep $\hat{x}$ in the appropriate space we have developed the *rounding* operator.

We will now explain the role of the threshold λ. By considering that each of the components of ν determines how much to move in each coordinate, it can be seen that the bigger the magnitude of a certain coordinate direction, the greater the reason to increase/decrease such component in the new point $\hat{x}$. Therefore, the necessity for a threshold on the magnitude of each component of the direction v arises, we have decided to call such threshold λ and although it is a user defined parameter, our experiments have shown that setting $\lambda \in [0.2, 0.4]$ usually leads to good results.

Finally, the reason for choosing *floor* and *ceil* functions over *round* is due to the following observation: *round* function is always equivalent to the result of the *floor* function if $\lambda < 0.5$, hence, if the user defines $\lambda \in (0, 0.5)$ the value of the i-th component will always be mapped to the smallest following integer. The situation is analogous for the *ceil* function when $\lambda \in [0.5, 1)$. This is of course an undesirable behavior according to the above discussion, thus, *floor* and *ceil* functions

are used instead of *round* since they provide more flexibility for mapping the real components of ν to an integer space.

Hence, to make the EDS method capable of solving MMOPs the application of the *rounding* operator is necessary for every new point that is computed.

3.9. An Example of the EDS Method

In the following we would like to exemplify how the various parameters of the EDS method have an impact on the performance and the reliability of the method. For this example we will use the bi-objective function described in **[?]**, which is defined as follows:

$$\begin{aligned} f_1(x) = &\quad (x_2 - a_2)^2 + (x_1 - a_1)^4 \\ f_2(x) = &\quad (x_1 - b_1)^2 + (x_2 - b_2)^4 \end{aligned} \tag{29}$$

for $a = (1, 1)$ and $b = -a$. This function has a convex Pareto front with a non-linear Pareto set. Figure **??** shows the resolution of a bi-objective problem using the EDS method, for this example we set $\tau = 2$ since our goal is to depict predictor and corrector steps. Blue dots represent the real Pareto front and set. The points computed by the EDS method are depicted in red color, green arrows represent the predictor directions while red arrows represent corrector directions. As can be observed in the picture, the predictor directions are not necessarily tangent to the Pareto set, it can also be observed that the computed points do not necessarily lie on the Pareto set but instead on the Pareto front. Also note that two predictors are computed per point, nevertheless not all of them lead to new correctors (see Section 3.4).

The computed Pareto front (PF) for this example with $\tau = 3$ can be seen in Figure **??**, a finer discretization of the objective space can be seen in Figure **??** where $\tau = 1$. An even finer discretization, where $\tau = 0.5$, is shown in Figure **??**.

The number of solutions computed for each of the τ values is displayed in Table 1. As it can be observed the lower the value of τ the more solutions we get.

τ value	Solutions
3	14
1	44
0.5	80

Table 1: Number of solutions computed for the different values of τ

The plots displayed in Picture **??** along with the results in Table 1 help to have a better understanding of the τ role.

3.9.1 On the Impact of the δ Thresholds

Here we would like to demonstrate how both δ thresholds impact on the overall performance of the EDS method. For this we will consider test problem (29) again. We set $\tau = 0.5$ and will show some combinations of the max_δ and min_δ in order to let the reader gain a bigger understanding of the role of such thresholds. Table 2 displays five different settings for the δ thresholds in the max_δ and min_δ columns along with the number of solutions, number of function evaluations and the $\triangle_2$ values for each of the combinations.

