2025 Fall CSE5025

Combinatorial Optimization

组合优化

Instructor: 刘晟材

Lecture 6-2: Heuristics and Metaheuristics for Combinatorial Optimization – Part II

Agenda for Today's Lecture

In this lecture, we will focus on

- Exploration vs. Exploitation
- Fitness Landscape Analysis
- More types of Combinatorial Optimization

Learning objectives for this lecture

- Know the basic tools to measure and maintain diversity
- Know the tools for analyzing fitness landscapes
- Know different variants of CO and the basic solution ideas for them

The Fundamental Dilemma

Exploitation (Intensification):

- Focusing on a promising region of the search space.
- Refining the current solution to find local optima.
- Risk: Getting trapped in poor local optima (premature convergence).
- Tools: Local Search, Hill Climbing.

Exploration (Diversification):

- Visiting entirely new, unexplored regions.
- Generating solutions that differ significantly from the current solution/population.
- Risk: Wasting computational resources on poor areas; behaving like Random Search.
- Tools: Random restarts, mutation, large neighborhood moves.

Measuring Diversity (1)

For Evolutionary Algorithms, to control exploration, we can quantify diversity as a proxy metric.

Genotypic Diversity (Distance in Solution Space, 基因型多样性):

- How different are the structures of two solutions S_A and S_B?

Hamming Distance (for Binary Vectors):

- Problem: Knapsack, Set Cover.
- Metric: Number of bits that differ, i.e., $d(\mathbf{x}^1, \mathbf{x}^2) = \sum |x_1^1 - x_2^2|$

Measuring Diversity (2)

Edit Distance (for Permutations):

- Problem: Scheduling.
- Metric: Minimum number of swaps to transform $\mathbf{x}^1$ to $\mathbf{x}^2$

Bond/Edge Distance (for Permutations)

- Problem: TSP.
- Metric: Number of unique edges, $n - |E_{\mathbf{x}^1} \cup E_{\mathbf{x}^2}|$, where $E_{\mathbf{x}^1}$ and $E_{\mathbf{x}^2}$ are the edges contained in solutions $\mathbf{x}^1$ and $\mathbf{x}^2$, respectively

Measuring Diversity (Phenotypic)

Phenotypic Diversity (Distance in Objective Space, 表现型多样性):

- How different are the costs or qualities of the solutions?
- Two solutions might look very different (Hamming distance) but have the same cost.

Fitness Distribution:

- Calculate the Variance or Entropy of the fitness values in a population.
- **Low Variance**: The population has converged (High Exploitation).
- **High Variance**: The population is spread out (High Exploration).

Measuring Diversify: Entropy-Based Metrics

Quantifying "Chaos": Shannon Entropy is a powerful tool to measure population diversity.

For a binary problem (e.g., Knapsack) with population size P:

- Let p_j be the proportion of individuals having $x_j = 1$ at index j.
- Entropy at bit j: $H_j = -p_j \log_2 p_j - (1 - p_j) \log_2 (1 - p_j)$.
- Total Population Entropy: $H = \sum_{j=1}^{n} H_j$

Usage:

- If $H \to 0$: All individuals are identical. Trigger **Exploration** (e.g., massive mutation).
- If H is high: Continue **Exploitation**.

Diversity Maintenance: Niching and Crowding

Explicit Diversity Maintenance: inspired by nature, different species occupy different "niches" to avoid competing for the same resource

Fitness Sharing:

- Degrade the fitness of an individual if it is too similar to others.
- $F_i' = \frac{F_i}{m_i}$, where m_i is the "niche count" (density of solutions around i).
- Effect: Forces the algorithm to maintain sub-populations at different peaks.

Crowding:

- When a new child solution is generated, it replaces the **most similar** individual in the parent population (if the child is better).
- Effect: Preserves diversity by preventing one good solution from cloning itself and taking over the whole population.

Diversity Maintenance: Strategic Oscillation

Violating Constraints for Exploration:

- **Intuition**: To move from one feasible region to another disjoint feasible region, we may need to cross an “infeasible sea”.
- Strategic Oscillation (策略震荡):
  - Deliberately allow the search to enter the infeasible region (e.g., exceed Knapsack capacity, allowing subtours for TSP).
  - Apply a penalty to the objective function, but do not reject the solution immediately.
  - Oscillate the search boundary: Go deep into infeasibility → Repair back to feasibility.
- **Effect**: This allows the trajectory to “tunnel” through barriers that strict feasibility checks would block.

Trade-off between Exploration & Exploitation: MAB

How do we automatically tradeoff between Exploration & Exploitation?

Multi-Armed Bandit view:

- We have K operators to choose (or regions to search). We want to maximize reward (improvement).
- Upper Confidence Bound (UCB):
- : Average improvement from operator i (Exploitation).
- n_i: Number of times operator i was used.
- N: Total steps taken.
- The term $\sqrt{\ln N / n_i}$ represents uncertainty (Exploration).
- **Strategy**: Always pick the operator with the highest UCB score. If we haven't used an operator for a while, its uncertainty grows, forcing us to try it.

The Golden Rule

There is **no single "correct" ratio** of Exploration vs. Exploitation.

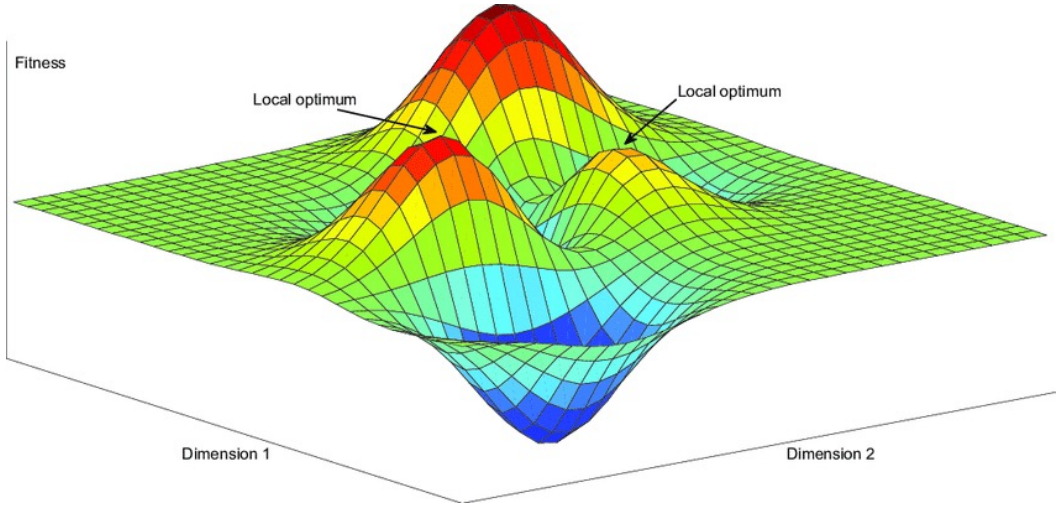
Dynamic Balance is Key:

- Start with high exploration.
- Transition to exploitation.
- Detect stagnation (using entropy or variance).
- Re-trigger exploration (restarts/large-moves).

