Multi-objective Optimization Problem Solving

Unit 5: MOOPs, MOEAs, Non-Pareto & Pareto Approaches, NSGA-II & Applications

Dr. Gopal Chandra Jana

Dept. of CSE, Sharda University
Soft Computing (CSA301) — Unit 5

July 21, 2026

Course Page:
https://www.gcjana.in/courses/shardauniversity/2601/soft-computing/

Outline

Learning Outcomes of Unit 5

After this unit, a student will be able to...

1. **Formulate** a multi-objective optimisation problem (MOOP) mathematically.
2. **Explain** Pareto dominance, Pareto-optimal sets and Pareto fronts.
3. **Identify** the issues that make MOOPs harder than single-objective problems.
4. **Distinguish** non-Pareto (aggregation/criterion) from Pareto-based MOEAs.
5. **Trace** NSGA-II: non-dominated sorting and crowding distance, with numerics.
6. **Survey** MOEA applications and the wider family of nature-inspired optimisers.

Mapping to the Syllabus

A. MOOP concepts & issues. **B.** MOEA: non-Pareto & Pareto approaches; applications. **C.** Hybrids & relatives (GA+PSO, DE, ABC, firefly, cuckoo) and NSGA-II.

From Single- to Multi-objective Optimisation

Single-objective (Unit 4)

One goal; the answer is a **single best** solution.

$$\min_{x} \; f(x)$$

A total order exists: any two solutions are comparable.

Multi-objective

Several **conflicting** goals optimised at once.

$$\min_{x} \; \big(f_1(x), f_2(x), \ldots, f_M(x)\big)$$

No single "best": improving one objective usually *worsens* another.

Everyday conflict — buying a car

- Minimise *cost* **and** minimise *fuel consumption*.
- A cheap car is thirsty; an economical car is dear.
- There is a **set** of sensible trade-offs, not one winner.

Key shift in thinking

The output of a MOOP is not a point but a **set of trade-off solutions** — the decision-maker then picks one.

Formal Statement of a MOOP

General multi-objective optimisation problem

$$\min_{\mathbf{x}\in\Omega} \ \mathbf{F}(\mathbf{x}) = \big(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})\big)$$

subject to

$$g_j(\mathbf{x}) \le 0, \ j = 1, \ldots, J; \qquad h_k(\mathbf{x}) = 0, \ k = 1, \ldots, K.$$

Two spaces

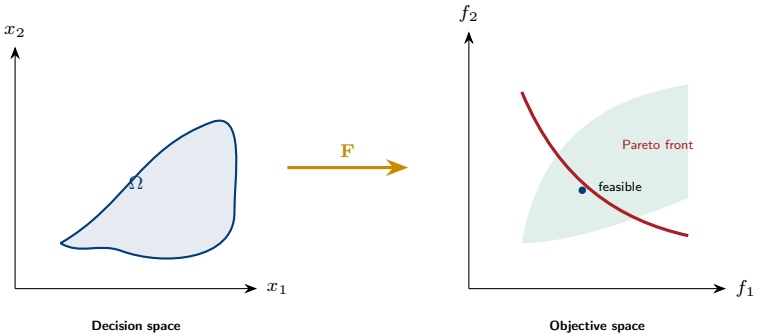
- **Decision space** Ω: the variables $\mathbf{x} = (x_1, \ldots, x_n)$.
- **Objective space**: the images $\mathbf{F}(\mathbf{x}) \in \mathbb{R}^M$.

Optimisation happens in decision space; *comparison* happens in objective space.

Notes

- $M = 1$ = single-objective; $M \ge 2$ = multi-objective; $M \ge 4$ often called *many*-objective.
- Maximisation converts to minimisation via $\max f = -\min(-f)$.
- Ω = feasible region defined by the constraints.

Decision Space vs. Objective Space



Reading the picture

The map $\mathbf{F}$ carries the feasible set Ω into the objective space; the **lower-left boundary** (for minimisation) is the **Pareto front** — the best attainable trade-offs.

Pareto Dominance — The Central Concept

Definition (minimisation)

Solution $\mathbf{a}$ **dominates** $\mathbf{b}$ (written $\mathbf{a} \prec \mathbf{b}$) iff

$$f_i(\mathbf{a}) \leq f_i(\mathbf{b}) \quad \forall i \quad \text{and} \quad f_i(\mathbf{a}) < f_i(\mathbf{b}) \quad \text{for at least one } i.$$

That is, $\mathbf{a}$ is *no worse in every* objective and *strictly better in at least one*.

