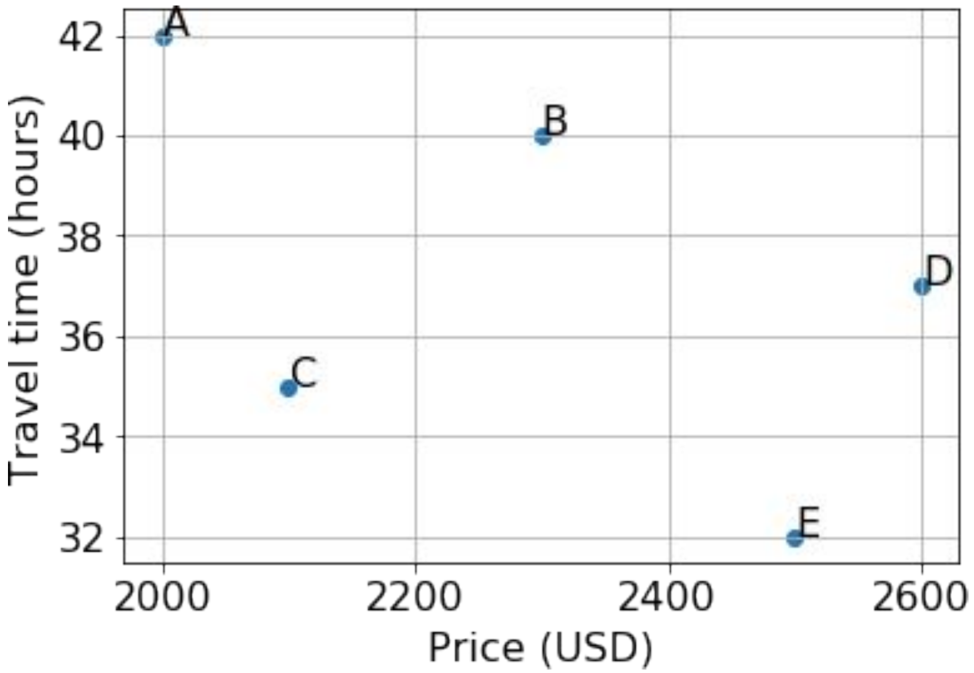
Multi-objectives optimization

Table of contents

Multi-objective optimization

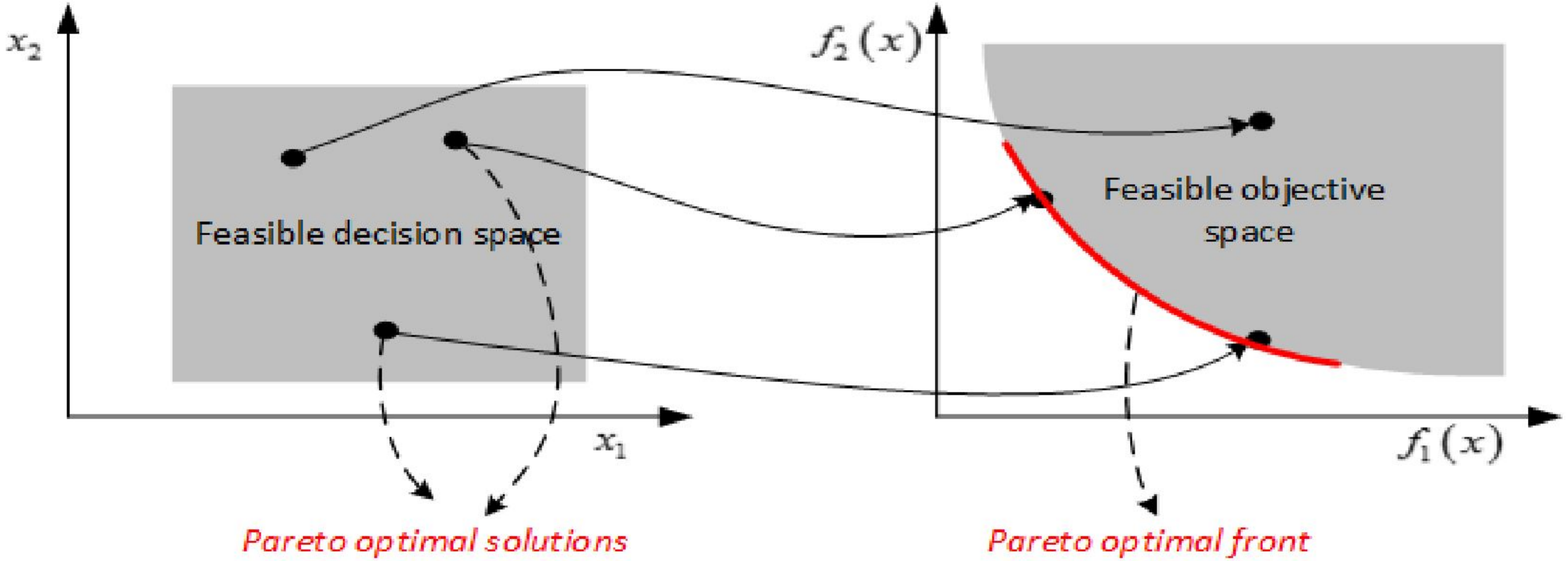
Suppose you need to fly on a long trip: Should you choose the cheapest ticket (more waiting time) or shortest flying time (more expensive)?

