# Multi-Phase Balance of Diversity and Convergence in Multi-Objective Optimization

Haitham Seada, Kalyanmoy Deb, *Fellow, IEEE*, and Mohamed Abouhawwash

**COIN Report Number 2018001**

***Abstract*—In multi-objective optimization, defining a good solution is a multi-factored process. Most existing evolutionary multi- or many-objective optimization (EMO) algorithms have utilized two factors: domination and crowding levels of each solution. Although these two coarse-grained factors are found to be adequate in many EMO algorithms, their relative importance in an algorithm has been a matter of great concern to many current studies. We argue that beside these issues, other more fine-grained factors are of importance. For example, since extreme objective-wise solutions are important in establishing a noise-free and stable normalization process, reaching extreme solutions is more crucial than finding other solutions. In this paper, we propose an integrated algorithm, B-NSGA-III, that produces much better convergence and diversity preservation. For this purpose, in addition to emphasizing extreme objective-wise solutions, B-NSGA-III tries to find solutions near intermediate undiscovered regions of the front. Recent theoretical developments are also exploited to identify and improve poorly converged non-dominated solutions. B-NSGA-III addresses critical algorithmic issues of convergence and diversity-preservation directly through recent progresses in literature and integrates all these critical fine-grained factors seamlessly in an alternating phases scheme. The proposed algorithm is shown to perform better than a number of commonly used existing methods.**

***Keywords*—Multiobjective, Evolutionary, Local Search, KK-TPM**

## I. Introduction

Balancing convergence and diversity has been drawing researchers' interest since the first ever multiobjective optimization algorithm [1]. Most early studies gave higher priority to convergence over diversity [2]. Gradually, researchers started to realize that strictly following this specific order might be too restrictive [3]. In this section we will discuss a selected set of notable efforts showing the progression of research in this topic.

### A. Towards a Better Balance

In 2001, Deb and Goel proposed Controlled Elitist NSGA-II [4]. Their algorithm used a geometric distribution with a user defined parameter $r$ that caps the number of allowed individuals in each front, in an exponentially decreasing order. Their approach is intended to preserve diversity by keeping a larger number of fronts represented in the population at all times. They showed that their approach can use this diversity preservation scheme to enhance convergence as well. A few years later, Bosman and Thierens discussed several ways by which EMO algorithms performs exploitations and exploration of both proximity (convergence) and diversity [5]. In their study they stated that "the exploitation of diversity should not precede the exploitation of proximity". Yet they warned that delaying diversity preservation can result in finding only discontinuous sections of the Pareto front instead of showing the entire trade-off. They also suggested a parameter-based approach to control how much emphasis is put on diversity.

From a different perspective, the desired balance can be attained through variation operators. The study conducted by Tan et al. [6] explored this idea in the context of binary chromosome representation. They proposed an Adaptive Variation Operator (AVO). AVO utilizes the chromosomal structure to tune crossover and mutation rates of each gene independently. Genes representing most significant bits are assigned higher rates in the beginning of optimization to encourage exploration. These rates decrease (either linearly or non-linearly) as optimization proceeds. The opposite is applied to genes representing the least significant bits, in order to encourage exploitation at the final stages of optimization. Thus, emphasizing diversity (a consequence of exploration) then moving gradually towards emphasizing convergence (a consequence of exploitation). They also combined crossover and mutation in a way intended to prevent mutation from disrupting the flow of information created by crossover. Despite the restricted context of this study, it represents the converse of the commonly adopted convergence-first idea at the time. However, It is important to emphasis that their results are based on the assumption that a chromosome is the direct binary representation of a number. Consequently, their conclusion cannot be extended, even over binary chromosomes in general.

Due to the recent success of many-objective optimization algorithms ( $> 3$ objectives) [3], [7], [8], [9], balancing convergence and diversity becomes even more challenging than before. As the number of objectives grows, the percentage of non-dominated solutions increases significantly. Consequently, one of the most widely used

Haitham Seada and Kalyanmoy Deb are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, 48824 USA. e-mails: {seadahai,kdeb}@msu.edu

Mohamed Abouhawwash is with the Department of Mathematics, Faculty of Science, Mansoura University, Egypt. e-mail: saleh1284@mans.edu.eg

Supplementary material of this study can be found at www.coin-laboratory.net/bnsga3

converging forces – non-domination – becomes ineffective. Although this effect can be generally considered a disadvantage, it can be beneficial in the realm of many objectives. As the dimensionality of the objective space increases, EMO algorithms become more prone to loosing parts of the Pareto front due to premature convergence. In such cases, maintaining mild convergence pressure allows for more exploration, which in turn can result in better diversity. This is the reason behind the reduced selection pressure adopted in NSGA-III [7].

In this context, Ke Li et al. proposed an interesting extension to MOEA/D [10]. Their approach treats subproblems and solutions as two different types of agents. Each subproblem creates a preference list of all available solutions sorted by their performance in the employed scalarization function (which is a metric of convergence in this specific subproblem). On the other hand, each solution creates a preference list of all available subproblems sorted by the perpendicular distance between the solution itself and each of them (which signifies diversity). After each agent – from both sides – creates its own preference list, a stable matching algorithm [11] dictates the final subproblem-solution associations. Those selected solutions move to the next generation.

Researchers continued to experiment with MOEA/D in the last few years. Wang et al. proposed a modification of its replacement strategy (Global replacement) [12]. The original version of MOEA/D simply assumes that a solution coming out of a specific subproblem can only replace another solution in the vicinity of this subproblem. Their approach on the other hand looks for the subproblem at which the new solution fits best, and performs replacement in the vicinity of this subproblem instead of the original one, hence the name "Global replacement". They showed how this simple modification performs better on a set of test problems. Another recent study by Yuan et al. [13] proposed a modified version of MOEA/D that achieves better balance. In their study they use a slightly relaxed mating restriction (controlled by a probability parameter $\delta$). Their approach – MOEA/D-DU – utilizes the perpendicular distances between solutions and directions ($d_2$) for enhanced diversity. Unlike the Penalty Boundary Based Intersection MOEA/D (MOEA/D-PBI) they are not incorporating $d_2$ in the scalarization function itself. They are still using Tchebycheff scalarization function. Instead, they compare each new solution (from subproblem $F_j$) with a non-decreasing sorted list of the $K$ closest solutions to $F_j$ (sorted by $d_2$). If the new solution $y$ has a better Tchebycheff value than solution $x_k$, $y$ replaces $x_k$ immediately. It is worth noting that MOEA/D-DU is a steady state algorithm. The study also included a similar extension of EFR [14].

Most of these studies do not propose truly *automatic* balancing approaches. We can generally classify them into two categories. The first category follows a predefined preference scheme to achieve the desired balance, either by focusing on convergence then diversity, or doing the opposite! Although following either way can show some merit on a selected set of problems, none of the two approaches can be considered appropriate in general. In addition, many problems would not fit in any of these two extremes. A parallel emphasis would be more robust and predictable over the whole spectrum of optimization problems. This parallel approach is adopted by the second category, however, this category uses an additional user-defined parameter to indicate the relative effort put to either convergence or diversity. Now it is the user's responsibility to find the right value for this parameter, which is a very challenging task given a new – possibly black-box – optimization problem. Thus, none of these approaches can be considered truly *automatic*.

One of the contributions of this study is the infinite seamless alternation of phases it follows, in order to reach the desired balance without adding explicit preferential parameters.

### B. Local Search

In a multi-objective optimization context, *local search* (LS) refers to optimizing an aggregate (combined) form of all the original objective functions. Several studies used LS in EMO. Ishibuchi et al. started this interesting combination in IM-MOGLS [15]. Their approach uses a simple weighted-sum aggregate (scalarization) function to combine all objectives. IM-MOGLS then starts an LS from each point in the population. Because of the possibly large amount of function evaluations consumed by each LS, another study proposed a modification where only some points are selected to start an LS [16]. After the proposal of NSGA-II Ishibuchi and Narukawa proposed an NSGA-II extension that uses LS to solve multiobjective 0/1 knapsack problems [17]. Other researchers followed the same path by adding LS to already existing EMO algorithms. Knowles and Corne proposed a similar extension to PAES [18].

Harada et al. proposed the notion of a Pareto descent direction, which is a direction "to which no other descent directions are superior in improving all objective functions" [19]. Their algorithm, Pareto Descent Method (PDM), solves a Linear Programming (LP) problem to get each of these directions. However, PDM is hardly considered an evolutionary algorithm. It does not involve any genetic operators either for interaction among individuals (i.e. recombination) or for randomly modifying each individual independently (i.e. mutation). PDM is rather an Evolutionary Strategy (ES)-like algorithm where mutation is replaced by an LS in the Pareto descent direction of each solution. Later, Bosman proposed and tested several multiobjective gradient-based algorithms in [20]. Starting at some solution, he also provided an analytical representation of the directions in which objective values can not deteriorate (they either improve or remain constant). And although Bosman's work considers EMO algorithms for hybridization, it need not be limited to the evolutionary domain.