Summary

Strategy	Mechanism	Role/Type
Restart	Reset solution randomly or apply large perturbation	Blind Exploration (Escaping stagnation)
Strategic Oscillation	Allow temporary constraint violation (infeasible → feasible)	Boundary Exploration (Tunneling)
Fitness Sharing	Penalize fitness of individuals in dense regions	Population Diversity (Niche)
Crowding	Offspring replaces the most similar parent	Genotypic Diversity (Structure)
Entropy Monitoring	Track population entropy; trigger mutation if $H < \epsilon$	Adaptive Control (Reactive)
UCB (Bandit)	Select operators (regions) based on performance + uncertainty	Operator/region Selection

Fitness Landscape Analysis (FLA)



Formal Definition: A triplet

- S : The set of all potential solutions (Solution/Search Space).
- N: The Neighborhood definition (N: $S \rightarrow 2^S$). This defines connectivity between solutions.
- f: The Fitness/objective function (f: $S \rightarrow \mathbb{R}$)

When to use: typically for **analyzing**, not for solving the problem

Crucial Distinction: Combinatorial vs. Continuous

Continuous Optimization:

- Neighborhood is **naturally defined by Euclidean distance** (ϵ-ball).
- Typically, we have gradients (∇f) and we know which direction is “up”.
- Landscape is continuous and differentiable.

Combinatorial Optimization:

- The space is discrete.
- There is no natural “distance” unless we define one.
- Connectivity depends entirely on the **move operator,** e.g., In TSP, Solution A and B might be “neighbors” under 3-opt, but far apart under 2-opt.
- **There is no gradient**. We must probe neighbors to find direction.

Intuition on Neighbors in Different Neighborhoods

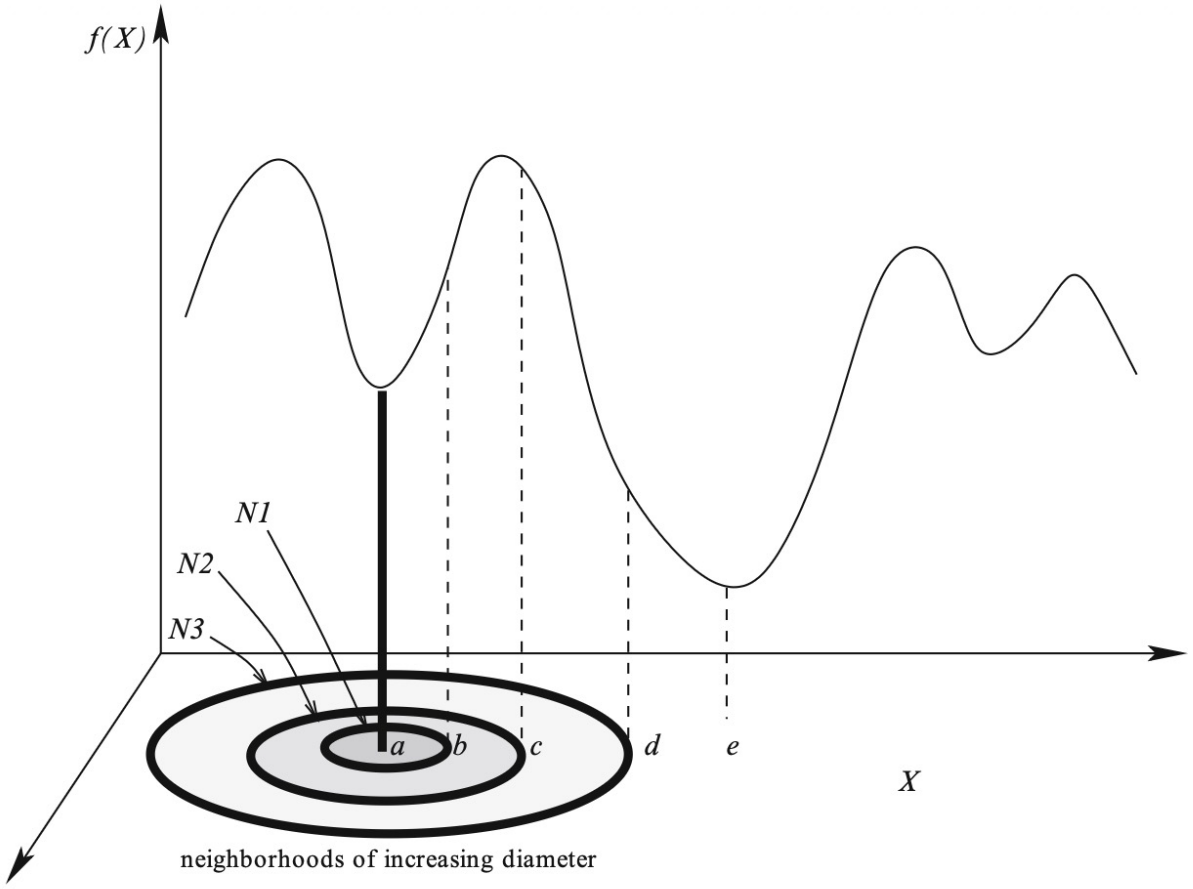


Fig. 2.5 Variable neighborhoods of different diameters. Neighboring points are on the circumferences at different distances. The figure is intended to help the intuition: The actual neighbors considered in the text are discrete

