Evolutionary Multi-Objective Optimisation: part 3

By: Dr. Vahid Ghasemi

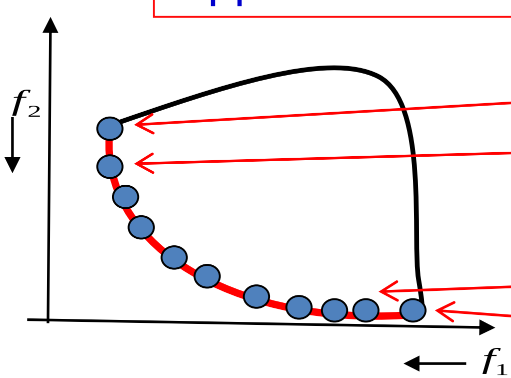
Outline

- MOEA/D
- Many-objective optimisation and NSGA-III

Decomposition-based MOEA

- **Multi-objective optimisation**: use a set of solutions to approximate the true Pareto front
  - Each solution is on the true Pareto front
  - Each solution covers a different region of the Pareto front
  - Solutions are uniformly distributed
- N solutions to approximate the true Pareto front is equal to solve N single-objective sub-problems
  - Each sub-problem finds a solution on the Pareto front

Approximation of the PF → N single obj opt suproblems



$$\min g^{ws}(x \mid \lambda^1) = 1 \times f_1(x) + 0 \times f_2(x) \qquad \lambda^1 = (1, 0)$$

$$\min g^{ws}(x \mid \lambda^2) = 0.9 \times f_1(x) + 0.1 \times f_2(x) \qquad \lambda^2 = (0.9, 0.1)$$

$$\vdots$$

$$\min g^{ws}(x \mid \lambda^{10}) = 0.1 \times f_1(x) + 0.9 \times f_2(x) \qquad \lambda^{10} = (0.1, 0.9)$$

$$\min g^{ws}(x \mid \lambda^{11}) = 0 \times f_1(x) + 1 \times f_2(x) \qquad \lambda^{11} = (0, 1)$$

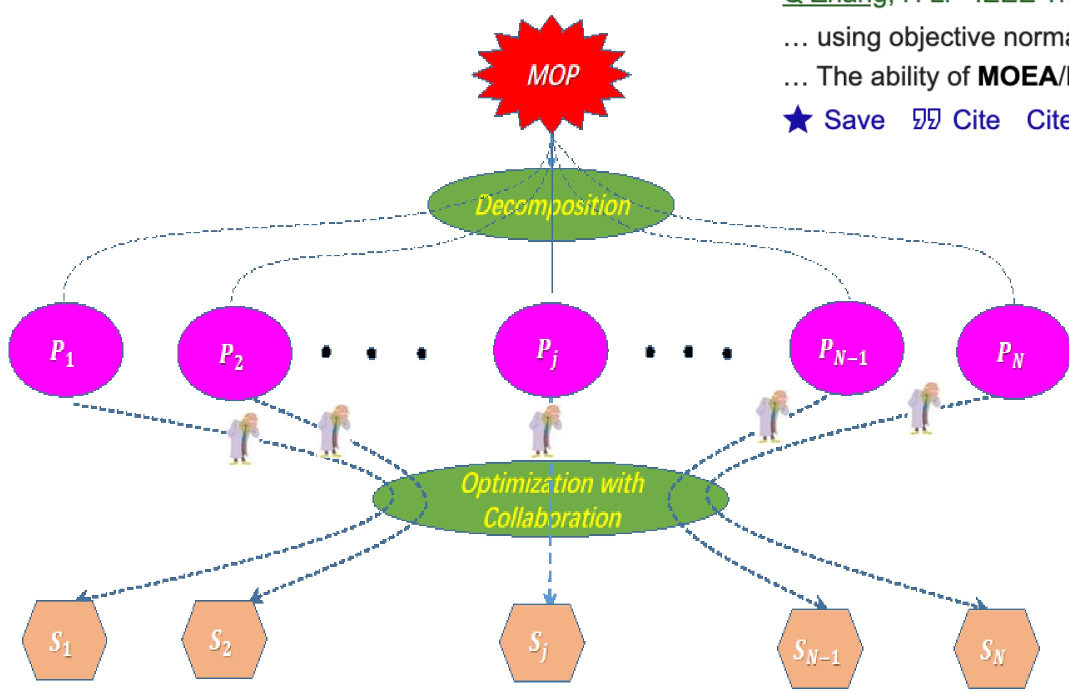
Decomposition-based MOEA

- Traditional decomposition
  - Define the single-objective sub-problems and solve them separately
    - $\min g_1 = f_1$
    - $\min g_2 = 0.9 * f_1 + 0.1 * f_2$
    - …
    - $\min g_{11} = f_2$
- Collaboration (EC methods)
  - N agents are used, each agent is for solving a different sub-problem
  - The N agents solve the sub-problems in a collaborative manner, based on the relationship between the sub-problems

