Lecture 8
Multi-objective Optimization

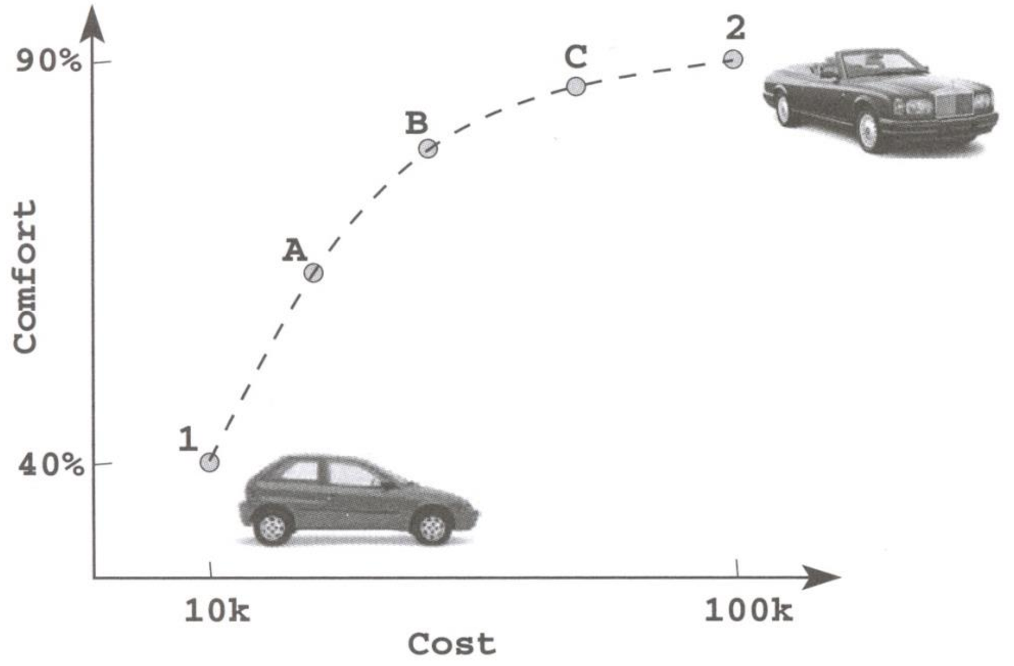
Algorithm

张子臻，中山大学计算机学院

zhangzizhen@gmail.com

Introduction

- Many real-world problems involve multiple objectives.
- These objectives are generally conflicting with each other.
  - A solution that is extreme with respect to one objective requires a compromise in other objectives.
  - A sacrifice in one objective is related to the gain in other objective(s).
- Illustrative example: Buying a car
  - Two extreme hypothetical cars 1 and 2.
  - Cars with a trade-off between cost and comfort: A, B, and C.
- Which solution out of all of the trade-off solutions is the best with respect to all objectives?
  - Without any further information those trade-offs are indistinguishable.
  - Thus, a number of optimal solutions is sought in multiobjective optimization.



Multi-objective Optimization

- A multi-objective optimization problem (MOOP) with *M* objectives is defined as follows: Given an n-dimensional decision variable vector x = {x_1,…, x_n} in the solution space X, find a vector x that minimizes/maximizes a given set of M objective functions: f(x) = {f_1(x), … , f_M(x)}.
- The solution space X is generally restricted by a series of constraints.
- MOOP in the general form:

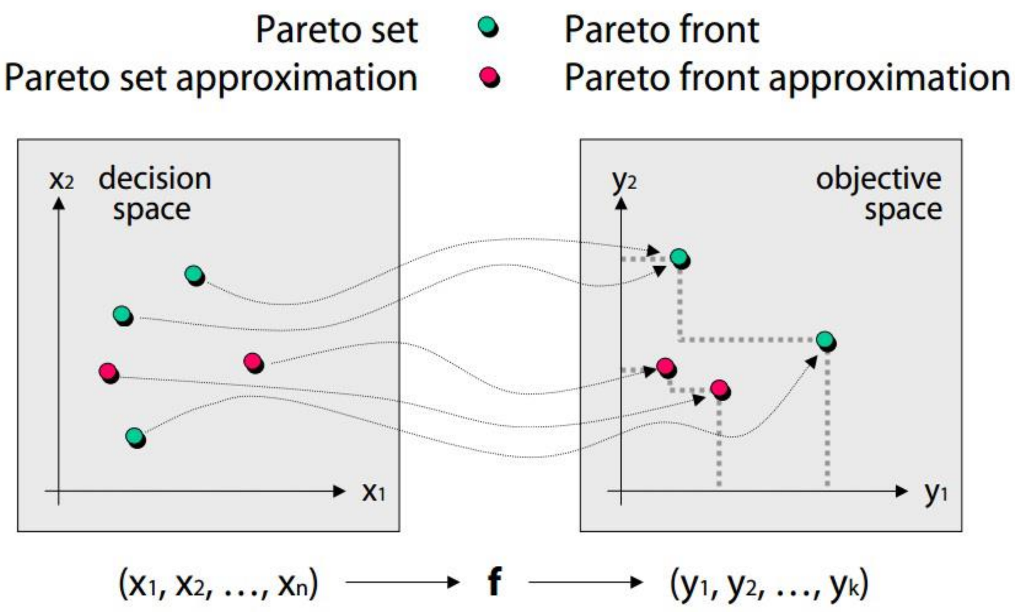
$$
\begin{aligned}
&\min f_m(x) && m = 1, 2, \ldots, M \\
&s.t && \\
&g_j(x) \geq 0 && j = 1, 2, \ldots, J \\
&h_k(x) = 0 && k = 1, 2, \ldots, K \\
&x_i^L \leq x_i \leq x_i^U && i = 1, 2, \ldots, n
\end{aligned}
$$

Definition

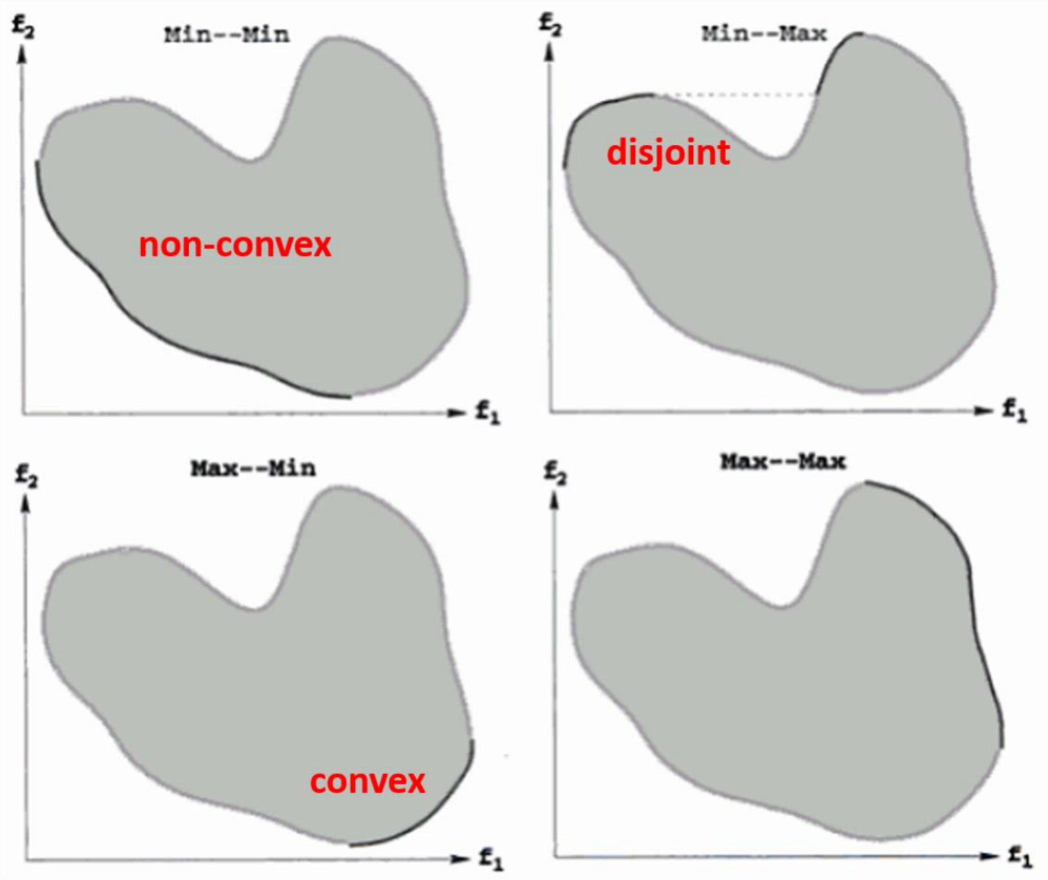
- (Domination) A feasible solution x is said to (weak) dominate another feasible solution y (or $x \preccurlyeq y$), if both the following condition hold:
  1. $f_i(x) \leq f_i(y), \forall\ i = 1, \ldots, M$.
  2. $f_i(x) < f_i(y), \exists\ i = 1, \ldots, M$.
- (Non-dominated set) Among a set of feasible solutions P, the non-dominated set of solutions P' are those solutions that are not dominated by any member of set P.
- (Pareto optimal set) The non-dominated set of the entire feasible search space S is called (globally) Pareto optimal set.
- (Pareto front) For a given Pareto optimal set, the corresponding objective function values in the objective space are called the Pareto front.

