UNIVERSITY OF PISA

MASTER'S DEGREE IN
ARTIFICIAL INTELLIGENCE AND DATA ENGINNERING

Symbolic and Evolutionary Artificial Intelligence

On the Impact of Scalarization in MOEA/D: Performance across Convex, Non-Convex, and Disconnected Pareto Fronts

Professor:

Marco Cococcioni

Student:

Giuseppe Soriano

ACADEMIC YEAR 2024/2025

Index

1. Introduction

Multi-objective optimization involves the simultaneous optimization of multiple, often conflicting, objective functions. In this context, the goal is not to identify a single optimal solution, but rather to approximate the set of Pareto-optimal solutions, each representing a different trade-off among the objectives. Evolutionary algorithms are widely adopted for this task due to their population-based nature, which allows exploration of diverse regions of the solution space in a single run.

Among the most studied approaches is the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D), which decomposes a multi-objective problem into a set of scalar optimization subproblems. Each subproblem is optimized simultaneously, typically using information from its neighboring subproblems. A key component of MOEA/D is the scalarization function, which transforms the multi-objective formulation into scalar ones. The most common scalarization techniques include the **weighted sum**, the **Chebyshev (Tchebycheff) approach**, and **boundary intersection methods**.

However, it is well known that the performance of scalarization techniques can vary significantly depending on the geometry of the Pareto front. In particular, convex scalarizations such as the weighted sum tend to perform poorly on non-convex Pareto fronts, potentially failing to identify large regions of the solution set. This motivates an empirical investigation into the behavior of different scalarization functions when applied to problems with non-convex Pareto fronts.

This work presents a comparative analysis of MOEA/D configured with two distinct scalarization strategies—Chebyshev and linear (weighted sum)—in contrast with the baseline performance of NSGA-II, a widely adopted dominance-based algorithm that does not rely on scalarization. The objective is to evaluate the strengths and limitations of each approach in addressing diverse Pareto front geometries, with a particular focus on how scalarization choices influence convergence, diversity, and overall solution quality in multi-objective optimization contexts. The comparison also includes a variant of MOEA/D where the traditional mutation operator is complemented by the Simulated Binary Crossover (SBX), aiming to investigate the impact of crossover-based genetic variation on the algorithm's effectiveness.

2. Background and Related Work

Multi-objective optimization deals with problems where two or more conflicting objectives must be optimized simultaneously. Instead of a single optimum, the outcome is a set of **Pareto-optimal solutions**, none of which can improve one objective without worsening another. The image of all Pareto-optimal objective vectors is called the **Pareto front**. Numerous algorithms, broadly termed **multi-objective evolutionary algorithms (MOEAs)**, have been developed to approximate the Pareto front. These algorithms can be categorized into different families based on how they balance the objectives:

- **Pareto dominance-based MOEAs:** Rely on the Pareto dominance relation to guide the search (e.g., NSGA-II, SPEA2, PAES).
- **Indicator-based MOEAs:** Use performance indicators (like hypervolume) to drive selection (e.g., SMS-EMOA, HypE).
- **Decomposition-based MOEAs:** Decompose the multi-objective problem into many single-objective subproblems (e.g., MOEA/D, MOGLS).

In this work, we focus on two representative algorithms: **NSGA-II** (a Pareto-based approach) and **MOEA/D** (a decomposition-based approach). These two methods are foundational in the literature and well-suited to highlight the contrast between dominance-based and decomposition-based strategies. Below, we detail the mechanics of each algorithm, including their mathematical underpinnings, and reference the seminal works where they were introduced.

2.1 NSGA-II: Non-dominated Sorting Genetic Algorithm II

NSGA-II is one of the most influential Pareto-based MOEAs, introduced by Deb *et al.* in 2002 [1] as an improvement over the earlier NSGA algorithm. NSGA-II's design addressed several shortcomings of first-generation MOEAs and became a standard baseline for multi-objective optimization. Key features of NSGA-II include:

- **Elitism:** It preserves the best solutions found so far by always keeping non-dominated individuals from generation to generation, rather than relying on an external archive.
- **Fast non-dominated sorting:** NSGA-II ranks individuals by Pareto dominance efficiently, without requiring any user-defined niche radius (unlike its predecessor).
- **Crowding distance diversity:** It uses a **crowding distance** metric to maintain solution diversity along the Pareto front, eliminating the need for a sharing parameter.
- **Simple parameter set:** NSGA-II does not introduce new parameters beyond the standard genetic algorithm ones (population size, crossover/mutation rates, etc.), making it easy to implement and tune.

Pareto Ranking (Non-dominated Sorting): In NSGA-II, the population is sorted into layers (Front 1, Front 2, etc.) based on dominance. Front 1 consists of all **non-dominated** individuals (Pareto-optimal within the population). Front 2 contains individuals that are dominated only byz individuals in Front 1, and so on. Each solution is assigned a rank (1 for the first front, 2 for the second, etc.). This ranking guides selection: lower-rank (better) fronts are prioritized for survival.

Crowding Distance: Within each Pareto front, NSGA-II computes a crowding distance for each solution to estimate how isolated it is from its neighbors in objective space. This promotes a spread-out Pareto front. The crowding distance for a solution is the sum of normalized objective function differences between its nearest neighbors on that front. For example, if solutions in a given front are sorted by objective m, the crowding distance d_i for solution i can be calculated as:

$$d_i = \sum_{m=1}^{M} \frac{f_m(i+1) - f_m(i-1)}{f_m^{\max} - f_m^{\min}},$$

where $f_m(i+1)$ and $f_m(i-1)$ are the m-th objective values of the neighboring solutions in the sorted order, and $f_m^{\max}$ and $f_m^{\min}$ are the maximum and minimum values of the m-th objective in that front. Boundary solutions (highest and lowest value in each objective) are assigned an infinite crowding distance to always preserve extreme trade-offs. A larger d_i means solution i resides in a less crowded region, which is desirable for maintaining diversity.

Selection and Elitist Survival: NSGA-II uses a combined population approach each generation. Given a parent population P_t of size N and an offspring population Q_t of size N (produced by crossover and mutation), NSGA-II proceeds as follows to form the next generation P_{t+1}:

1. **Combine Populations:** Form $R_t = P_t \cup Q_t$ (of size $2N$) and perform non-dominated sorting on R_t to identify fronts $f_1, f_2, \ldots$.
2. **Rank-based Selection:** Initialize $P_{t+1} = \emptyset$. Fill P_{t+1} with the solutions from f_1, then f_2, etc., until adding another entire front would exceed N. Let f_k be the last front that *partially* fits.
3. **Diversity Selection:** If P_{t+1} has fewer than N solutions after filling whole fronts up to f_{k-1}, sort the solutions in f_k by descending crowding distance and select the most widely spaced solutions to fill the remaining slots (up to N). This ensures a well-distributed selection from the last included front.
4. **Next Generation:** Discard the rest and set P_{t+1} as the new parent population. Then generate Q_{t+1} from P_{t+1} via genetic operators and repeat.

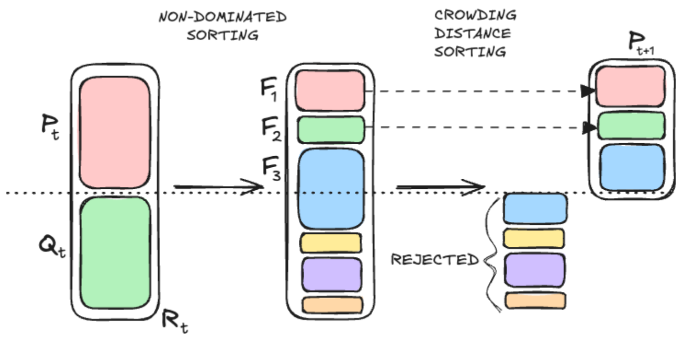


Figure 1: Schematic representation of the NSGA-II algorithm.

By always preferring lower-rank (non-dominated) solutions and secondarily those in less crowded regions, NSGA-II strikes a balance between **convergence** (favoring Pareto-optimal solutions) and **diversity** (covering the Pareto front evenly). Deb *et al.* report that these mechanisms allow NSGA-II to find a well-spread set of Pareto-optimal solutions in a single run [1]. Since its introduction, NSGA-II has been widely adopted as a benchmark and has inspired many subsequent algorithms in the Pareto-based MOEA family (including extensions like NSGA-III for many-objective problems).

2.2 MOEA/D: Multi-Objective Evolutionary Algorithm based on Decomposition

MOEA/D was proposed by Zhang and Li in 2007 [2] as a new paradigm for MOEAs based on **problem decomposition**. Instead of relying on Pareto dominance comparisons, MOEA/D decomposes a multi-objective problem into many scalar optimization subproblems and solves them concurrently. The algorithm maintains a population of N individuals, each associated with a distinct **weight vector** $\lambda^{(i)} = (\lambda_1, \dots, \lambda_M)$ (where M is the number of objectives). These weight vectors define different trade-off directions, effectively specifying N scalarized objective functions.

Scalarization Functions: A core idea in MOEA/D is that any Pareto-optimal solution can be optimal for some scalarization of the objectives. Common scalarization strategies used in MOEA/D include:

- *Weighted Sum:* Each subproblem optimizes a weighted linear combination of objectives. The scalar objective is $g^{WS}(x \mid \lambda) = \sum_{i=1}^{M} \lambda_i f_i(x)$, with $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$. The weight vector λ emphasizes certain objectives over others. Weighted sum is simple, but it can only attain solutions on **convex** sections of the Pareto front; if the true Pareto front is concave (non-convex), some Pareto-optimal solutions will never be the minimum of any linear combination of objectives (they are unreachable by weighted sum optimization).
- *Tchebycheff (Chebyshev) approach:* Each subproblem minimizes the maximum weighted deviation from the ideal objective values. Let $\mathbf{z}^* = (z_1^*, \dots, z_M^*)$ be the **ideal point** (best attainable value for each objective, often the minimum of $f_i(x)$ in the current population). The Chebyshev scalarization is: $g^{Cheb}(x \mid \lambda, \mathbf{z}^*) = \max_{1 \leq i \leq M}\{\, \lambda_i \,|\, f_i(x) - z_i^* \,|\, \}$. Each solution is evaluated by its worst (relative) objective performance, with weights λ_i. By adjusting λ, this formulation can target different parts of the Pareto front. An important property is that **for any Pareto-optimal solution, there exists a weight vector** λ (often in combination with an appropriate z^*) **that makes that solution optimal for the Chebyshev metric**. Thus, unlike the weighted sum, the Chebyshev approach can discover **non-convex Pareto fronts** (including concave regions) given a well-distributed set of weight vectors. This is a major motivation for using Chebyshev (and related boundary intersection methods) in MOEA/D when tackling problems like ZDT2 which have a concave front.