The reader can notice that different studies used different ways to combine objectives. Some used Achievement

Scalarization Functions (ASF) [21], while others tried to reach a weighted combination of the objectives based on finding a promising direction that may improve their values. Using both approaches is dependent on the underlying search directions/weights. However these directions may harm diversity if they are calculated based on potentially decreasing objective values only. In addition, just using a weighted sum approach yields the algorithm unable to attain non-convex sections of the Pareto front. We recommend using the former approach, ASF, as a general way of combining objectives in any algorithm, as it theoretically enables the algorithm to reach any point, regardless of the convexity of the region at which this point lies on the front. In addition, given a reference directions based algorithm like B-NSGA-III, ASF can use those readily available directions.

On the other hand, we also noticed that using ASF can significantly distort the contour lines of the aggregate function if the direction is too flat or steep. Consequently, ASF shows poor performance when used to reach objective-wise extreme points. This means that ASF is not a universal formulation that can be used unconditionally.

In this study, we are not concerned with the specific single objective optimization algorithm used by the single objective optimizers used in these studies. We are more concerned with the formulation of the aggregate function itself, and how LS can employed in the course of a bigger algorithm, in order to serve the ultimate purpose of automatically balancing convergence and diversity. One of the contributions of this study is using different formulations according to the current state/phase of the optimization process.

### C. Karush Kuhn Tucker Proximity Measures

Since its introduction, Karush Kuhn Tucker (KKT) conditions have been used extensively by researchers [22], [23], [24], [25]. KKT conditions are necessity conditions that each optimal point must satisfy. The opposite is not true, however. A KKT point is not necessarily optimal. Despite this fact many researchers and even popular softwares use some form of KKT conditions to check the optimality of their solutions. Thus, it would be more accurate if we said that these studies/softwares search for KKT points, instead of actual optimal points. To ensure optimality another set of more involved conditions (sufficiency conditions) should be used. But, sufficiency conditions are more difficult to apply in real world problems. For example, KKT conditions are sufficient (ensure global optimality) if the problem satisfies moderate convexity assumptions, a rare case in practice.

Another problem researchers face is that KKT conditions are "singular" i.e. they hold only at KKT points. They do not provide any clue regarding how far an arbitrary solution can be from being a true KKT point. Thus in their original form, KKT conditions cannot be used as a convergence metric. Dutta et al. [26] tackled this problem and proposed their KKT proximity metric (KKTPM) for single objective optimization problems. The key to overcome the singularity problem was to relax the complementary slackness condition of the original KKT formulation. Deb and Abouhawwash [27] extended this work to multi-objective optimization. The computational complexity of these approaches remained an obstacle towards using KKTPM efficiently in optimization, as each single KKTPM calculation required solving a quadratic programming problem to get the corresponding Lagrange multipliers. In order to overcome this obstacle, the authors proposed *approximate* KKTPM which is still reliable yet much faster [28]. In this study, we use *approximate* KKTPM to identify weakly converged solutions, then try to help them improve.

In a precursor study, [29], we proposed a preliminary version of our multi-phased algorithm. Here we extensively modify and extend the algorithm as follows. In [29], too much resources were wasted trying to fill *uncoverable* gaps in problems having discontinuous Pareto optimal fronts. Our modified algorithm, B-NSGA-III, never *re-tries* to cover a gap unless a new promising starting point is found during evolution. B-NSGA-III uses a normalized version of the Biased Weighted Sum Local Search (BWS-LS) objective function proposed in [29]. This modification allows for using a fixed value for $\epsilon$ regardless of the dimensionality (number of objectives) of the problem in hand. We also observed that the way [29] picks a *possibly dominated* starting point for local search in *Phase-2* wastes Solution Evaluations (SEs). Here, such a starting point is restricted to be non-dominated. B-NSGA-III also has the option of using numerical partial derivatives to calculate KKTPM in problems having non-differential objectives and/or constraints. In addition, our simulations and results have been extended significantly to adequately test the proposed idea. Finally, B-NSGA-III is now available as an Open Source Software for researchers willing do further investigate this path[1].

## II. Proposed B-NSGA-III

B-NSGA-III retains the general outline of U-NSGA-III [9]. Starting with a randomly generated initial population, B-NSGA-III generates an equal number of offspring individuals (solutions/points) using niche-based tournament selection, Simulated Binary Crossover and polynomial mutation [30]. The two populations are then combined and the ideal point is updated. The combined population goes through non-dominated sorting [31] and the next population is formed by collecting individuals front by front starting at the first front. Since population size is fixed, the algorithm will typically reach a situation where the number of individuals needed to complete the next population is less than the number of individuals available in the front currently being considered. B-NSGA-III collects only as many individuals as

[1] https://www.coin-laboratory.com/evolib

it needs using a *niching* procedure. This niching procedure normalizes the objectives of all fronts considered. Then, using a fixed set of evenly distributed reference directions (in the normalized objective space) preference is given to those solutions representing the least represented reference directions in the objective space so far. Interested readers are encouraged to consult [9] for more details.

U-NSGA-III maintains a constant preference of convergence over diversity. A solution in front $n+1$ will never be considered for inclusion in the next population unless all solutions in front $n$ are already included. This convergence-always-first scheme has been recently criticized by several researchers [3], [32]. B-NSGA-III breaks this constant emphasis on convergence. Every $\alpha$ generations, the proposed algorithm changes its survival selection strategy, by favoring solutions solely representing some reference directions over other – possibly dominating – redundant solution. A redundant solution is a solution that is not the best representative of its niche i.e. there exists another solution – in the same front – that is closer to the reference direction representing their niche.

Another difference between U-NSGA-III and B-NSGA-III is that U-NSGA-III treats all individuals/regions of the search space equally at all generations. And although it might seem better from a generic point of view, we claim the opposite. One of the most important resources in optimization is the number of function evaluations (FEs) consumed to reach a solution. In a multi-objective optimization scenario, we use the term Solution Evaluation (SE) instead, as evaluating a single solution involves evaluating more than one function. Being fair the way U-NSGA-III is, can lead to wasting SEs on easy sections of the Pareto front. Those wasted SEs could have been put into better use, if they were directed towards reaching more difficult sections of the front. Obviously, in order to achieve the maximum possible utilization of SEs, a truly dynamic algorithm that gives more attention to more difficult sections/points of the front is needed. However, designing such an algorithm needs an oracle that knows deterministically the difficulty of attaining each point on the front relative to others. Unfortunately, for an arbitrary optimization problem, perfecting such an oracle is a far-fetched dream so far, despite some studies [33]. However, recent studies show some clues that can drive creating a non-deterministic version of the targeted oracle. We summarize these clues in the following points:

1) Researchers have repeatedly shown the important role normalization plays especially in achieving better coverage of the Pareto front. Usually, the extreme points of the current population dictate normalization parameters. During evolution, as new extreme points appear all previously normalized objective values become outdated, and normalization is repeated. Hence, the importance of extreme points in optimization. And as pointed in [34], the earlier we reach extreme points, the better normalization we have and the better coverage we attain.
2) Reaching some parts of the Pareto front may require more effort than others. Several test as well as real world problems exhibit such behavior [35], [36]. Usually such difficult regions appear as gaps in the first front. In reference directions based optimization algorithms (like MOEA/D, NSGA-III and U-NSGA-III), gaps can be identified by looking for reference directions having no associations so far.
3) In multiobjective optimization, all non-dominated solutions are considered equally good, thus deserving equal attention. This is not ideal though. Being non-dominated with respect to each other does not mean that two solutions are equally converged. The recently published approximate KKTPM now enables us to efficiently differentiate non-dominated solutions based on their proximity from local optima.

These clues are realized in B-NSGA-III through several phases. In $\alpha$ generations, the algorithm switches back and forth among three different phases. Each phase uses one some LS operator to fulfill its goal. The following subsection discusses these phases in greater detail.

### A. Alternating Phases

One naive approach is to use sequential phases. Given their relative importance the first phase may seek extreme points. Once found, the algorithm moves to subsequent phases and never looks back. But, as shown in [34], reaching true extreme points is not a trivial task. Even using LS, several optimizations might be needed to attain one extreme point. And since we can never safely assume that we have reached the true extreme points, this sequential design is not recommended. The same argument is valid for covering gaps. In an earlier generation, although your solutions may not provide the desired spread, it might be the case that there are no gaps within the small region they cover. As generations proceed and solutions expand, gaps may appear. This is a frequent pattern that is likely to repeat through an optimization run. Again, the sequential pattern is prone to failure, as we can never know if more gaps will appear in the future.

Another more involved yet simple approach is to move from one phase to the next after a fixed number of generations (or SEs). Once, the algorithm reaches the final phase it goes back to the first cyclically. This cyclic approach is more appealing, but how many generations (or SEs) to wait before switching from one phase to the next? Obviously, it is never easy to tell. In addition, using this rigid design obligates the algorithm to spend resources (SEs) in *possibly unnecessary* phases, just because it is their turn in the alternation cycle.