Ticket	Travel time (hours)	Price ($)
A	42	2000
B	40	2300
C	35	2100
D	37	2600
E	32	2500



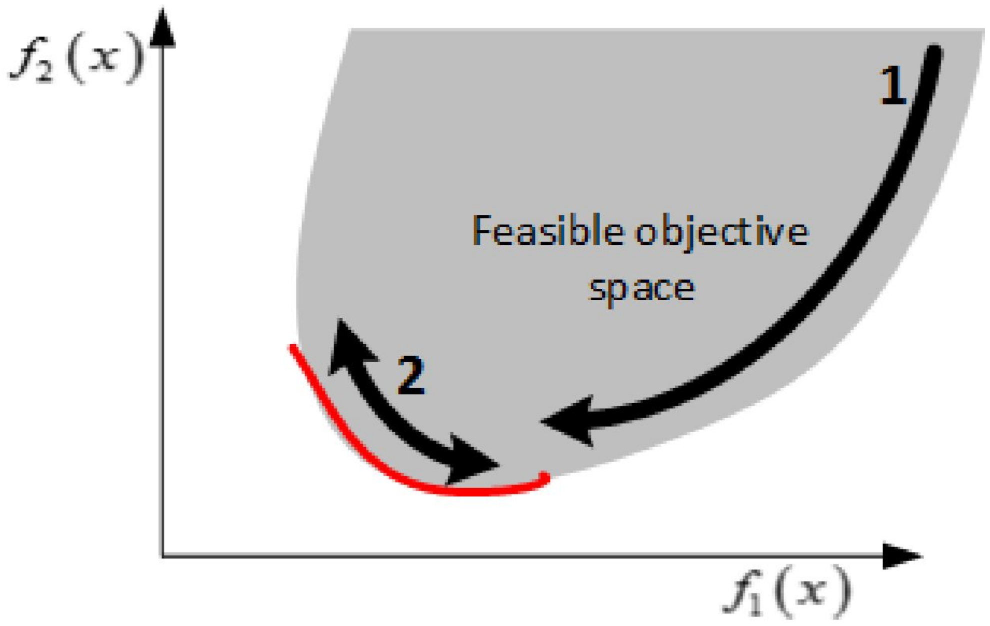
Multi-objective optimization

- Definition: Non-dominated solution set is a set of all the solutions that are not dominated by any member of the solution set.
- The non-dominated set of the entire feasible decision space is called the Pareto-optimal set
- The boundary defined by the set of all point mapped from the Pareto optimal set is called the Pareto-optimal front



Multi-objective optimization

- The goal in Multi-Objective Optimization (MOO)
- Find a set of solutions as close as possible to Pareto-optimal front
- To find a set of solutions as diverse as possible