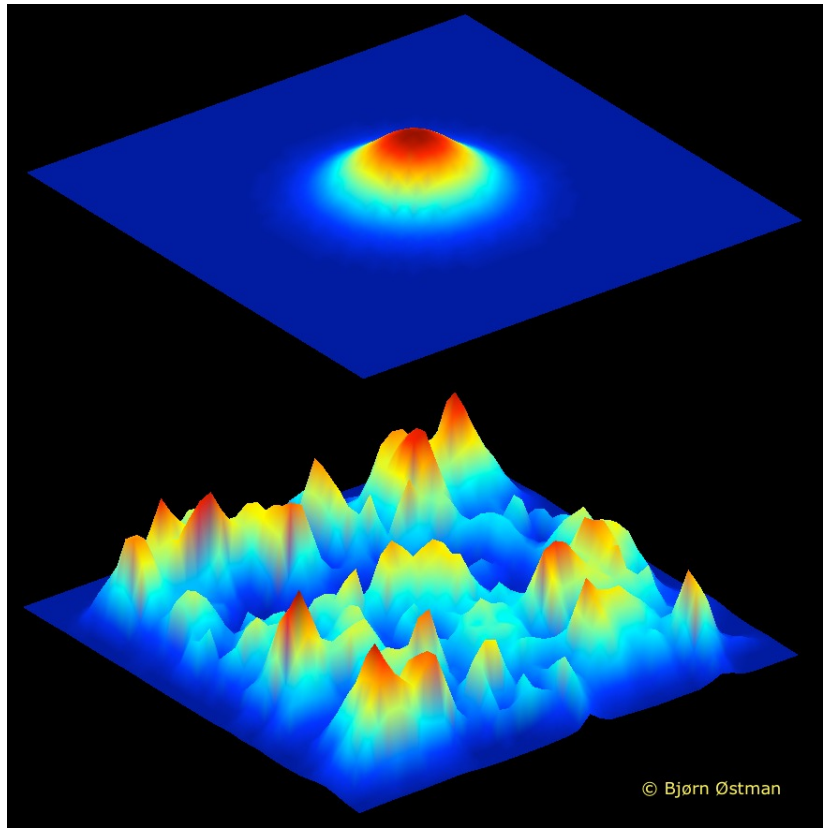
Landscape Properties: Ruggedness (崎岖度)

Smooth Landscape:

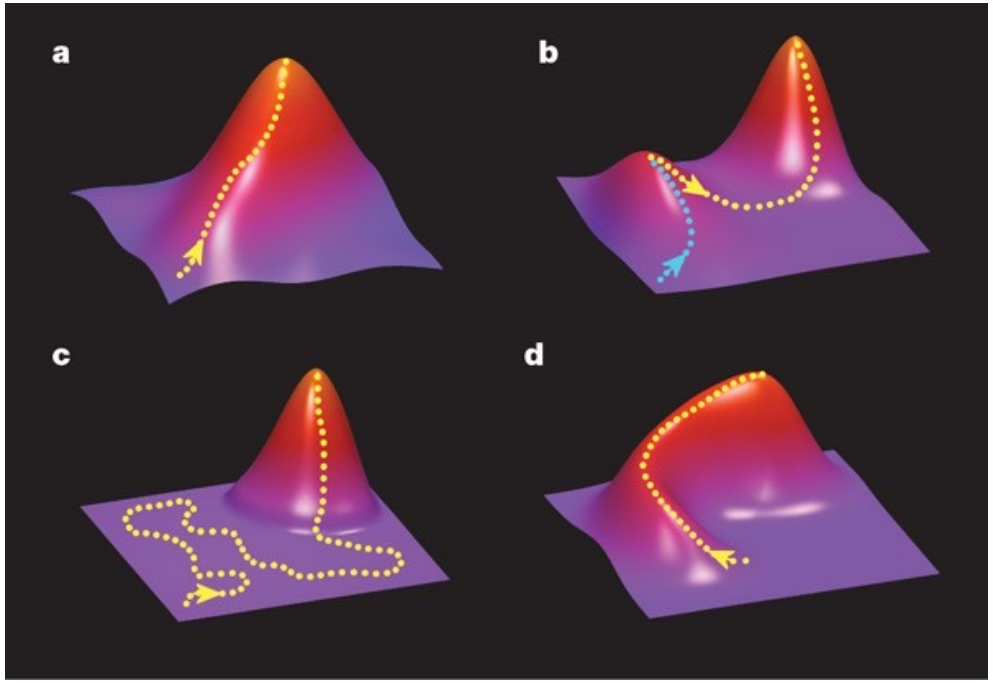
- Small changes in the solution → Small changes in fitness.
- Correlation between neighbors is high.
- Easy for Local Search.

Rugged Landscape:

- Small changes in solution → Massive, random jumps in fitness.
- Many Local Optima.
- Correlation is low.
- Hard for Local Search (looks like random noise).



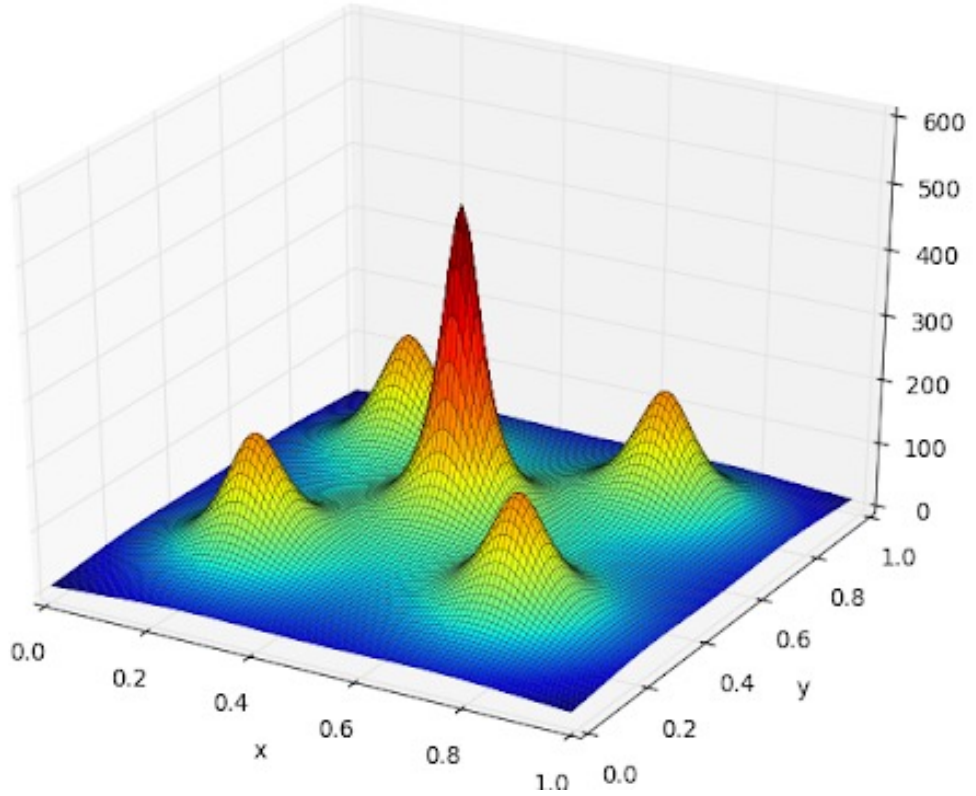
Landscape Properties: Neutrality (中立性)