MOEA/D works by assigning each weight vector $\lambda^{(i)}$ to one individual in the population, which is intended to converge to a Pareto-optimal solution favoring that weight profile. A notion of **neighborhood** in the weight space is used: for each weight vector, a set of T closest weight vectors (usually in Euclidean distance) is defined as its neighbors. The subproblems corresponding to neighboring weight vectors are expected to have optimal solutions in nearby regions of objective space. Therefore, MOEA/D encourages **collaboration among neighboring subproblems** by mainly allowing genetic variation and solution replacement within those neighborhoods.

Algorithm Outline: Initially, a set of N weight vectors $\{\lambda^{(1)}, \dots, \lambda^{(N)}\}$ is chosen (e.g., uniformly spread over the objective simplex for balanced coverage). The initial population $\{x^{(1)}, \dots, x^{(N)}\}$ is typically generated at random or by some initialization heuristic, and the **ideal point** $\mathbf{z}^*$ is set to the component-wise minima of the objective values found. MOEA/D then evolves the population for a number of generations. At each generation, for each subproblem $i = 1, \dots, N$:

1. **Mating Selection:** Randomly pick a few indices from the neighbor set $B(i)$ of subproblem i (which typically includes i itself and its T nearest weight indices). Using the corresponding parent solutions, generate an **offspring** solution (call it y) by applying crossover and mutation.
2. **Update Ideal Point:** Evaluate $f(y) = (f_1(y), \dots, f_M(y))$. Update the ideal point z^* such that for each objective j: if $f_j(y) < z_j^*$, then set $z_j^* = f_j(y)$ (i.e. record any new best objective values found).
3. **Solution Replacement:** For each subproblem j in the neighbor set $B(i)$ (including i itself), **compare the new solution y to the current solution** $x^{(j)}$ using *j's scalarization function*. That is, compute the scalarized objective $g(x^{(j)} \mid \lambda^{(j)}, \mathbf{z}^*)$ and $g(y \mid \lambda^{(j)}, \mathbf{z}^*)$. If y is *better* (lower) in this scalar value, then replace $x^{(j)}$ with y as the new solution for subproblem j.

By restricting replacement to a subproblem's neighborhood, MOEA/D maintains diversity across different weight vectors while still allowing good solutions to propagate locally. Each subproblem searches in a semi-isolated manner, but the **sharing of offspring among neighbors** helps disseminate useful genetic material to nearby regions of the Pareto front. Notably, MOEA/D's computational cost per generation is relatively low, since each offspring is compared only with a limited set of neighbors rather than the entire population. Indeed, Zhang and Li note that MOEA/D has **lower per-generation complexity than NSGA-II**, which involves $O(N^2)$ sorting operations. Empirical results in the original MOEA/D paper showed that this algorithm could outperform or at least equal NSGA-II in various test problems, especially as the number of objectives grows [2].

Weight Vector Design and Scalarization Choice: The performance of MOEA/D heavily depends on a good spread of weight vectors and an appropriate scalarization method for the problem at hand. If the Pareto front is convex, a uniform spread of weight vectors with the simple weighted sum may suffice to approximate the front. For problems with **non-convex fronts (e.g., ZDT2)**, using the Chebyshev approach (or other advanced methods like Penalty Boundary Intersection) is beneficial because it can locate solutions in concave regions that weighted sum would miss. In practice, MOEA/D can incorporate various scalarization techniques; the chosen two in this work (linear weighted sum and Chebyshev) provide a clear contrast in how well they handle concave trade-offs. The **Chebyshev MOEA/D** is expected to cover the Pareto front more completely on non-convex problems, whereas **Weighted-Sum MOEA/D** may struggle to find evenly distributed solutions in the non-convex middle part of the front – this is precisely the issue under investigation.

MOEA/D has become a cornerstone in multi-objective optimization research. Its modular framework allows hybridization and extension; for example, researchers have studied alternative ways to update neighbors, dynamic weight adjustment, or embedding local search. The original MOEA/D formulation by Zhang and Li (2007) [2] demonstrated that even a straightforward decomposition (using fixed weights and basic genetic operators) is highly effective. That work opened up a new line of research into decomposition-based MOEAs, distinguishing MOEA/D as a **milestone algorithm** comparable to NSGA-II in its influence.

3. Experimental Setup

The experimental setup was designed to perform a rigorous comparison between NSGA-II and MOEA/D algorithms. Particular care was taken to maintain architectural consistency between the implementations in order to ensure fairness and modularity. Specifically, the MOEA/D algorithm was implemented from scratch by the author, reusing the structural conventions and modular decomposition adopted in the NSGA-II codebase provided by Prof. Marco Cococcioni. This included consistent interfaces, modular function decomposition (e.g., objective evaluation, initialization, crossover, mutation), and data structures for individuals and populations.

3.1 Benchmark Problems

To evaluate the performance of the algorithms under study, three well-established benchmark problems from the ZDT family have been selected. All of them are formulated as bi-objective minimization tasks defined over the domain $[0, 1]^n$, with $n = 30$ decision variables. Despite their similar structure, each problem exhibits a distinct shape of the Pareto front, allowing the investigation of how different algorithms and scalarization strategies cope with varying levels of complexity in multi-objective optimization.

The true Pareto fronts for all problems were generated analytically by sampling 1,000 uniformly spaced solutions in the decision space according to known conditions.

ZDT1

ZDT1 defines a convex Pareto front and is commonly used to assess the ability of algorithms to maintain a uniform spread of solutions across a smooth and continuous front. The problem is defined as follows:

$$\begin{aligned} f_1(x) &= x_1, \\ g(x) &= 1 + 9 \cdot \frac{\sum_{i=2}^{n} x_i}{n-1}, \\ f_2(x) &= g(x) \cdot \left(1 - \sqrt{\frac{x_1}{g(x)}}\right). \end{aligned}$$

The optimal front corresponds to the condition $x_1 \in [0, 1]$ and $x_i = 0$ for $i = 2, \ldots, n$.

ZDT2

ZDT2 presents a non-convex Pareto front, making it particularly useful for evaluating the ability of algorithms to approximate regions that cannot be reached via simple linear combinations of objectives. The formulation is given by:

$$\begin{aligned} f_1(x) &= x_1, \\ g(x) &= 1 + 9 \cdot \frac{\sum_{i=2}^{n} x_i}{n-1}, \\ f_2(x) &= g(x) \cdot \left(1 - \left(\frac{x_1}{g(x)}\right)^2\right). \end{aligned}$$

As in ZDT1, the Pareto-optimal set is obtained when $x_1 \in [0, 1]$ and all other variables are zero.

ZDT3

ZDT3 introduces an additional layer of complexity by featuring a **disconnected Pareto front**, composed of multiple non-contiguous segments. This structure challenges algorithms to ensure diversity across several isolated regions. The problem is defined as:

$$\begin{aligned} f_1(x) &= x_1, \\ g(x) &= 1 + 9 \cdot \frac{\sum_{i=2}^{n} x_i}{n-1}, \\ f_2(x) &= g(x) \cdot \left(1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)} \cdot \sin(10\pi x_1)\right). \end{aligned}$$

The optimal solutions satisfy $x_1 \in [0, 1]$, $x_i = 0$ for $i = 2, \ldots, n$, and lie on a front with five disconnected segments.

These three problems collectively cover a range of Pareto front geometries—convex, non-convex, and discontinuous—providing a robust basis for performance comparison.

3.2 Algorithmic Structure and Design Choices

NSGA-II

The NSGA-II implementation provided by the instructor was used without conceptual modification. This version follows the canonical structure proposed by Deb et al. (2002), with the following components:

- **Fast Non-Dominated Sorting**: Used to classify the population into Pareto fronts.
- **Crowding Distance**: Used to ensure diversity within each front.
- **Binary Tournament Selection**: Based on rank and crowding distance.
- **Simulated Binary Crossover (SBX)** and **Polynomial Mutation**: Used to generate offspring.
- **Elitist Replacement**: Parent and offspring populations are merged and sorted; the best N individuals are retained.

MOEA/D

This MOEA/D implementation strictly follows the decomposition-based optimization paradigm as described in the original MOEA/D paper. Its main components are:

- **Weight Vectors (λ_i)**: Each subproblem corresponds to a different weight vector in the objective space, uniformly generated for the bi-objective case.
- **Neighborhood Structure ($B(i)$)**: For each subproblem i, a neighborhood of $T = 10$ nearby subproblems is defined using Euclidean distance between weight vectors.
- **Ideal Point (z^*)**: Represents the best known objective values, updated throughout the evolution.
- **Scalarization**: Two strategies were employed:
  - **Chebyshev Scalarization**:

$$\text{Cheby}(f, z, \lambda) = \max_j \lambda_j \cdot |f_j - z_j|$$

  - **Linear Scalarization**:

$$\text{Linear}(f, z, \lambda) = \sum_j \lambda_j \cdot |f_j - z_j|$$

- **Variation Operators**: Polynomial mutation applied to a parent selected from the neighborhood $B(i)$.
- **Localized Replacement**: Offspring may replace neighboring solutions if they improve the scalarized objective with respect to their associated weight vector.
- **Archive Management**: An archive of non-dominated solutions of size N is maintained and updated at each generation. If the archive exceeds size N, solutions are removed based on crowding distance to preserve diversity.

MOEA/D with Crossover

This variant extends the base MOEA/D algorithm by incorporating crossover in the variation stage, while still adhering strictly to the original MOEA/D formulation.