Three relations

- $\mathbf{a}$ **dominates** $\mathbf{b}$: $\mathbf{a}$ is clearly preferable.
- $\mathbf{b}$ dominates $\mathbf{a}$: the reverse.
- **Non-dominated** (incomparable): each is better in some objective — a genuine trade-off.

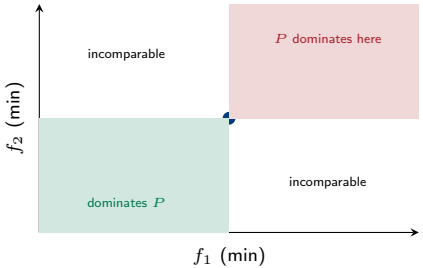


Figure: The four quadrants around a point P.

Numerical Example — Who Dominates Whom?

Seven solutions (minimise both f_1 and f_2)

Solution	A	B	C	D	E	F	G
f_1	2	3	5	4	6	7	1
f_2	8	5	4	6	2	7	9

Checking dominance

- B$(3,5)$ vs D$(4,6)$: B $\leq$ D in both, strictly $<$ in both $\Rightarrow$ **B dominates D**.
- B$(3,5)$ vs F$(7,7)$: $\Rightarrow$ **B dominates F**.
- C$(5,4)$ vs E$(6,2)$: C better in f_1, worse in f_2 $\Rightarrow$ **non-dominated**.

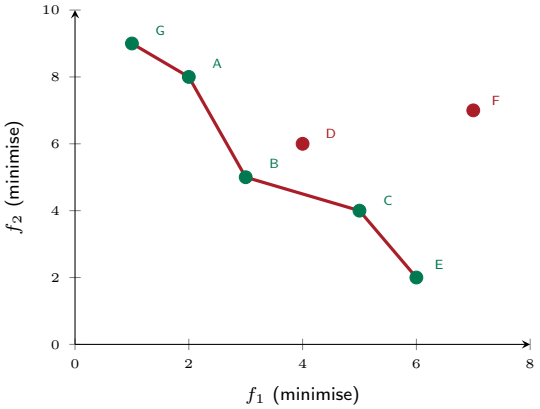
Result

Only **D** and **F** are dominated (both by B; F also by C, D, E).

Pareto-optimal set $= \{A, B, C, E, G\}$.

We plot and confirm this on the next slide.

The Pareto Front — Plotting the Example



What the plot shows

- Green points $\{G, A, B, C, E\}$ form the **Pareto front** — the non-dominated trade-off surface.
- Red points $\{D, F\}$ are **dominated** — they lie "above-right" of the front.

Terminology

Pareto-optimal set lives in **decision** space; its image, the *Pareto front*, lives in **objective** space.

Goals of Multi-objective Optimisation

Two competing goals

1. **Convergence**: drive solutions *onto* (or close to) the true Pareto front — accuracy.
2. **Diversity**: spread solutions *evenly along* the front — coverage.

Both matter

A tight cluster on the front is accurate but useless (few choices). A wide but far-off spread offers choices that are all poor. A good MOEA achieves **both** at once.

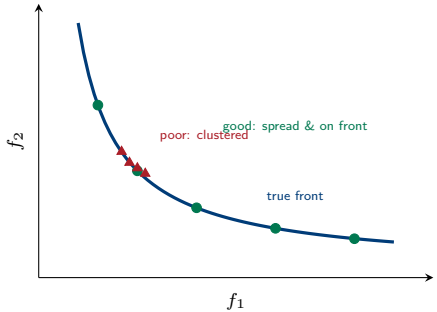


Figure: Convergence *and* diversity are both required.

Issues in Solving MOOPs

Why MOOPs are hard

- **No total order**: solutions are only *partially* ordered by dominance.
- The answer is a **set** (possibly infinite), not a point.
- Must balance **convergence** and **diversity** simultaneously.
- Objectives may be on **different scales** $\Rightarrow$ normalisation needed.
- **Many objectives** ($M \geq 4$): almost everything becomes non-dominated.
- Expensive/black-box objectives; noisy evaluations.