Property 2: Neutrality

- **Plateaus**: Large areas where $f(S) = f(S'), S' \in N(S)$.
- **Effect**: Local search wanders blindly (Random Walk).
- Common in problems with discrete integer costs, like MAX-SAT.

Landscape Properties: Deception



Property 3: Deception

- **Deceptive Landscape**: The landscape structure leads the algorithm away from the global optimum.

Analyzing Ruggedness: Random Walks (1)

How do we measure ruggedness without checking every point?

Random Walk Analysis:

1) Start at random solution $\mathbf{x}_0$ $(t = 0)$ with fitness value f_0.

2) Randomly pick neighbor $\mathbf{x}_{t+1} \in N(\mathbf{x}_t)$. Record fitness sequence $f_0, f_1, \ldots, f_m$.

3) Calculate autocorrelation coefficient $\rho(k)$ at step size k.

- One possible definition of $\rho(k)$:

$$\rho(k) = \frac{\sum_{t=1}^{m-k} (f_t - \bar{f})(f_{t+k} - \bar{f})}{\sum_{t=1}^{m} (f_t - \bar{f})^2}$$

- $\bar{f}$: The mean fitness of the random walk sequence.
- The denominator is essentially the variance $\sigma^2 \times m$.

Analyzing Ruggedness: Random Walks (2)

4) **Interpretation**:

- $\rho(1) \approx 1$: Very Smooth (Neighbor fitness is highly predictive of current fitness)
- $\rho(1) \approx 0$: Very Rugged /Random (Neighbor fitness unrelated)

5) **Correlation Length (L):** How many steps until the correlation drops to nearly zero?

- Small $L \rightarrow$ Very rugged (Hard).

Fitness Distance Correlation (1)

Does “getting closer to the optimum” mean “getting better fitness”?

Fitness Distance Correlation (FDC) Method:

1) Assume we know (or approximate) the global optimal $\mathbf{x}_{opt}$.

2) Sample m solutions (typically local optima): $\mathbf{x}_1, \dots, \mathbf{x}_m$.

3) For each solution, measure Fitness $f(\mathbf{x}_i)$ and distance to optimum $d(\mathbf{x}_i, \mathbf{x}_{opt})$.

4) Calculate the **Pearson Correlation** coefficient r between $f(\mathbf{x}_i)$ and $d(\mathbf{x}_i, \mathbf{x}_{opt})$

$$r = \frac{Cov(f,d)}{\sigma_f \sigma_d} = \frac{\sum_{i=1}^{n}(f_i - \bar{f})(d_i - \bar{d})}{\sqrt{\sum_{i=1}^{n}(f_i - \bar{f})^2}\sqrt{\sum_{i=1}^{n}(d_i - \bar{d})^2}}$$

- Note: Pearson Correlation is often used to measure the **linear relationship** between two paired data.

Fitness Distance Correlation (2)

5) Interpretation (Minimization):

- $r \approx 1$ (**Positive**): Good! Closer to optimum = Lower Cost. (Big Valley structure).
- $r \approx 0$: No linear correlation.
- $r < 0$ (**Negative**): Deceptive! Better fitness leads away from the optimum.

Summary

Landscape Property	Analysis tool	Diagnosis	Impact on Search Strategy
Ruggedness	Random Walks	Low: Uncorrelated, “noisy” peaks.	Local Search gets stuck immediately. Requires strong diversification (Restarts, Large Neighborhoods).
Neutrality	Proportion of Neutral Neighbors	High Neutrality: Large plateaus (flat areas).	Require allow drift on plateaus
Global Structure (Deceptiveness)	Fitness Distance Correlation (FDC)	Negative FDC: the landscape leads search away from the global optimum	Usually needs random jumps or population diversity to ignore the misleading path.

No Free Lunch (NFL) Theorem

The NFL Theorem (Wolpert & Macready, 1997)

- **Statement**: When averaged over **all possible** optimization problems, all search algorithms perform **exactly the same**.
- In other words: A complex Evolutionary Algorithm performs no better than **random search** in the general case.

Key Insights

- No "Silver Bullet": There is no single universally "best" algorithm.
- **Conservation of Performance**: If Algorithm *A* outperforms Algorithm *B* on a specific set of problems (e.g., TSP), it must perform worse on the remaining set of problems.
- **The Role of Bias**: An algorithm's success depends on how well its inductive bias (assumptions) matches the specific structure of the problem.

Understanding NFL from FLA

The "White Noise" Scenario: The set of "all possible problems" is dominated by functions with no structure (random fitness landscapes).

Why NFL Holds: In a random landscape ($\rho(1) = 0$), the fitness of a known point gives zero information about the fitness of its neighbors.

- **Consequence:** Heuristics (exploitation) become mathematically impossible; "intelligent" moves become random guesses.

The Value of FLA: FLA helps us verify that real-world problems have **structure** (non-random correlations), justifying why we can ignore the NFL theorem in practice and design advanced algorithms.

Lecture 7-1: Advanced Topics: Variants of Combinatorial Optimization

Moving Beyond the Standard CO Model

Standard **CO problems** (like TSP, Knapsack) assume:

- **Deterministic**: All problem parameters (c_{ij}, w_i, v_i) is known exactly.
- **Static**: Data does not change over time.
- **Single Objective**: We only care about one metric (e.g., cost).