- **Weight Vectors (λ_i)**: Same as in MOEA/D.
- **Neighborhood Structure ($B(i)$)**: Same as in MOEA/D.
- **Ideal Point (z^*)**: Same as in MOEA/D.
- **Scalarization**: Same as in MOEA/D.
- **Variation Operators**: Simulated Binary Crossover (SBX) followed by polynomial mutation, applied to parents selected from the neighborhood $B(i)$.
- **Localized Replacement**: Same as in MOEA/D.
- **Archive Management**: Same as in MOEA/D.

3.3 Parameter Configuration

For all algorithms, the following parameters were fixed:

- Population size: $N = 50$
- Generations: $G = 1000$
- Number of independent runs: 30 (random seed fixed from 1 to 30)
- Crossover and mutation indices: $\mu = 20$, $\mu_m = 20$

In NSGA-II and MOEA/D with crossover, the following genetic operators were used:

- Crossover probability: 90%

- Mutation probability: 10%

while in MOEA/D without crossover, only mutation was applied with 100% probability.

Neighborhood size $T = 10$ was used in both versions of MOEA/D, ensuring that updates are localized but sufficiently diverse.

3.4 Evaluation Metrics

The performance of each algorithm was assessed using four widely adopted multi-objective optimization metrics: **Generational Distance (GD)**, **Inverted Generational Distance (IGD)**, the Δ **diversity indicator**, and the **Hypervolume (HV)**. These metrics provide complementary insights into the convergence and diversity of the obtained solution sets. At the end of each run, only the first non-dominated front was retained for evaluation. The hypervolume computation is proposed in two versions, the approach derived from the *PlatEMO* framework and, for two-objective problems, the rectangle-based computation method.

A detailed mathematical definition of these metrics, including discussion on their significance and how they were computed in this work, is provided in Section 4.1.

3.5 Reproducibility

All code was executed in MATLAB with modularized folder structures (`+kpi/`, `+moead/`, `+moead_modified/`, `+nsga2/`, `+utility/`), allowing isolated evaluation of each method. Seeds were fixed per run, and all outputs (including final solution fronts and metric values) were saved for further inspection.

This consistent and transparent setup ensures both **fairness** in comparison and **reproducibility** of all results obtained in this experimental study.

4. Results and Discussion

This section presents the evaluation of the algorithms under comparison on the selected benchmark problems. Each algorithm was independently executed for 30 runs, and the results were assessed in terms of convergence and diversity using a standard set of multi-objective performance indicators. The first non-dominated front obtained at the end of each run was extracted and evaluated.

4.1 Evaluation Metrics

To ensure a comprehensive analysis of performance, four widely used metrics in evolutionary multi-objective optimization were employed: **Generational Distance (GD)**, **Inverted Generational Distance (IGD)**, **Diversity Indicator (Δ)**, and **Hypervolume (HV)**. Each of these provides insight into a specific aspect of the quality of the approximated Pareto front. All metric values were computed consistently across all 30 independent runs per algorithm. The average values for each metric were stored and used for analysis in the subsequent discussion.

Generational Distance (GD)

The **Generational Distance** measures how close the obtained approximation front A is to the true Pareto front Z, and is defined as:

$$GD(A) = \frac{1}{n} \left(\sum_{i=1}^{n} d_i^p \right)^{1/p}$$

Where:

- $A = a_1, \ldots, a_n$ is the set of non-dominated solutions produced by the algorithm;
- $Z = z_1, \ldots, z_m$ is the reference Pareto front;
- d_i is the Euclidean distance between a solution $a_i \in A$ and the nearest point in Z;
- p is typically set to 2 (Euclidean norm).

A lower GD indicates better convergence to the optimal front.

Inverted Generational Distance (IGD)

The **Inverted Generational Distance** instead evaluates how well the reference front is covered by the approximation set A, and is defined as:

$$IGD(A) = \frac{1}{m} \left(\sum_{i=1}^{m} d_i^p \right)^{1/p}$$

In this case, d_i is the distance between a reference point $z_i \in Z$ and the closest point in A. A low IGD reflects both good convergence and distribution.

Diversity Indicator (Δ)

The **Diversity Indicator**, denoted as Δ, was computed as the maximum of the GD and IGD values for each run:

$$\Delta = \max(\text{GD}, \text{IGD})$$

This simplified formulation emphasizes the worst-case degradation in either convergence or spread, penalizing algorithms that fail on one of the two aspects.

Hypervolume (HV)

The **Hypervolume (HV)** indicator quantifies the portion of the objective space that is weakly dominated by a set of solutions and bounded by a fixed **reference point** (commonly referred to as the *nadir point*). It is widely recognized for capturing both convergence and diversity of the approximation set.

In this study, two complementary methods were employed to compute the HV:

1. Rectangle-Based Method (Exact for $M = 2$) For **bi-objective problems**, HV can be computed **exactly** via a geometric decomposition of the dominated region into **rectangles**. This method assumes that:

- The input front is **non-dominated** and sorted in **ascending order** with respect to the first objective.
- The **reference point** lies in the **worst corner** of the objective space (typically beyond the last solution in each objective).

The intuition is as follows:

- Between each pair of consecutive points on the Pareto front, a **rectangle** is defined whose width is the difference in the first objective, and whose height is the distance to the reference point along the second objective.
- A final rectangle closes the region between the **last point** and the **reference point**, completing the dominated area.
- The **total hypervolume** is the **sum of the areas** of these axis-aligned rectangles.

This method is:

- **Deterministic** and **fast** for $M = 2$,
- Used in this work as a **reference computation** for two-objective problems,
- Particularly valuable when a large number of comparisons need to be made (e.g., across multiple runs).

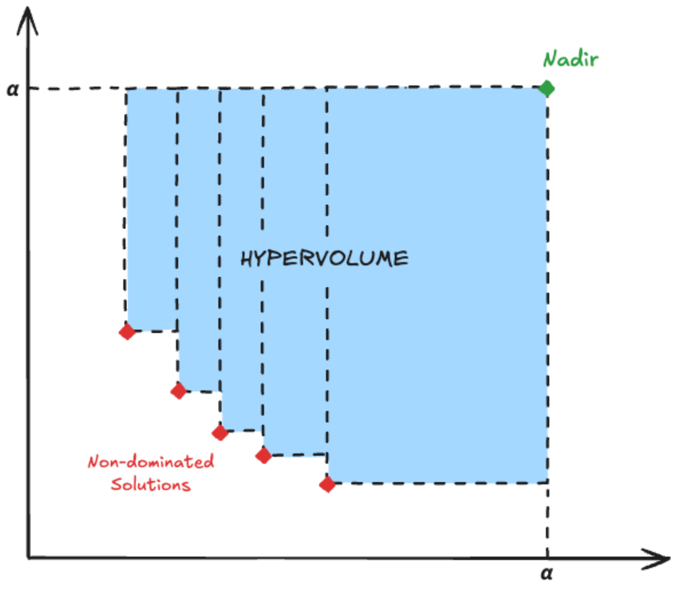


Figure 2: Hypervolume computation using rectangles for bi-objective problems.

2. PlatEMO-Based Method (General Case) To ensure accurate and consistent computation of the Hypervolume (HV) across different problem dimensions, this study employs a dual-strategy method derived from the PlatEMO platform [3]. The implementation automatically selects the most appropriate strategy depending on the dimensionality of the objective space.

Case 1: Monte Carlo Estimation for $M > 2$

For problems with more than two objectives, PlatEMO adopts a Monte Carlo-based estimation technique. This method approximates the hypervolume by generating a large number of uniformly distributed random points within the objective space bounded by the ideal and reference points. Specifically:

- A number of n_{sample} points are sampled uniformly between the component-wise minimum of the current approximation set (the ideal point) and the reference point (typically a nadir).
- For each sample point, the number of solutions that dominate it is counted.
- The contribution of each solution is then estimated based on the number of samples it dominates, weighted by coefficients α_i computed analytically as:

$$HV \approx \sum_{i=1}^{N} \alpha_i \cdot \text{Vol}_i$$

Where:

  - α_i accounts for the combinatorial weight of the subset in the decomposition,
  - Vol_i is the estimated hypervolume contribution of the i-th solution.

This method strikes a balance between **efficiency and scalability**, making it suitable for high-dimensional problems where exact computation is prohibitively expensive. While inherently approximate, the large sample size and the analytical form of the weights ensure good accuracy for comparative evaluations.

Case 2: Exact Computation for $M = 2$ via the HSO Algorithm

When the number of objectives is two, PlatEMO switches to an exact hypervolume computation based on the **Hypervolume Slicing Objectives (HSO)** algorithm, originally proposed by White et al. [4]. This method is particularly efficient and accurate in the bi-objective setting.

The HSO algorithm operates by recursively slicing the objective space along one objective at a time and reducing the dimensionality at each step. For a minimization problem with two objectives:

1. The approximation set is first **sorted in ascending order of the first objective**.
2. The algorithm traverses this sorted list to generate **vertical slices**, each representing a "strip" of the hypervolume bounded in the first objective.
3. For each slice, the remaining portion (in the second objective) is analyzed, discarding any dominated points, and the area of the resulting rectangle is computed.
4. The total hypervolume is then obtained as the **sum of the areas of all these rectangles**.

More formally, this corresponds to decomposing the dominated region into a union of disjoint rectangles, each defined by two consecutive points and the reference point. The approach is **deterministic**, **non-iterative**, and efficient for two-objective problems.

The underlying structure of the implementation in PlatEMO reflects the key components of the original HSO algorithm as formalized by White et al. [4], although the corresponding operations are embedded within the recursive MATLAB function `hypesub(...)` rather than being defined as separate functions:

- The generation of cross-sectional slices of the dominated region (conceptually similar to the `slice` operation) is performed via the sorting step `sortrows(A, M)`, which orders the solutions along the last objective to enable recursive slicing.
- The logic of point insertion and pruning (analogous to the `insert` operation in the original algorithm) is implicitly handled by iteratively selecting the first `i` points in each recursive call (`S(1:i,:)`), effectively maintaining a sorted and non-dominated subset at each level.
- The recursive traversal over objectives, which corresponds to the main `hso` procedure, is implemented through successive invocations of `hypesub`, reducing the dimensionality one objective at a time until the base case is reached (typically $M = 1$ or $M = 2$), where contributions can be computed exactly.