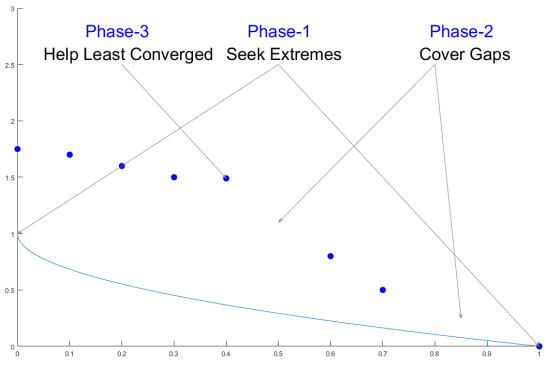



Fig. 1: *Phase-1*, *Phase-2* and *Phase-3* in action

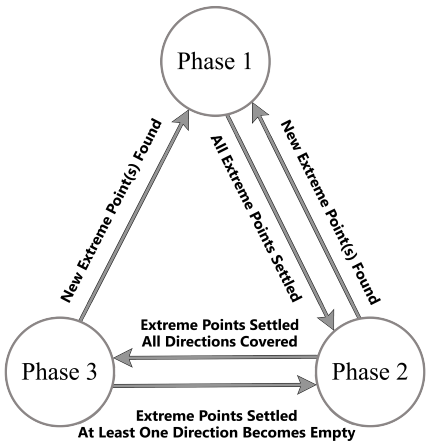



Fig. 2: Alternation of Phases

B-NSGA-III alternates among three phases *dynamically* and *adaptively*. Figure 1 shows the three phases in action. During *Phase-1*, the algorithm seeks extreme points. *Phase-2* is where an attempt is made to cover gaps found in the non-dominated front. Finally, during *Phase-3* the focus is shifted towards helping poorly converged non-dominated solutions. In order to avoid the shortcomings of the two aforementioned approaches, B-NSGA-III watches for specific incidents that trigger transitions from one phase to another. Those transitions are completely unrestricted i.e. B-NSGA-III can move from any phase to the other if the appropriate trigger is observed. Figure 2 shows all possible transitions along with their triggers.

In the first $\alpha$ generation, the algorithm puts itself in *Phase-1*. Algorithm 1 shows the details of this phase. For an $M$ objectives problem, *Phase-1* uses an Extreme-LS operator (discussed later) to search for its $M$ extreme points. If all extreme points remain unchanged from one $\alpha$ generation to the next, B-NSGA-III assumes *temporarily* that these settled points are the true extreme points, and moves to *Phase-2*. While being in phases 2 or 3, finding a better extreme point through evolution indicates that those extreme points previously settled are not the true ones. Consequently, B-NSGA-III returns to *Phase-1* in search for better extreme points again.

**Algorithm 1** Phase 1

**Input:** parent population $(P)$, offspring $(O)$ population size $(N)$, reference directions $(D)$, ideal point $(I)$, intercepts $(T)$, maximum number of function evaluations $(FeMax)$, maximum number of local search operations per iteration $\beta$, augmentation factor $\epsilon$
**Output:** None
1: $All \leftarrow P \cup O$
2: $E \leftarrow getExtremePoints(All)$
3: **for** i = 1 to $M$ **do**
4: $E_i \leftarrow localSearch_{BWS}(E_i, I, T, FeMax, \epsilon)$, $i = 1, \ldots, M$
5: $O(randomIndex) \leftarrow E_i$, $1 \leq randomIndex \leq |O|$
6: $i \leftarrow i + 1$
7: **end for**

As mentioned earlier, B-NSGA-III gives a chance to possibly dominated solutions that solely represent their niche, every $\alpha$ generations. This is shown in Algorithm 2, line 3 and expanded in Algorithm 3. The points surrounding each *non-empty* reference direction are collected from the merged population (parents and offspring) (line 2), and the best ranked point is selected to represent this direction/niche (line 4). If more than one point share the same rank, the point closest to the direction is selected (line 5). Obviously, as opposed to U-NSGA-III points in B-NSGA-III compete only with their niche peers, which means that an inferiorly ranked point from one niche, can be included because it is the best representative of its niche, while a superior point from another niche is left out because a better representative if its niche exists.

Once in *Phase-2*, B-NSGA-III looks for reference directions having no associations in the first front (*empty directions*). These directions represent gaps in the non-dominated front (Algorithm 2, line 5). If several such directions exist, one is picked randomly (line 9) and the closest first front point to this direction is saved (line 10) to be used later as a starting point in local search. Notice that lines 11 to 16 ensures that B-NSGA-III will not retry to cover a gap until a closer starting point than the one previously used is found. If no empty directions exist, B-NSGA-III moves to *Phase-3*, looking for the least converged point among those selected so far. This point should have the highest KKTPM among all (line 18). An ASF-LS operator (discussed later) is employed in both cases (line 21), either to cover a gap (*Phase-2*) or to bring a poorly converged point closer to the front (*Phase-3*). In order to keep SEs as low as possible, a maximum of $\beta$

**Algorithm 2** Phases 2 and 3

**Input:** parents ($P$), offspring ($O$), number of objectives ($M$), population size ($N$), reference directions ($D$), ideal point ($I$), intercepts ($T$), maximum number of function evaluations ($FeMax$), maximum number of local search operations per iteration $\beta$, last point used to cover direction $d$ ($prev_d$) for all $d$ in $D$
**Output:** New Population ($\hat{P}$)

$F \leftarrow nonDominatedSorting(All)$
$All \leftarrow P \vee O$
$\hat{P} \leftarrow getBestWithinNiche(d, O)\ \forall d \in D$
**if** $stagnant(E)$ **then**
  $D_{empty} \leftarrow \{d \in D \mid (\nexists x)[x \in F_1 \wedge x \in d_{surroundings}]\}$
  $s_{kktpm} \leftarrow calculateKKTPM(s) \quad \forall s \in \hat{P}$
  **for** i = 1 to $\beta$ **do**
    **if** $D_{empty} \neq \phi$ **then** ▶ Phase-2
      $d \leftarrow randomPick(D_{empty})$
      $s \leftarrow \{x \in F_1 \mid (\nexists y)[y \in F_1 \wedge \perp_d (y) <\perp_d (x)]\}$
      **if** $prev_d = null$ or $\perp_d (s) <\perp_d (prev_d)$ **then**
        $prev_d \leftarrow s$
      **else**
        $s \leftarrow null$
        $D_{empty} \leftarrow D_{empty} \setminus \{d\}$
      **end if**
    **else** ▶ Phase-3
      $s \leftarrow \{x \in \hat{P} \mid (\nexists y)[y \in \hat{P} \wedge y_{kktpm} > x_{kktpm}]\}$
    **end if**
    **if** $s \neq null$ **then**
      $\hat{s} \leftarrow localSearch_{ASF}(s, I, T, FeMax)$
      $\hat{P} \leftarrow \hat{P} \vee \{\hat{s}\}$
      $\beta \leftarrow \beta + 1$
    **end if**
  **end for**
**end if**

LS operations are allowed, even if the number of gaps is more than $\beta$. Notice the ability of the algorithm to move directly from *Phase-1* to *Phase-3* if no gaps are found.

It is worth noting that phases 2 and 3 may run simultaneously. If the number of empty directions (gaps) is less than $\beta$, B-NSGA-III moves to *Phase-3* and uses the remaining budget to help poorly converged solutions. Obviously, *Phase-1* has the highest priority followed by *Phase-2* then *Phase-3*. The following two subsections discuss both Extreme-LS and ASF-LS operators in detail.

Finally, since all the three phases are not guaranteed to completely fill the next population, a final pass is made to fill up the next population using points that B-NSGA-III has discarded so far. Algorithm 4 shows that the best ranked points – out of those not included yet in the next population – are given higher priority.

**Algorithm 3** getBestWithinNiche(...)

**Input:** merged parents and offspring ($All$), reference directions ($D$)
**Output:** selected Individuals (one from each niche) ($\hat{P}$)

$\hat{P} \leftarrow \phi$
$S \leftarrow getSurroundings(d, All)$
**for all** $d \in D$ **do**
  $X_d \leftarrow \{x \in S \mid (\nexists y)[y \in S \ \wedge y_{rank} < x_{rank}]\}$
  $x_d \leftarrow \{x \in X_d \mid (\nexists y)[y \in S \wedge \perp_d (y) <\perp_d (x)]\}$
  $\hat{P} \leftarrow \hat{P} \cup \{x_d\}$
**end for**

**Algorithm 4** fillUpPop(...)

**Input:** merged parents and offspring ($All$), population size ($N$), partially full new population ($\hat{P}$)
**Output:** completely full new population ($\hat{P}$)

$All \leftarrow All \setminus \hat{P}$
**while** $|\hat{P}| < N$ **do**
  $Z \leftarrow \{x \in All \mid (\nexists y)[y \in All \wedge y_{rank} < x_{rank}]\}$
  $z \leftarrow pickRandom(Z)$
  $All \leftarrow All \setminus \{z\}$
  $\hat{P} \leftarrow \hat{P} \vee \{z\}$
**end while**

### B. Two Local Search Operators

As mentioned earlier, B-NSGA-III uses two different local search operators. In each, all the objectives are combined into some aggregate function (scalarization). Any single objective optimizer can be used to minimize these aggregate functions. Here we chose to use Matlab's® `fmincon()` optimization routine, a point-to-point deterministic optimizer. Point-to-point optimizers use less function evaluations compared to set-based methods (e.g. evolutionary algorithms). But, they are also less guaranteed to reach global optima. Yet, in an alternating multi-phased algorithm like B-NSGA-III the embedded single objective optimizer is not expected to reach the global optimum in one shot. That's why `fmincon()` fits our criteria for an embedded single objective optimizer. Earlier we discussed the role of our two LS operators. Next we discuss their formulations and how they fit into their designated roles.