Setting	max_δ	min_δ	Solutions	Feval	$\triangle_2$	Avg. Feval/Sols
1	10^{-1}	10^{-3}	81	423	0.15	5.2
2	10^{-1}	10^{-2}	77	320	0.19	4.1
3	10^{-1}	10^{-1}	77	269	0.19	3.4
4	10^{-2}	10^{-3}	26	164	10.16	6.3
5	10^{-3}	10^{-3}	7	45	15.53	6.4

Table 2: Impact of the δ thresholds on the overall performance of the EDS method

As can be observed by the results in Table 2 the choice of the values of the δ thresholds have a large impact on the performance of the EDS method. In some cases this impact is so large that the EDS method is unable to compute further correctors and hence stops it execution as was the case for the last two settings in Table 2. Also note that, as expected, the more tight these thresholds are the more function evaluations are used per computed solution. Therefore a good balance between max_δ and min_δ thresholds has to be considered in order to obtain reliable results.

3.9.2 On the Impact of the Size of the Neighborhood $N(x)$

As mentioned in Section 3.7 neighboring information can be used in order to save function evaluations and, therefore, improve the performance of the EDS method. Here we would like to demonstrate, trough some examples, how the choice of the size of the neighborhood $N(x)$ impacts the performance of the EDS method.

Once again we will use function (29) for our examples. We set $\tau = 0.5$, max_$\delta = 10^{-1}$ and min_$\delta = 10^{-3}$. For this example we put special emphasis on the number of function evaluations, the number of neighboring solutions used in the computation and the quality of the computed solutions (measured by the $\triangle_2$ indicator). Table 3 summarizes the results obtained for five different neighborhood sizes.

First of all note that the first setting is almost same as the first setting in Table 2 but here we increased the tolerance for smaller step sizes in the corrector, hence, this setting may have slightly more correctors than the setting in Table 2, this explains with this second setting uses more function evaluations the former. Note, by the data in Table 3, that by increasing the size of the neighborhood more points can be reused, nevertheless, the bigger the size of the neighborhood is the less accurate mapping (10) and hence, the more function evaluations needed to reach a new critical point by means of the corrector phase. Note the negative impact this effect has on the quality of the solution computed. This effect also explains why although the number of neighboring solutions reused increases in the last three settings, the number of function evaluations required by the method

Size of $N(x)$	Solutions	Feval	# Neighbors	$\triangle_2$	Avg. Feval/Sols
0	80	502	0	0.153	6.2
0.01	84	458	28	0.158	5.4
0.02	80	396	163	0.140	4.9
0.03	89	357	177	0.164	4
0.04	74	344	249	0.164	4.6
0.05	78	334	268	0.171	4.2

Table 3: Impact of the size of $N(x)$ on the overall performance of the EDS method

does not improve much. In an extreme case, where the size of the neighborhood is too large, the corrector phase may not be able to compute further critical points, leading to a premature termination of the EDS method.

4. Experimental Results

In this chapter we present the numerical results obtained with the EDS method on unconstrained, box constrained and mixed-integer problems up to three objectives. The EDS method is compared against the DZZ method [**?**] and NSGA-II algorithm [**?**]. We chose to compare against the DZZ method since it is a state-of-the-art algorithm in the field of mixed-integer multi-objective optimization. As for NSGA-II, it is one of the most widely used algorithms for solving MOPs. Although we are aware that there is no "fair" comparison between NSGA-II and EDS algorithms given that the former is a *global* optimization method while the latter is of *local* nature, we consider this comparison necessary in order to show the quality of the solutions obtained by our method.

4.1. General Setting

We will now describe the general setting of our experiments. DZZ version used in this experiment was provided by Dr. Honggang Wang from Rutgers University, being this the better version up to date. NSGA-II algorithm was downloaded from [**?**] which is version of NSGA-II with support for mixed-integer problems [**?**]. For each of the problems a similar number of solutions was computed by each algorithm in order to make a fairer comparison. For the case of NSGA-II a budget of approximately four times the number of function evaluations spent by EDS was assigned as stopping criteria. In the case of the DZZ method the First Pareto Solution (FPS) is given in order to restrict the comparison to the continuation method (Zig and Zag steps). For each of the test problems the three algorithms were run ten times and the better solution of each of them is used for the comparison.