4.2 Results Analysis

The comparative evaluation of MOEA/D with Chebyshev scalarization, MOEA/D with linear scalarization, and NSGA-II was carried out on three classical benchmark problems: ZDT1, ZDT2, and ZDT3. Each algorithm was executed over 30 independent runs, using the same set of random seeds across all configurations to ensure consistency and comparability. The performance was assessed using four key metrics—Generational Distance (GD), Inverted Generational Distance (IGD), Hypervolume (HV) (both Platemo version and Rectangles method), and the Delta metric, whose definitions and properties have been introduced in the previous section.

The following table reports the average values of each metric across the 30 runs for every algorithm and problem:

Problem	Algorithm	GD	IGD	Δ	HV Platemo	HV Rect
ZDT1	moead_cheby	0.0033	0.0096	0.0103	0.9293	0.8645
	moead_linear	0.0043	0.0091	0.0112	0.9540	0.8649
	moead_sbx_cheby	0.0033	0.0104	0.0119	0.9394	0.8632
	moead_sbx_linear	0.0004	0.0090	0.0090	0.8646	0.8646
	nsga2	0.0005	0.0098	0.0098	0.8643	0.8643
ZDT2	moead_cheby	0.0031	0.0097	0.0105	0.5979	0.5316
	moead_linear	0.0011	0.0177	0.0177	0.5205	0.5205
	moead_sbx_cheby	0.0011	0.0106	0.0110	0.5489	0.5308
	moead_sbx_linear	0.0017	0.0290	0.0290	0.5048	0.5048
	nsga2	0.0004	0.0099	0.0099	0.5316	0.5316
ZDT3	moead_cheby	0.0049	0.1932	0.1932	1.3901	1.3254
	moead_linear	0.0063	0.1941	0.1941	1.4262	1.3256
	moead_sbx_cheby	0.0051	0.1937	0.1937	1.4013	1.3250
	moead_sbx_linear	0.0023	0.1974	0.1974	1.3148	1.3148
	nsga2	0.0020	0.1946	0.1946	1.3196	1.3196

In the case of **ZDT1**, which features a convex and continuous Pareto front, all algorithms perform consistently well in terms of convergence and diversity. NSGA-II exhibits the best GD and Δ values, indicating its ability to place solutions very close to the true front while maintaining a uniform distribution. The modified MOEA/D with linear scalarization (`moead_sbx_linear`) performs exceptionally, achieving the best overall values across all metrics, including GD, IGD, and HV (identical across both versions). This suggests that the addition of crossover significantly mitigates the limitations of linear scalarization on convex fronts. Standard MOEA/D variants without crossover show slightly inferior IGD values, confirming the beneficial effect of recombination. Interestingly, the Hypervolume computed via PlatEMO overestimates the performance of MOEA/D-linear, likely due to better coverage near the extreme regions, while the rectangle-based HV—aligned with true front coverage—supports a more consistent ranking.

When we shift our focus to **ZDT2**, characterized by a **non-convex** Pareto front, the behavior of the algorithms diverges. As expected, linear scalarization struggles to reconstruct non-convex shapes: `moead_linear` shows poor IGD and HV values despite a good GD, revealing that some solutions converge, but the overall coverage is poor. This limitation also affects the sbx-enhanced version (`moead_sbx_linear`), which maintains low GD but has the highest IGD and lowest HV. In contrast, both NSGA-II and MOEA/D-Chebyshev (standard and sbx) provide superior performance across all metrics. NSGA-II achieves the best GD and IGD values, whereas MOEA/D-Chebyshev attains the highest HV (especially using PlatEMO), underlining its robustness in dealing with front curvature and shape complexity. The crossover variant (`moead_sbx_cheby`) also brings minor improvements over the pure mutation version, suggesting additional benefit from recombination.

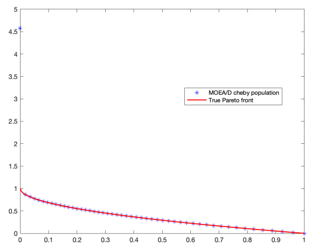
The scenario becomes even more challenging with **ZDT3**, due to the **disconnected** nature of the Pareto front. Here, convergence and diversity are harder to maintain across multiple disjoint regions. All algorithms show an increase in GD and IGD as expected. NSGA-II and `moead_sbx_linear` perform best in terms of GD, with values around 0.002, suggesting better convergence to the various front segments. However, MOEA/D with Chebyshev scalarization (standard and sbx) achieves the highest Hypervolume, demonstrating its superior ability to **spread solutions** across multiple disconnected segments. Linear MOEA/D variants, while performing well in convergence (especially when combined with crossover), fail to achieve high HV values, reflecting limited diversity across all regions. This confirms that scalarization-based methods require careful design to avoid biasing solutions toward specific areas of the front.

It is important to interpret the Δ **metric** carefully. By definition, $\Delta = \max(\text{GD}, \text{IGD})$ for each run. However, the table reports **average values over 30 runs**, and the reported Δ values are the mean of the per-run maximums. This means that the average Δ may not match the greater of average GD and average IGD. For example, in ZDT1 for `moead_cheby`, $\Delta = 0.0103$, which does not correspond exactly to $\max(0.0033, 0.0096) = 0.0096$, since in some runs GD was higher, and in others IGD was. This nuance is essential for correct interpretation of aggregated results.

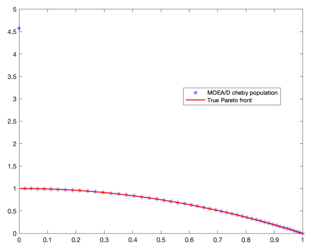
Overall, the experiments confirm theoretical expectations. **NSGA-II** consistently demonstrates strong performance across all problems due to its dominance-based selection and global elitism mechanisms. **MOEA/D with Chebyshev scalarization** adapts well to non-convex and disconnected fronts, leveraging its decomposition strategy to maintain spread and diversity. The **addition of crossover** (SBX) in MOEA/D proves beneficial, especially for linear scalarization, where it compensates for the inability to handle non-convexity and improves performance even on convex and discontinuous problems. **MOEA/D with linear scalarization**, while effective on ZDT1, consistently underperforms on ZDT2 and ZDT3 due to its intrinsic limitation in spanning the entire front geometry, unless recombination is introduced to increase diversity.

Visual Comparison of Pareto Fronts

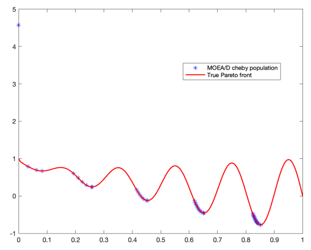
The graphical analysis includes a series of plots that compare the true Pareto front (shown in red) with the non-dominated solutions obtained by each algorithm. For each benchmark problem, three images are shown—one per algorithm—allowing direct visual inspection of coverage and convergence. These plots provide valuable qualitative support to the numerical metrics, especially in identifying regions of the front that are poorly approximated or entirely missed by certain algorithms.



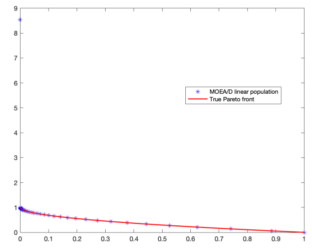
ZDT1 - MOEA/D-Chebyshev



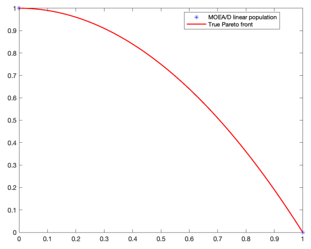
ZDT2 - MOEA/D-Chebyshev



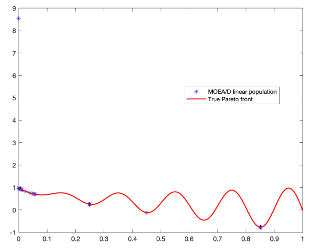
ZDT3 - MOEA/D-Chebyshev



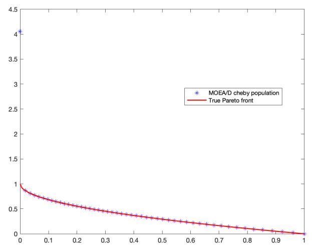
ZDT1 - MOEA/D-Linear



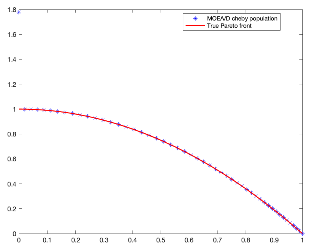
ZDT2 - MOEA/D-Linear



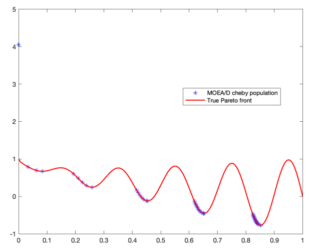
ZDT3 - MOEA/D-Linear



ZDT1 - MOEA/D-Chebyshev with SBX Crossover

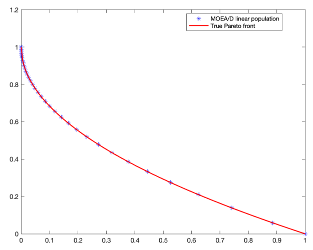


ZDT2 - MOEA/D-Chebyshev with SBX Crossover

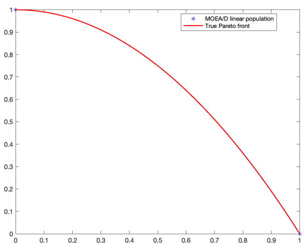


ZDT3 - MOEA/D-Chebyshev with SBX Crossover

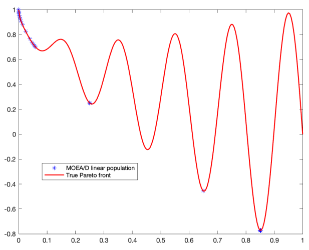
Figure 3: Comparison of final Pareto front approximations obtained by each algorithm on ZDT1–ZDT3. (1/2)



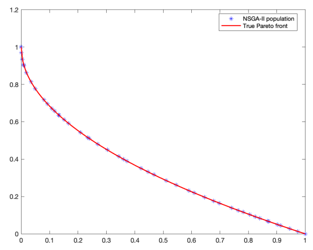
ZDT1 - MOEA/D-Linear with SBX Crossover



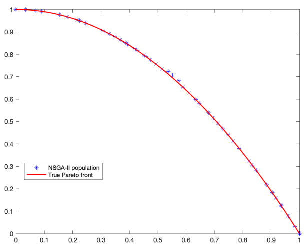
ZDT2 - MOEA/D-Linear with SBX Crossover



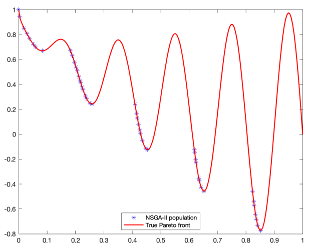
ZDT3 - MOEA/D-Linear with SBX Crossover



ZDT1 - NSGA-II



ZDT2 - NSGA-II



ZDT3 - NSGA-II

Figure 4: Comparison of final Pareto front approximations obtained by each algorithm on ZDT1–ZDT3. (2/2)

Appendix A

The numerical results from each of the 30 independent runs are reported per problem and per algorithm. This data provides insight into the variability of the algorithmic behavior and supports more fine-grained statistical analysis.