Why classical methods struggle

- A weighted sum returns **one** point per weight vector — many runs needed.
- It **misses** solutions on *non-convex* parts of the front.
- Choosing weights *a priori* needs knowledge we often lack.

The evolutionary advantage

A population-based MOEA finds **many** trade-off solutions in a **single run** — the natural fit for MOOPs.

Quick Check — Round 1 (MOOP Concepts)

[Q1]

State the Pareto dominance condition for minimisation of two objectives.

[Q2]

For minimisation, does $(3, 7)$ dominate $(3, 9)$? Justify.

[Q3]

Name the two goals every MOEA must achieve simultaneously.

[Q4] True/False

"A weighted-sum method can capture every point on a non-convex Pareto front."

Think for a minute — answers on the next slide.

Answers — Round 1

[A1]

$\mathbf{a}$ dominates $\mathbf{b}$ iff $f_i(\mathbf{a}) \leq f_i(\mathbf{b})$ for *all* i **and** $f_i(\mathbf{a}) < f_i(\mathbf{b})$ for *at least one* i.

[A2]

Yes. $(3, 7)$ is equal in f_1 $(3 = 3)$ and strictly better in f_2 $(7 < 9)$ — no worse everywhere, better somewhere. So $(3, 7) \prec (3, 9)$.

[A3]

Convergence (get onto the true Pareto front) and **Diversity** (spread evenly across it).

[A4]

False. The weighted sum can only reach points on the *convex hull* of the front; solutions in non-convex "dips" are unreachable for any positive weights.

Classical (Non-Pareto) Approaches — Overview

Idea: convert the MOOP into single-objective problems

Also called **a priori** or **scalarisation** methods — preferences are set *before* the search.

Main techniques

- **Weighted sum**: $F = \sum_i w_i f_i$
- **Weighted metric (L_p)**: $F = \left(\sum_i w_i |f_i - z_i^*|^p \right)^{1/p}$
- **ε-constraint**: optimise f_1, bound the rest $f_j \leq \varepsilon_j$
- **Goal programming**: minimise deviation from targets
- **Lexicographic**: rank objectives by priority

Pros

Simple; reuses standard single-objective solvers; one clear answer per run.

Cons

- One run $\Rightarrow$ one point; need many runs for the front.
- Weighted sum misses **non-convex** regions.
- Requires **a-priori** preference/weights.
- Sensitive to objective **scaling**.

Numerical Example — Weighted-Sum Scalarisation

Minimise $F = w_1 f_1 + w_2 f_2$ over four candidates

	f_1	f_2	F at $(0.5, 0.5)$	F at $(0.8, 0.2)$	F at $(0.2, 0.8)$
A	2	8	5.0	3.2	6.8
B	3	5	**4.0**	3.4	4.6
C	5	4	4.5	4.8	4.2
E	6	2	**4.0**	5.2	**2.8**

Reading the columns

- **Equal weights** $(0.5, 0.5)$: B and E tie at 4.0.
- **Favour** f_1 $(0.8, 0.2)$: A wins (3.2).
- **Favour** f_2 $(0.2, 0.8)$: E wins (2.8).

Lesson

Different weights ⇒ different single winners.
To map the whole front you must sweep many weight vectors — and convex-only coverage still limits you.

The ε-Constraint Method

Idea

Keep **one** objective; turn the others into **constraints**:

$$\min_{\mathbf{x}} f_1(\mathbf{x}) \quad \text{s.t. } f_j(\mathbf{x}) \le \varepsilon_j,\ j \neq 1.$$

Sweeping ε

Varying the bounds ε_j traces out **different points** on the Pareto front — *including* non-convex regions (its advantage over weighted sum).

Cost

Still *a priori*, one point per run, and choosing sensible ε values needs range information.

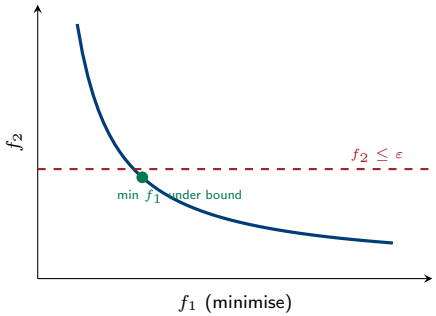


Figure: Bound $f_2 \le \varepsilon$, then minimise f_1; slide the bound to sweep the front.