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

Math definitions

- Multi-objective optimization
  - Minimize $f_m(x)$, $\quad m \in \{1, 2, \dots, M\}$
  - Subject to $g_j(x) \geq 0$, $\quad j \in \{1, 2, \dots, J\}$
  - $h_k(x) = 0$, $\quad k \in \{1, 2, \dots, K\}$
- A solution $x_1 \in R^n$ dominates another solution $x_2 \in R^n$ $(x_1 \succ x_2)$ when
  - $\forall i \in \{1, 2, \dots, M\}$, $\quad f_i(x_1) \leq f_i(x_2)$
  - $\exists i \in \{1, 2, \dots, M\}$, $\quad f_i(x_1) < f_i(x_2)$

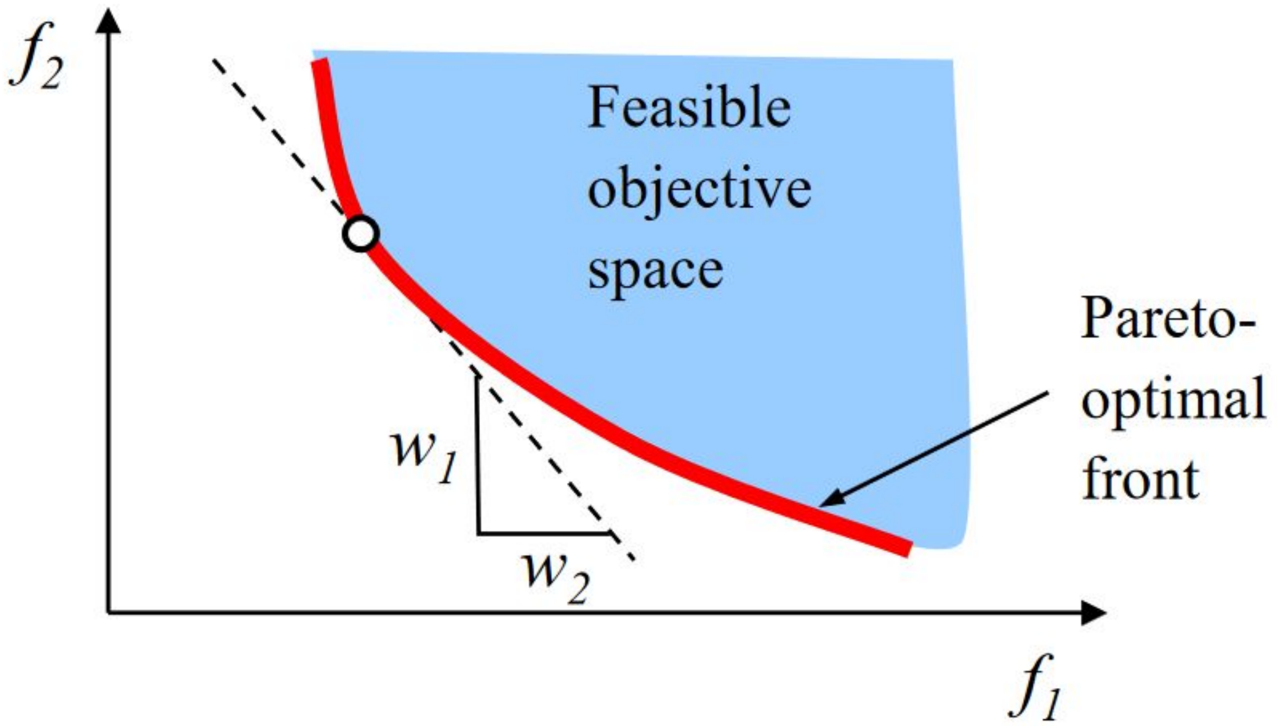
Methods

- Weighted Sum method
- Constraint method
- Weighted Metric method
- Evolutionary Algorithm (EA)