ZDT1 Results:

Algorithm	Run	GD	IGD	Δ	HV Platemo	HV Rect
moead_cheby	1	0.0004	0.0089	0.0089	0.8654	0.8654
moead_linear	1	0.0275	0.0086	0.0275	1.7435	0.8651
moead_mod_cheby	1	0.0007	0.0111	0.0111	0.8627	0.8627
moead_mod_linear	1	0.0006	0.0089	0.0089	0.8649	0.8649
nsga2	1	0.0004	0.0115	0.0115	0.8627	0.8627
moead_cheby	2	0.0073	0.0095	0.0095	0.9997	0.8646
moead_linear	2	0.0334	0.0104	0.0334	1.3073	0.8622
moead_mod_cheby	2	0.0007	0.0090	0.0090	0.8654	0.8654
moead_mod_linear	2	0.0004	0.0088	0.0088	0.8646	0.8646
nsga2	2	0.0004	0.0097	0.0097	0.8644	0.8644
moead_cheby	3	0.0009	0.0096	0.0096	0.8646	0.8646
moead_linear	3	0.0009	0.0095	0.0095	0.8638	0.8638
moead_mod_cheby	3	0.0005	0.0106	0.0106	0.8629	0.8629
moead_mod_linear	3	0.0006	0.0094	0.0094	0.8628	0.8628
nsga2	3	0.0005	0.0097	0.0097	0.8641	0.8641
moead_cheby	4	0.0008	0.0098	0.0098	0.8643	0.8643
moead_linear	4	0.0004	0.0101	0.0101	0.8640	0.8640
moead_mod_cheby	4	0.0005	0.0104	0.0104	0.8635	0.8635
moead_mod_linear	4	0.0006	0.0088	0.0088	0.8652	0.8652
nsga2	4	0.0004	0.0092	0.0092	0.8651	0.8651
moead_cheby	5	0.0009	0.0108	0.0108	0.8633	0.8633
moead_linear	5	0.0316	0.0093	0.0316	1.8530	0.8643
moead_mod_cheby	5	0.0004	0.0108	0.0108	0.8628	0.8628
moead_mod_linear	5	0.0003	0.0091	0.0091	0.8644	0.8644
nsga2	5	0.0004	0.0096	0.0096	0.8645	0.8645
moead_cheby	6	0.0275	0.0107	0.0275	1.3721	0.8630
moead_linear	6	0.0011	0.0098	0.0098	0.8632	0.8632
moead_mod_cheby	6	0.0007	0.0105	0.0105	0.8633	0.8633
moead_mod_linear	6	0.0003	0.0090	0.0090	0.8650	0.8650
nsga2	6	0.0005	0.0099	0.0099	0.8644	0.8644
moead_cheby	7	0.0010	0.0091	0.0091	0.8651	0.8651
moead_linear	7	0.0005	0.0090	0.0090	0.8654	0.8654
moead_mod_cheby	7	0.0005	0.0107	0.0107	0.8629	0.8629
moead_mod_linear	7	0.0003	0.0095	0.0095	0.8635	0.8635
nsga2	7	0.0006	0.0102	0.0102	0.8642	0.8642
moead_cheby	8	0.0013	0.0092	0.0092	0.8651	0.8651
moead_linear	8	0.0045	0.0087	0.0087	0.9799	0.8655

Algorithm	Run	GD	IGD	Δ	HV Platemo	HV Rect
moead_mod_cheby	8	0.0005	0.0100	0.0100	0.8636	0.8636
moead_mod_linear	8	0.0003	0.0087	0.0087	0.8650	0.8650
nsga2	8	0.0004	0.0099	0.0099	0.8642	0.8642
moead_cheby	9	0.0027	0.0103	0.0103	0.8754	0.8640
moead_linear	9	0.0006	0.0101	0.0101	0.8643	0.8643
moead_mod_cheby	9	0.0008	0.0100	0.0100	0.8635	0.8635
moead_mod_linear	9	0.0003	0.0092	0.0092	0.8636	0.8636
nsga2	9	0.0003	0.0097	0.0097	0.8645	0.8645
moead_cheby	10	0.0005	0.0092	0.0092	0.8652	0.8652
moead_linear	10	0.0004	0.0090	0.0090	0.8651	0.8651
moead_mod_cheby	10	0.0004	0.0106	0.0106	0.8631	0.8631
moead_mod_linear	10	0.0005	0.0086	0.0086	0.8652	0.8652
nsga2	10	0.0004	0.0096	0.0096	0.8647	0.8647
moead_cheby	11	0.0014	0.0098	0.0098	0.8645	0.8645
moead_linear	11	0.0013	0.0090	0.0090	0.8651	0.8651
moead_mod_cheby	11	0.0007	0.0121	0.0121	0.8610	0.8610
moead_mod_linear	11	0.0003	0.0092	0.0092	0.8646	0.8646
nsga2	11	0.0004	0.0104	0.0104	0.8637	0.8637
moead_cheby	12	0.0008	0.0098	0.0098	0.8641	0.8641
moead_linear	12	0.0009	0.0094	0.0094	0.8649	0.8649
moead_mod_cheby	12	0.0006	0.0110	0.0110	0.8627	0.8627
moead_mod_linear	12	0.0006	0.0091	0.0091	0.8638	0.8638
nsga2	12	0.0004	0.0090	0.0090	0.8651	0.8651
moead_cheby	13	0.0009	0.0094	0.0094	0.8648	0.8648
moead_linear	13	0.0005	0.0088	0.0088	0.8653	0.8653
moead_mod_cheby	13	0.0008	0.0104	0.0104	0.8632	0.8632
moead_mod_linear	13	0.0004	0.0093	0.0093	0.8645	0.8645
nsga2	13	0.0004	0.0093	0.0093	0.8648	0.8648
moead_cheby	14	0.0130	0.0092	0.0130	1.1061	0.8647
moead_linear	14	0.0012	0.0092	0.0092	0.8648	0.8648
moead_mod_cheby	14	0.0150	0.0101	0.0150	1.5339	0.8633
moead_mod_linear	14	0.0006	0.0091	0.0091	0.8651	0.8651
nsga2	14	0.0004	0.0099	0.0099	0.8637	0.8637
moead_cheby	15	0.0096	0.0093	0.0096	1.2545	0.8650
moead_linear	15	0.0019	0.0088	0.0088	0.8654	0.8654
moead_mod_cheby	15	0.0030	0.0109	0.0109	0.8626	0.8626
moead_mod_linear	15	0.0004	0.0082	0.0082	0.8661	0.8661
nsga2	15	0.0005	0.0101	0.0101	0.8642	0.8642
moead_cheby	16	0.0061	0.0094	0.0094	1.0354	0.8652
moead_linear	16	0.0012	0.0087	0.0087	0.8654	0.8654
moead_mod_cheby	16	0.0006	0.0092	0.0092	0.8648	0.8648
moead_mod_linear	16	0.0004	0.0095	0.0095	0.8646	0.8646
nsga2	16	0.0004	0.0097	0.0097	0.8645	0.8645
moead_cheby	17	0.0006	0.0088	0.0088	0.8654	0.8654

Algorithm	Run	GD	IGD	Δ	HV Platemo	HV Rect
moead_linear	17	0.0009	0.0087	0.0087	0.8655	0.8655
moead_mod_cheby	17	0.0052	0.0099	0.0099	1.0149	0.8639
moead_mod_linear	17	0.0005	0.0090	0.0090	0.8653	0.8653
nsga2	17	0.0006	0.0108	0.0108	0.8631	0.8631
moead_cheby	18	0.0016	0.0098	0.0098	0.8639	0.8639
moead_linear	18	0.0004	0.0085	0.0085	0.8660	0.8660
moead_mod_cheby	18	0.0006	0.0098	0.0098	0.8642	0.8642
moead_mod_linear	18	0.0004	0.0087	0.0087	0.8658	0.8658
nsga2	18	0.0006	0.0104	0.0104	0.8637	0.8637
moead_cheby	19	0.0007	0.0105	0.0105	0.8633	0.8633
moead_linear	19	0.0009	0.0086	0.0086	0.8656	0.8656
moead_mod_cheby	19	0.0006	0.0099	0.0099	0.8639	0.8639
moead_mod_linear	19	0.0004	0.0087	0.0087	0.8653	0.8653
nsga2	19	0.0012	0.0093	0.0093	0.8649	0.8649
moead_cheby	20	0.0006	0.0092	0.0092	0.8654	0.8654
moead_linear	20	0.0005	0.0087	0.0087	0.8656	0.8656
moead_mod_cheby	20	0.0226	0.0094	0.0226	1.4570	0.8645
moead_mod_linear	20	0.0003	0.0090	0.0090	0.8642	0.8642
nsga2	20	0.0006	0.0105	0.0105	0.8640	0.8640
moead_cheby	21	0.0017	0.0091	0.0091	0.8652	0.8652
moead_linear	21	0.0004	0.0087	0.0087	0.8657	0.8657
moead_mod_cheby	21	0.0358	0.0111	0.0358	1.7339	0.8618
moead_mod_linear	21	0.0004	0.0090	0.0090	0.8640	0.8640
nsga2	21	0.0005	0.0095	0.0095	0.8646	0.8646
moead_cheby	22	0.0011	0.0103	0.0103	0.8633	0.8633
moead_linear	22	0.0013	0.0090	0.0090	0.8650	0.8650
moead_mod_cheby	22	0.0007	0.0113	0.0113	0.8622	0.8622
moead_mod_linear	22	0.0006	0.0095	0.0095	0.8644	0.8644
nsga2	22	0.0005	0.0089	0.0089	0.8653	0.8653
moead_cheby	23	0.0005	0.0107	0.0107	0.8631	0.8631
moead_linear	23	0.0020	0.0090	0.0090	0.8649	0.8649
moead_mod_cheby	23	0.0012	0.0108	0.0108	0.8632	0.8632
moead_mod_linear	23	0.0006	0.0090	0.0090	0.8648	0.8648
nsga2	23	0.0005	0.0100	0.0100	0.8641	0.8641
moead_cheby	24	0.0014	0.0100	0.0100	0.8644	0.8644
moead_linear	24	0.0018	0.0089	0.0089	0.8653	0.8653
moead_mod_cheby	24	0.0008	0.0099	0.0099	0.8642	0.8642
moead_mod_linear	24	0.0003	0.0089	0.0089	0.8655	0.8655
nsga2	24	0.0005	0.0090	0.0090	0.8648	0.8648
moead_cheby	25	0.0010	0.0087	0.0087	0.8651	0.8651
moead_linear	25	0.0007	0.0093	0.0093	0.8636	0.8636
moead_mod_cheby	25	0.0018	0.0109	0.0109	0.8631	0.8631
moead_mod_linear	25	0.0005	0.0091	0.0091	0.8637	0.8637
nsga2	25	0.0004	0.0101	0.0101	0.8641	0.8641