VEGA — The First Non-Pareto MOEA

Vector Evaluated GA (Schaffer, 1985)

For M objectives, split the mating pool into M equal parts; each part is selected using **one** objective only, then all parts are shuffled, crossed and mutated together.

Mechanism (for $M = 2$, population N)

1. Fill sub-population 1 by selecting on f_1 ($N/2$ individuals).
2. Fill sub-population 2 by selecting on f_2 ($N/2$ individuals).
3. Merge, then apply crossover & mutation as usual.

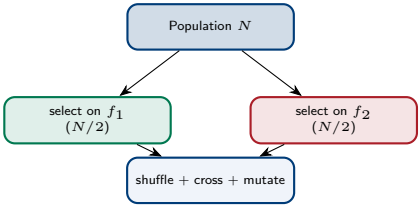


Figure: VEGA selects each sub-population on a single objective.

Weakness

Drifts toward **extreme** solutions (champions of one objective); the middle of the front is under-populated.

Pareto-based Approaches — The Big Idea

Rank by dominance, not by aggregation

Use **Pareto dominance directly** to rank the population, so the whole front is approximated in **one run** — no weights, no a-priori preference.

Two ingredients every Pareto MOEA needs

1. A **fitness / ranking** based on dominance (drives **convergence**).
2. A **diversity** mechanism — niching, sharing, or crowding (spreads solutions).

A short family tree

Algorithm	Year / idea
MOGA	1993, rank = #dominators
NPGA	1994, tournament by dominance
NSGA	1994, non-dominated sorting
SPEA / SPEA2	1999/2001, strength + archive
NSGA-II	2002, fast sort + crowding
MOEA/D	2007, decomposition

NSGA-II is the workhorse we study in detail next.

Diversity Preservation: Sharing & Crowding

Fitness sharing (niching)

Penalise crowded solutions by dividing fitness by a *niche count*:

$$f_i' = \frac{f_i}{\sum_j \mathrm{sh}(d_{ij})}, \quad \mathrm{sh}(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{\mathrm{share}}}\right)^{\alpha} & d < \sigma_{\mathrm{share}} \\ 0 & \text{else} \end{cases}$$

Needs a niche radius σ_{share} (hard to set).

Crowding distance (NSGA-II)

Parameter-free: estimate how "crowded" a solution is by the size of the cuboid formed by its nearest neighbours on the same front (next section).

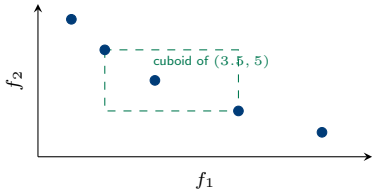


Figure: The neighbour cuboid measures local crowding.

Quick Check — Round 2 (Non-Pareto vs Pareto)

[Q1]

Give one advantage and one disadvantage of the weighted-sum method.

[Q2]

Which classical method can reach non-convex parts of the Pareto front, and how?

[Q3]

What is the main weakness of VEGA?

[Q4]

Name the two ingredients every Pareto-based MOEA must provide.

Answers — Round 2

[A1]

Advantage: simple — reuses a standard single-objective solver, one clear answer per run. *Disadvantage*: one run gives one point, it needs a-priori weights, and it **misses non-convex** regions of the front.

[A2]

The **ε-constraint** method: minimise one objective while bounding the others ($f_j \leq \varepsilon_j$); sweeping ε reaches non-convex points too.

[A3]

VEGA drifts toward **extreme** (single-objective champion) solutions, leaving the *middle* of the Pareto front poorly covered.

[A4]

(i) A **dominance-based ranking** for convergence, and (ii) a **diversity** mechanism (sharing / crowding) for spread.

NSGA-II — The Elitist Workhorse

Deb et al. (2002)

NSGA-II is the most widely used MOEA. Its three pillars:

1. **Fast non-dominated sorting** — rank the population into fronts $\mathcal{F}_1, \mathcal{F}_2, \ldots$
2. **Crowding distance** — a parameter-free diversity measure.
3. **Elitism** — combine parents + offspring ($2N$) and keep the best N.

Selection rule (crowded-comparison $\prec_n$)

Prefer the **lower front rank**; if equal rank, prefer the **larger crowding distance** (the less crowded solution).