*1) Extreme-LS: Phase-1* uses Extreme-LS to find extreme points. This operator is formulated simply as a Biased Weighted Sum (BWS) aggregate function of all objectives (Equation 1). $\tilde{f}_k(x)$ represents the normalized value of objective $k$. When seeking the $i^{th}$ extreme point, the term *Biased* refers to the significantly smaller weight (we call it *augmentation factor*) multiplied by the $i_{th}$ objective, compared to the weights of all other objectives. Although, weighted sum aggregate functions are straightforward and easy to implement, they (including ours) can only reach points lying on convex sections of the Pareto front. While, this makes them less plausible as a

generic formulation, they perfectly serve their purpose in B-NSGA-III, since extreme points by definition can never lie in a non-convex section of the Pareto front. Unlike [29], the operator we are proposing here is normalized based on the number of objectives. This allows using the same *augmentation factor* for different dimensions. It is important to note that adding the *i-th* objective term to the formula helps avoiding weakly dominated points.

$$\underset{\mathbf{x}}{\text{Minimize}} \quad \text{BWS}_\text{i}(\mathbf{x}) = \epsilon \tilde{f}_i(x) + \sum_{j=1,\ j\neq i}^{M} \frac{w_j \tilde{f}_j(x)}{M-1}, \quad (1)$$

where $\epsilon$ is set as one percent of $\min_{j=1, j\neq i}^{M} w_j$.

*2) Achievement Scalarization Function LS (ASF-LS):* As mentioned earlier, a generic LS operator that is required to get an arbitrary Pareto point cannot rely on BWS. That's why we use ASF to formulate our second LS operator, ASF-LS. The formulation in Equation 2 shows that ASF-LS targets the intersection between the provided direction and the Pareto front. Since, B-NSGA-III is a reference direction based algorithm, ASF-LS can follow these already existing directions. And because of its ability to reach points lying on both convex and non-convex sections of the Pareto front, this is the operator employed in both *Phase-2* and *Phase-3* of B-NSGA-III. It is worth noting that according to our earlier experiments, ASF-LS does not perform as well if used to find extreme points. This can be attributed to the steep gradient of the aggregate ASF function (at these points) on one side of the global optimum, which usually misleads the single objective optimizer. Hence, we need both operators in B-NSGA-III, each playing its designated role.

$$\begin{aligned} &\underset{\mathbf{x}}{\text{Minimize}} \quad \text{ASF}(\mathbf{x}, \mathbf{z}^r, \mathbf{w}) = \max_{i=1}^{M} \left( \frac{\tilde{f}_i(x) - u_i}{w_i} \right), \\ &\text{subject to} \quad g_j(\mathbf{x}) \leq 0, \quad j = 1, 2, \ldots, J. \end{aligned} \quad (2)$$

## III. Results

The set of problems used here are carefully selected and/or modified to exhibit different types of difficulties. Both unconstrained and constrained problems are taken into account. Some of them have disconnected Pareto fronts. Others, challenge the convergence ability of the multiobjective algorithm, while others test its ability to cover the entire front evenly (diversity). Some of these problems are differentiable while others are not. We even modified some problems to exhibit certain types of behavior, different from what they were originally designed for. Our problems cover a wide range of dimensionality as well, namely 2, 3, 5 and 10 objectives. B-NSGA-III is compared to two other state-of-the-art reference direction based algorithms, U-NSGA-III and MOEA/D. Since the original study [3] did not mention a specific way to handle constraints, we use MOEA/D with unconstrained problems only. U-NSGA-III on the other hand is applied to all problems. Population size is kept at a small value to make a strict test of the algorithms. We use Inverted Generational Distance (IGD) and Generational Distance (GD) to test both overall performance and convergence respectively. Sometimes, we also show how KKTPM progresses during optimization. All results presented here are the medians of 31 independent runs of each algorithm on each problem. Table I shows the parameters used by B-NSGA-III in each problem. The other two algorithms use the same values for common parameters.

TABLE I: Parameters used by B-NSGA-III. $M$ is the number of objectives. $N$ is population size. SEs stands for Solution Evaluations. $\alpha$ is the frequency explained in Section II while $\beta$ is the maximum limit of function evaluations the single objective optimizer (`fmincon()`) can use. The final column shows the values of $\epsilon$ in Equation 1.

| Problem | $M$ | $N$ | SEs | $\alpha$ | $\text{FE}_{\text{max}}$ | $\epsilon$ |
|---|---|---|---|---|---|---|
| ZDT3 | 2 | 72 | 8000 | 10 | 200 | 0.1 |
| ZDT4 | 2 | 48 | 30000 | 10 | 200 | 0.1 |
| ZDT6 | 2 | 48 | 5000 | 10 | 200 | 0.1 |
| OSY | 2 | 48 | 15000 | 10 | 200 | 0.1 |
| TNK | 2 | 24 | 5000 | 10 | 200 | 0.1 |
| DTLZ4 | 3 | 36 | 7000 | 10 | 200 | 0.1 |
| DTLZ4 | 5 | 120 | 25000 | 10 | 200 | 0.1 |
| DTLZ4 | 10 | 276 | 50000 | 10 | 200 | 0.1 |
| DTLZ7 | 3 | 92 | 20000 | 10 | 200 | 0.1 |
| WFG1 | 3 | 92 | 20000 | 10 | 200 | 0.1 |

TABLE II: Each problem test one or more aspects of the algorithm

| Problem | Properties |
|---|---|
| ZDT3 | disconnected Pareto Front (PF) |
| ZDT4 | large number of local PFs, distant initial population |
| ZDT6 | biased density objective space |
| OSY | PF sections vary in difficulty |
| TNK | disconnected PF |
| DTLZ4 | biased density objective space, distant initial population |
| DTLZ7 | disconnected PF, PF sections vary in difficulty |
| WFG1 | non-differentiable, local PFs, differently scaled objectives |

Before going through the details of our simulations results, we emphasize the alternating nature of B-NSGA-III by showing it in action. Figure 3 shows a typical alternation of phases performed by B-NSGA-III while solving ZDT4 [36].

### A. Multi-objective problems

We start our experiments by solving three unconstrained bi-objective problems, ZDT3, ZDT4 and ZDT6 [35] to test the ability of our algorithm to deal with disconnected Pareto fronts, local Pareto fronts and variable density objective spaces respectively. As shown in Figure 4, in ZDT3, B-NSGA-III and U-NSGA-III achieve better convergence compared to MOEA/D. The slight

[2]Using numerical derivatives.

TABLE III: p-values of a two sided Wilcoxon rank sum test.

| Problem | M | B-NSGA-III vs. U-NSGA-III | B-NSGA-III vs. MOEA/D |
|---|---|---|---|
| ZDT3 | 2 | $3.98 \times 10^{-1}$ | $6.47 \times 10^{-11}$ |
| ZDT4 | 2 | $1.40 \times 10^{-11}$ | $1.68 \times 10^{-9}$ |
| ZDT6 | 2 | $1.40 \times 10^{-11}$ | $1.99 \times 10^{-5}$ |
| OSY | 2 | $1.51 \times 10^{-7}$ | — |
| OSY[2] | 2 | $7.57 \times 10^{-6}$ | — |
| TNK | 2 | $4.32 \times 10^{-4}$ | — |
| DTLZ4 | 3 | $2.40 \times 10^{-5}$ | $1.51 \times 10^{-7}$ |
| DTLZ4 | 5 | $1.40 \times 10^{-11}$ | $1.40 \times 10^{-11}$ |
| DTLZ4 | 10 | $6.47 \times 10^{-11}$ | $1.40 \times 10^{-11}$ |
| DTLZ7 | 3 | $1.11 \times 10^{-7}$ | $1.08 \times 10^{-4}$ |
| WFG1 | 3 | $5.38 \times 10^{-2}$ | $1.40 \times 10^{-11}$ |

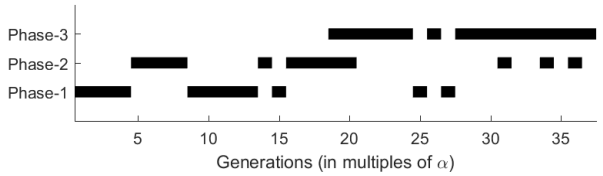



Fig. 3: A typical alternation of phases of B-NSGA-III while solving ZDT4

difference observed in IGD is not statistically significant (see Table III). In both ZDT4 and ZDT6, B-NSGA-III is a clear winner in terms of both convergence and diversity as shown in Figures 5 and 6.