Algorithm	Run	GD	IGD	Δ	HV Platemo	HV Rect
moead_cheby	26	0.0009	0.0092	0.0092	0.8650	0.8650
moead_linear	26	0.0071	0.0090	0.0090	1.1119	0.8649
moead_mod_cheby	26	0.0004	0.0112	0.0112	0.8624	0.8624
moead_mod_linear	26	0.0003	0.0091	0.0091	0.8644	0.8644
nsga2	26	0.0004	0.0103	0.0103	0.8638	0.8638
moead_cheby	27	0.0009	0.0092	0.0092	0.8651	0.8651
moead_linear	27	0.0006	0.0088	0.0088	0.8657	0.8657
moead_mod_cheby	27	0.0006	0.0102	0.0102	0.8636	0.8636
moead_mod_linear	27	0.0005	0.0094	0.0094	0.8638	0.8638
nsga2	27	0.0004	0.0096	0.0096	0.8639	0.8639
moead_cheby	28	0.0009	0.0094	0.0094	0.8645	0.8645
moead_linear	28	0.0005	0.0086	0.0086	0.8657	0.8657
moead_mod_cheby	28	0.0005	0.0110	0.0110	0.8619	0.8619
moead_mod_linear	28	0.0003	0.0088	0.0088	0.8642	0.8642
nsga2	28	0.0010	0.0100	0.0100	0.8638	0.8638
moead_cheby	29	0.0113	0.0099	0.0113	1.3505	0.8641
moead_linear	29	0.0015	0.0098	0.0098	0.8645	0.8645
moead_mod_cheby	29	0.0005	0.0113	0.0113	0.8624	0.8624
moead_mod_linear	29	0.0003	0.0086	0.0086	0.8651	0.8651
nsga2	29	0.0005	0.0089	0.0089	0.8652	0.8652
moead_cheby	30	0.0006	0.0092	0.0092	0.8651	0.8651
moead_linear	30	0.0013	0.0092	0.0092	0.8650	0.8650
moead_mod_cheby	30	0.0007	0.0097	0.0097	0.8643	0.8643
moead_mod_linear	30	0.0004	0.0083	0.0083	0.8660	0.8660
nsga2	30	0.0004	0.0098	0.0098	0.8641	0.8641

ZDT2 Results:

Algorithm	Run	GD	IGD	Δ	HV Platemo	HV Rect
moead_cheby	1	0.0003	0.0093	0.0093	0.5319	0.5319
moead_linear	1	0.0014	0.0166	0.0166	0.5209	0.5209
moead_mod_cheby	1	0.0003	0.0105	0.0105	0.5308	0.5308
moead_mod_linear	1	0.0004	0.0089	0.0089	0.5324	0.5324
nsga2	1	0.0003	0.0101	0.0101	0.5310	0.5310
moead_cheby	2	0.0061	0.0101	0.0101	0.6665	0.5314
moead_linear	2	0.0004	0.0377	0.0377	0.4975	0.4975
moead_mod_cheby	2	0.0004	0.0110	0.0110	0.5307	0.5307
moead_mod_linear	2	0.0020	0.0111	0.0111	0.5281	0.5281
nsga2	2	0.0004	0.0096	0.0096	0.5313	0.5313
moead_cheby	3	0.0004	0.0097	0.0097	0.5318	0.5318
moead_linear	3	0.0004	0.0095	0.0095	0.5319	0.5319
moead_mod_cheby	3	0.0004	0.0100	0.0100	0.5313	0.5313
moead_mod_linear	3	0.0058	0.0484	0.0484	0.4711	0.4711

Algorithm	Run	GD	IGD	Δ	HV Platemo	HV Rect
nsga2	3	0.0004	0.0097	0.0097	0.5317	0.5317
moead_cheby	4	0.0006	0.0098	0.0098	0.5317	0.5317
moead_linear	4	0.0005	0.0089	0.0089	0.5319	0.5319
moead_mod_cheby	4	0.0003	0.0102	0.0102	0.5314	0.5314
moead_mod_linear	4	0.0042	0.0260	0.0260	0.5105	0.5105
nsga2	4	0.0004	0.0095	0.0095	0.5318	0.5318
moead_cheby	5	0.0006	0.0098	0.0098	0.5320	0.5320
moead_linear	5	0.0004	0.0229	0.0229	0.5145	0.5145
moead_mod_cheby	5	0.0004	0.0109	0.0109	0.5303	0.5303
moead_mod_linear	5	0.0005	0.0094	0.0094	0.5315	0.5315
nsga2	5	0.0004	0.0094	0.0094	0.5319	0.5319
moead_cheby	6	0.0276	0.0106	0.0276	1.0393	0.5302
moead_linear	6	0.0006	0.0113	0.0113	0.5296	0.5296
moead_mod_cheby	6	0.0004	0.0099	0.0099	0.5315	0.5315
moead_mod_linear	6	0.0012	0.0095	0.0095	0.5309	0.5309
nsga2	6	0.0003	0.0102	0.0102	0.5319	0.5319
moead_cheby	7	0.0010	0.0111	0.0111	0.5299	0.5299
moead_linear	7	0.0040	0.0465	0.0465	0.4830	0.4830
moead_mod_cheby	7	0.0004	0.0104	0.0104	0.5312	0.5312
moead_mod_linear	7	0.0012	0.0299	0.0299	0.5076	0.5076
nsga2	7	0.0004	0.0095	0.0095	0.5316	0.5316
moead_cheby	8	0.0017	0.0105	0.0105	0.5292	0.5292
moead_linear	8	0.0003	0.0334	0.0334	0.5037	0.5037
moead_mod_cheby	8	0.0004	0.0100	0.0100	0.5311	0.5311
moead_mod_linear	8	0.0034	0.0563	0.0563	0.4633	0.4633
nsga2	8	0.0003	0.0094	0.0094	0.5320	0.5320
moead_cheby	9	0.0026	0.0104	0.0104	0.5434	0.5320
moead_linear	9	0.0011	0.0106	0.0106	0.5299	0.5299
moead_mod_cheby	9	0.0004	0.0094	0.0094	0.5320	0.5320
moead_mod_linear	9	0.0012	0.0261	0.0261	0.5107	0.5107
nsga2	9	0.0003	0.0106	0.0106	0.5315	0.5315
moead_cheby	10	0.0004	0.0093	0.0093	0.5319	0.5319
moead_linear	10	0.0004	0.0107	0.0107	0.5310	0.5310
moead_mod_cheby	10	0.0004	0.0100	0.0100	0.5310	0.5310
moead_mod_linear	10	0.0018	0.0132	0.0132	0.5264	0.5264
nsga2	10	0.0003	0.0096	0.0096	0.5320	0.5320
moead_cheby	11	0.0012	0.0091	0.0091	0.5320	0.5320
moead_linear	11	0.0009	0.0273	0.0273	0.5036	0.5036
moead_mod_cheby	11	0.0004	0.0114	0.0114	0.5316	0.5316
moead_mod_linear	11	0.0022	0.0146	0.0146	0.5230	0.5230
nsga2	11	0.0004	0.0105	0.0105	0.5308	0.5308
moead_cheby	12	0.0009	0.0098	0.0098	0.5317	0.5317
moead_linear	12	0.0006	0.0101	0.0101	0.5316	0.5316
moead_mod_cheby	12	0.0004	0.0098	0.0098	0.5314	0.5314