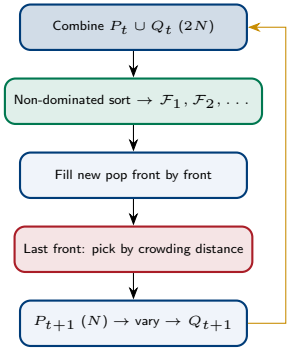


Figure: One NSGA-II generation (elitist $(\mu + \lambda)$ survival).

Step 1 — Fast Non-dominated Sorting

Algorithm (per individual p)

For each p compute: n_p = number of solutions that *dominate* p, and S_p = set of solutions p *dominates*.

- Front $\mathcal{F}_1$ = all p with $n_p = 0$ (dominated by none).
- Remove $\mathcal{F}_1$; decrement n_q for each $q \in S_p$; new zeros form $\mathcal{F}_2$; repeat.

Apply to our 8-point set

Points: A$(2,8)$ B$(3,5)$ C$(5,4)$ D$(4,6)$ E$(6,2)$ F$(7,7)$ G$(1,9)$ H$(8,9)$.

- $\mathcal{F}_1 = \{A, B, C, E, G\}$ ($n_p = 0$).
- $\mathcal{F}_2 = \{D\}$ (only B dominated it).
- $\mathcal{F}_3 = \{F\}$; $\mathcal{F}_4 = \{H\}$.

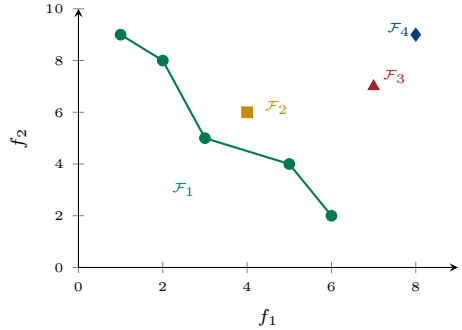


Figure: The population sorted into four fronts.

Step 2 — Crowding Distance (Definition)

Per objective m, on a single front

Sort the front by objective m. Boundary solutions get ∞ (always kept). For an interior solution i:

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

summed over all objectives m. Larger $d_i \Rightarrow$ more isolated $\Rightarrow$ preferred.

Why normalise by range

Dividing by $(f_m^{\max} - f_m^{\min})$ makes the measure **scale-independent**, so no single objective dominates the distance just because its numbers are larger.

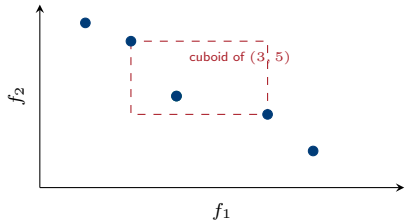


Figure: The cuboid around an interior point defines its crowding distance.

Numerical Example — Crowding Distance (Front $\mathcal{F}_1$)

Front $\mathcal{F}_1$ sorted by f_1

Point	G	A	B	C	E
(f_1, f_2)	$(1, 9)$	$(2, 8)$	$(3, 5)$	$(5, 4)$	$(6, 2)$

Ranges: $f_1^{\max} - f_1^{\min} = 6 - 1 = 5, \quad f_2^{\max} - f_2^{\min} = 9 - 2 = 7.$

Interior point B $(3, 5)$

$$d_{f_1} = \frac{f_1(C) - f_1(A)}{5} = \frac{5 - 2}{5} = 0.600$$

$$d_{f_2} = \frac{f_2(A) - f_2(C)}{7} = \frac{8 - 4}{7} = 0.571$$

$$d_B = 0.600 + 0.571 = \mathbf{1.171}$$

Full front

Point	d
G (boundary)	∞
A	0.971
B	**1.171**
C	1.029
E (boundary)	∞

Boundaries kept first; then B (largest d) is preferred.

Crowding Distance — The Other Interior Points

Point A $(2, 8)$ — neighbours G$(1, 9)$ and B$(3, 5)$

$$d_A = \underbrace{\frac{f_1(B) - f_1(G)}{5}}_{(3-1)/5=0.400} + \underbrace{\frac{f_2(G) - f_2(B)}{7}}_{(9-5)/7=0.571} = \mathbf{0.971}$$

Point C $(5, 4)$ — neighbours B$(3, 5)$ and E$(6, 2)$

$$d_C = \underbrace{\frac{f_1(E) - f_1(B)}{5}}_{(6-3)/5=0.600} + \underbrace{\frac{f_2(B) - f_2(E)}{7}}_{(5-2)/7=0.429} = \mathbf{1.029}$$

Ranking within the front (for truncation)

By crowded-comparison: boundaries **G, E** (∞) first, then **B** (1.171), **C** (1.029), **A** (0.971). If we had to *drop* one, we would drop **A** (most crowded).