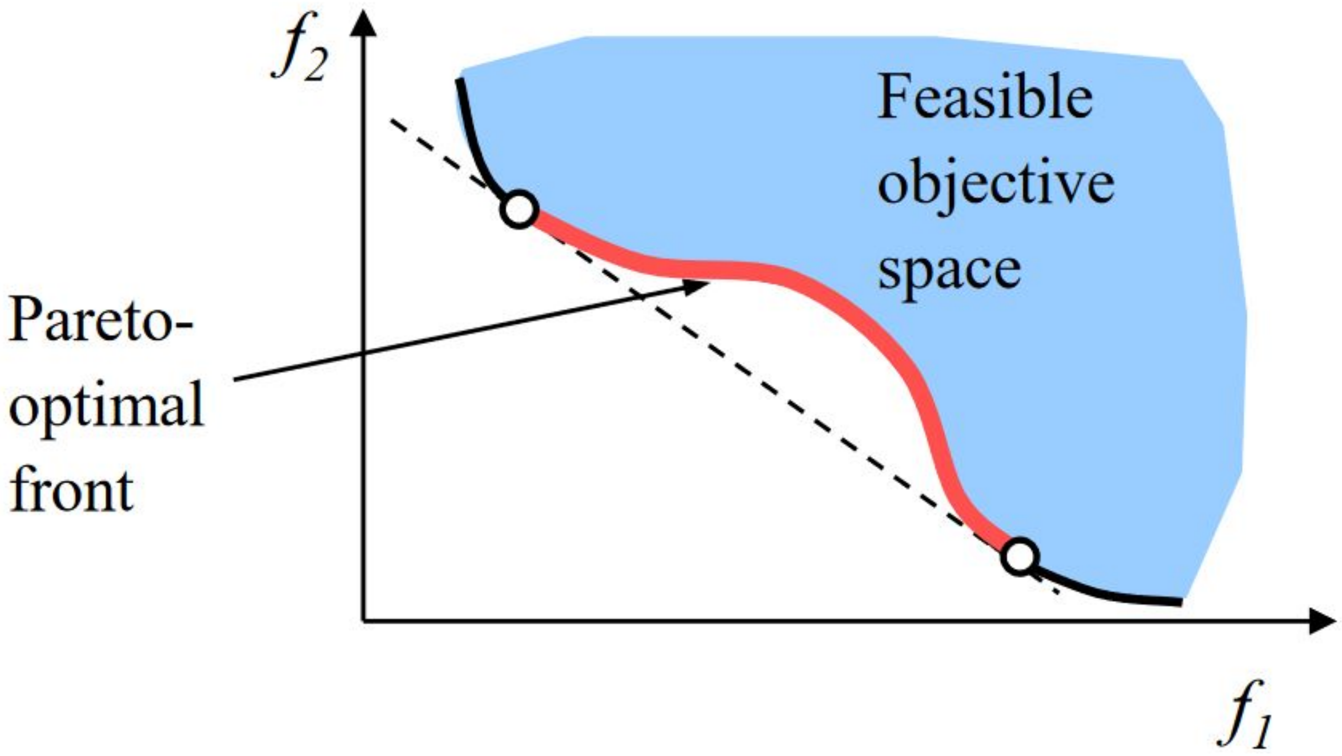
Weighted Sum

- Scalarize a set of objectives into a single objective with predefined weight.
  - Minimize $F(x) = \sum_{m=1}^{M} w_m f_m(x)$
  - Subject to $g_j(x) \geq 0, \quad j \in \{1, 2, \ldots, J\}$
  - $h_k(x) = 0, \quad k \in \{1, 2, \ldots, K\}$
- Pros: Simple, easy to implement
- Cons:
  - Can only find 1 solution for each set of weight
  - Cannot find all solutions in case of non-convex objective space

Weighted Sum (Convex case)



Weighted Sum (Non-convex case)