Pareto Optimality

- Pareto set & Pareto front

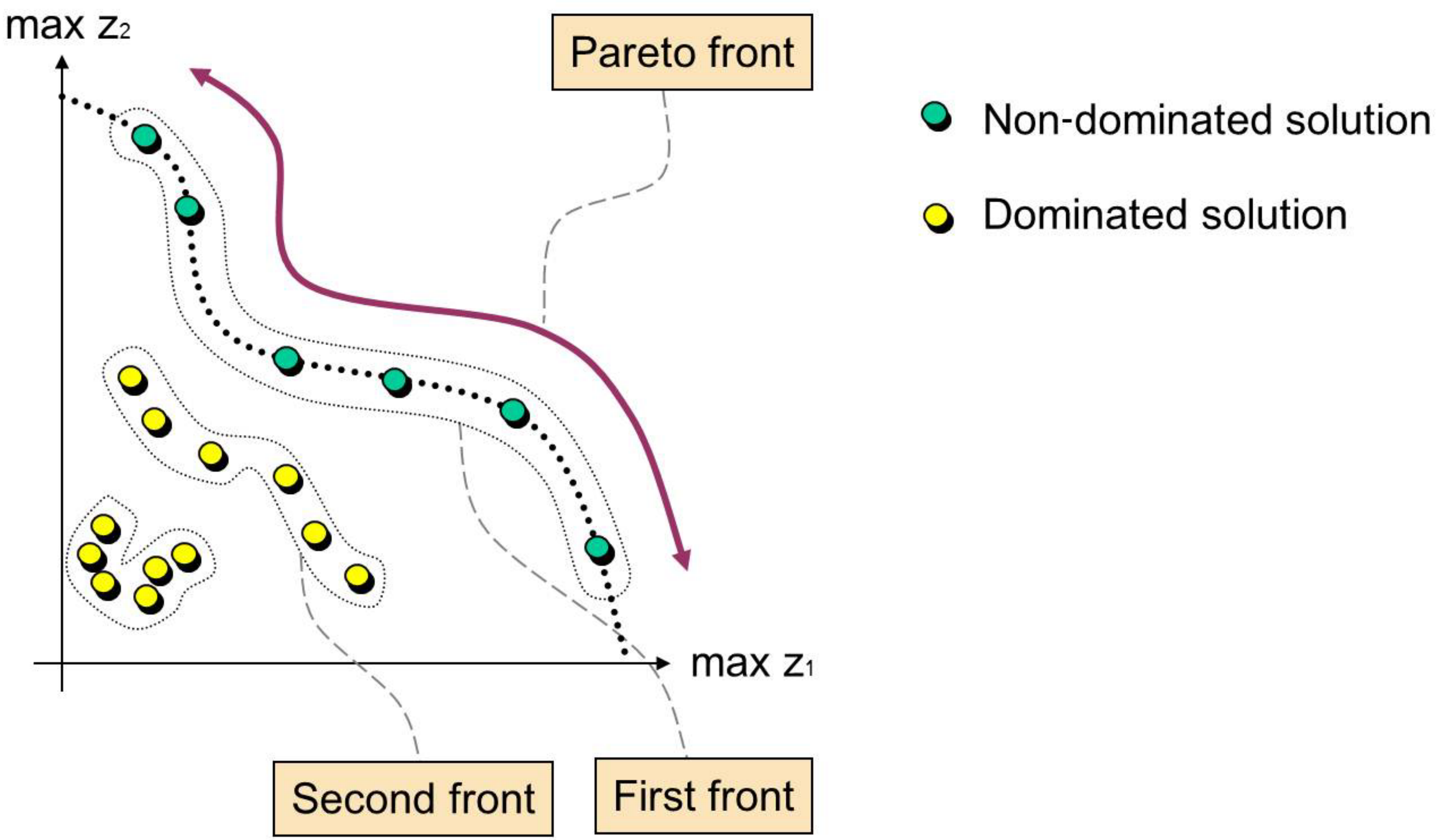


- Shapes of the Pareto front



Pareto Optimality

- Fronts can be classified into different levels.



Non-dominated Sorting

- The non-dominated sorting is to classified solutions into different levels.
- It is essentially a topological sorting process.

An $O(MN^2)$ non-dominated sorting algorithm

```
INPUT: Feasible solution set P;
for each p ∈ P do
    S_p = ∅;
    n_p = 0;
    for each q ∈ P do
        if p ⪯ q then
            S_p = S_p ∪ {q};
        else if q ⪯ p then
            n_p = n_p + 1;
        end if
    end for
    if n_p = 0 then
        p_rank = 1;
        F_1 = F_1 ∪ {p};
    end if
end for
i = 1;
while F_i ≠ ∅ do
    Q = ∅;
    for each p ∈ F_i do
        for each q ∈ S_p do
            n_q = n_q − 1;
            if n_q = 0 then
                q_rank = i + 1;
                Q = Q ∪ {q};
            end if
        end for
    end for
    i = i + 1;
    F_i = Q;
end while
```

Classical Method: Weighted Sum Method

- The Weighted sum method scalarizes a set of objectives into a single objective by pre-multiplying each objective with a user-defined weight.

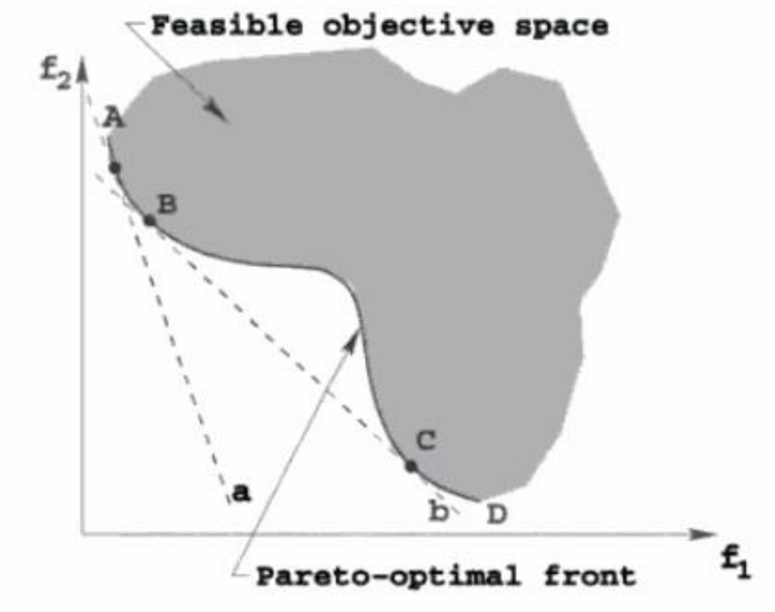
$$\min F(x) = \sum_{m=1}^{M} w_m f_m(x)$$

$$\begin{aligned} \textit{subject to} \quad & g_j(x) \geq 0 & j = 1, 2, \ldots, J \\ & h_k(x) = 0 & k = 1, 2, \ldots, K \\ & x_i^L \leq x_i \leq x_i^U & i = 1, 2, \ldots, n \end{aligned}$$

- **Q**: How to set up an appropriate weight vector w?
- **A**: By relative importance of objectives in the problem.
- Different objectives take different orders of magnitude. Sometimes we need a process to normalize each objective.

Weighted Sum Method

- Theorem: If the weight is positive for all objectives, the solution to the problem represented by weighted sum model is Pareto-optimal.
- If *x* is a Pareto-optimal solution of a convex multi-objective optimization problem, then there exists a non-zero positive weight vector *w* such that *x* is a solution to the problem give by weighted sum model.