MOEA/D

- **Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)**
  - A population of agents, each searching for the optimum of each sub-problem
  - Each agent collaborates with other related agents in crossover
  - #agents = #sub-problems

MOEA/D: A multiobjective evolutionary al
Q Zhang, H Li - IEEE Transactions on evolutionary co
… using objective normalization can deal with dispara
… The ability of **MOEA/D** with small population, the s
★ Save 99 Cite Cited by 6736 Related articles

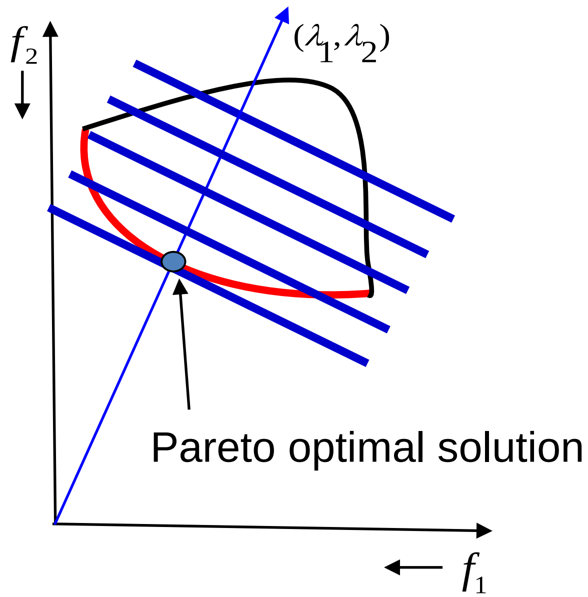


MOEA/D

- **Design Issues**
  - How to decompose the problem and define sub-problems?
    - Each desired Pareto solution is the optimum of a different sub-problem
  - How to design search methods of each agent?
    - Memory: which individuals to store during the search
    - Selection, crossover, mutation operators
    - Neighbourhood structure
    - Collaboration mechanisms between agents

Problem Decomposition

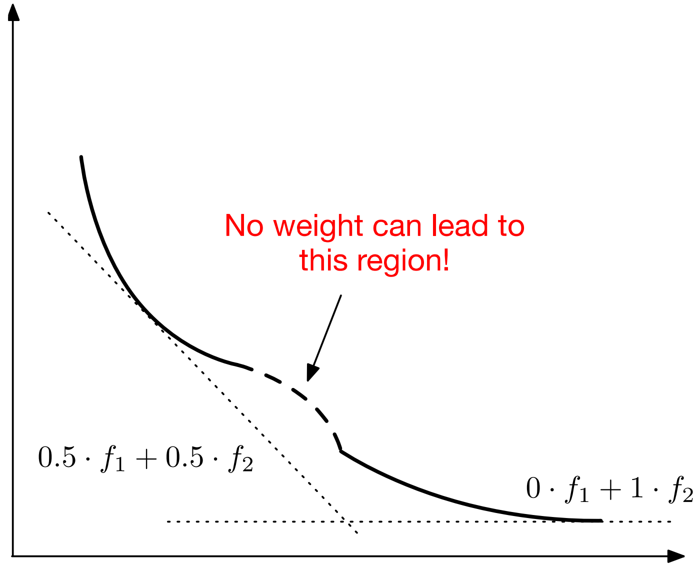
- **Weighted sum approach**
  - Each weight vector (λ_1, λ_2) leads to a different sub-problem



$$\min g^{ws}(x \mid \lambda) = \lambda_1 f_1(x) + \lambda_2 f_2(x)$$

$$\text{where } \lambda_1 + \lambda_2 = 1 \text{ and } \lambda_1, \lambda_2 \geq 0.$$

- Works for convex PF.
- Doesn't work if the PF is not convex.