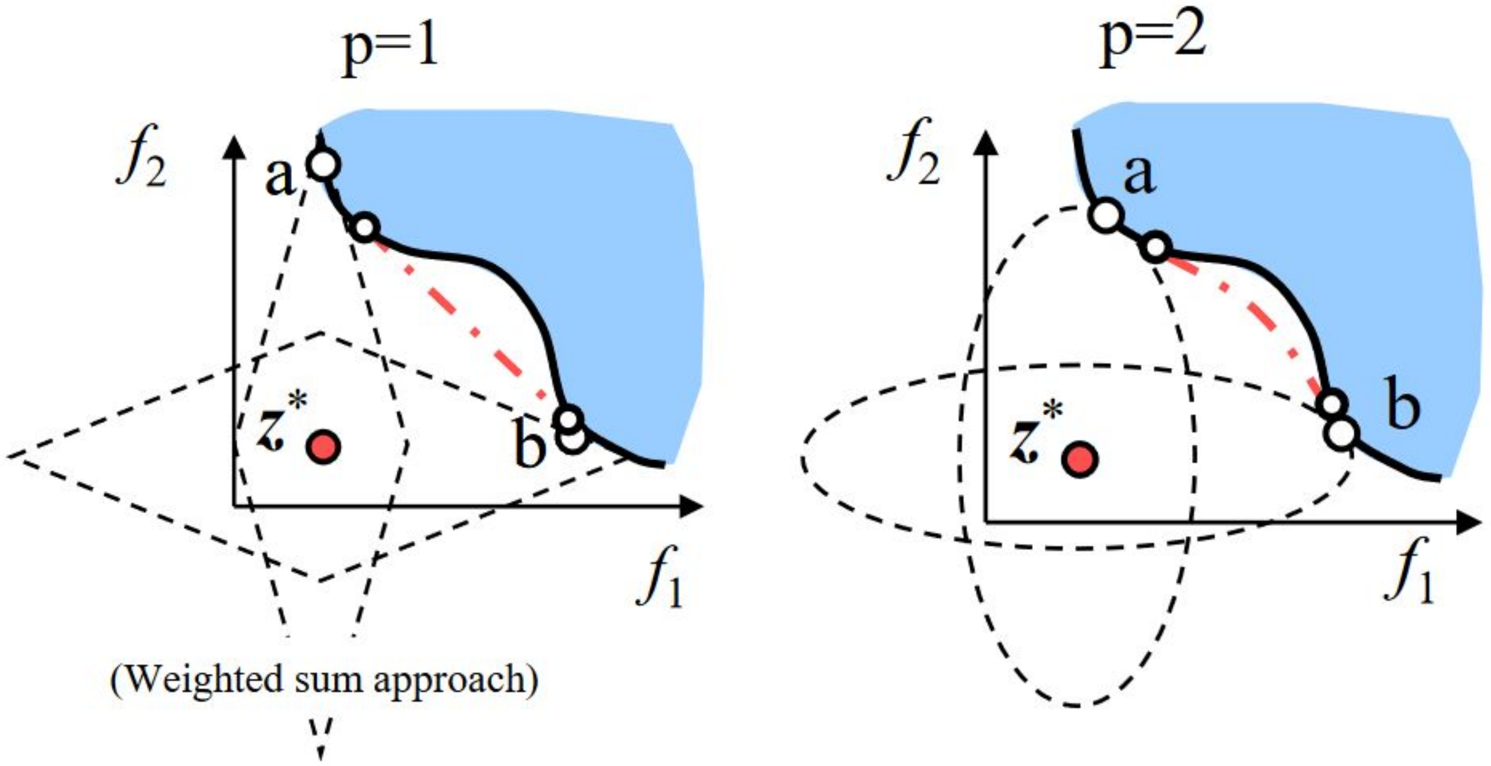
Constraint method

- Optimize one objective while consider other objectives as constraints.
  - Minimize $f_\mu(x)$
  - Subject to $g_j(x) \geq 0, \quad j \in \{1, 2, \dots, J\}$
  - $h_k(x) = 0, \quad k \in \{1, 2, \dots, K\}$
  - $f_m \leq \epsilon_m, \quad m \in \{1, 2, \dots, M\}, m \neq \mu$
- Pros: Can applicable to both convex and non-convex problem
- Cons: Must carefully choose constraint ε