Classical Method: Weighted Metric Method

$$\min l_p(x) = \Big(\sum_{m=1}^{M} w_m |f_m(x) - z_m^*|^p\Big)^{1/p}$$

$$\begin{aligned} \textit{subject to} \quad & g_j(x) \geq 0 & j = 1, 2, \ldots, J \\ & h_k(x) = 0 & k = 1, 2, \ldots, K \\ & x_i^L \leq x_i \leq x_i^U & i = 1, 2, \ldots, n \end{aligned}$$

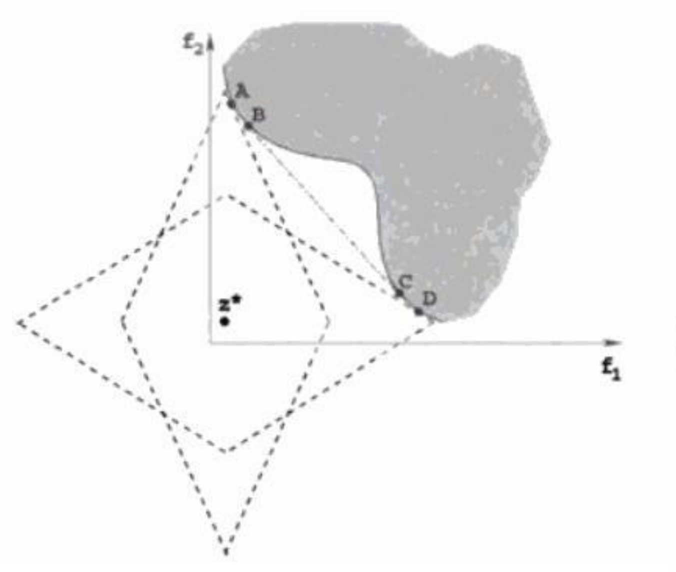
- When p = 1 is used, the resulting problem is equivalent to the weighted sum approach.
- When p = 2 is used, a weighted Euclidean distance of any point in the objective space from the *ideal* (or *utopian*) point z* is minimized.
- When p = ∞, the problem is called weighted Tchebyche problem.

$$\min l_\infty(x) = \max_{m=1,\ldots,M} w_m |f_m(x) - z_m^*|$$

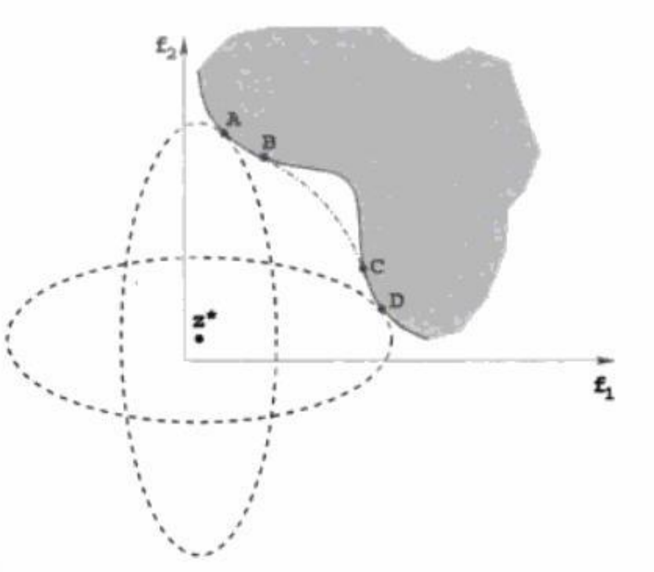
$$\begin{aligned} \textit{subject to} \quad & g_j(x) \geq 0 & j = 1, 2, \ldots, J \\ & h_k(x) = 0 & k = 1, 2, \ldots, K \\ & x_i^L \leq x_i \leq x_i^U & i = 1, 2, \ldots, n \end{aligned}$$

Weighted Metric Method

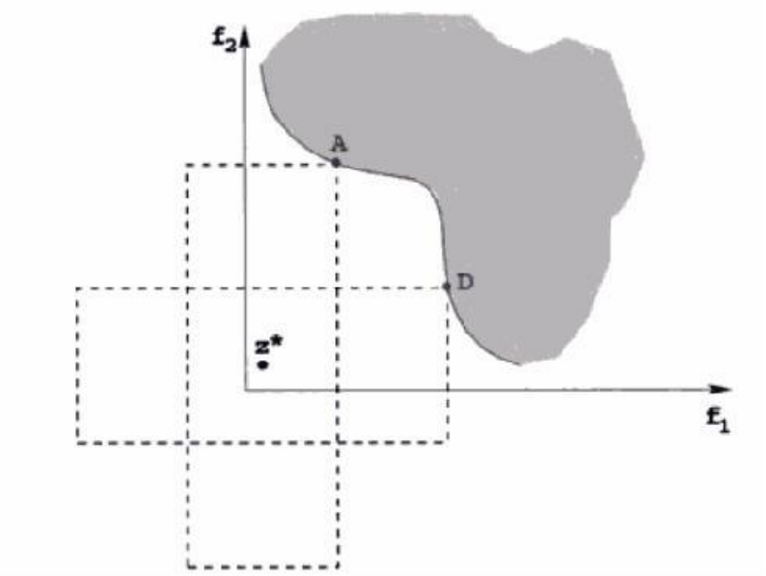
- Let x* be a Pareto-optimal solution. Then there exists a positive weighting vector such that x is a solution of the weighted Tchebyche problem, where the reference point is the ideal (*utopian*) objective vector.



The weighted metric method with $p = 1$.



The weighted metric method with $p = 2$.

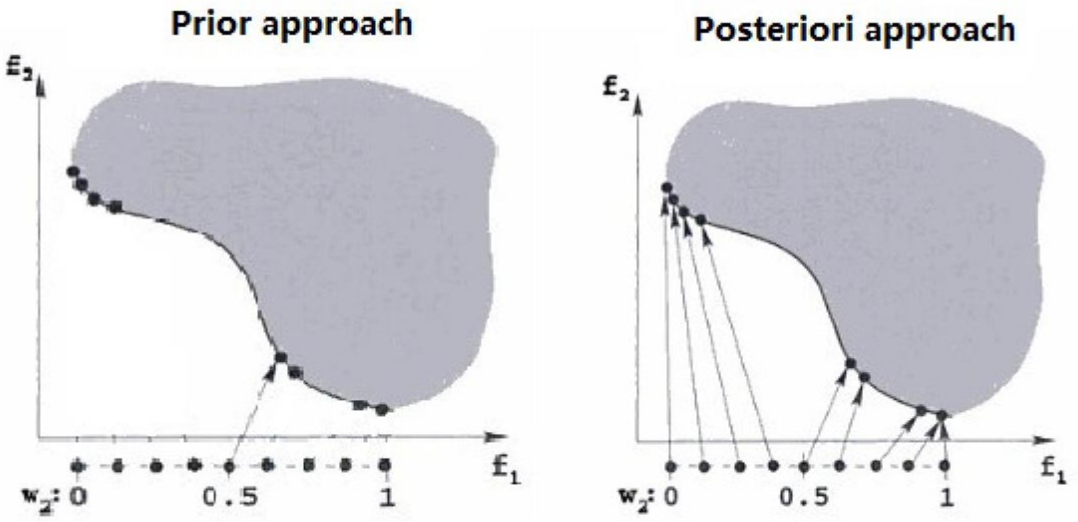


The weighted metric method with $p = \infty$.

Motivation for Finding Multiple Pareto-optimal Solutions

- **Q**: What should we do with multiple Pareto-optimal solution?
- **A**: From a practical point of view, we only need one solution to implement.
- **Q**: Which one of these multiple solutions are we interested in?
- **A**: It depends on the trade-off relationship among objectives.