For each of the solved problems a table summarizing key information for the comparison is presented. We put special emphasis on the number of function evaluations used by each algorithm displayed in the *Feval* column and on the value of the $\triangle_2$ and $\triangle_3$ performance indicators [**?**]. A ratio of function evaluations per solution is shown on the column *Funeval/sol*.

Finally, for the computation of the $\triangle_p$ performance indicator the real Pareto front was used in all of the cases except for the last function presented in this section, where the Pareto front was approximated by performing ten *long* runs of the NSGA-II algorithm and keeping the

non-dominated points among all of the runs.

4.2. DTLZ2 Continuous Problem

We now consider the tri-objective problem DTLZ1[**?**] which is a multimodal function with box constraints. It is defined as follows:

$$\begin{aligned} f_1(x) = &\quad \frac{1}{2}x_1x_2(1+g(x)) \\ f_2(x) = &\quad \frac{1}{2}x_1(1-x_2)(1+g(x)) \\ f_3(x) = &\quad \frac{1}{2}(1-x_1)(1+g(x)) \\ g(x) = &\quad 100(10+\sum_{i=3}^{n}(x_i-0.5)^2 - cos(20\pi(x_i-0.5))) \\ s.t. &\quad 0 \leq x_i \leq 1, \quad i=1,\ldots,n. \end{aligned} \tag{30}$$

For this example we set $n = 3$. As in the previous example DZZ is not applicable to this function, hence the comparison is only made between the EDS method and NSGA-II algorithm. Results are shown in Table 4.

Algorithm	Int. Var.	Real Var.	Sols	Feval	$\triangle_2$	$\triangle_3$	Funeval/Sol
EDS	0	3	870	5991	0.0085	0.0082	7
NSGA-II	0	3	900	23400	0.0093	0.0094	26

Table 4: Summarized results for DTLZ1 function

Once again, as indicated by the results in Table 4 the EDS method performs better than the NSGA-II algorithm, despite the latter uses about four times more function evaluations. Figure **??** shows the Pareto fronts computed by both methods.

4.3. ZDT1 Integer Function

Now we test on a discretized version of the ZDT1 function which proposed by Dr. Wang for his experiments with the DZZ method on [**?**]. We will use the same setting as the one proposed in [**?**], the function is defined in the following way:

$$\begin{aligned} f_1(x) = &\quad \frac{x_1}{100} \\ f_2(x) = &\quad g(x)\left(1-\sqrt{\frac{f_1(x)}{g(x)}}\right) \\ g(x) = &\quad 1+9\cdot\frac{x_2-x_1{}^2}{100} \\ s.t. &\quad 0 \leq x_i \leq 100, \quad i=1,\ldots,n. \end{aligned} \tag{31}$$

where $n = 2$ and $x \in \mathbb{Z}^n$. This function has a convex Pareto front, the solution set is contained within the domain $[0, 100] \times [0, 100]$. The results for the three methods are displayed in Table 5.

Algorithm	Int. Var.	Real Var.	Sols	Feval	$\triangle_2$	$\triangle_3$	Funeval/Sol
EDS	0	2	122	268	0.0168	0.0324	2
NSGA-II	0	2	120	1080	0.0165	0.0295	9
DZZ	0	2	100	400	0.0100	0.0214	4

Table 5: Summarized results for ZDT1 Integer function

It can be observed by the results presented in Table 5 that the three methods computed good solutions, being the $\triangle_p$ values of the three quite similar. Nevertheless, it can also be observed that the EDS method was the most efficient of the three methods, using up to four times less function evaluations than NSGA-II and half the number of function evaluations of the DZZ method. The Pareto fronts obtained by each method are shown in Figure **??**.