NSGA-II — One Generation End-to-End

The loop

1. Combine $R_t = P_t \cup Q_t$ (size $2N$).
2. Non-dominated sort R_t into $\mathcal{F}_1, \mathcal{F}_2, \ldots$
3. Add whole fronts to P_{t+1} until adding the next would overflow N.
4. For the **splitting** front, sort by **crowding distance** and take just enough to reach N.
5. Tournament-select, crossover, mutate $\rightarrow Q_{t+1}$.

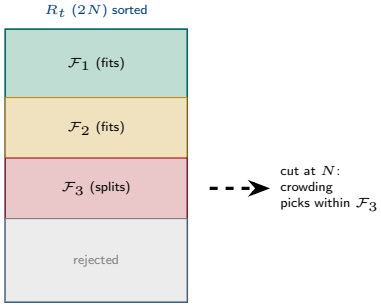


Figure: Fronts fill P_{t+1}; the splitting front is trimmed by crowding distance.

Numerical Example — NSGA-II Truncation Step

Setup

Suppose $N = 6$. After sorting R_t we have $\mathcal{F}_1 = \{G, A, B, C, E\}$ (5 points) and $\mathcal{F}_2 = \{D, \mathrm{X}\}$ (2 points), where $\mathcal{F}_2$ crowding distances are $d_D = \infty,\ d_X = 0.4$.

Fill the next population

1. $\mathcal{F}_1$ (5 points) fits entirely $\Rightarrow$ take all 5. Slots left: $6 - 5 = 1$.
2. $\mathcal{F}_2$ has 2 points but only **1 slot** $\Rightarrow$ it *splits*.
3. Sort $\mathcal{F}_2$ by crowding distance (descending): $D(\infty) > X(0.4)$.
4. Take **D**; drop X.

Resulting P_{t+1}

$$P_{t+1} = \{G, A, B, C, E\} \cup \{D\}$$

exactly $N = 6$ solutions — the best front kept whole, the tie broken in favour of the **less crowded** point.

Take-away

Rank first, crowding second. Elitism (using $R_t = P_t \cup Q_t$) guarantees good solutions are never lost.

NSGA-II vs. SPEA2 vs. MOEA/D

	NSGA-II	SPEA2	MOEA/D
Ranking	Non-dominated fronts	Strength (dominators)	Decomposition (weights)
Diversity	Crowding distance	k-NN density	Neighbourhood of sub-problems
Elitism	$P_t \cup Q_t$, keep best N	External archive	Best per subproblem
Best when	2–3 objectives	2–3 objectives	many objectives
Cost	$O(MN^2)$ per gen	higher (archive)	lower per subproblem

Guidance

NSGA-II is the default for 2–3 objectives; **MOEA/D** scales better to *many* objectives; **SPEA2** keeps a fine-grained external elite archive.

Quick Check — Round 3 (NSGA-II)

[Q1]

Name the three pillars of NSGA-II.

[Q2]

In non-dominated sorting, which individuals form the first front $\mathcal{F}_1$?

[Q3]

On a front, point P has neighbours with $f_1 = 4$ and $f_1 = 10$; the front's f_1 range is 12. What is P's f_1 contribution to its crowding distance?

[Q4]

In the crowded-comparison operator, if two solutions have the *same* front rank, which is preferred?

Answers — Round 3

[A1]

Fast non-dominated sorting, **crowding-distance** diversity, and **elitism** (combine $P_t \cup Q_t$, keep the best N).

[A2]

All individuals with $n_p = 0$ — those **dominated by no one** (the current non-dominated set).

[A3]

Contribution $= \dfrac{10 - 4}{12} = \dfrac{6}{12} = \mathbf{0.5}$.

[A4]

The one with the **larger crowding distance** — i.e. the solution in the *less crowded* region, to preserve diversity.

Applications of MOEAs

Engineering & design

- Structural design: minimise weight *and* stress.
- Aerodynamic / antenna shape optimisation.
- VLSI floor-planning: area vs. wire length.
- Control: performance vs. energy vs. robustness.