The same outcome can be observed with constrained problems, see Figure 7 and Supplementary Figures S1-a, S1-b and S1-c. For these two problems, TNK and OSY [2], we exclude MOEA/D since the original study did not provide a specific way to handle constraints. Notice that for simple problems requiring no special attention to either convergence or diversity, B-NSGA-III and U-NSGA-III behave similarly, which is expected. However as problems get more difficult the merits of B-NSGA-III becomes evident. Notice, how B-NSGA-III was able to cover the entire Pareto front providing a nice distribution in Figures 5c, 6c and 7c using a limited number of function evaluations (see Table I). Both, U-NSGA-III and MOEA/D need more solution evaluations to catch up with B-NSGA-III, if they ever do.

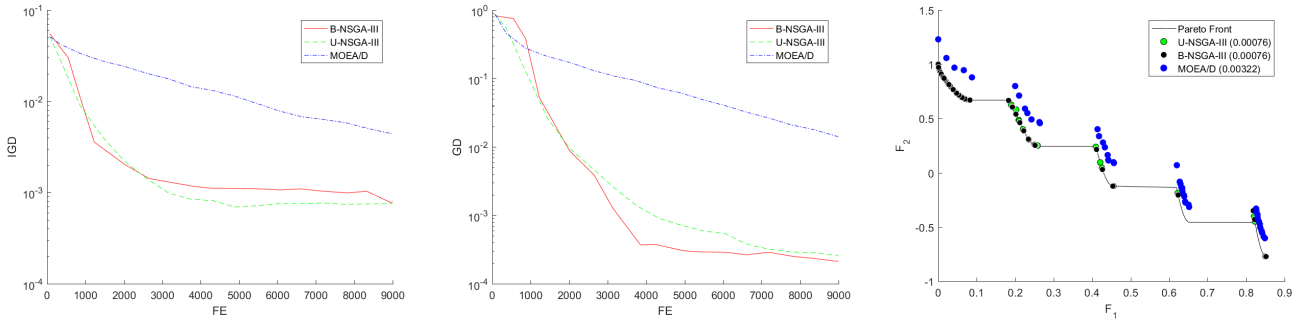
Looking at KKTPM yields another perspective that can confirm the convergence edge B-NSGA-III has. Supplementary Figures S2-a, S2-b and S2-c show how the median KKTPM of the non-dominated solutions progresses throughout optimization in ZDT4, ZDT6 and OSY, respectively. Obviously, B-NSGA-III can reach Pareto optimal points (either local or global) much faster than U-NSGA-III and MOEA/D. Combined with the previous GD plots, the reader can easily conclude that B-NSGA-III hits global Pareto optimal points much earlier than U-NSGA-III and MOEA/D. Fluctuations can be seen in initial generations (see Supplementary Figure S2-b because of the rapidly changing number of non-dominated solutions at early generations.

For three objectives, we use DTLZ4 and DTLZ7 problems [36]. DTLZ4 tests the ability of an optimization algorithm to diversify solutions in a variable density objective space. A randomly generated set of Pareto optimal solutions will be dense near the $f_M$-$f_1$ plane, and as you move away from it, solutions get sparser. Degree of density/sparseness can be controlled via the parameter $\gamma$ (called $\alpha$ in the original study and changed here not to be confused with our $\alpha$ parameter shown in Table I). However, DTLZ4 does not test the ability of an algorithm to converge. Actually, all randomly generated solutions are relatively close to the Pareto front (compared to other problems from the same family, like DTLZ1). And since B-NSGA-III is designed to tackle both convergence and diversity simultaneously, we need a problem that is able to test both capabilities at the same time. We can get such a problem by multiplying the $g(x)$ function of DTLZ4 by a constant factor $D$. The larger $D$ is, the more distant randomly generated solutions will be from the Pareto front. In this study we use $\gamma = 20$ and $D = 100$. On the other hand, the Pareto front of DTLZ7 has four disconnected regions. One region is relatively easier than the others. An optimization algorithm can easily get attracted to the easily attainable region and ignore some/all others.

Figures 8a, 8b and 8c shows the overall performance of B-NSGA-III compared to U-NSGA-III and MOEA/Don DTLZ4. B-NSGA-III is better in terms of both overall performance and convergence. Median Pareto fronts (those having median IGD values) shown in Figures 9a, 9b and 9c confirm the overall superiority of B-NSGA-III and U-NSGA-III over MOEA/D, especially in terms of diversity. The superiority of B-NSGA-III is even more evident on DTLZ7 as show in Figures 10 and 11.

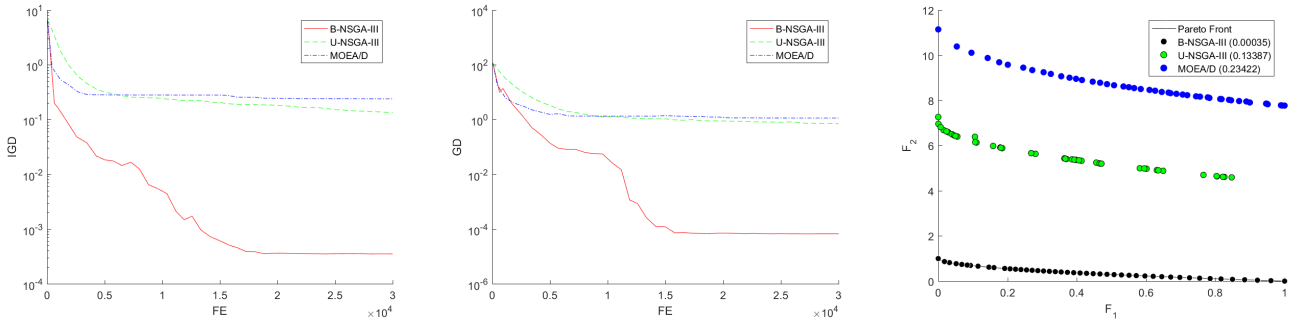
### B. Many-Objective Optimization

A many-objective optimization problem, is a problem having more than 3 objectives. Both MOEA/D and U-NSGA-III are known to be very efficient with this category of problems. In this section, we consider a difficult problem (our modified version of DTLZ4) and compare the performance of B-NSGA-III to these two powerful algorithms. Here, we use two instances of the problem, one with $5$ objectives and the other with $10$. For the $5$ objectives instance, Figure 12a is a Parallel Coordinate Plot (PCP) showing the $25\%$, median and $75\%$ quantile values of each objective among the three algorithms. The values plotted for each algorithm are taken from the final population of the run having the median IGD value of its 31 independent runs. Unmarked lines represent the benchmark quantile values of the selected Pareto optimal set. Lines marked with circles, squares and triangles represent B-NSGA-III, U-NSGA-III



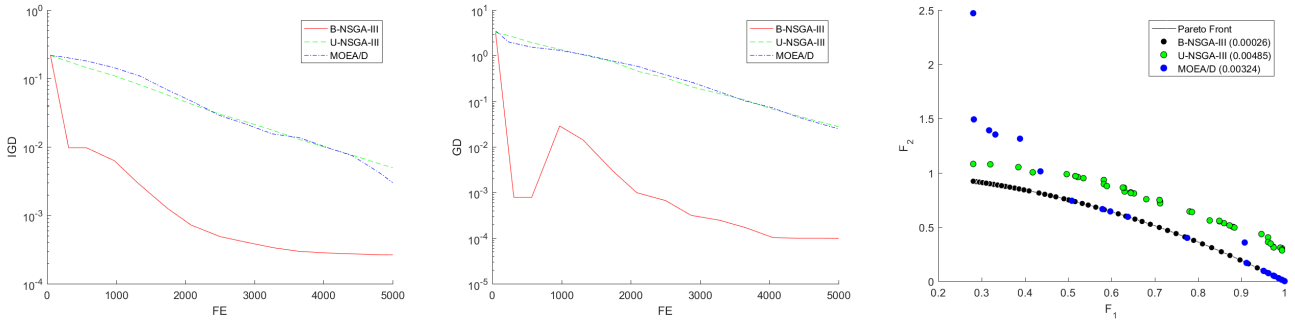

(a) IGD Comparison on ZDT3 (b) GD Comparison on ZDT3 (c) fronts Comparison on ZDT3

Fig. 4: Performance of B-NSGA-III, U-NSGA-III, and MOEA/D on bi-objective ZDT3



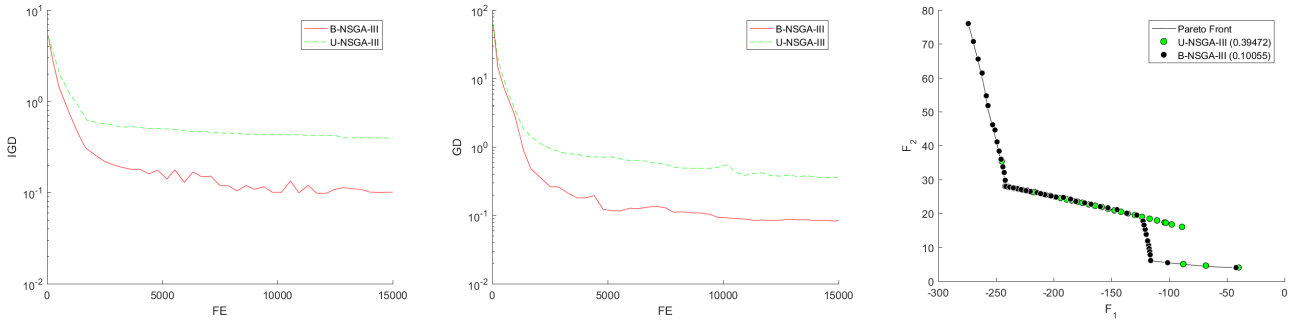

(a) IGD Comparison on ZDT4 (b) GD Comparison on ZDT4 (c) fronts Comparison on ZDT4