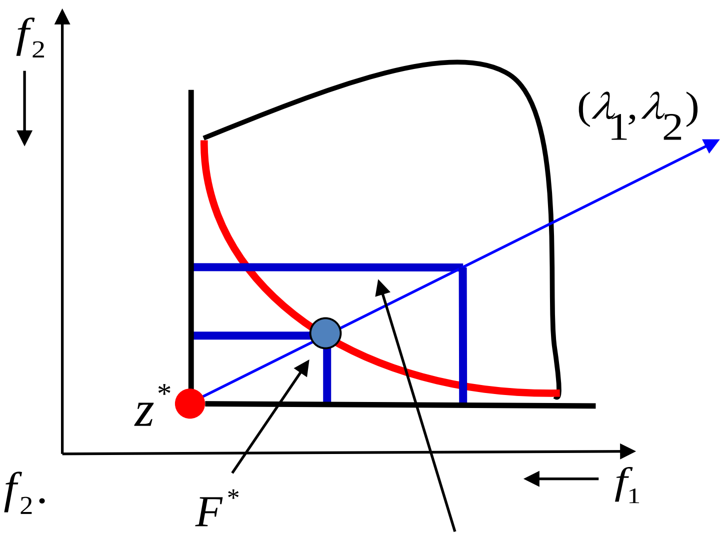
Problem Decomposition

- **Tchbycheff Approach**
  - For any Pareto optimal solution $\mathbf{x}^*$, there is a weight vector such that $\mathbf{x}^*$ is optimal to the problem.

$$\min g^T(x \mid \lambda, z^*)$$

$$g^T(x \mid \lambda, z^*) = \max\{\lambda_1 \mid f_1(x) - z_1^* \mid, \lambda_2 \mid f_2(x) - z_2^* \mid\}$$

$$z^* = (z_1^*, z_2^*) \text{ is an Utopian point. } z_1^* < \min f_1,\ z_2^* < \min f_2.$$

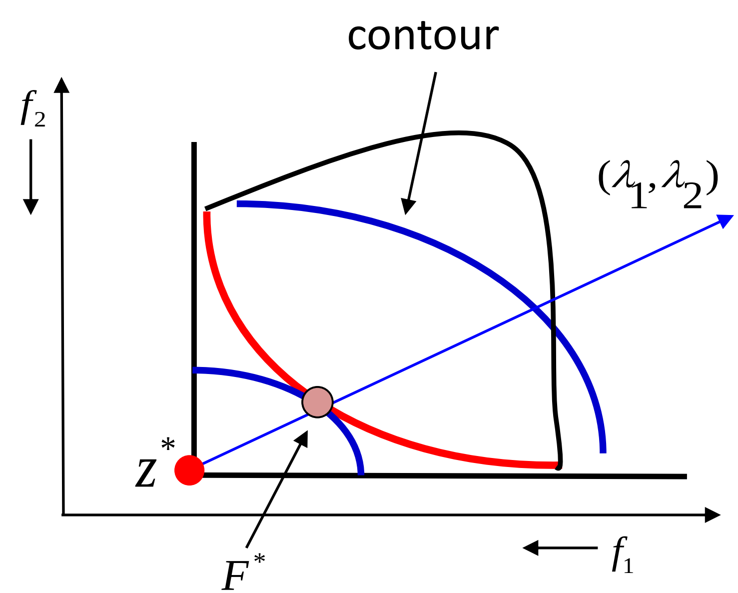


- **Weighted L_p Approach**

$$\min g(x \mid \lambda,\ z^*)$$

$$\text{where } g(x \mid \lambda, z^*) = \{\lambda_1 \mid f_1(x) - z_1^* \mid^p + \lambda_2 \mid f_2(x) - z_2^* \mid^p\}^{\frac{1}{p}}$$

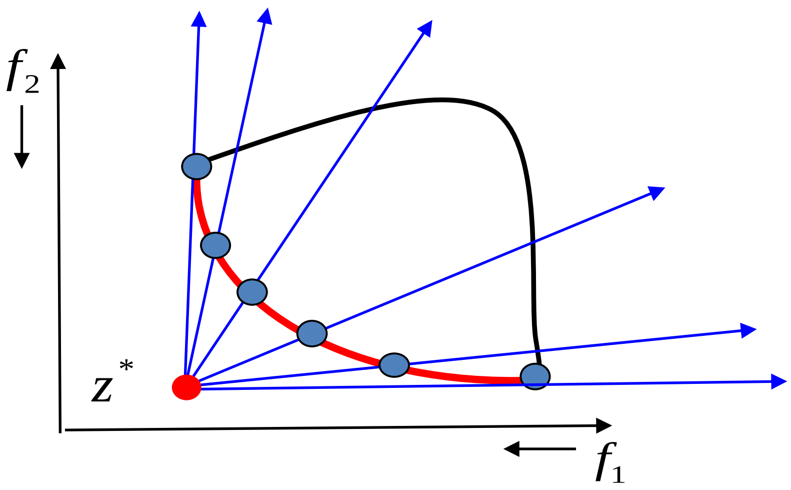
$$z^* = (z_1^*, z_2^*) \text{ is an Utopian point.}$$