Operations & logistics

- Scheduling: makespan vs. cost vs. tardiness.
- Vehicle routing: distance vs. time vs. emissions.
- Supply chains: cost vs. service level.

Data science & ML

- Feature selection: accuracy vs. #features.
- Model tuning: accuracy vs. complexity vs. latency.
- Neural architecture search (accuracy vs. size).

Finance, energy, environment

- Portfolio: return vs. risk (Markowitz frontier).
- Power dispatch: cost vs. emissions.
- Water / land-use planning: yield vs. sustainability.

Case Study — Portfolio Optimisation (Return vs. Risk)

Two conflicting objectives

$$\max \text{ return } R(\mathbf{w}), \qquad \min \text{ risk } \sigma(\mathbf{w})$$

over portfolio weights $\mathbf{w}$ with $\sum_i w_i = 1,\ w_i \geq 0$.

Why a MOEA?

The output is the **efficient frontier** — a whole set of return/risk trade-offs. An investor then picks a point matching their risk appetite. A single weighted sum would give just one portfolio.

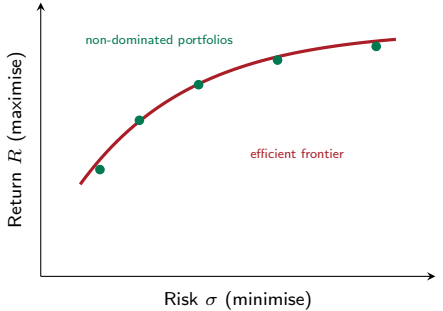


Figure: The efficient (Pareto) frontier of return vs. risk.

Performance Metrics for MOEAs

How do we compare fronts?

- **Hypervolume (HV)**: volume of objective space dominated by the front, relative to a reference point — captures **both** convergence & diversity (bigger is better).
- **Generational Distance (GD)**: average distance to the true front (convergence).
- **Spread (Δ)**: how evenly points are distributed (diversity).
- **IGD**: inverted GD — distance from the true front to the obtained set.

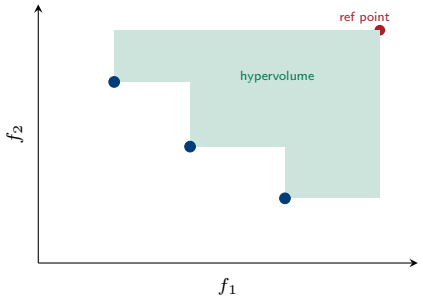


Figure: Hypervolume = area (here) dominated by the front up to a reference point.

Hybrids & Relatives — The Wider Family

Hybrids of GA and swarm

- **GA + PSO**: GA's recombination with PSO's fast convergence.
- **Memetic algorithms**: GA + *local search* (Lamarckian refinement) — "genes that also learn".

Other evolutionary optimisers

- **Differential Evolution (DE)**: vector differences drive mutation; strong on continuous problems.
- **Genetic Programming**: evolves *programs*/trees.

Nature-inspired swarm relatives

- **Artificial Bee Colony (ABC)**: employed/onlooker/scout bees.
- **Firefly Algorithm**: attraction by brightness $\propto$ fitness.
- **Cuckoo Search**: brood parasitism + Lévy flights.
- Also: bat, grey-wolf, whale, ant-lion optimisers.

Multi-objective versions

MOPSO, **MODE**, **NSGA-II/III**, **MOEA/D** extend these ideas to Pareto optimisation.

Differential Evolution — A Quick Numeric

DE/rand/1 mutation

A trial vector is built from three distinct population members:

$$\mathbf{v} = \mathbf{x}_{r1} + F\,(\mathbf{x}_{r2} - \mathbf{x}_{r3})$$

with scale factor $F \in [0, 2]$ (often ≈ 0.5–0.9).

Worked (1-D, $F = 0.5$)

Let $x_{r1} = 6,\ x_{r2} = 10,\ x_{r3} = 2$:

$$v = 6 + 0.5\,(10 - 2) = 6 + 4 = \mathbf{10}$$

Then crossover with the target and keep the better of trial/target (greedy selection).

Why DE works well

- **Self-scaling**: the step size adapts to the population spread (the difference $\mathbf{x}_{r2} - \mathbf{x}_{r3}$ shrinks as it converges).
- Few parameters (F, crossover rate CR, population N).
- Strong, robust performance on continuous optimisation.

MODE applies the same idea with Pareto ranking for multi-objective problems.