Fig. 5: Performance of B-NSGA-III, U-NSGA-III, and MOEA/D on bi-objective ZDT4



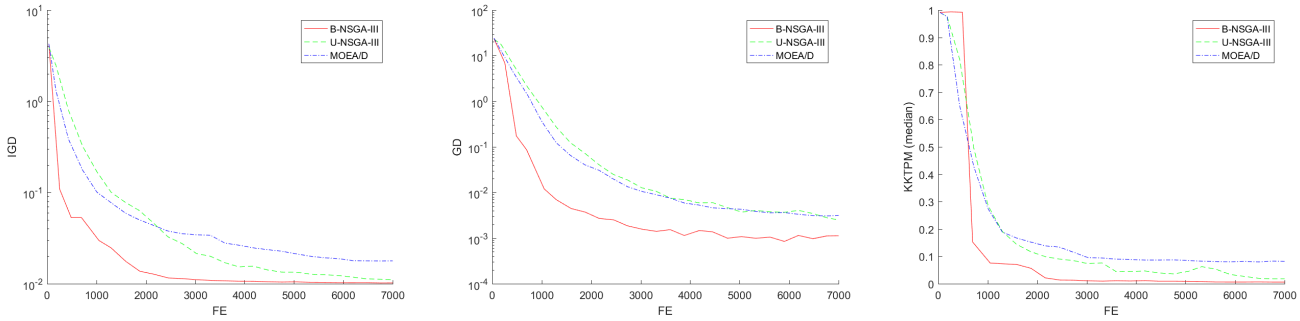

(a) IGD Comparison on ZDT6 (b) GD Comparison on ZDT6 (c) fronts Comparison on ZDT6

Fig. 6: Performance of B-NSGA-III, U-NSGA-III, and MOEA/D on bi-objective ZDT6



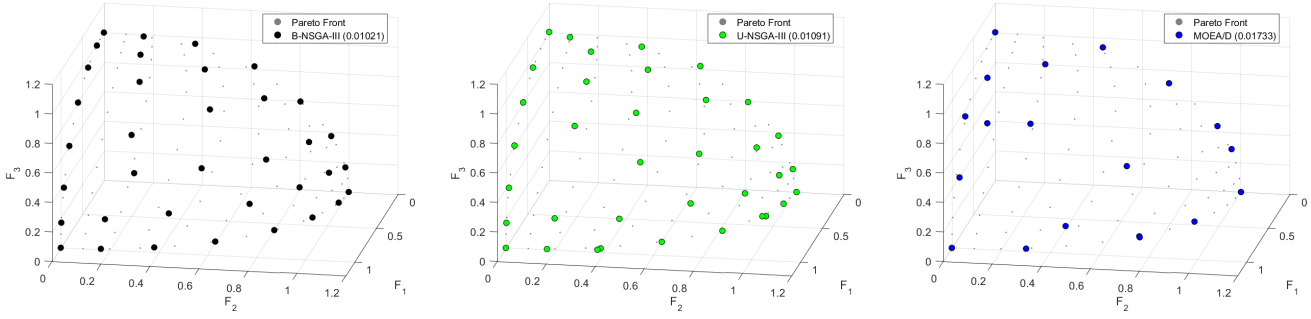

(a) IGD Comparison on OSY (b) GD Comparison on OSY (c) fronts Comparison on OSY

Fig. 7: Performance of B-NSGA-III, U-NSGA-III, and MOEA/D on bi-objective OSY



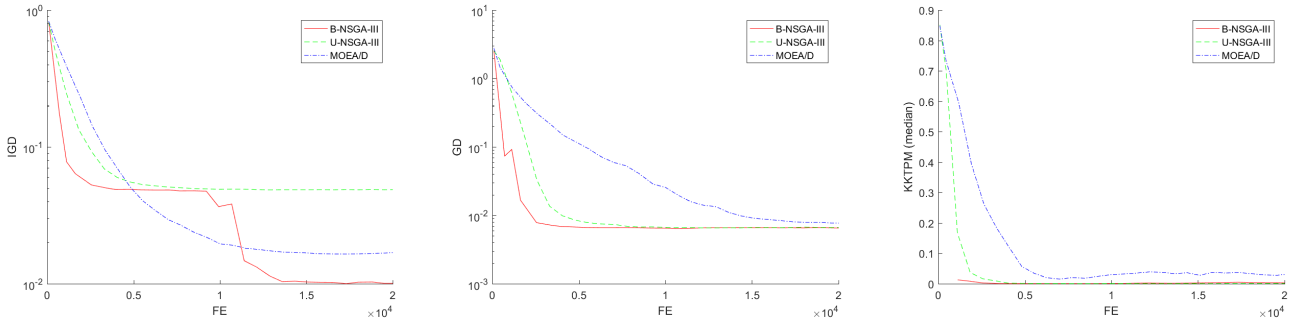

(a) IGD Comparison on DTLZ4 (3 obj.) (b) GD Comparison on DTLZ4 (3 obj.) (c) KKTPM Comparison on DTLZ4 (3 obj.)

Fig. 8: Performance of B-NSGA-III, U-NSGA-III, and MOEA/D on DTLZ4 (3 objectives).



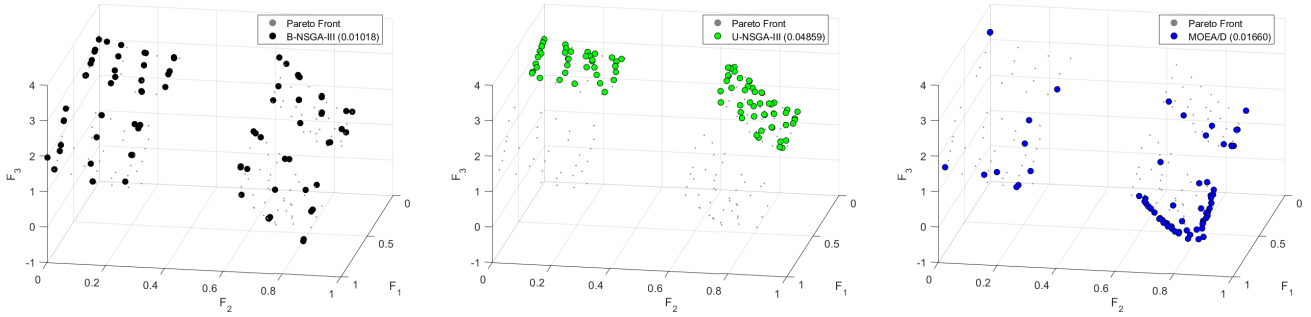

(a) B-NSGA-III Median fornt (b) U-NSGA-III Median fornt (c) MOEA/D Median fornt

Fig. 9: Median final fronts of U-NSGA-III, and MOEA/D on DTLZ4 (3 objectives).



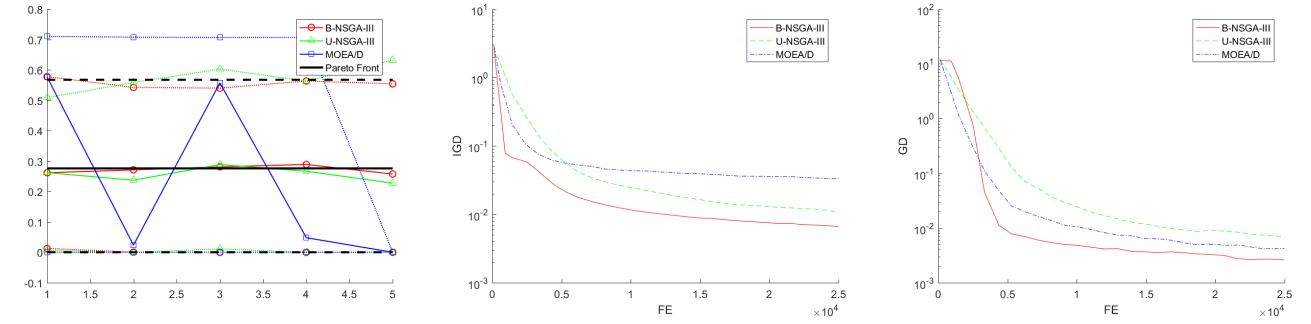

(a) IGD Comparison on DTLZ7 (3 obj.) (b) GD Comparison on DTLZ7 (3 obj.) (c) KKTPM Comparison on DTLZ7 (3 obj.)