4.4. ZDT2 Integer Function

Our next test is conducted on a discretized version of the ZDT2 function proposed in [**?**]. Once again we will use the same setting as the one proposed in [**?**]. The function is defined in the as follows:

$$\begin{aligned} f_1(x) = &\quad \frac{x_1}{100} \\ f_2(x) = &\quad g(x) \cdot \left(1 - \sqrt{\frac{f_1(x)}{g(x)}}\right)^2 \\ g(x) = &\quad 1 + \left(\frac{x_2}{100}\right)^{\frac{1}{4}} \\ s.t. &\quad 0 \leq x_i \leq 100, \quad i = 1, \ldots, n. \end{aligned} \tag{32}$$

where $n = 2$ and $x \in \mathbb{Z}^n$. This function has a concave Pareto front, the solution set is contained within the square $[0, 100] \times [0, 100]$. The results for the three methods are displayed in Table 6.

Algorithm	Int. Var.	Real Var.	Sols	Feval	$\triangle_2$	$\triangle_3$	Funeval/Sol
EDS	0	2	24	49	0.044	0.065	2
NSGA-II	0	2	20	200	0.15	0.21	10
DZZ	0	2	26	106	0.031	0.051	4

Table 6: Summarized results for ZDT2 Integer function

The results presented in Table 6 demonstrate, once again, that the results obtained by the EDS and the DZZ methods are similar with respect to the $\triangle_p$ indicator. Nevertheless, the EDS method uses half of the function evaluations that the DZZ method uses. NSGA-II is again outperformed by both methods. The Pareto fronts obtained by each method are shown in Figure **??**.

4.5. A Mixed-Integer Model

Finally, in this section we present one mixed-integer problem solved by the EDS method. A comparison between EDS and NSGA-II algorithms is presented. No comparison against the DZZ method is possible since this is a tri-objective. Its definition is as follows:

$$\begin{aligned} f_1(x) &= ||x - a_1||_2^2 \\ f_2(x) &= ||x - a_2||_2^2 \\ f_3(x) &= ||x - a_3||_2^2, \end{aligned} \tag{33}$$

for $a_1 = (20, \ldots, 20) \in \mathbb{R}^d$, $a_2 = -a_1$ and $a_3 = (\underbrace{20, \ldots, 20}_{\text{m times}}, \underbrace{-20, \ldots, -20}_{\text{n-m times}}) \in \mathbb{R}^d$ for $m = ceil(d/2)$, $d = d_1 + d_2$, $d_1 = 3$, $d_2 = 2$, and $x \in \mathbb{R}^{d_1} \times \mathbb{Z}^{d_2}$. This function has a convex Pareto front. Since comparison against DZZ is possible only the comparison against NSGA-II is presented. The results of such comparison are shown in Table 7.

Algorithm	Int. Var.	Real Var.	Sols	Feval	$\triangle_2$	$\triangle_3$	Funeval/Sol
EDS	2	3	434	6995	135.88	150.74	9
NSGA-II	2	3	450	27000	136.69	145.36	38

Table 7: Summarized results for Binh3 MI function

As the data in Table 7 shows, both methods, the EDS method and the NSGA-II algorithm deliver similar quality solutions. Nevertheless it is important to note that the EDS method used four times less function evaluations than NSGA-II to reach the same quality of solutions. The high values on the $\triangle_p$ values of each method are due to the scale of the objective space, going up to 8000 units in one of the objective functions. The fronts computed by both methods are shown in Figure **??**.

It can be observed in the above pictures that the EDS method had some difficulties computing the narrow parts of the Pareto front, nevertheless, it can also be observed that the solutions of the EDS method are more evenly distributed than those computed by the NSGA-II. This explains why the $\triangle_p$ values of both methods are similar. The higher values of the $\triangle_3$ indicator on the EDS method are due to the fact that as p increases in the p-norm, outliers are penalized more (see [**?**]).

5. Conclusions and Future Work

Here we have presented a novel continuation method for the treatment of MMOPs. The Enhanced Directed Search (EDS), as we call our method, follows the ideas proposed by the original Directed Search (DS) method [**?**]. Nevertheless, as indicated by its name, the EDS method has several improvements over the classical DS method. Namely that the EDS method has a better, more reliable mechanism for determining critical points than the one in the DS method (the δ threshold). Furthermore, the EDS method is capable of solving MMOPs through a strategy of rounding which, according the experiments conducted in this work, proved to be efficient and effective. In

addition, the performance of the EDS method can be improved through the use of neighborhood information.