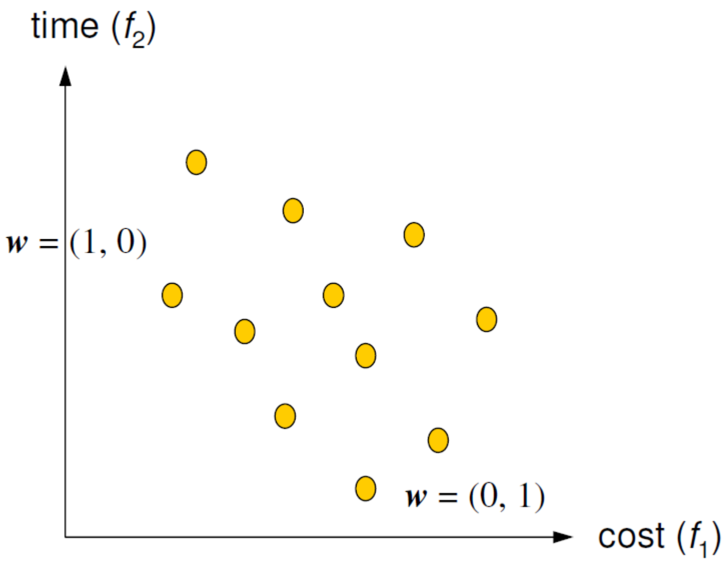
- A priori approach v.s. Posterior approach



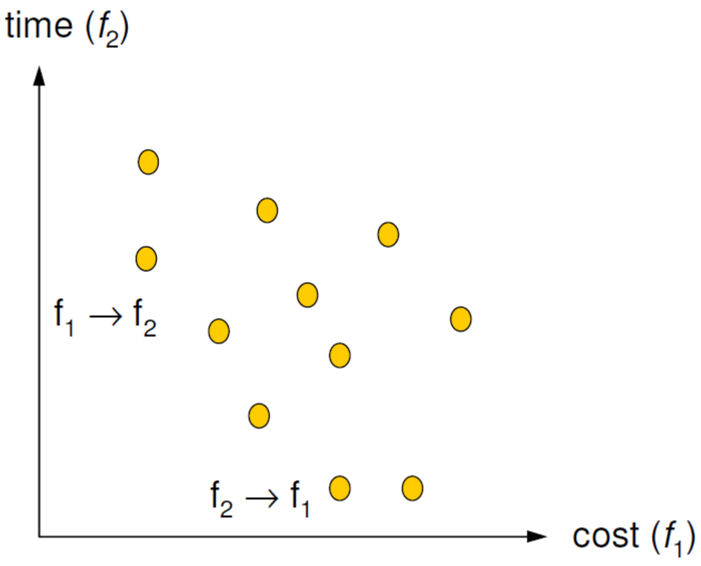
A Priori Method

- A priori method - Aggregation

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{k} w_i f_i(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in S, \end{array}$$

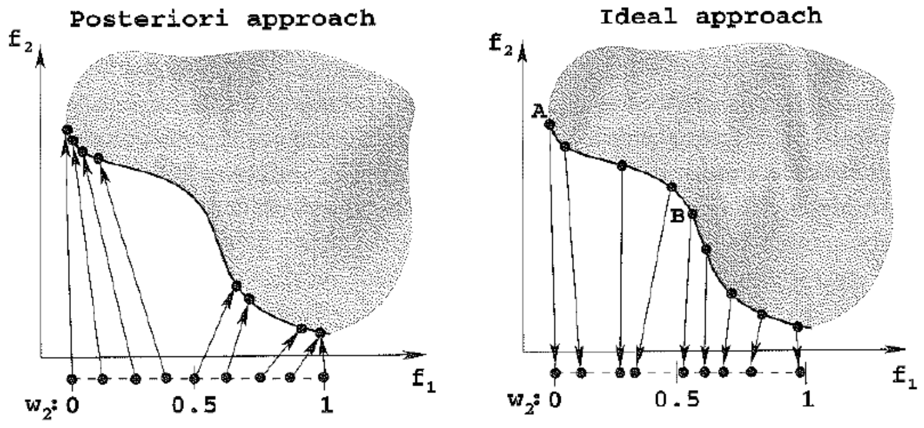


- A priori method - Lexicographical ordering
  - Optimize f_1 primarily and f_2 secondarily
  - Optimize f_2 primarily and f_1 secondarily



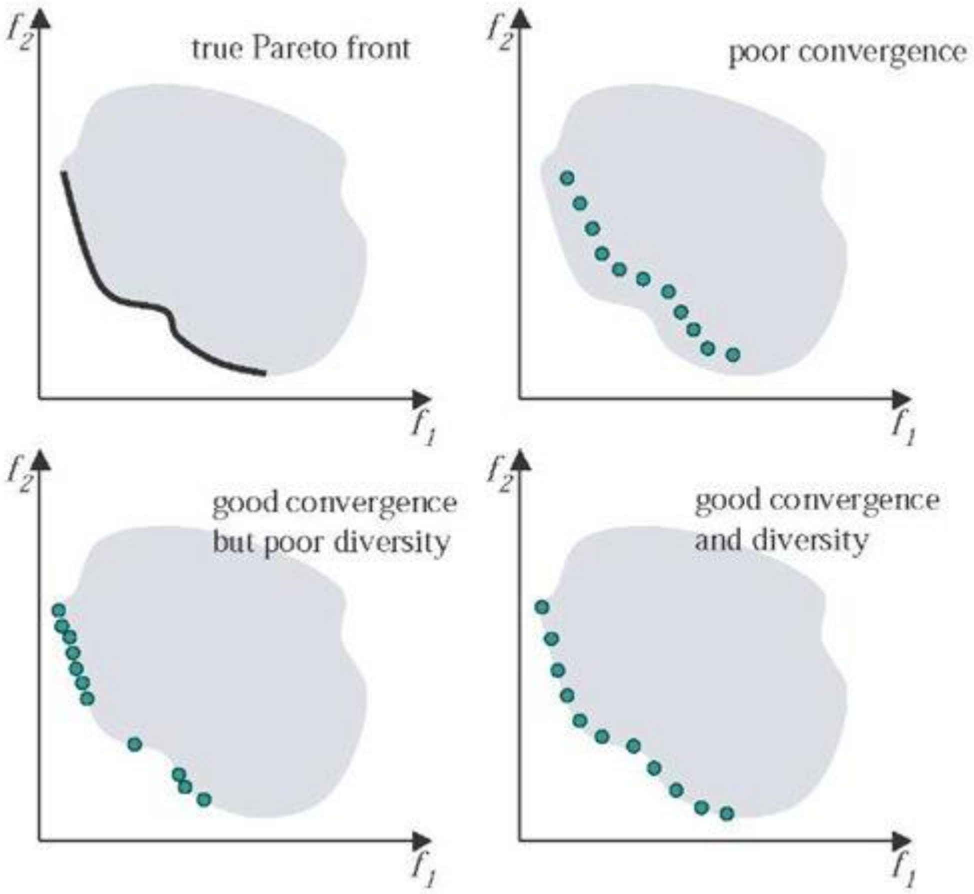
Posteriori Method

- Disadvantages of a priori methods
  - It is difficult to give the preference information in advance.
  - The decision maker needs to adjust the preference to obtain alternative solutions.
  - The condition becomes harder when there are multiple decision makers.
- Posteriori method
  - Find a set of Pareto optimal solutions first.
  - Choose one solution from the set by using other high-level information.
- Classical posteriori approach and the ideal approach



Goals of Multiobjective Optimization

- Goals of MOP are also multiobjective.
  - (Quality, Convergence, Proximity) To find a set of solutions as close as possible to the Pareto-optimal set.
  - (Diversity, Spread) To find a set of solutions as diverse as possible.

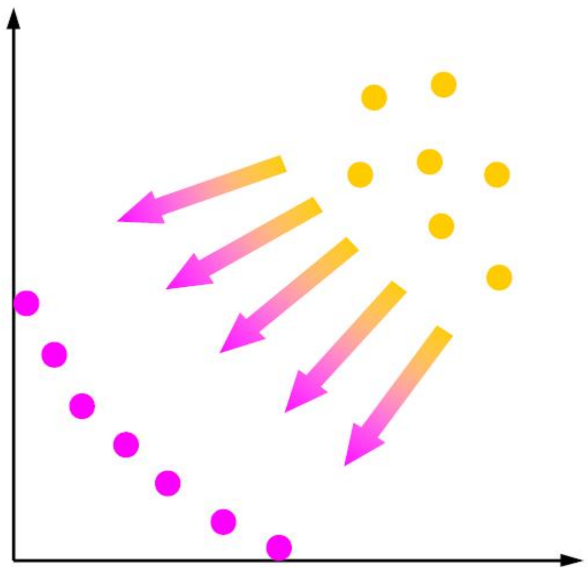


Multiobjective Optimization using Evolutionary Algorithm

- Evolutionary algorithm is a good option in achieving these goals.
- In general, an evolutionary algorithm is a population-based meta-heuristic optimization algorithm.