Fig. 10: Performance of B-NSGA-III, U-NSGA-III, and MOEA/D on DTLZ7 (3 objectives).



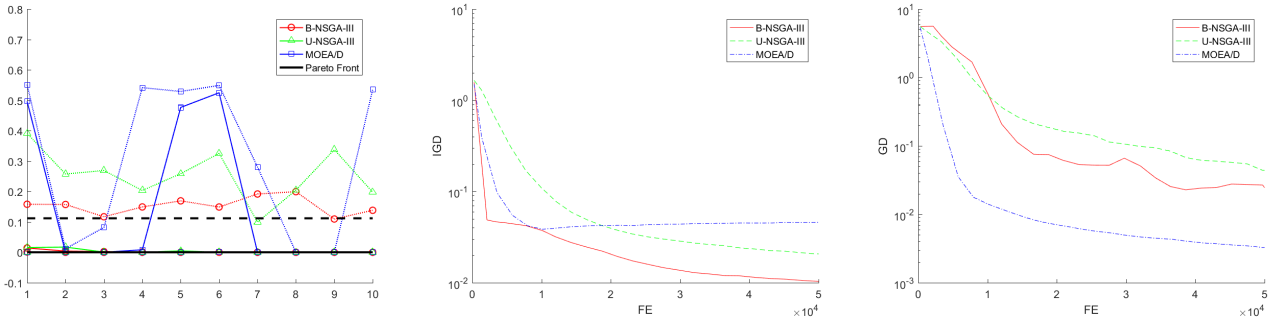

(a) B-NSGA-III Median fornt (b) U-NSGA-III Median fornt (c) MOEA/D Median fornt

Fig. 11: Median final fronts of U-NSGA-III, and MOEA/D on DTLZ7 (3 objectives).




(a) Median, 25% and 75% quartile values of DTLZ4 (5 obj.) (b) IGD comparison on DTLZ4 (5 obj.) (c) GD comparison on DTLZ4 (5 obj.)

Fig. 12: Performance of B-NSGA-III, U-NSGA-III, and MOEA/D on DTLZ4 (5 objectives).

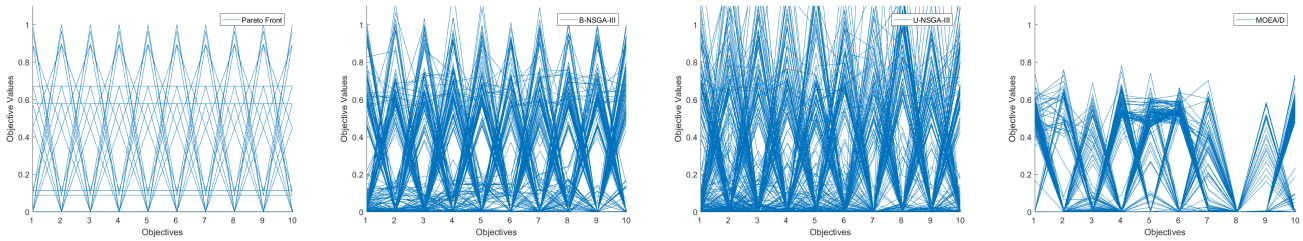



(a) Median, 25% and 75% quartile values of DTLZ4 (10 obj.)

(b) IGD comparison on DTLZ4 (10 obj.)

(c) GD comparison on DTLZ4 (10 obj.)

Fig. 13: Performance of B-NSGA-III, U-NSGA-III, and MOEA/D on DTLZ4 (10 objectives).




(a) Reference PCP (True PF)

(b) B-NSGA-III

(c) U-NSGA-III

(d) MOEA/D

Fig. 14: PCPs of B-NSGA-III, U-NSGA-III, and MOEA/D on DTLZ4 (10 objectives).

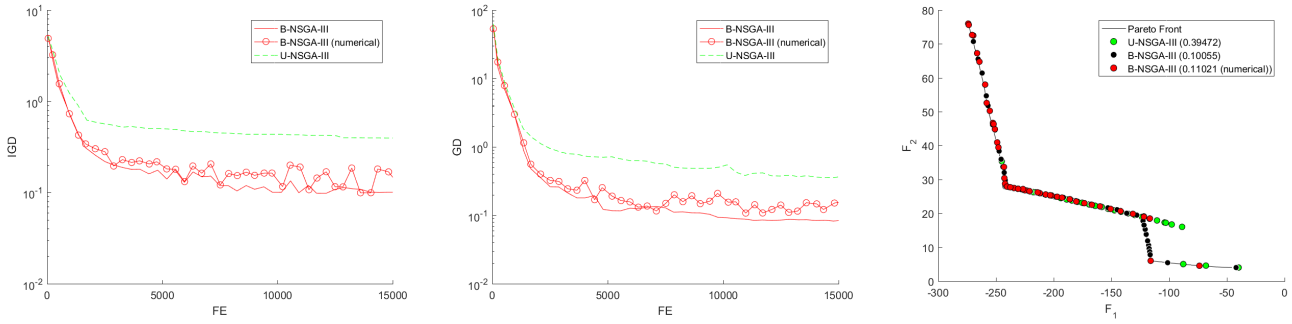
and MOEA/D respectively. The closer the lines representing one algorithm to the benchmark lines, the better this algorithm performs. All three algorithms successfully attain the 25% quantile values. However median and 75% quantile results vary. Obviously, B-NSGA-III achieves the best approximation of the two benchmark lines among the three algorithms. U-NSGA-III is still acceptable while MOEA/D misses the target lines completely. In addition, IGD and GD median comparisons are shown in Figures 12b and 12c respectively. Obviously, B-NSGA-III outperforms the other two algorithms in terms of overall performance and convergence.

The same conclusions can be drawn for the 10 objectives problem instance. Since most of the Pareto points are on the edges of the hyper simplex, the median of each objective is the same as the minimum (Zero). Figure 13a shows that unlike both U-NSGA-III and MOEA/D, B-NSGA-III achieves a close approximation of the benchmark lines. Those results are supported by IGD and GD results in Figures 13b and 13c respectively. In order to further confirm our results, we have included the PCPs of all non-dominated solutions of the median run of each algorithm in Figure 14. Obviously, the PCP of B-NSGA-III in Figure 14b is closer to the benchmark PCP (Figure 14a) than both the PCPs of U-NSGA-III and MOEA/D.

### C. Using numerical gradients

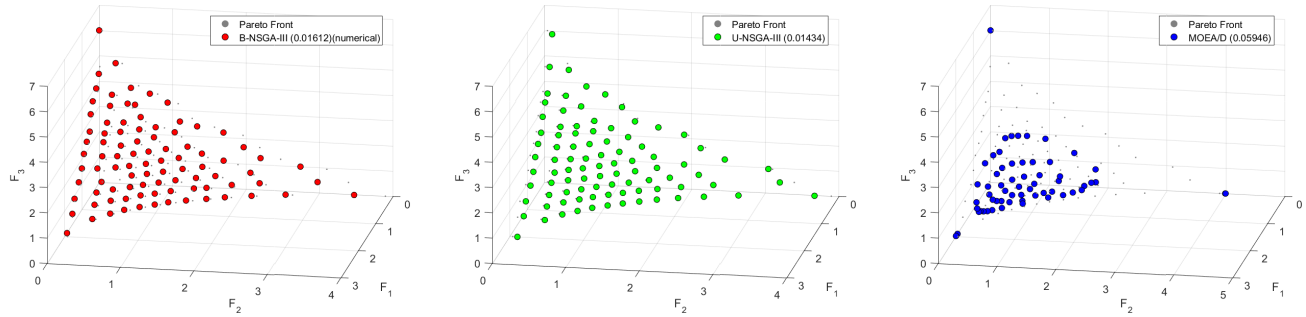
Up to this point, our approach has been using exact analytical gradients – provided by the user – to calculate KKTPM. However for many problems – especially real-world problems – these gradients may not be available in their analytical form. In such cases we resort to using numerical gradients. In most cases numerical gradients yield acceptable gradient approximations, but this comes on the expense of consuming more SEs. In order to investigate the effect of using numerical gradients on our approach, first we use numerical gradients with a problem for which we know the exact analytical gradients, namely OSY. Then we apply the same approach to WFG1 [37], a problem with non-differentiable objective functions.

Figure 15 clearly show how using numerical gradients has a minor negative effect on the performance of B-NSGA-III. As the reader can observe, B-NSGA-III using numerical gradients still outperforms U-NSGA-III. In addition, although using numerical gradients consumes more SEs, it is clear from both figures that the magnitude of sacrifice is minimal. B-NSGA-III with numerical gra-




(a) IGD comparison on OSY (b) GD comparison on OSY (c) Median runs on OSY

Fig. 15: Performance of B-NSGA-III (exact & numerical) and U-NSGA-III on OSY problem.




(a) Median B-NSGA-III run on WFG1 (b) Median U-NSGA-III run on WFG1 (c) Median MOEA/D run on WFG1

Fig. 16: Performance of B-NSGA-III (using numerical gradients), U-NSGA-III, and MOEA/D on WFG1 problem.