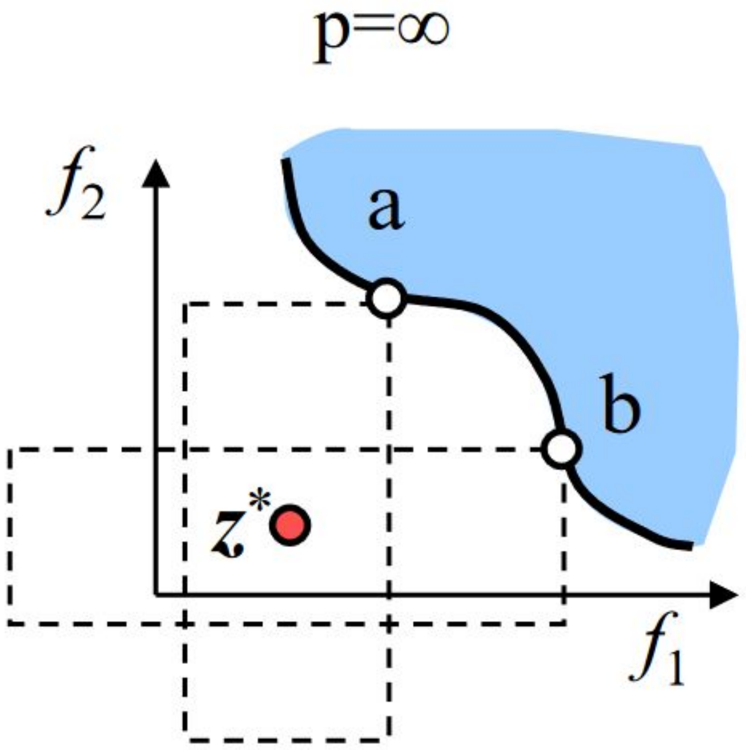
Weighted Metric

- Combine multiple objectives using the weighted distance metric of any solution from the ideal solution z^*.
  - Minimize $l_p(x) = \left(\sum_{m=1}^{M} w_m |f_m(x) - z^*|^p\right)^{\frac{1}{p}}$
  - Subject to $g_j(x) \geq 0, \quad j \in \{1, 2, \dots, J\}$
  - $h_k(x) = 0, \quad k \in \{1, 2, \dots, K\}$
- Pros: Can find all Pareto-optimal solution with ideal solution z^*
- Cons:
  - Require z^* to be found by independently optimize each objective function
  - For small *p*, not all Pareto-optimal solution can be founded
  - As *p* increases, the problem becomes non-differentiable.

Weighted Metric



Weighted Metric

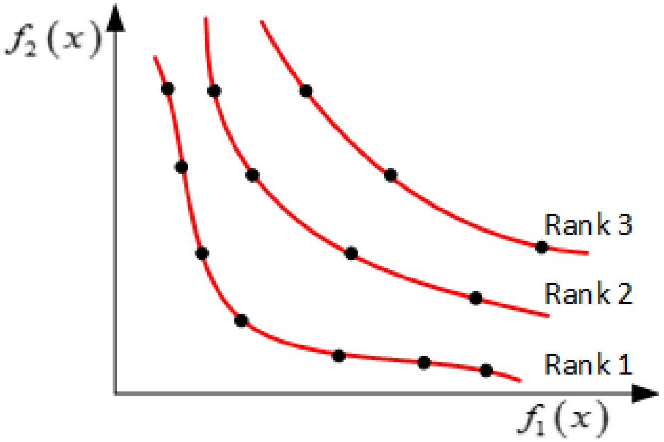


(Weighted Tchebycheff problem)

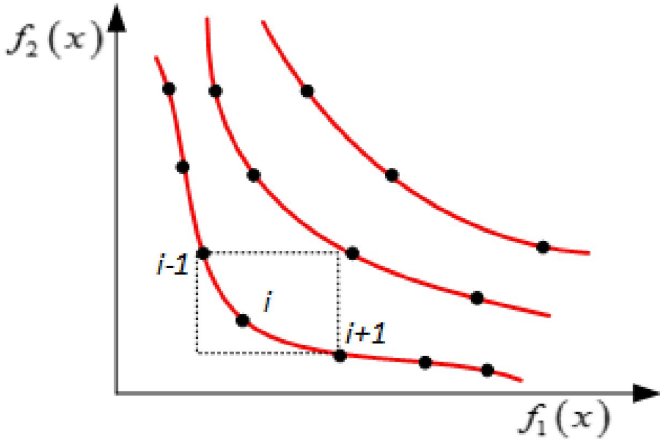
Evolutionary Algorithm

- Advantages over classical methods
  - In MOO, we desire a set of solutions as answer
  - Classical methods operate on a candidate solution
  - To obtain a set of solution in classical methods, we must run the program with different parameters -> not efficient
  - Evolutionary Algorithm (EA) fundamentally operates on a set of candidate solutions
- There are several different multi-objectives evolutionary algorithm, the most popular one is Non-dominated Sorting Genetic Algorithm II (NSGA-II)