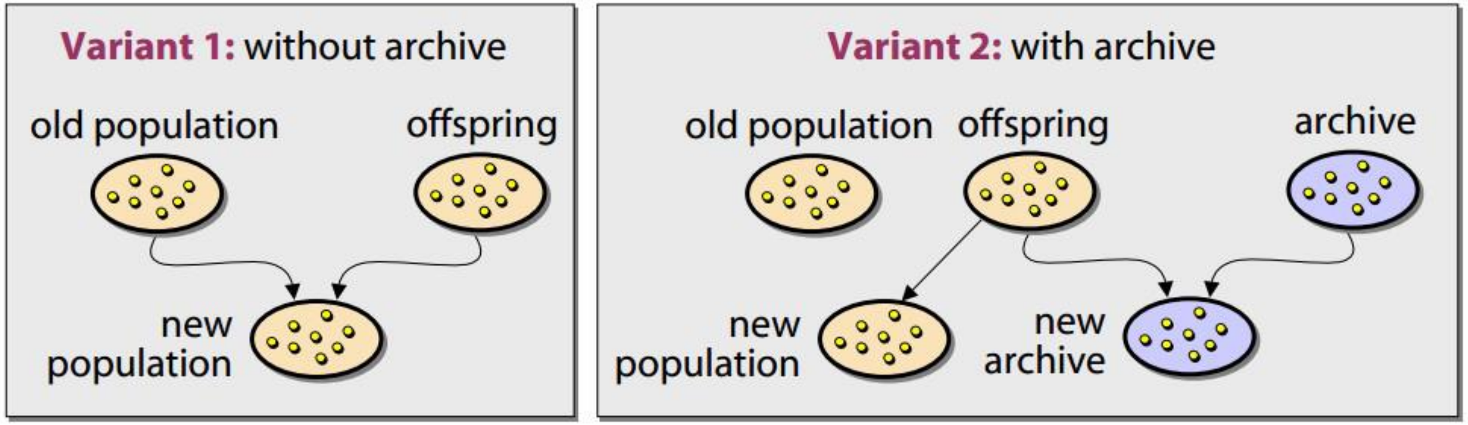
A Generic Evolutionary Algorithm Framework

```
Initialize population P0;
t = 0;
while Termination criteria not reached do
    Parent_selection();
    Recombination();
    Mutation();
    Assign_fitness();
    Selection();
    t = t + 1;
end while
```



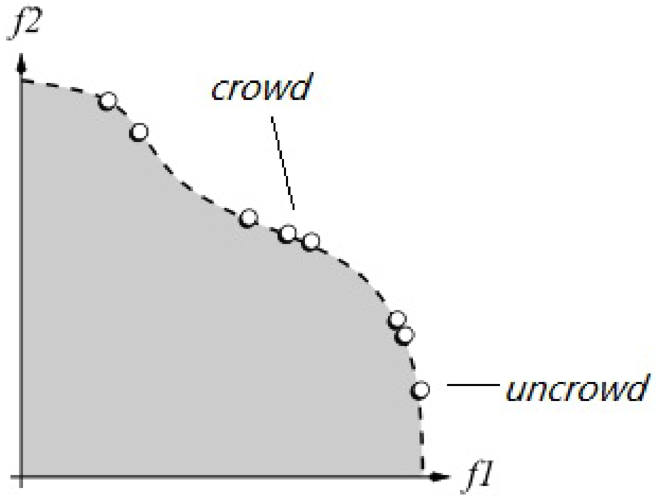
Challenges

- Q: How to guide search towards the Pareto-optimal front?
- A: Use Environmental selection. Elitism is important in evolutionary algorithms.
- Q: How to maintain the diversity of the population?
- A: Use Density estimation approaches.
- Q: How to preserve Pareto-optimal solutions?
- A: Use Population or Population + Archive strategy.



Density Estimation

- To maintain a diverse Pareto-optimal set, density estimation of individual in the population is introduced.



- Approaches:
  - Sharing function approach
  - Crowing distance approach
  - k-th nearest neighbor approach

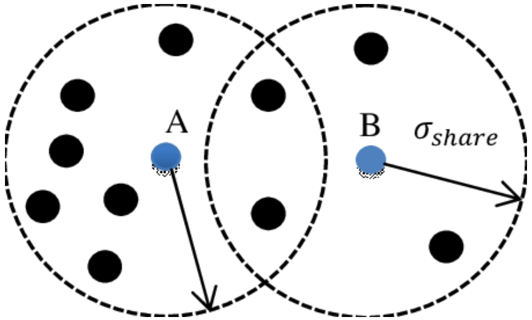
Sharing Function

- Idea: diversity in the population is preserved by degrading the fitness of similar solutions.
- The algorithm of calculating the shared fitness value of i-th individual in the population of size *N*:
  - Calculate sharing function value with all solutions in the population according to

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^{\alpha}, & \text{if } d_{ij} \leq \sigma_{share} \\ 0, & \text{otherwise.} \end{cases}$$

  - Calculate niche count nc_i as follows

$$nc_i = \sum_{j=1}^{N} Sh(d_{ij})$$



  - Calculate shared fitness as

$$f_i' = f_i / nc_i$$

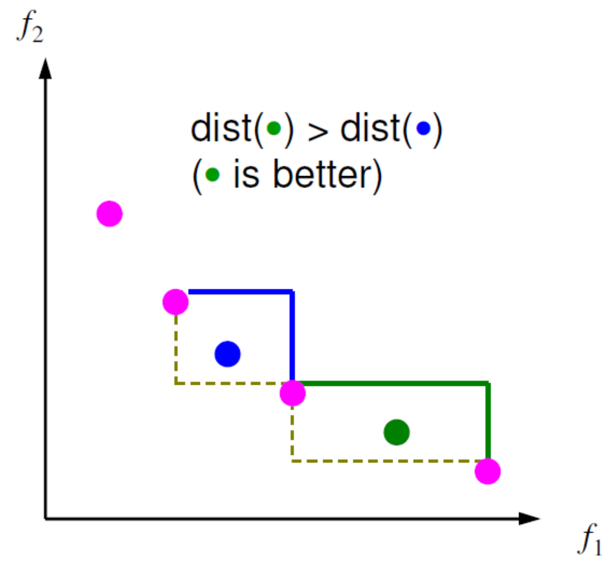
  - If d = 0 then Sh(d) = 1 meaning that two solutions are identical. If d ≥ σ_{share} then Sh(d) = 0 meaning that two solutions do not have any sharing effect on each other.

Density Estimation

- Crowing distance approach
  - To get an estimate of the density of solutions surrounding a particular solution in the population, calculate the average distance of two points on either side of this point along each of the objectives.
  - This quantity $i_{distance}$ serves as an estimate of the perimeter of the cuboid formed by using the nearest neighbors as the vertices (called *crowding distance*).

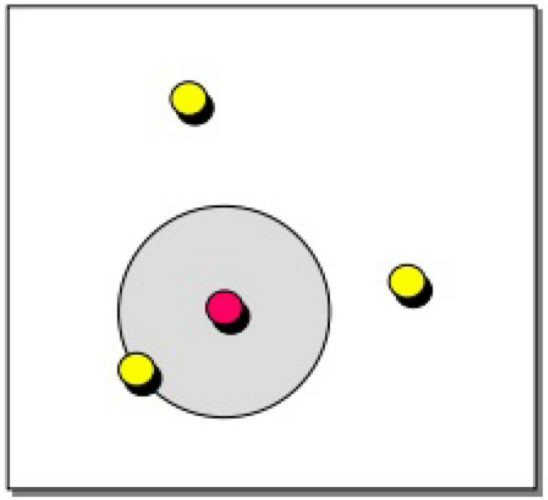
Crowding distance assignment

INPUT: Non-dominated set $\mathcal{I}$
$l = \mathcal{I}$
for each i **do**
  set $\mathcal{I}[i]_{distance} = 0$;
end for
for each objective m **do**
  $\mathcal{I} = sort(\mathcal{I}, m)$;
  $\mathcal{I}[1]_{distance} = \mathcal{I}[l]_{distance} = \infty$;
  for $i = 2$ to $l - 1$ **do**
    $\mathcal{I}[i]_{distance} = \mathcal{I}[i]_{distance} + (\mathcal{I}[i+1].m - \mathcal{I}[i-1].m)/(f_m^{max} - f_m^{min})$;
  end for
end for



Density Estimation

- k-th nearest neighbor approach
  - For each individual i, the distances (in objective space) to all individuals j are calculated and stored in a list.
  - After sorting the list in increasing order, the k-th element gives the distance, denoted as k_i.