Algorithm	Run	GD	IGD	Δ	HV Platemo	HV Rect
moead_mod_linear	12	0.0005	0.0147	0.0147	0.5268	0.5268
nsga2	12	0.0004	0.0106	0.0106	0.5325	0.5325
moead_cheby	13	0.0004	0.0099	0.0099	0.5313	0.5313
moead_linear	13	0.0004	0.0249	0.0249	0.5114	0.5114
moead_mod_cheby	13	0.0003	0.0119	0.0119	0.5295	0.5295
moead_mod_linear	13	0.0019	0.0489	0.0489	0.4734	0.4734
nsga2	13	0.0003	0.0097	0.0097	0.5311	0.5311
moead_cheby	14	0.0135	0.0093	0.0135	0.7781	0.5314
moead_linear	14	0.0004	0.0174	0.0174	0.5211	0.5211
moead_mod_cheby	14	0.0213	0.0108	0.0213	1.0735	0.5298
moead_mod_linear	14	0.0004	0.0091	0.0091	0.5318	0.5318
nsga2	14	0.0003	0.0109	0.0109	0.5318	0.5318
moead_cheby	15	0.0103	0.0098	0.0103	0.9601	0.5320
moead_linear	15	0.0007	0.0096	0.0096	0.5311	0.5311
moead_mod_cheby	15	0.0004	0.0108	0.0108	0.5309	0.5309
moead_mod_linear	15	0.0066	0.0215	0.0215	0.5131	0.5131
nsga2	15	0.0004	0.0098	0.0098	0.5320	0.5320
moead_cheby	16	0.0059	0.0109	0.0109	0.6999	0.5296
moead_linear	16	0.0029	0.0314	0.0314	0.4968	0.4968
moead_mod_cheby	16	0.0003	0.0104	0.0104	0.5303	0.5303
moead_mod_linear	16	0.0004	0.0106	0.0106	0.5311	0.5311
nsga2	16	0.0008	0.0103	0.0103	0.5301	0.5301
moead_cheby	17	0.0005	0.0088	0.0088	0.5328	0.5328
moead_linear	17	0.0011	0.0093	0.0093	0.5312	0.5312
moead_mod_cheby	17	0.0004	0.0105	0.0105	0.5304	0.5304
moead_mod_linear	17	0.0004	0.0700	0.0700	0.4577	0.4577
nsga2	17	0.0004	0.0107	0.0107	0.5310	0.5310
moead_cheby	18	0.0004	0.0086	0.0086	0.5329	0.5329
moead_linear	18	0.0045	0.0200	0.0200	0.5158	0.5158
moead_mod_cheby	18	0.0004	0.0105	0.0105	0.5303	0.5303
moead_mod_linear	18	0.0015	0.0819	0.0819	0.4366	0.4366
nsga2	18	0.0003	0.0103	0.0103	0.5318	0.5318
moead_cheby	19	0.0007	0.0091	0.0091	0.5320	0.5320
moead_linear	19	0.0005	0.0097	0.0097	0.5326	0.5326
moead_mod_cheby	19	0.0003	0.0106	0.0106	0.5314	0.5314
moead_mod_linear	19	0.0020	0.0626	0.0626	0.4518	0.4518
nsga2	19	0.0004	0.0099	0.0099	0.5317	0.5317
moead_cheby	20	0.0005	0.0095	0.0095	0.5321	0.5321
moead_linear	20	0.0006	0.0178	0.0178	0.5211	0.5211
moead_mod_cheby	20	0.0006	0.0111	0.0111	0.5304	0.5304
moead_mod_linear	20	0.0024	0.0418	0.0418	0.4830	0.4830
nsga2	20	0.0004	0.0090	0.0090	0.5323	0.5323
moead_cheby	21	0.0013	0.0089	0.0089	0.5327	0.5327
moead_linear	21	0.0005	0.0216	0.0216	0.5159	0.5159

Algorithm	Run	GD	IGD	Δ	HV Platemo	HV Rect
moead_mod_cheby	21	0.0003	0.0106	0.0106	0.5305	0.5305
moead_mod_linear	21	0.0004	0.0415	0.0415	0.4894	0.4894
nsga2	21	0.0005	0.0097	0.0097	0.5316	0.5316
moead_cheby	22	0.0006	0.0094	0.0094	0.5323	0.5323
moead_linear	22	0.0012	0.0166	0.0166	0.5209	0.5209
moead_mod_cheby	22	0.0004	0.0109	0.0109	0.5302	0.5302
moead_mod_linear	22	0.0009	0.0103	0.0103	0.5291	0.5291
nsga2	22	0.0004	0.0100	0.0100	0.5311	0.5311
moead_cheby	23	0.0005	0.0111	0.0111	0.5306	0.5306
moead_linear	23	0.0006	0.0112	0.0112	0.5290	0.5290
moead_mod_cheby	23	0.0006	0.0117	0.0117	0.5299	0.5299
moead_mod_linear	23	0.0010	0.0130	0.0130	0.5261	0.5261
nsga2	23	0.0004	0.0094	0.0094	0.5317	0.5317
moead_cheby	24	0.0013	0.0093	0.0093	0.5319	0.5319
moead_linear	24	0.0010	0.0169	0.0169	0.5193	0.5193
moead_mod_cheby	24	0.0004	0.0109	0.0109	0.5301	0.5301
moead_mod_linear	24	0.0018	0.0436	0.0436	0.4773	0.4773
nsga2	24	0.0004	0.0095	0.0095	0.5320	0.5320
moead_cheby	25	0.0008	0.0099	0.0099	0.5313	0.5313
moead_linear	25	0.0007	0.0115	0.0115	0.5289	0.5289
moead_mod_cheby	25	0.0004	0.0104	0.0104	0.5309	0.5309
moead_mod_linear	25	0.0007	0.0890	0.0890	0.4393	0.4393
nsga2	25	0.0004	0.0094	0.0094	0.5324	0.5324
moead_cheby	26	0.0008	0.0090	0.0090	0.5324	0.5324
moead_linear	26	0.0011	0.0105	0.0105	0.5304	0.5304
moead_mod_cheby	26	0.0004	0.0102	0.0102	0.5308	0.5308
moead_mod_linear	26	0.0004	0.0100	0.0100	0.5308	0.5308
nsga2	26	0.0004	0.0099	0.0099	0.5319	0.5319
moead_cheby	27	0.0005	0.0108	0.0108	0.5308	0.5308
moead_linear	27	0.0026	0.0247	0.0247	0.5098	0.5098
moead_mod_cheby	27	0.0004	0.0119	0.0119	0.5296	0.5296
moead_mod_linear	27	0.0008	0.0110	0.0110	0.5293	0.5293
nsga2	27	0.0003	0.0094	0.0094	0.5320	0.5320
moead_cheby	28	0.0010	0.0091	0.0091	0.5317	0.5317
moead_linear	28	0.0008	0.0114	0.0114	0.5283	0.5283
moead_mod_cheby	28	0.0004	0.0113	0.0113	0.5307	0.5307
moead_mod_linear	28	0.0012	0.0130	0.0130	0.5258	0.5258
nsga2	28	0.0003	0.0092	0.0092	0.5321	0.5321
moead_cheby	29	0.0112	0.0091	0.0112	1.0189	0.5324
moead_linear	29	0.0006	0.0109	0.0109	0.5299	0.5299
moead_mod_cheby	29	0.0004	0.0103	0.0103	0.5313	0.5313
moead_mod_linear	29	0.0033	0.0127	0.0127	0.5249	0.5249
nsga2	29	0.0004	0.0100	0.0100	0.5316	0.5316
moead_cheby	30	0.0005	0.0086	0.0086	0.5325	0.5325

Algorithm	Run	GD	IGD	Δ	HV Platemo	HV Rect
moead_linear	30	0.0007	0.0093	0.0093	0.5326	0.5326
moead_mod_cheby	30	0.0004	0.0105	0.0105	0.5316	0.5316
moead_mod_linear	30	0.0012	0.0100	0.0100	0.5301	0.5301
nsga2	30	0.0009	0.0099	0.0099	0.5308	0.5308

ZDT3 Results:

Algorithm	Run	GD	IGD	Δ	HV Platemo	HV Rect
moead_cheby	1	0.0025	0.1929	0.1929	1.3255	1.3255
moead_linear	1	0.0290	0.1940	0.1940	2.2038	1.3254
moead_mod_cheby	1	0.0021	0.1936	0.1936	1.3251	1.3251
moead_mod_linear	1	0.0021	0.1944	0.1944	1.3255	1.3255
nsga2	1	0.0021	0.1934	0.1934	1.3256	1.3256
moead_cheby	2	0.0079	0.1935	0.1935	1.4605	1.3254
moead_linear	2	0.0349	0.1941	0.1941	1.7703	1.3251
moead_mod_cheby	2	0.0019	0.1933	0.1933	1.3258	1.3258
moead_mod_linear	2	0.0023	0.1958	0.1958	1.3243	1.3243
nsga2	2	0.0023	0.1935	0.1935	1.3251	1.3251
moead_cheby	3	0.0023	0.1933	0.1933	1.3256	1.3256
moead_linear	3	0.0020	0.1935	0.1935	1.3252	1.3252
moead_mod_cheby	3	0.0020	0.1937	0.1937	1.3251	1.3251
moead_mod_linear	3	0.0022	0.1948	0.1948	1.2700	1.2700
nsga2	3	0.0017	0.1937	0.1937	1.3246	1.3246
moead_cheby	4	0.0026	0.1929	0.1929	1.3252	1.3252
moead_linear	4	0.0017	0.1938	0.1938	1.3261	1.3261
moead_mod_cheby	4	0.0027	0.1933	0.1933	1.3256	1.3256
moead_mod_linear	4	0.0027	0.1927	0.1927	1.3221	1.3221
nsga2	4	0.0018	0.1936	0.1936	1.3250	1.3250
moead_cheby	5	0.0025	0.1933	0.1933	1.3252	1.3252
moead_linear	5	0.0386	0.1936	0.1936	2.3171	1.3254
moead_mod_cheby	5	0.0021	0.1939	0.1939	1.3249	1.3249
moead_mod_linear	5	0.0020	0.1928	0.1928	1.3198	1.3198
nsga2	5	0.0020	0.1935	0.1935	1.3250	1.3250
moead_cheby	6	0.0292	0.1935	0.1935	1.8343	1.3249
moead_linear	6	0.0020	0.1937	0.1937	1.3257	1.3257
moead_mod_cheby	6	0.0021	0.1941	0.1941	1.3239	1.3239
moead_mod_linear	6	0.0019	0.1963	0.1963	1.3252	1.3252
nsga2	6	0.0023	0.1933	0.1933	1.3256	1.3256
moead_cheby	7	0.0026	0.1934	0.1934	1.3251	1.3251
moead_linear	7	0.0021	0.1934	0.1934	1.3256	1.3256
moead_mod_cheby	7	0.0020	0.1938	0.1938	1.3243	1.3243
moead_mod_linear	7	0.0021	0.1952	0.1952	1.3251	1.3251
nsga2	7	0.0021	0.1933	0.1933	1.3251	1.3251