Quick Check — Round 4 (Applications & Hybrids)

[Q1]

In portfolio optimisation, what are the two conflicting objectives, and what is the resulting front called?

[Q2]

Which single metric captures *both* convergence and diversity of a MOEA's output?

[Q3]

In DE, compute the mutant for $x_{r1} = 4,\ x_{r2} = 9,\ x_{r3} = 3,\ F = 0.5$.

[Q4] MCQ

Which algorithm uses Lévy flights?
(a) Firefly (b) Cuckoo Search (c) ABC (d) NSGA-II

Answers — Round 4

[A1]

Maximise return and **minimise risk**; the trade-off curve is the **efficient frontier** (a Pareto front).

[A2]

Hypervolume (HV) — the volume of objective space dominated by the front w.r.t. a reference point.

[A3]

$v = x_{r1} + F(x_{r2} - x_{r3}) = 4 + 0.5(9 - 3) = 4 + 3 = \mathbf{7}$.

[A4]

(b) Cuckoo Search — it combines brood parasitism with Lévy-flight random walks.

Summary of Unit 5

What we covered

1. **MOOPs** — several conflicting objectives; the answer is a *set*.
2. **Pareto dominance** — non-dominated set, Pareto-optimal set, Pareto front.
3. **Issues** — partial order, convergence *and* diversity, scaling, many-objective difficulty.
4. **Non-Pareto MOEAs** — weighted sum, ε-constraint, VEGA (and their limits).
5. **Pareto MOEAs** — MOGA, NPGA, NSGA, SPEA2, MOEA/D.
6. **NSGA-II** — fast non-dominated sorting, crowding distance, elitism; full numerics.
7. **Applications** — design, scheduling, ML, portfolio; performance metrics (HV, GD, spread).
8. **Hybrids & relatives** — GA+PSO, memetic, DE, ABC, firefly, cuckoo.

MOEAs find the whole trade-off front in a single run.

Practice Questions (Exam Oriented)

Short answer (2–5 marks)

1. Define a MOOP formally and state the Pareto dominance condition.
2. Differentiate the Pareto-optimal set from the Pareto front.
3. Why does the weighted-sum method fail on non-convex fronts?
4. State the crowding-distance formula and explain the range normalisation.
5. List the three pillars of NSGA-II.

Long answer (8–10 marks) — include numerics

6. Given a set of points, identify all dominated solutions and draw the Pareto front.
7. Perform non-dominated sorting of a given population into fronts.
8. Compute the crowding distances of a given front and rank the solutions.
9. Carry out one NSGA-II truncation step for a stated population size N.
10. Compare NSGA-II, SPEA2 and MOEA/D; state when each is preferred.

Think & Explore (Take-Home)

Numeric drill

For points $\{(1,6),(2,3),(3,4),(5,2),(4,7)\}$ (minimise both):

- Find the Pareto-optimal set.
- Sort the rest into fronts.
- Compute crowding distances for front $\mathcal{F}_1$.

Coding (self-learning)

Run NSGA-II on the ZDT1 / DTLZ test problems using `pymoo`; plot the obtained front and report the hypervolume.

Reading & reflection

- K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*.
- Deb et al. (2002), the NSGA-II paper.
- Explore `pymoo` documentation (algorithms + metrics).
- Revisit Units 1–4: this unit ties fuzzy, neural and evolutionary ideas together.

Reminder

End-Semester Exam covers Units 1–5; Assignment 2 covers Units 3, 4 & 5 — handwritten, step-by-step, single PDF via the course page.

References

Textbooks (as per the course page)

1. K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, Wiley. **[Main text for Unit 5]**
2. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley.
3. J.-S. R. Jang, C.-T. Sun and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice Hall.

Seminal papers

4. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, 6(2):182–197, 2002.
5. J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," *ICGA*, 1985.
6. Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, 2007.

NPTEL: *Soft Computing*, IIT Kharagpur · Course page:
https://www.gcjana.in/courses/shardauniversity/2601/soft-computing/

Thank You

Any Questions?

"The best is the enemy of the good" — but in multi-objective optimisation, there is no single "best": only a **front** of good trade-offs.

End of the Soft Computing (CSA301) lecture series — Units 1 to 5.

Post your doubts on the course page →
https://www.gcjana.in/courses/shardauniversity/2601/soft-computing/#Post-doubts