MOEA Approaches

- Weighted-Based Genetic Algorithm (WBGA)
- Non-dominated Sorting Genetic Algorithm II (NSGA-II)
- Strength Pareto Evolutionary Algorithm II (SPEA-II)
- Multiobjective Optimization Evolutionary Algorithm with Decomposition (MOEA/D)
- …

Weighted-Based Genetic Algorithm (WBGA)

- The framework of WBGA is very similar to EA procedure.
- Consider a multi-objective maximization problem. The user first need to assign a set of weight vector w. Normalize each weight: $\bar{w}_i = \frac{w_i}{\sum_{j=1}^{M} w_j}$

x_w	1	2	3	4	5
Weight vector	(0.1,0.9)	(0.3,0.7)	(0.5,0.5)	(0.7,0.3)	(0.9,0.1)

- Each individual $x^{(i)}$ in the population is associated with a weight vector.

Individual	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$	$x^{(7)}$	$x^{(8)}$
x_w	3	1	2	4	2	5	1	3

- Assign fitness to individuals: $F(x^{(i)}) = \sum_{j=1}^{M} w_j^{x_w^{(i)}} \frac{f_j(x^{(i)}) - f_j^{min}}{f_j^{max} - f_j^{min}}$
- Use the Sharing function to calculate the niche count nc_i for individual $x^{(i)}$
- Calculate the shared fitness: $F_i' = F_i / nc_i$

Non-dominated Sorting Genetic Algorithm II (NSGA-II)

- Fast non-dominated sorting approach
  - Computational complexity: $O(MN^2)$.

- Diversity preservation: crowded comparison approach
  - Every solution in the population has two attributes:
    1. non-domination rank (i^{rank})
    2. crowding distance ($i^{distance}$)
  - Crowded comparison operator: A partial order $\prec_n$ is defined as:

$$i \prec_n j \text{ if}(i^{rank} < j^{rank}) \text{ or } ((i^{rank} = j^{rank}) and (i^{distance} > j^{distance}))$$

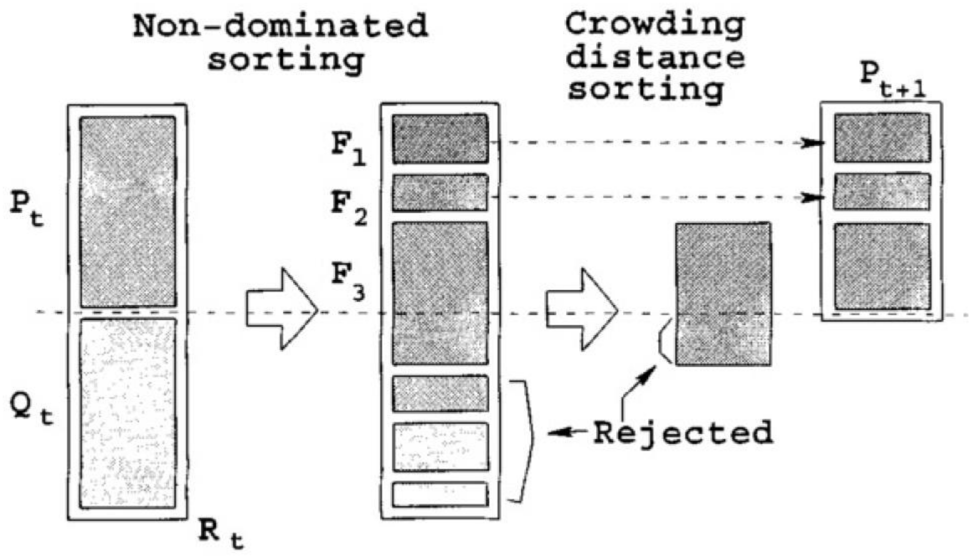
- Elitist evolutionary model
  - Only the best solutions survive to subsequent generations.

Non-dominated Sorting Genetic Algorithm II (NSGA-II)

- The overall complexity of the algorithm in each iteration is $O(MN^2)$.

NSGA-II in t-th iteration

Q_t = **make-new-pop**(P_t);
$R_t = P_t \cup Q_t$;
$\mathcal{F}$ = **fast-non-dominated-sort**(R_t);
$P_{t+1} = \emptyset$ and $i = 1$;
while $|P_{t+1}| + |\mathcal{F}_i| \leq N$ **do**
    crowing-distance-assignment($\mathcal{F}_i$);
    $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$;
    $i = i + 1$;
end while
Sort($\mathcal{F}_i, \prec_n$);
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$;
$t = t + 1$



NSGA-II procedure

NSGA-II as a Constraint Handling Approach

- In the presence of constraints each solution in the population can be either feasible or infeasible, so that there are the following three possible situations:
  1. Both solutions are feasible.
  2. One is feasible and other is not.
  3. Both are infeasible.
- Constrained-domination: A solution i is said to constrained-dominate a solution j, if any of the following conditions is true
  1. Solution i is feasible and solution j is not.
  2. Solutions i and j are both infeasible, but solution i has a smaller overall constraint violation.
  3. Solutions i and j are feasible, and solution i dominates solution j.
- Binary tournament selection with modified domination concept is used to choose the better solution out of the two solutions i and j, randomly picked up from the population.

MOEA/D

- Multiobjective Optimization Evolutionary Algorithm with Decomposition, “D” for problem Decomposition.
- The MOEA/D decomposes a multi-objective optimization problem into N scalar optimization subproblems.
- These subproblems are optimized simultaneously from the evolution of a population of solutions.
- In each generation, the population is composed of the best solutions found for each subproblem.
- Each subproblem is optimized considering only neighboring sub-problem information.
- It has low computational complexity compared to popular methods: NSGA-II, MOGLS.

Decomposition

- Weight vector-based aggregation function
  - Linear weighted sum
  - Tchebycheff
  - Boundary intersection
  - Penalty-based Boundary Intersection
- Every weight vector defines a subproblem, and every subproblem keeps the best solution.
  - A large number of weight vector
  - A very small sub-population (typically, the size is 1)

MOEA/D Framework

Input:
- MOP (1);
- a stopping criterion;
- N: the number of the subproblems considered in MOEA/D;
- a uniform spread of N weight vectors: $\lambda^1, \ldots, \lambda^N$;
- T: the number of the weight vectors in the neighborhood of each weight vector.

Output: EP.

Step 1.1) Set EP $= \emptyset$.

Step 1.2) Compute the Euclidean distances between any two weight vectors and then work out the T closest weight vectors to each weight vector. For each $i = 1, \ldots, N$, set $B(i) = \{i_1, \ldots, i_T\}$, where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the T closest weight vectors to λ^i.

Step 1.3) Generate an initial population $x^1, \ldots, x^N$ randomly or by a problem-specific method. Set $FV^i = F(x^i)$.

Step 1.4) Initialize $z = (z_1, \ldots, z_m)^T$ by a problem-specific method.

For $i = 1, \ldots, N$, do

Step 2.1) Reproduction: Randomly select two indexes k, l from $B(i)$, and then generate a new solution y from x^k and x^l by using genetic operators.

Step 2.2) Improvement: Apply a problem-specific repair/ improvement heuristic on y to produce y'.

Step 2.3) Update of z: For each $j = 1, \ldots, m$, if $z_j < f_j(y')$, then set $z_j = f_j(y')$.

Step 2.4) Update of Neighboring Solutions: For each index $j \in B(i)$, if $g^{te}(y'|\lambda^j, z) \le g^{te}(x^j|\lambda^j, z)$, then set $x^j = y'$ and $FV^j = F(y')$.

Step 2.5) Update of EP:

Remove from EP all the vectors dominated by $F(y')$.

Add $F(y')$ to EP if no vectors in EP dominate $F(y')$.

Aggregation Vectors

- Simplex-Lattice Design

$$\lambda_j^i \in \left\{\frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H}\right\}, \forall i = 1, \ldots, N, j = 1, \ldots, M$$

where M is the number of objectives, H is a positive integer (user-defined).