Algorithm	Run	GD	IGD	Δ	HV Platemo	HV Rect
moead_cheby	8	0.0030	0.1933	0.1933	1.3252	1.3252
moead_linear	8	0.0061	0.1936	0.1936	1.4399	1.3255
moead_mod_cheby	8	0.0021	0.1935	0.1935	1.3253	1.3253
moead_mod_linear	8	0.0018	0.2022	0.2022	1.3251	1.3251
nsga2	8	0.0019	0.1934	0.1934	1.3251	1.3251
moead_cheby	9	0.0039	0.1933	0.1933	1.3372	1.3258
moead_linear	9	0.0024	0.1942	0.1942	1.3257	1.3257
moead_mod_cheby	9	0.0025	0.1945	0.1945	1.3228	1.3228
moead_mod_linear	9	0.0021	0.1968	0.1968	1.3260	1.3260
nsga2	9	0.0025	0.1937	0.1937	1.3251	1.3251
moead_cheby	10	0.0021	0.1934	0.1934	1.3249	1.3249
moead_linear	10	0.0025	0.1942	0.1942	1.3256	1.3256
moead_mod_cheby	10	0.0017	0.1934	0.1934	1.3255	1.3255
moead_mod_linear	10	0.0023	0.1936	0.1936	1.3253	1.3253
nsga2	10	0.0020	0.1935	0.1935	1.3250	1.3250
moead_cheby	11	0.0033	0.1932	0.1932	1.3254	1.3254
moead_linear	11	0.0027	0.1956	0.1956	1.3245	1.3245
moead_mod_cheby	11	0.0122	0.1940	0.1940	1.4833	1.3237
moead_mod_linear	11	0.0022	0.2013	0.2013	1.3228	1.3228
nsga2	11	0.0020	0.1937	0.1937	1.3244	1.3244
moead_cheby	12	0.0025	0.1934	0.1934	1.3251	1.3251
moead_linear	12	0.0028	0.1950	0.1950	1.3260	1.3260
moead_mod_cheby	12	0.0020	0.1935	0.1935	1.3251	1.3251
moead_mod_linear	12	0.0025	0.1988	0.1988	1.3247	1.3247
nsga2	12	0.0022	0.1934	0.1934	1.3253	1.3253
moead_cheby	13	0.0024	0.1924	0.1924	1.3256	1.3256
moead_linear	13	0.0027	0.1954	0.1954	1.3258	1.3258
moead_mod_cheby	13	0.0021	0.1935	0.1935	1.3258	1.3258
moead_mod_linear	13	0.0018	0.2014	0.2014	1.2888	1.2888
nsga2	13	0.0023	0.1934	0.1934	1.3252	1.3252
moead_cheby	14	0.0146	0.1934	0.1934	1.5671	1.3256
moead_linear	14	0.0106	0.1938	0.1938	1.6672	1.3255
moead_mod_cheby	14	0.0432	0.1937	0.1937	1.9722	1.3251
moead_mod_linear	14	0.0024	0.1940	0.1940	1.3251	1.3251
nsga2	14	0.0019	0.1934	0.1934	1.3250	1.3250
moead_cheby	15	0.0113	0.1928	0.1928	1.7151	1.3254
moead_linear	15	0.0034	0.1930	0.1930	1.3255	1.3255
moead_mod_cheby	15	0.0032	0.1940	0.1940	1.3243	1.3243
moead_mod_linear	15	0.0020	0.1966	0.1966	1.3246	1.3246
nsga2	15	0.0016	0.1934	0.1934	1.3254	1.3254
moead_cheby	16	0.0076	0.1933	0.1933	1.4952	1.3255
moead_linear	16	0.0026	0.1933	0.1933	1.3258	1.3258
moead_mod_cheby	16	0.0020	0.1936	0.1936	1.3252	1.3252
moead_mod_linear	16	0.0029	0.1917	0.1917	1.3000	1.3000

Algorithm	Run	GD	IGD	Δ	HV Platemo	HV Rect
nsga2	16	0.0017	0.1933	0.1933	1.3255	1.3255
moead_cheby	17	0.0018	0.1934	0.1934	1.3254	1.3254
moead_linear	17	0.0030	0.1933	0.1933	1.3243	1.3243
moead_mod_cheby	17	0.0062	0.1947	0.1947	1.4458	1.3224
moead_mod_linear	17	0.0023	0.2109	0.2109	1.3167	1.3167
nsga2	17	0.0023	0.1935	0.1935	1.3250	1.3250
moead_cheby	18	0.0026	0.1933	0.1933	1.3249	1.3249
moead_linear	18	0.0020	0.1934	0.1934	1.3252	1.3252
moead_mod_cheby	18	0.0026	0.1932	0.1932	1.3252	1.3252
moead_mod_linear	18	0.0020	0.1959	0.1959	1.3243	1.3243
nsga2	18	0.0020	0.1937	0.1937	1.3248	1.3248
moead_cheby	19	0.0026	0.1934	0.1934	1.3254	1.3254
moead_linear	19	0.0026	0.1935	0.1935	1.3261	1.3261
moead_mod_cheby	19	0.0197	0.1936	0.1936	2.1852	1.3252
moead_mod_linear	19	0.0025	0.2131	0.2131	1.3034	1.3034
nsga2	19	0.0021	0.2105	0.2105	1.2424	1.2424
moead_cheby	20	0.0023	0.1933	0.1933	1.3254	1.3254
moead_linear	20	0.0020	0.1939	0.1939	1.3261	1.3261
moead_mod_cheby	20	0.0035	0.1937	0.1937	1.3248	1.3248
moead_mod_linear	20	0.0021	0.1940	0.1940	1.3256	1.3256
nsga2	20	0.0019	0.1931	0.1931	1.3257	1.3257
moead_cheby	21	0.0028	0.1932	0.1932	1.3258	1.3258
moead_linear	21	0.0021	0.1937	0.1937	1.3262	1.3262
moead_mod_cheby	21	0.0131	0.1935	0.1935	1.8184	1.3252
moead_mod_linear	21	0.0023	0.1945	0.1945	1.3254	1.3254
nsga2	21	0.0020	0.1936	0.1936	1.3245	1.3245
moead_cheby	22	0.0025	0.1918	0.1918	1.3254	1.3254
moead_linear	22	0.0027	0.1944	0.1944	1.3258	1.3258
moead_mod_cheby	22	0.0021	0.1934	0.1934	1.3256	1.3256
moead_mod_linear	22	0.0022	0.1944	0.1944	1.3258	1.3258
nsga2	22	0.0020	0.1933	0.1933	1.3253	1.3253
moead_cheby	23	0.0022	0.1934	0.1934	1.3252	1.3252
moead_linear	23	0.0029	0.1947	0.1947	1.3257	1.3257
moead_mod_cheby	23	0.0022	0.1934	0.1934	1.3257	1.3257
moead_mod_linear	23	0.0028	0.1947	0.1947	1.2926	1.2926
nsga2	23	0.0018	0.2105	0.2105	1.2422	1.2422
moead_cheby	24	0.0029	0.1933	0.1933	1.3256	1.3256
moead_linear	24	0.0036	0.1946	0.1946	1.3250	1.3250
moead_mod_cheby	24	0.0026	0.1935	0.1935	1.3252	1.3252
moead_mod_linear	24	0.0030	0.1927	0.1927	1.2770	1.2770
nsga2	24	0.0021	0.1932	0.1932	1.3256	1.3256
moead_cheby	25	0.0023	0.1931	0.1931	1.3258	1.3258
moead_linear	25	0.0026	0.1944	0.1944	1.3255	1.3255
moead_mod_cheby	25	0.0046	0.1938	0.1938	1.3328	1.3254

Algorithm	Run	GD	IGD	Δ	HV Platemo	HV Rect
moead_mod_linear	25	0.0025	0.2045	0.2045	1.2919	1.2919
nsga2	25	0.0024	0.1934	0.1934	1.3252	1.3252
moead_cheby	26	0.0028	0.1933	0.1933	1.3253	1.3253
moead_linear	26	0.0091	0.1971	0.1971	1.5718	1.3248
moead_mod_cheby	26	0.0017	0.1937	0.1937	1.3252	1.3252
moead_mod_linear	26	0.0021	0.1940	0.1940	1.3092	1.3092
nsga2	26	0.0020	0.1934	0.1934	1.3252	1.3252
moead_cheby	27	0.0022	0.1933	0.1933	1.3254	1.3254
moead_linear	27	0.0021	0.1935	0.1935	1.3258	1.3258
moead_mod_cheby	27	0.0017	0.1935	0.1935	1.3252	1.3252
moead_mod_linear	27	0.0021	0.1963	0.1963	1.3249	1.3249
nsga2	27	0.0020	0.1937	0.1937	1.3245	1.3245
moead_cheby	28	0.0027	0.1931	0.1931	1.3259	1.3259
moead_linear	28	0.0020	0.1931	0.1931	1.3260	1.3260
moead_mod_cheby	28	0.0022	0.1935	0.1935	1.3258	1.3258
moead_mod_linear	28	0.0021	0.1941	0.1941	1.3249	1.3249
nsga2	28	0.0024	0.1935	0.1935	1.3253	1.3253
moead_cheby	29	0.0133	0.1933	0.1933	1.8119	1.3255
moead_linear	29	0.0028	0.1938	0.1938	1.3256	1.3256
moead_mod_cheby	29	0.0023	0.1934	0.1934	1.3251	1.3251
moead_mod_linear	29	0.0027	0.1957	0.1957	1.3056	1.3056
nsga2	29	0.0021	0.1937	0.1937	1.3246	1.3246
moead_cheby	30	0.0025	0.1933	0.1933	1.3248	1.3248
moead_linear	30	0.0021	0.1941	0.1941	1.3261	1.3261
moead_mod_cheby	30	0.0027	0.1939	0.1939	1.3253	1.3253
moead_mod_linear	30	0.0022	0.2078	0.2078	1.3225	1.3225
nsga2	30	0.0019	0.1936	0.1936	1.3249	1.3249

References

[1] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002, doi: 10.1109/4235.996017.

[2] 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, doi: 10.1109/TEVC.2007.892759.

[3] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, “PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum],” *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017, doi: 10.1109/MCI.2017.2742868.

[4] L. While, P. Hingston, L. Barone, and S. Huband, “A faster algorithm for calculating hypervolume,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 29–38, 2006, doi: 10.1109/TEVC.2005.851275.