NSGA-II



Non-dominated sorting



Crowding distance

Consider a pair of individuals p_1 and p_2 with non-dominated fronts NF_1 and NF_2, and crowding distances CD_1 and CD_2. Individual p_2 to be preferred over p_1 ($p_2 \succ p_1$) iff $\begin{cases} NF_1 < NF_2 \\ NF_1 = NF_2 \text{ and } CD_1 > CD_2 \end{cases}$

NSGA-II

$\underline{\texttt{fast-non-dominated-sort}(P)}$	
for each $p \in P$	
$\quad S_p = \emptyset$	
$\quad n_p = 0$	
$\quad$ for each $q \in P$	
$\quad\quad$ if $(p \prec q)$ then	If p dominates q
$\quad\quad\quad S_p = S_p \cup \{q\}$	Add q to the set of solutions dominated by p
$\quad\quad$ else if $(q \prec p)$ then	
$\quad\quad\quad n_p = n_p + 1$	Increment the domination counter of p
$\quad$ if $n_p = 0$ then	p belongs to the first front
$\quad\quad p_{\text{rank}} = 1$	
$\quad\quad \mathcal{F}_1 = \mathcal{F}_1 \cup \{p\}$	
$i = 1$	Initialize the front counter
while $\mathcal{F}_i \neq \emptyset$	
$\quad Q = \emptyset$	Used to store the members of the next front
$\quad$ for each $p \in \mathcal{F}_i$	
$\quad\quad$ for each $q \in S_p$	
$\quad\quad\quad n_q = n_q - 1$	
$\quad\quad\quad$ if $n_q = 0$ then	q belongs to the next front
$\quad\quad\quad\quad q_{\text{rank}} = i + 1$	
$\quad\quad\quad\quad Q = Q \cup \{q\}$	
$\quad i = i + 1$	
$\quad \mathcal{F}_i = Q$	

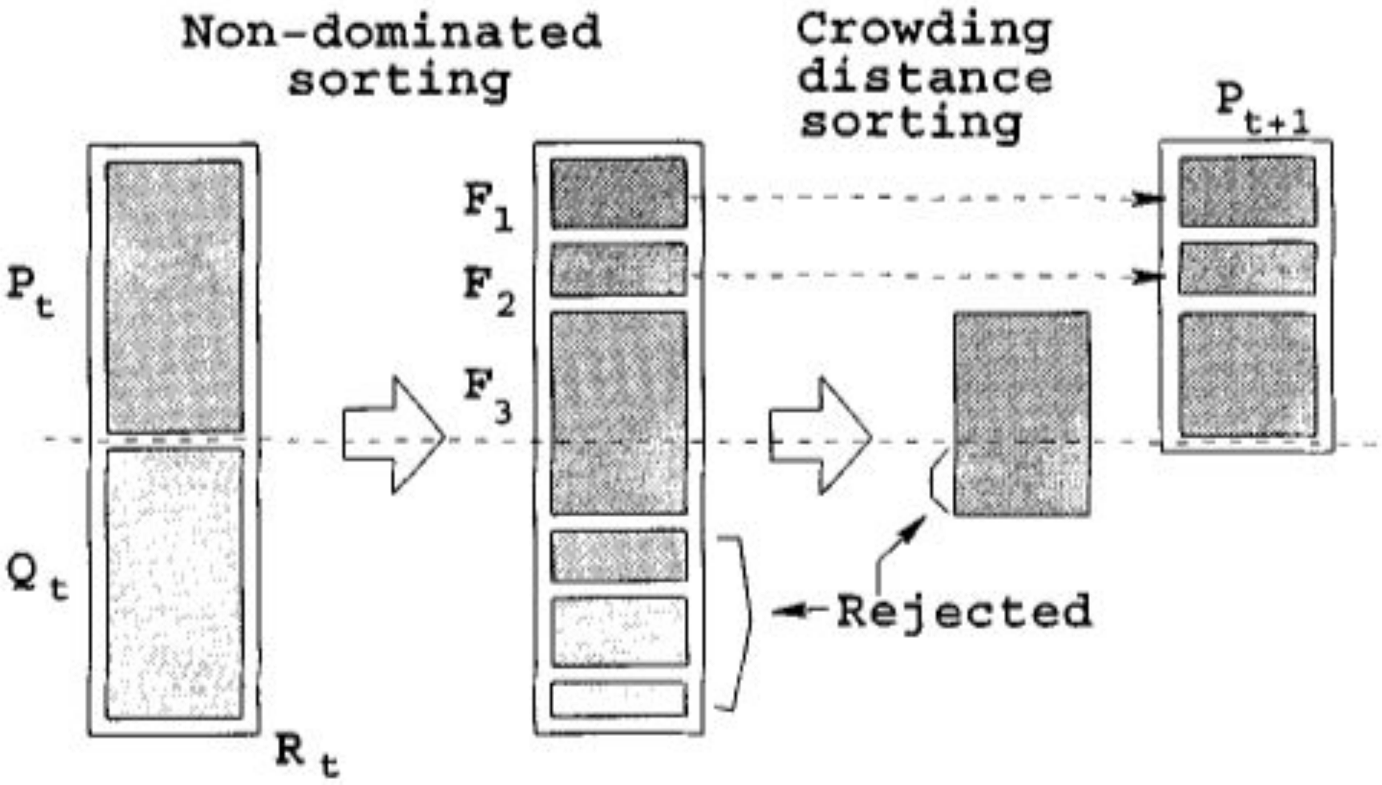
NSGA-II

```
crowding-distance-assignment(ℐ)
```