The experiments conducted here also demonstrated that the EDS method performs better than its closest competitor, the DZZ method. A major difference between these two methods is that the former can be used for problems where $k > 2$. This provides clear evidence that the EDS can be applied to problems that can be solved by the DZZ and obtain as good results as it and furthermore, that the EDS method can be applied for solving problems that are impossible to solve for the DZZ method.

For future work it is intended to test on some other corrector techniques as well as a detailed analysis of convergence of the EDS method. A version of the EDS method that does not rely so much on problem dependent parameters is also desirable. The treatment of constrained problems is equally interesting. For the latter task some ideas are being currently explored, nevertheless, they are left out of the scope of this paper. We believe this task to be key for the success of the method in real-world applications.

Another interesting topic that arises from the development of the EDS method is that of parallelization. Given the way predictors and correctors are computed (Sections 3.2 and 3.3) and thanks to the successfully implementation of the data structure proposed in [**?**] we believe that parallelizing the computation of predictors and correctors is an attainable task. A parallel implementation of the EDS method should boost the speed and the overall performance of it, allowing it to deal in a more efficient way with problems whose function evaluation time is high.

Finally, we would like to stress that the EDS method (as all continuation methods) is of local nature. It is thus conceivable to hybridize the algorithm with a global strategy such as specialized MOEAs in order to obtain a fast and reliable procedure. Furthermore, the use of neighboring information can be exploited more with the use of memetic strategies where the solutions computed by the MOEA can be reused.

All in all, we strongly believe that the EDS method has some serious potential for real-world applications, nevertheless, further experiments and some additional features are needed in order to guarantee its success when dealing with real-world problems.

References

[1] N. Z. Gebraeel, M. A. Lawley, R. Li, and J. K. Ryan. Residual-life distributions from component degradation signals: a bayesian approach. *IIE Transactions*, 37(6):543–557, 2005.

[2] M.A. Zaidan, A.R. Mills, and R.F. Harrison. Bayesian framework for aerospace gas turbine engine prognostics. In IEEE, editor, *Aerospace Conference*, pages 1–8, 2013.

[3] Z. Zhao, L. Bin, X. Wang, and W. Lu. Remaining useful life prediction of aircraft engine based on degradation pattern learning. *Reliability Engineering & System Safety*, 164:74–83, 2017.

[4] J. Lee, F. Wu, W. Zhao, M. Ghaffari, L. Liao, and D. Siegel. Prognostics and health management design for rotary machinery systems - reviews, methodology and applications. *Mechanical Systems and Signal Processing*, 42(12):314–334, 2014.

[5] W. Yu and H. Kuffi. A new stress-based fatigue life model for ball bearings. *Tribology Transactions*, 44(1):11–18, 2001.

[6] J. Liu and G. Wang. A multi-state predictor with a variable input pattern for system state forecasting. *Mechanical Systems and Sygnal Processing*, 23(5):1586–1599, 2009.

[7] A. Mosallam, K. Medjaher, and N. Zerhouni. Nonparametric time series modelling for industrial prognostics and health management. *The International Journal of Advanced Manufacturing Technology*, 69(5):1685–1699, 2013.

[8] M. Pecht and Jaai. A prognostics and health management roadmap for information and electronics rich-systems. *Microelectronics Reliability*, 50(3):317–323, 2010.

[9] J. Liu, M. Wang, and Y. Yang. A data-model-fusion prognostic framework for dynamic system state forecasting. *Engineering Applications of Artificial Intelligence*, 25(4):814–823, 2012.

[10] Y. Qian, R. Yan, and R. X. Gao. A multi-time scale approach to remaining useful life prediction in rolling bearing. *Mechanical Systems and Signal Processing*, 83:549–567, 2017.