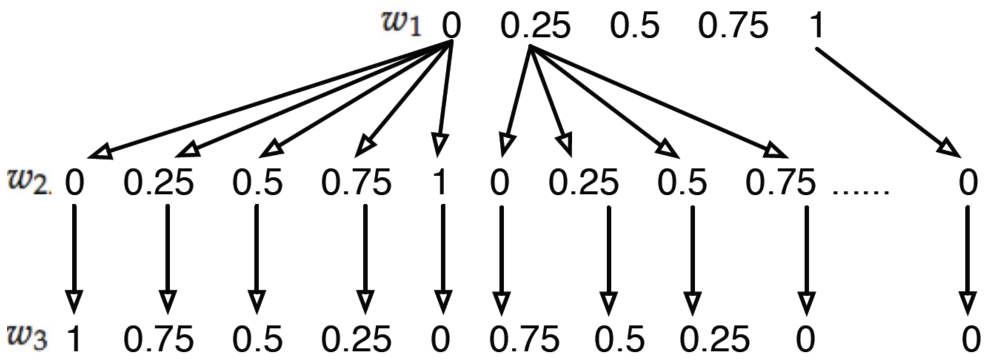
- The number of vectors (equal to the size of the population) is given by:

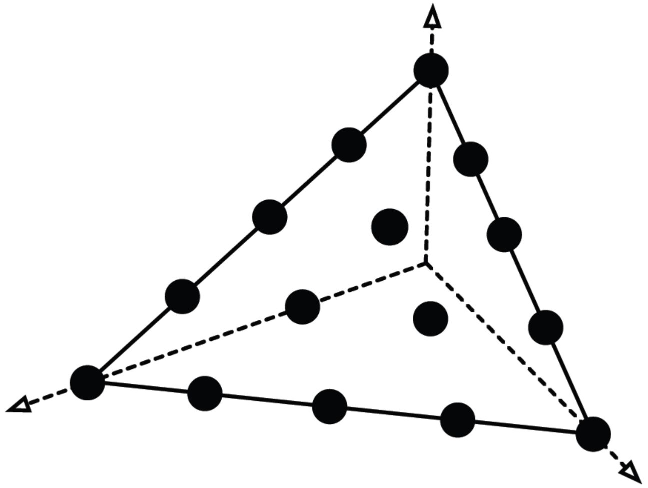
$$N = C_{H+M-1}^{M-1}$$

- For example, assume that M = 3 objectives and different H values, then
  - $H = 1, N = C_3^2 = 3,\ \lambda \in \{(1,0,0), (0,1,0), (0,0,1)\}$
  - $H = 2, N = C_4^2 = 6,\ \lambda \in \{(1,0,0), (0,1,0), (0,0,1), (0,1/2,1/2), (1/2,0,1/2), (1/2,1/2,0)\}$
  - $H = 3, N = C_5^2 = 10,\ \lambda \in \{(1,0,0), (0,1,0), (0,0,1), (0,1/3,2/3), (1/3,0,2/3), (1/3,2/3,0), (0,2/3,1/3), (2/3,0,1/3), (2/3,1/3,0), (1/3,1/3,1/3)\}$

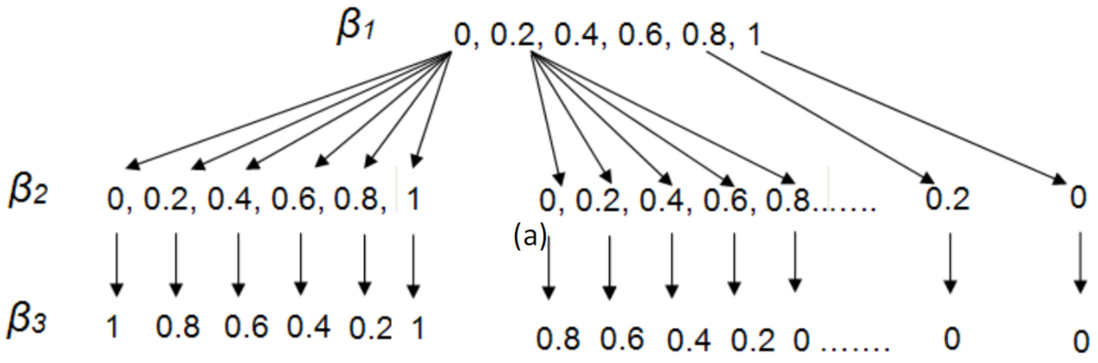
Aggregation Vectors

- M = 3, H = 4





- M = 3, H = 5



Performance Measures of MOOP

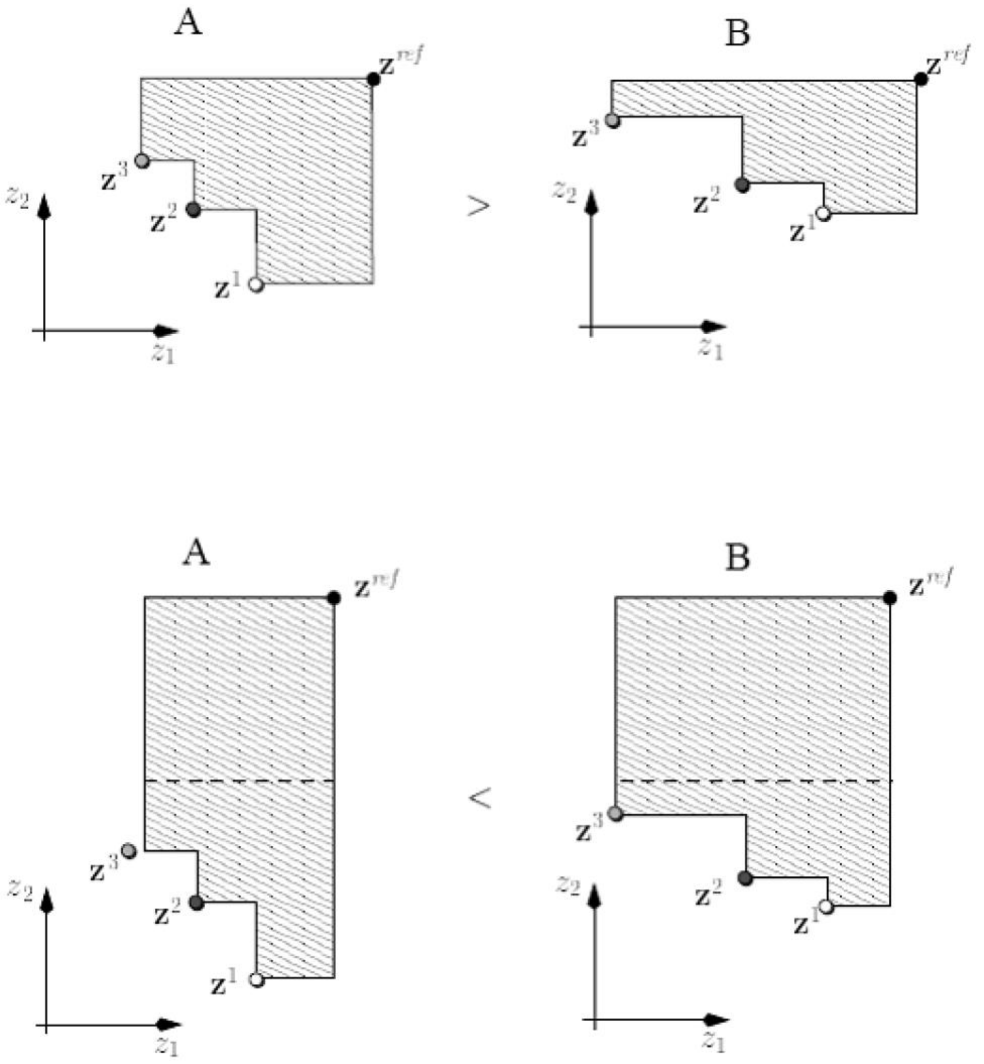
- The result of a MOEA run is not a single scalar value, but a collection of vectors forming a non-dominated set.
- Comparing two MOEA algorithms requires comparing the non-dominated sets they produce. However, there is no straightforward way to compare different non-dominated sets.
- Three goals that can be identified and measured:
  1. The distance of the resulting non-dominated set to the Pareto-optimal front should be minimized.
  2. A good (in most cases uniform) distribution of the solutions found is desirable.
  3. The extent of the obtained non-dominated front should be maximized, i.e., for each objective, a wide range of values should be present.

Performance Metrics

- Number of fronts
- Generational distance (GD)
- Spacing (SP)
- Spread
- Hypervolume (HV) (*S-metric*)
- Coverage (*C-metric*)
- Inverted Generational Distance (IGD) (*D-metric*)
- Epsilon indicator

S-Metric

- It calculates the hypervolume of the multi-dimensional region enclosed by a set A and a reference point Z^{ref}.
- The hypervolume expresses the size of the region that is dominated by A.
- The bigger the value of this measure the better the quality of A is, and vice versa.