$l = \lvert\mathcal{I}\rvert$	number of solutions in $\mathcal{I}$
for each i, set $\mathcal{I}[i]_{\text{distance}} = 0$	initialize distance
for each objective m	
$\quad \mathcal{I} = \text{sort}(\mathcal{I}, m)$	sort using each objective value
$\quad \mathcal{I}[1]_{\text{distance}} = \mathcal{I}[l]_{\text{distance}} = \infty$	so that boundary points are always selected
$\quad$ for $i = 2$ to $(l - 1)$	for all other points

$$\mathcal{I}[i]_{\text{distance}} = \mathcal{I}[i]_{\text{distance}} + (\mathcal{I}[i+1].m - \mathcal{I}[i-1].m)/(f_m^{\max} - f_m^{\min})$$

NSGA-II


Non-dominated sorting
Crowding distance sorting
P_{t+1}
P_t
Q_t
R_t
F_1
F_2
F_3
Rejected

NSGA-II

$R_t = P_t \cup Q_t$	combine parent and offspring population
$\mathcal{F} = \texttt{fast-non-dominated-sort}(R_t)$	$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots)$, all nondominated fronts of R_t
$P_{t+1} = \emptyset$ and $i = 1$	
until $\lvert P_{t+1} \rvert + \lvert \mathcal{F}_i \rvert \leq N$	until the parent population is filled
$\quad \texttt{crowding-distance-assignment}(\mathcal{F}_i)$	calculate crowding-distance in $\mathcal{F}_i$
$\quad P_{t+1} = P_{t+1} \cup \mathcal{F}_i$	include ith nondominated front in the parent pop
$\quad i = i + 1$	check the next front for inclusion
Sort$(\mathcal{F}_i, \prec_n)$	sort in descending order using $\prec_n$
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - \lvert P_{t+1} \rvert)]$	choose the first $(N - \lvert P_{t+1} \rvert)$ elements of $\mathcal{F}_i$
$Q_{t+1} = \texttt{make-new-pop}(P_{t+1})$	use selection, crossover and mutation to create a new population Q_{t+1}
$t = t + 1$	increment the generation counter

Challenges

- Many-objectives optimization
  - Number of objective functions is high (>=4)
  - Solutions are more likely to be non-dominated -> no room for evolution process
  - Hard to visualize solutions
- Multi-task multi-objective optimization
  - Optimize simultaneously multiple multi-objective optimization problem at once

Summary

- Multi-objective Optimization
  - Optimize simultaneously several objective functions with tradeoff between functions
  - Search for a set of optimal solution as answer
- Methods to solve Multi-objective Optimization Problem
  - Weighted Sum Method
  - Constraint Method
  - Weighted Metric Method
  - Evolutionary Algorithm
- Non-dominated Sorting Genetic Algorithm II (NSGA-II)
  - Fast non-dominated sorting
  - Crowding distance sorting