Real-world Scenarios:

- Travel times fluctuate (Traffic).
- Demands are predictions, not facts.
- We want to minimize **Cost AND Risk AND Duration**.

We need **Advanced Variants**

Variant 1: Stochastic CO

Stochastic CO: Uncertainty

- Some parameters (e.g., edge weights c_{ij}) are random variables with known probability distributions ξ.
- We cannot maximize $f(x, \xi)$ directly because ξ is random (thus f is random).

Typical goal: Maximize the **Expected Value:**

$$\max_{x \in X} E_{\xi}[f(x, \xi)]$$

Example: Stochastic VRP

- Demands d_i are random.
- We plan a route. If actual demand > truck capacity, we fail (recourse cost).
- Objective: Min (Routing Cost + Expected Failure Penalty).

Stochastic CO: Solution Ideas

We usually cannot compute the expected value E_ξ exactly.

Sample Average Approximation (SAA)

- Generate N random scenarios (samples) $\xi_1, \ldots, \xi_N$
- Solve the deterministic average problem:

$$\max_{\mathbf{x}} \frac{1}{N} \sum_{k=1}^{N} f(\mathbf{x}, \xi_k)$$

- As $N \to \infty$, this converges to the true stochastic solution.

Variant 2: Robust Optimization

Robust CO: Uncertainty

- We do NOT know the probability distribution (or we don't trust it).
- We only know the Uncertainty Set U, e.g., travel time $t_{ij} \in [10,20]$. It could be anything in this interval.

Goal: Worst-case Optimization (Min-Max)

- Find a solution that performs best in the worst possible scenario.

$$\min_{\mathbf{x}} \left(\max_{u \in U} f\ (\mathbf{x}, u) \right)$$

- **Intuitive Understanding**: I want a guarantee. My plan must work even if everything goes wrong.

Robust CO: Solution Ideas

Challenge: the problem is a bi-level optimization problem.

Approach: Adversarial Constraint Generation (The "Iterative" Way)

- Idea: We cannot list all possible scenarios $u \in U$. We generate them "on the fly".
- Step 1 (Master): Solve the master problem for **a small set of** scenarios. Get candidate solution $\mathbf{x}^*$ and get its objective value θ^*. Since we only consider a subset of scenarios, θ^* is a lower bound of the true optimum: $\theta^* \leq \text{OPT}$.
- Step 2 (Adversary): Find the true worst-case u^* for $\mathbf{x}^*$in the full set of scenarios U, and calculate its actual objective value θ^*_{true}. $\mathbf{x}^*$ is a valid solution, then we must have $\theta^*_{true} \geq OPT$.
- Step 3 (check): If $\theta^*_{true} \leq \theta^*$: **Converged**, return $\mathbf{x}^*$ and θ^*_{true}. Proof: because we have $\theta^*_{true} \geq OPT$ and $\theta^* \leq \text{OPT}$ and $\theta^*_{true} \leq \theta^*$, hence $\theta^*_{true} = OPT = \theta^*$.
  - else: add a new constraint to the master problem and repeat.

Variant 3: Multi-Objective CO (MOCO)

Definition of MOCO:

- We have k **conflicting** objectives: $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})$
- There is usually no single optimal solution (minimal weight with maximal value).

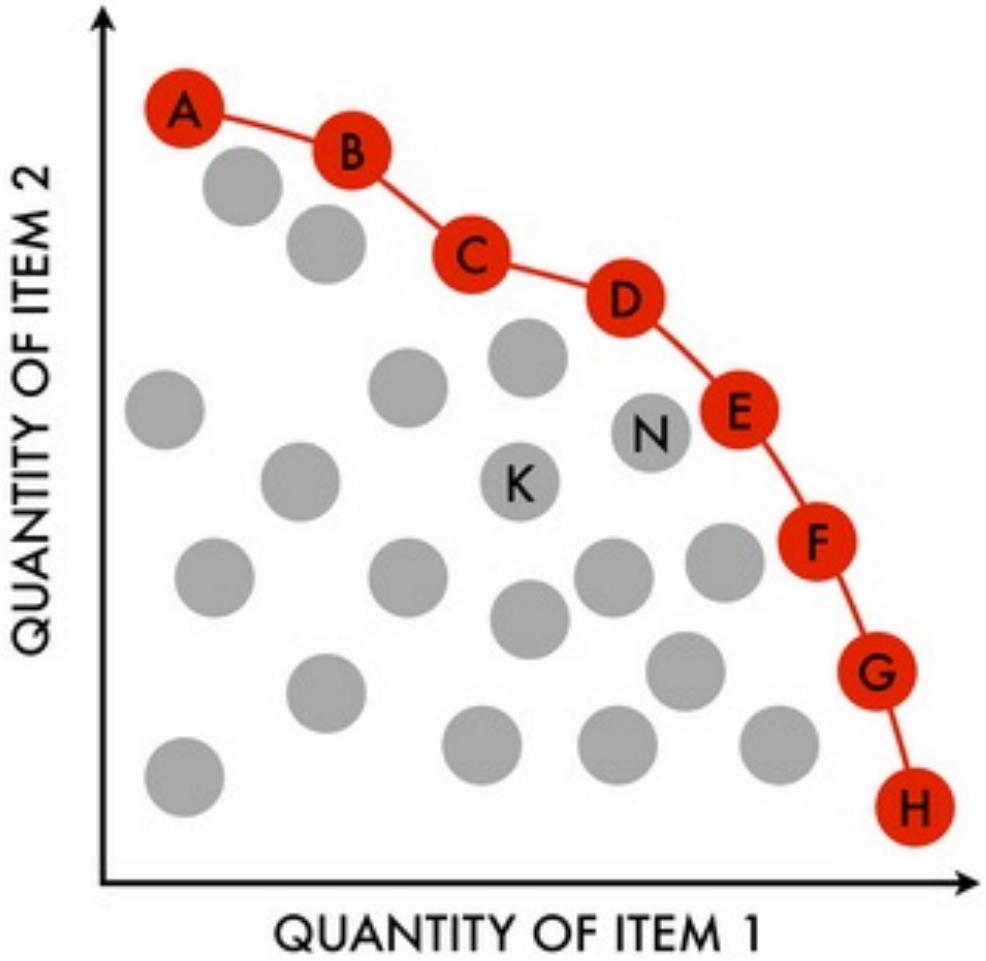
Dominance Relation: Solution A dominates Solution B if:

- A is no worse than B in all objectives.
- A is strictly better than B in at least one objective.