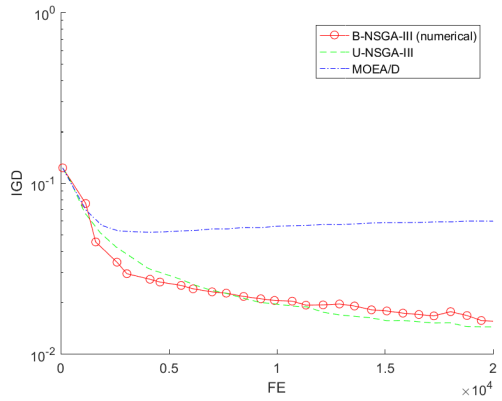



Fig. 17: IGD of the three algorithms on WFG1

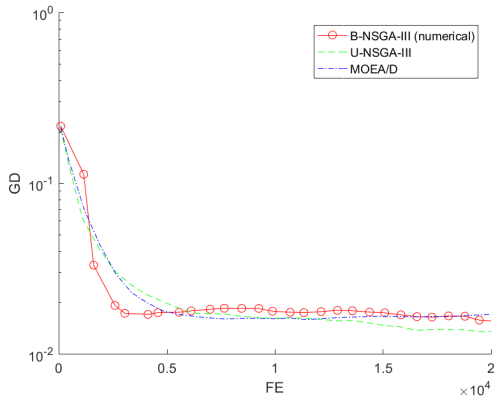



Fig. 18: GD of the three algorithms on WFG1

dients even catches up with the exact gradients version in some cases.

The reader can easily reach a similar conclusion by looking at Figures 16, 17 and 18. Here we use a slightly modified version of WFG1. We noticed that in order to reach about 100 Pareto Front points in the original version of WFG1, we can only use a code whose precision goes beyond $\approx 10^{-50}$! This value represents the difference between objective values of two neighboring points in such a front. This is true because of the very

small power used at the third transformation of WFG1 ($0.02$). To the best of our knowledge, most of the codes available are not even close to this level of precision. In order to solve this problem we changed the power of the third transformation from $0.02$ to $0.2$.

Since, all the objectives of WFG1 are non-differentiable, the only possible option for B-NSGA-III is to use numerical gradients. Even with the burden of additional solution evaluations, B-NSGA-III (with numerical gradients) still outperforms both MOEA/D and U-NSGA-III. It is worth noting that, although Figures 16a and 16b look similar, the points obtained by B-NSGA-III are more converged (closer to the front) than those obtained by U-NSGA-III. This difference in convergence can be visually observed through the profile views of the same two plots (not included due to space limitations).

The good performance of numerical gradient based B-NSGA-III can be attributed to the very conservative approach B-NSGA-III follows with KKTPM calculations, which is summarized in the following points:

- Since our modified niching approach is applied every $\alpha$ generations, KKTPM calculations are only possible in $1/\alpha$ of the total number of generations. Typically, in our case – since $\alpha = 10$ – KKTPM values may be calculated in only $10\%$ of all generations at max.
- Even when it comes to these $10\%$ of generations, *Phases-I* and *II* do not require any KKTPM calculations. And as we showed earlier, in most problems B-NSGA-III spends the majority of its time in these two phases.
- *Phase-III* is where most KKTPM calculations are required. In this phase, KKTPM values are calculated only for those individuals selected so far, which is usually less than $N$ (population size).

## IV. Conclusions

In this study we have proposed B-NSGA-III, a multi-phased many-objective evolutionary optimization algorithm capable of automatically balancing convergence and diversity of population members. B-NSGA-III does not use explicit preferential parameters or weights to distribute its effort among the two aspects. It rather waits for signals based on which it changes from one phase to the other. This alternation of phases is completely unrestricted and the exact way it happens adapts automatically to the problem in hand. Two types of local search operators are used during optimization. The first is designed to find extreme points while the second is more suited to move arbitrary points towards the Pareto optimal front. Approximate KKTPM is used to identify weakly dominated points that need to be locally enhanced. On a wide range of problems with different attributes and difficulties, B-NSGA-III shows superior performance to a number of state-of-the-art algorithms for an equal number of solution evaluations.

The specific algorithmic setup proposed in this study should be viewed as one among many possibilities by which several optimization techniques, metrics and algorithms can communicate and cooperate to reach better results. We argue that our open source implementation of B-NSGA-III can serve as a starting point for researchers willing to further exploit these possibilities towards balanced many-objective optimization.

## References


[1] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st international Conference on Genetic Algorithms*, pp. 93–100, L. Erlbaum Associates Inc., 1985.

[2] K. Deb, *Multi-objective optimization using evolutionary algorithms*, vol. 16. John Wiley & Sons, 2001.

[3] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[4] K. Deb and T. Goel, "Controlled elitist non-dominated sorting genetic algorithms for better convergence," in *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 67–81, Springer, 2001.

[5] P. A. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE transaction on evolutionary computation*, vol. 7, no. 2, pp. 174–188, 2003.

[6] K. C. Tan, S. C. Chiam, A. Mamun, and C. K. Goh, "Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization," *European Journal of Operational Research*, vol. 197, no. 2, pp. 701–713, 2009.

[7] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[8] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: handling constraints and extending to an adaptive approach," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 602–622, 2014.

[9] H. Seada and K. Deb, "A unified evolutionary optimization procedure for single, multiple, and many objectives," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 358–369, 2016.

[10] K. Li, Q. Zhang, S. Kwong, M. Li, and R. Wang, "Stable matching-based selection in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 6, pp. 909–923, 2014.

[11] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[12] Z. Wang, Q. Zhang, M. Gong, and A. Zhou, "A replacement strategy for balancing convergence and diversity in moea/d," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pp. 2132–2139, IEEE, 2014.

[13] Y. Yuan, H. Xu, B. Wang, B. Zhang, and X. Yao, "Balancing convergence and diversity in decomposition-based many-objective optimizers," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 180–198, 2016.

[14] Y. Yuan, H. Xu, and B. Wang, "Evolutionary many-objective optimization using ensemble fitness ranking," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 669–676, ACM, 2014.

[15] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 28, no. 3, pp. 392–403, 1998.

[16] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE transaction on evolutionary computation*, vol. 7, no. 2, pp. 204–223, 2003.

[17] H. Ishibuchi and K. Narukawa, "Performance evaluation of simple multiobjective genetic local search algorithms on multiobjective 0/1 knapsack problems," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 1, pp. 441–448, IEEE, 2004.

[18] J. D. Knowles and D. W. Corne, "M-paes: A memetic algorithm for multiobjective optimization," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1, pp. 325–332, IEEE, 2000.

[19] 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*, pp. 659–666, ACM, 2006.

[20] P. A. Bosman, "On gradients and hybrid evolutionary algorithms for real-valued multiobjective optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 16, no. 1, pp. 51–69, 2012.

[21] A. P. Wierzbicki, "The use of reference objectives in multiobjective optimization," in *Multiple criteria decision making theory and application*, pp. 468–486, Springer, 1980.

[22] H.-C. Wu, "The karush–kuhn–tucker optimality conditions in an optimization problem with interval-valued objective function," *European Journal of Operational Research*, vol. 176, no. 1, pp. 46–59, 2007.

[23] T. Antczak, "A new approach to multiobjective programming with a modified objective function," *Journal of Global Optimization*, vol. 27, no. 4, pp. 485–495, 2003.

[24] G. Bigi and M. Castellani, "Second order optimality conditions for differentiable multiobjective problems," *RAIRO-Operations Research*, vol. 34, no. 4, pp. 411–426, 2000.

[25] K. Miettinen, *Nonlinear multiobjective optimization*, vol. 12. Springer Science & Business Media, 2012.

[26] J. Dutta, K. Deb, R. Tulshyan, and R. Arora, "Approximate kkt points and a proximity measure for termination," *Journal of Global Optimization*, vol. 56, no. 4, pp. 1463–1499, 2013.

[27] K. Deb and M. Abouhawwash, "An optimality theory based proximity measure for set based multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, In press.

[28] K. Deb, M. Abouhawwash, and H. Seada, "A computationally fast convergence measure and implementation for single-, multiple-, and many-objective optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 4, pp. 280–293, 2017.

[29] H. Seada, M. Abouhawwash, and K. Deb, "Towards a better balance of diversity and convergence in nsga-iii: First results," in *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 545–559, Springer, Cham, 2017.

[30] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.

[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transaction on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[32] M. Li, J. Zheng, K. Li, Q. Yuan, and R. Shen, "Enhancing diversity for average ranking method in evolutionary many-objective optimization," *Parallel Problem Solving from Nature, PPSN XI*, pp. 647–656, 2010.

[33] H.-L. Liu, F. Gu, and Q. Zhang, "Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 450–455, 2014.

[34] A. K. A. Talukder, K. Deb, and S. Rahnamayan, "Injection of extreme points in evolutionary multiobjective optimization algorithms," in *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 590–605, Springer, 2017.

[35] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[36] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pp. 105–145, 2005.

[37] S. Huband, L. Barone, L. While, and P. Hingston, "A scalable multi-objective test problem toolkit," in *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 280–295, Springer, 2005.