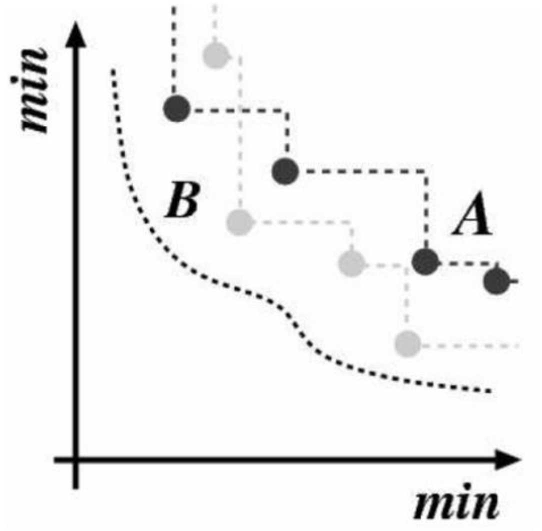
S-Metric

- Pros:
  - Given two non-dominated sets A and B, if each point in B is dominated by a point in A, then A will always be evaluated as being better than B.
  - Independent: the hypervolume calculated for the given set is not dependent on any other, or any reference set.
- Cons:
  - Requires defining some upper boundary of the region.
  - It has a large computational overhead, especially for high-dimensional cases.

Reading: Nicola Beume, Carlos M. Fonseca, Manuel López-Ibáñez, Luís Paquete, and J. Vahrenhold. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13(5):1075–1082, 2009.

C-Metric

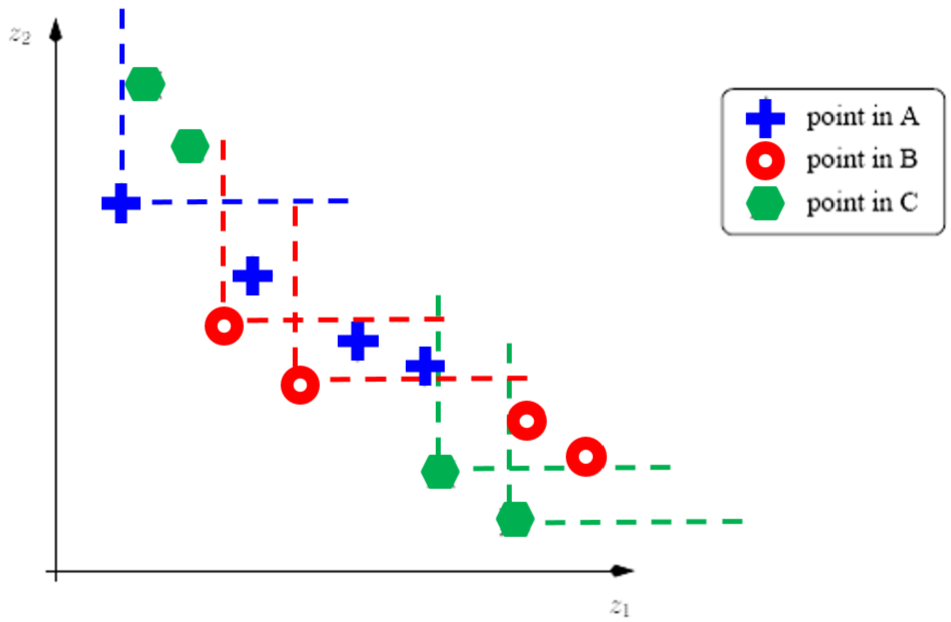
- Coverage of two sets C(X,Y): given two sets of non-dominated solutions X and Y found by the compared algorithms, the measure C(X,Y) returns a ratio of a number of solutions of Y that are dominated by or equal to any solution of X to the whole set Y.
  - It returns values from the interval [0,1].
  - The value C(X,Y) = 1 means that all solutions in Y are covered by solutions of the set X.
  - And vice versa, the value C(X,Y) = 0 means that none of the solutions in Y are covered by the set X.
  - Always both orderings have to be considered, since C(X,Y) is not necessarily equal to 1-C(Y,X).
  - Roughly speaking, C(X,Y)>C(Y,X) indicates that X is better than Y.



$C(A,B) = 0.25,\ C(B,A) = 0.75$

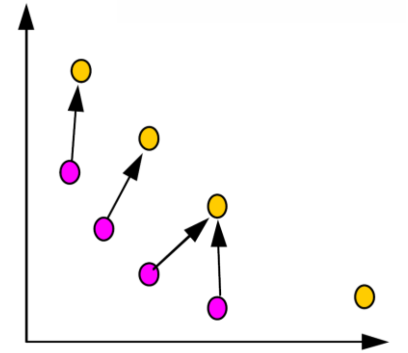
C-Metric

- It has low computational overhead.
- If two sets are of different cardinality and/or the distributions of the sets are non-uniform, then it gives unreliable results.
- It is cycle-inducing: if three sets are compared using C-metric, they may not be ordered.
- Example:
  - C(A,B) = 0, C(B,A) = 3/4
  - C(A,C) = 1/2, C(C,A) = 0
  - C(B,C) = 0, C(C,B) = 1/2
  - B is considered better than A
  - A is considered better than C
  - C is considered better than B



D-Metric

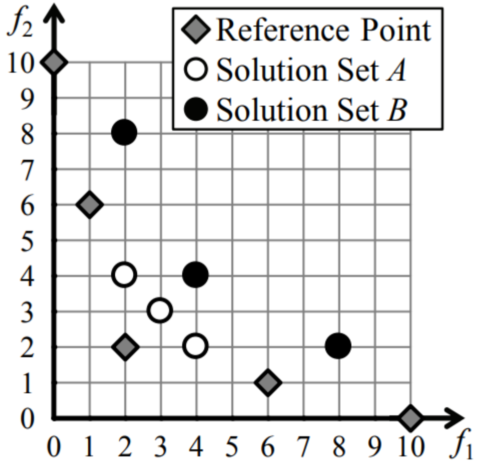
- D-metric measures the mean distance over the points in a reference set, of the nearest point in an approximation set.



$$D(A, P^*) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|}$$

where A is the approximation set, P* is a reference set (true PF), d(v,A) denotes the nearest distance from v to solutions in A.

- Pros:
  - It is cheap to compute.
  - It can differentiate between different levels of complete outperformance given an appropriate choice of reference set.
- Cons:
  - It is not Pareto compliant.
  - The score is strongly dependent upon the distribution of points in the reference set.



Example with misleading IGD.

Applications

- Popular benchmark problems
  - Continuous functions
  - Knapsack
  - Traveling salesman problem
  - Permutation flow shop scheduling

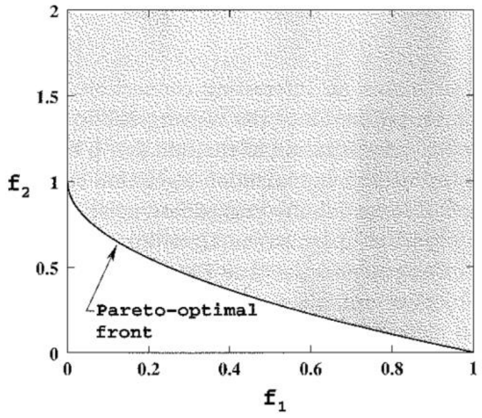


Figure 213 The search space near the Pareto-optimal region for ZDT1.

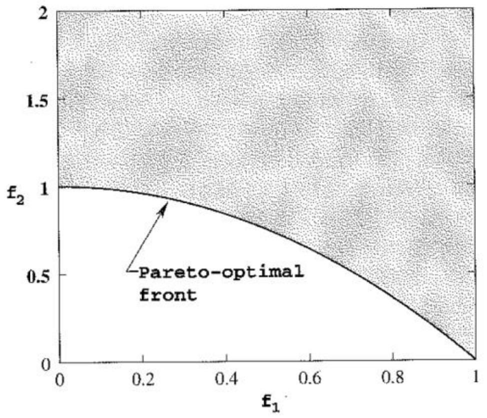


Figure 214 The search space near the Pareto-optimal region for ZDT2.

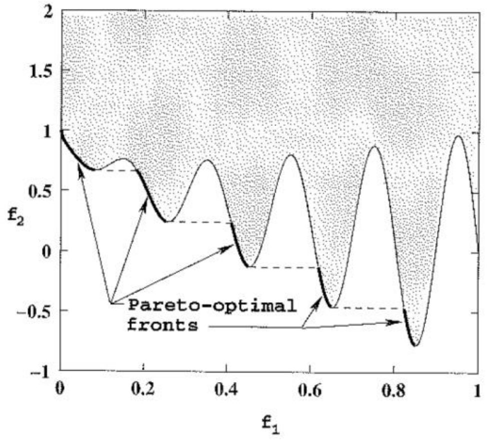


Figure 215 The search space near the Pareto-optimal region for ZDT3.

Thank you!