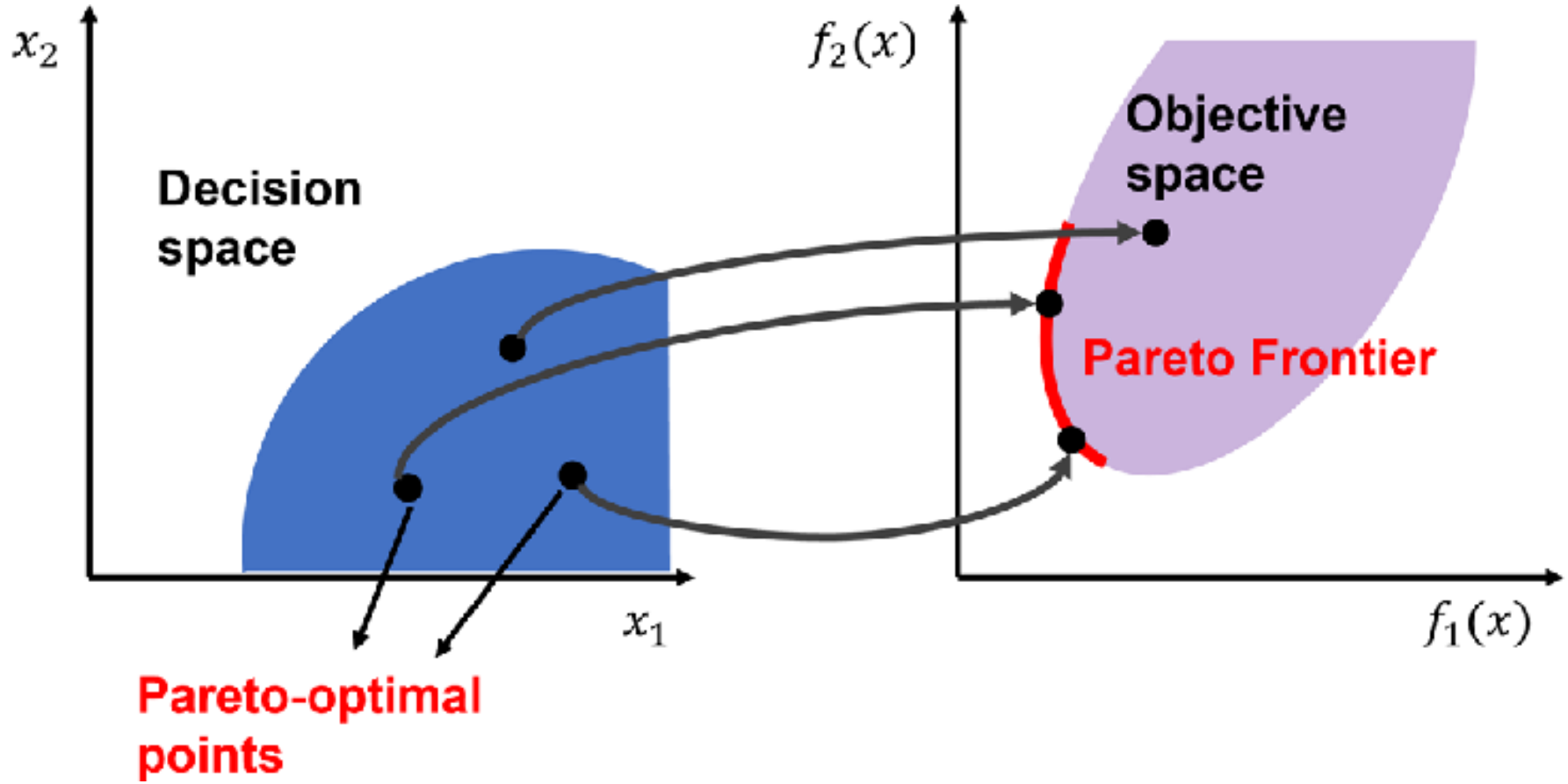
A solution is called **Pareto-optimal** if it is a non-dominated solution, i.e., not dominated by any other solution.

Goal: find the Pareto Front (The set of all non-dominated solutions)

Visualization of Pareto Front



Decision (Solution) Space vs. Objective Space



MOCO: Solution Ideas (1)

Weighted Sum Approach: Convert the multi-objective problem into a single-objective problem by assigning a weight w_i to each objective:

$$w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \cdots + w_k f_k(\mathbf{x})$$

Weights: represent the relative importance

Process: Solve this new single-objective problem using standard algorithms

By varying weights, we hope to find different points on the Pareto Front.

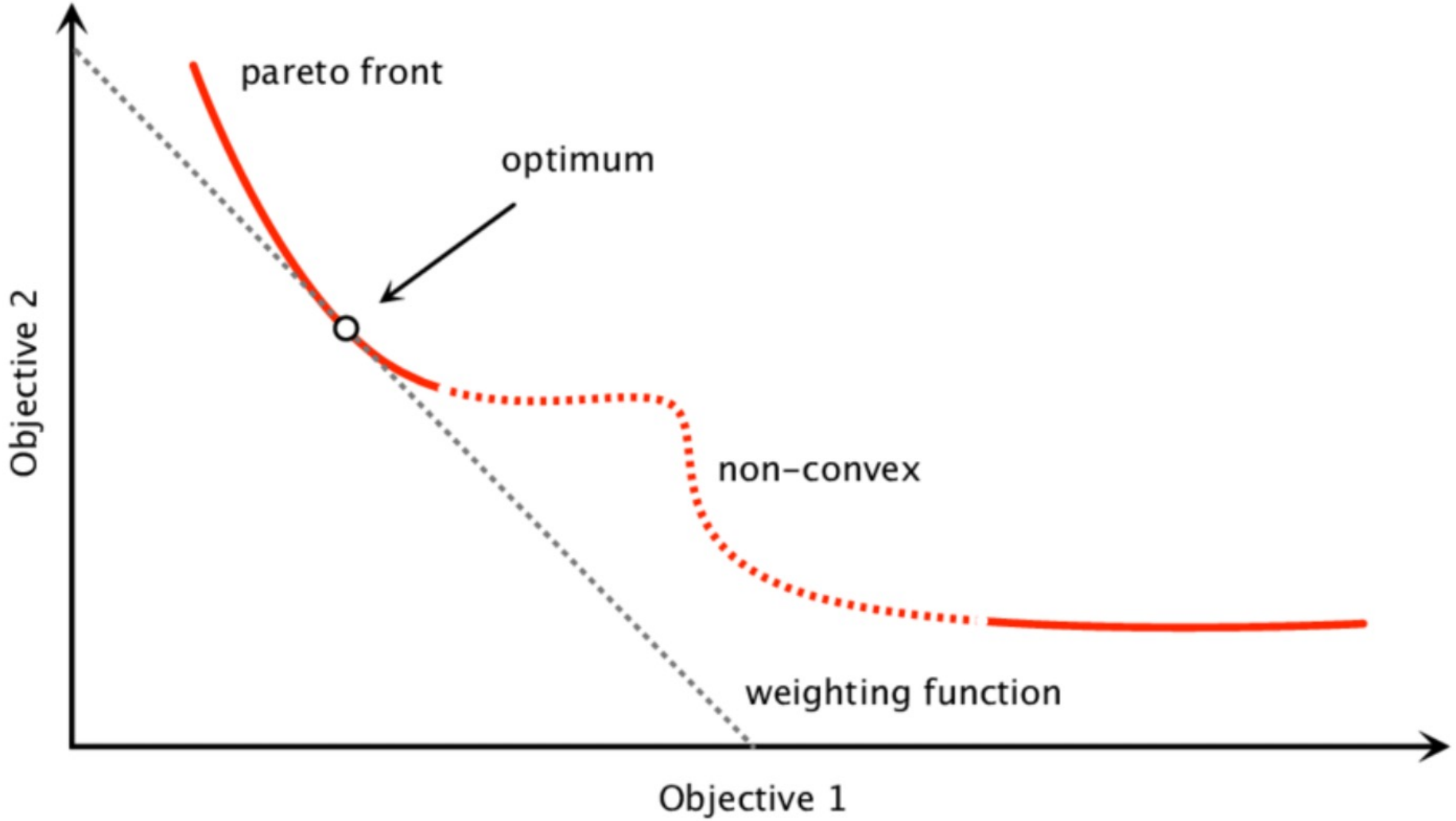
Geometric Interpretation of Weighted Sum

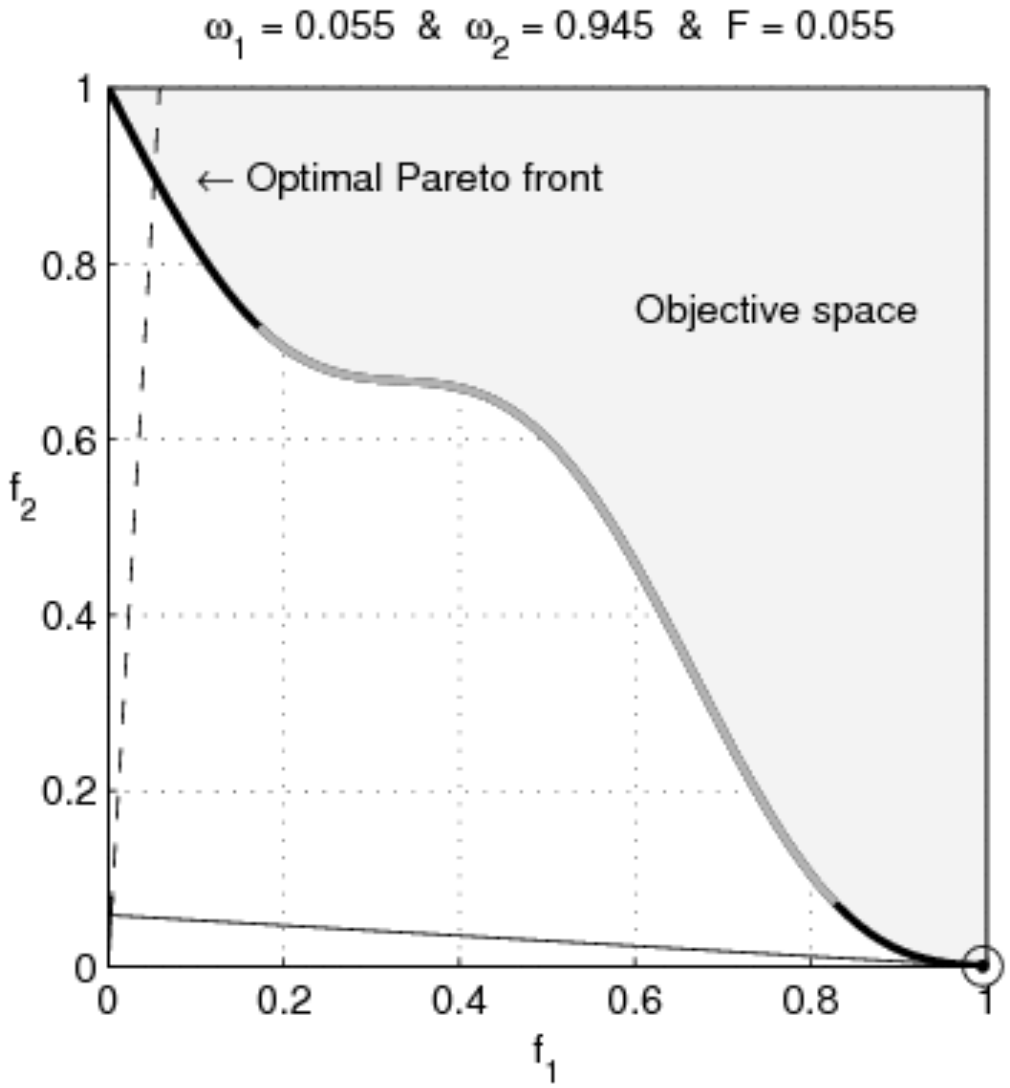
In the objective space (f_1 vs f_2), the equation $w_1 \cdot f_1 + w_2 \cdot f_2 = C$ defines a line (or hyperplane) with slope $-w_1/w_2$.

Minimizing the weighted sum is equivalent to pushing this line towards the origin from infinity.

The optimal solution is the **last contact point** where this line touches the feasible region.

Visualization of Weighted Sum




$\omega_1 = 0.055$ & $\omega_2 = 0.945$ & F = 0.055
← Optimal Pareto front
Objective space
1
0.8
0.6
0.4
0.2
0
f_2
0
0.2
0.4
0.6
0.8
1
f_1

The Limitation of Weighted Sum

Critique: Does setting $w_1 = 0.5, w_2 = 0.5$ guarantee a balanced solution? No.

The Non-Convexity Issue (crucial for CO):

- In CO, the shape of the Pareto Front is often non-convex (concave).
- The "Unsupported" Solutions:
  - Imagine a Pareto Front with a "cave" or indentation.
  - The weighted sum line will touch the "outer" points but can NEVER touch the points inside the concave region, no matter what weights you choose.
  - **Weighted Sum misses some parts of the Pareto Front!**

Sensitivity:

- Small changes in weights can cause the optimal solution to jump drastically from one end of the Pareto Front to the other.

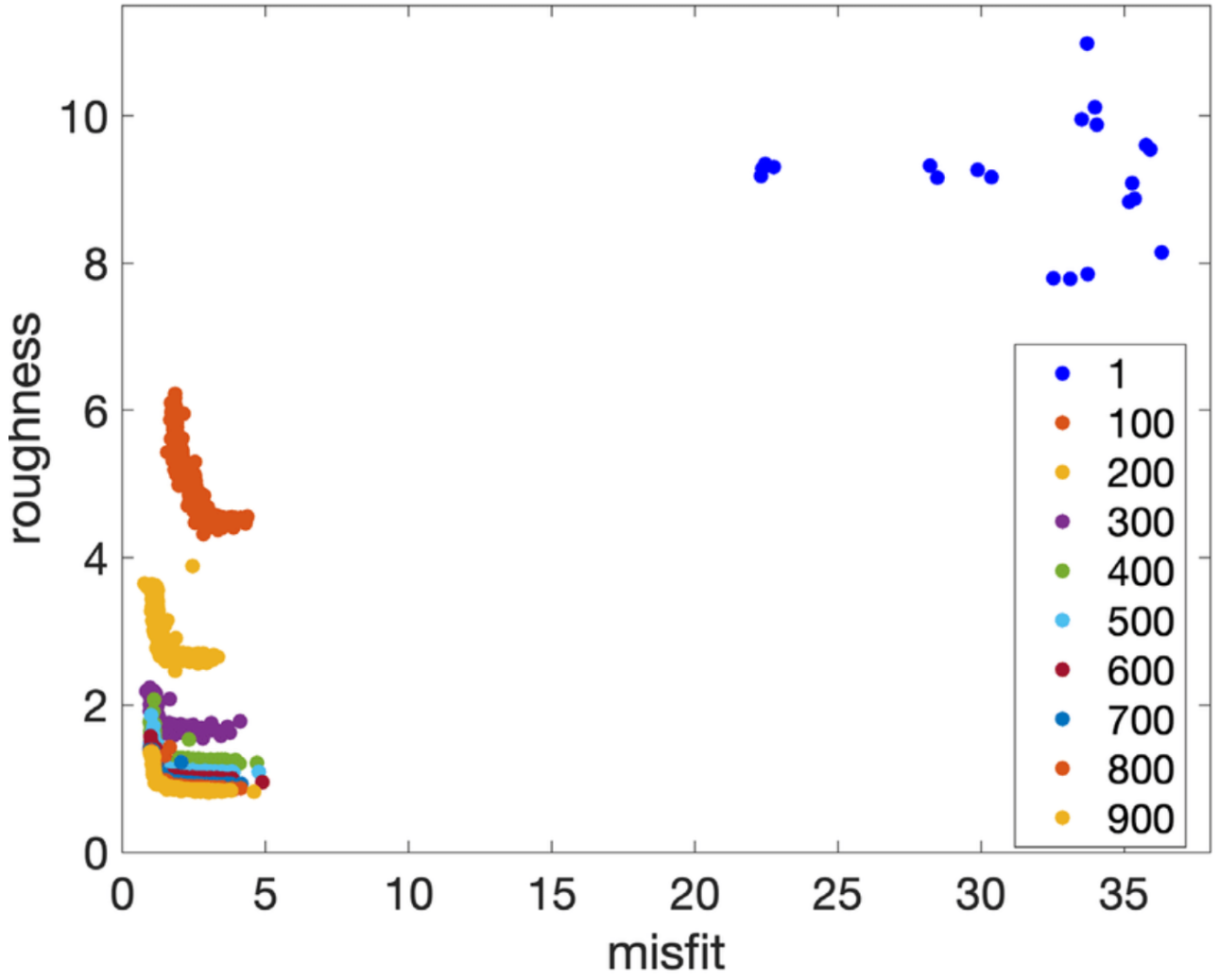
MOCO: Solution Ideas (2)

Evolutionary Algorithms (e.g., NSGA-II): Maintain a population of solutions that approximates the entire Pareto Front simultaneously.

Core Mechanisms in the **Survival Selection in** NSGA-II

- Non-Dominated Sorting:
  - Instead of fitness value, rank individuals by "**Layers of Dominance**".
    - Rank 1: Solutions not dominated by anyone (Current Pareto Front).
    - Rank 2: Solutions dominated **only** by Rank 1.
- Crowding Distance (Diversity):
  - If two solutions have the same rank, prefer the more isolated one (far from others).
  - This forces the population to spread out evenly across the entire front, revealing the true trade-offs.

Visualization of NSGA-II (different generations)



Summary

CO Variant	Nature of Difficulty	Optimization Goal	Some Solution Methods
Stochastic CO	Parameters are Random Variables	Maximize Expected Value	Sample Average Approximation
Robust CO	Parameters belong to an Uncertainty Set. No probabilities known	Min-Max (Worst-Case)	Constraint Generation
Multi-objective CO	Multiple Conflicting Objectives (e.g., Cost vs. Time). No single optimal solution.	Find the Pareto Front.	Weighted Sum, Evolutionary Algorithms