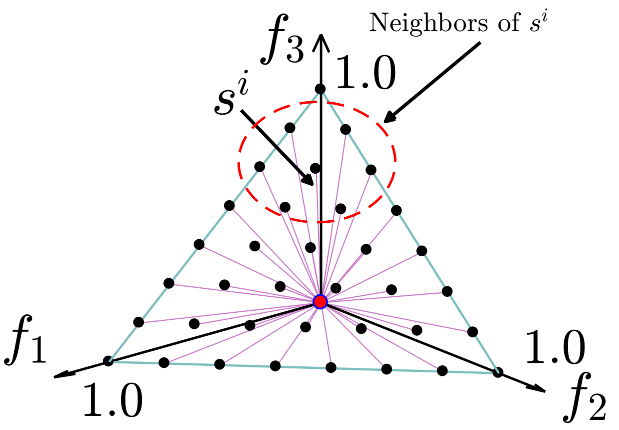


✓ $p = 1$: weighted sum
✓ $p = \infty$: Tchbycheff

Weight Vectors and Utopian Points

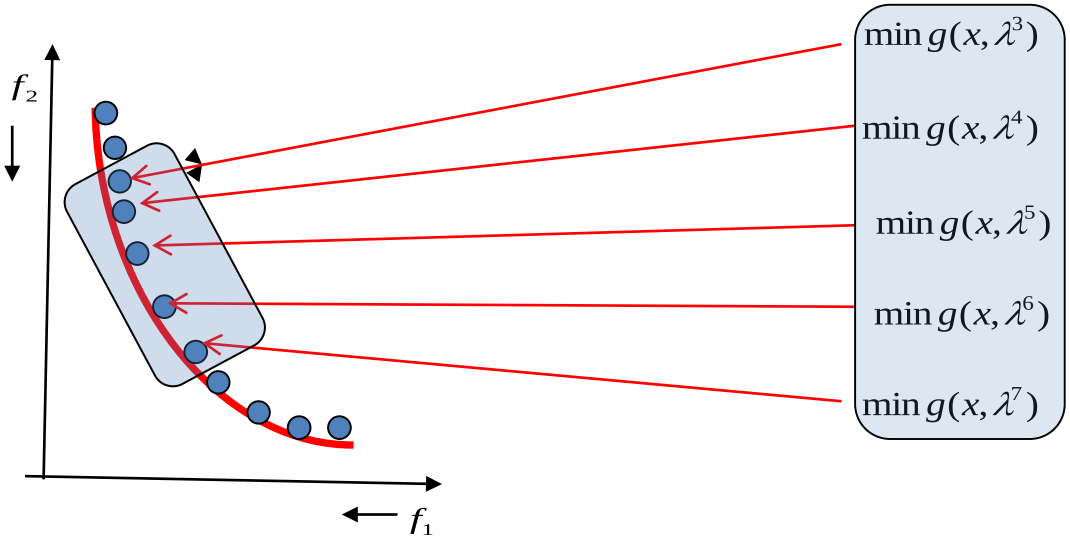
- A simple way for weight vectors:
  - All the weight vectors are uniformly distributed in the unit simplex
  - $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m)$, $\sum_{k=1}^{m} \lambda_k = 1$, $0 \leq \lambda_k \leq 1$
  - Each weight vector is a direction line
- Utopian points: problem specific, based on the estimation of the true Pareto front





Neighbourhood Structure

- Sub-problems with similar objective functions (weight vectors, direction lines) are most likely to have similar optimal solutions
- Define neighbourhoods based on similarity of weight vectors



Neighborhood of subproblem 5

Search Method

- **Memory**: Each agent i records only one candidate solution $\mathbf{x}^i$, the best solution found so far for its sub-problem
- At each generation, for each agent:
  - **Mating selection**: Borrow solutions from neighbours (collaboration)
  - **Breeding**: generate a new solution from the mating pool (parent selection + crossover/mutation)
  - **Replacement**:
    - Evaluate the new solution on its own objective
    - Replace its old solution by the new solution if the new one is better for its own objective
    - Pass the new solution to some of its neighbours (collaboration, neighbourhood)
      - For each neighbour receiving the new solution, evaluate the new solution on the objective of the neighbour
      - Replace the old solution of the neighbour with the new solution, if the new one is better for the neighbour's objective

MOEA/D (Original Form)

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) Initialization:

Step 1.1) Set $\mathrm{EP} = \emptyset$.

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.

Step 2) Update:

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 $g^{te}(y'|\lambda^j, z) \leq g^{te}(x^j|\lambda^j, z)$, then set $x^j = y'$ and $FV^j = F(y')$.

Step 2.5) Update of EP:

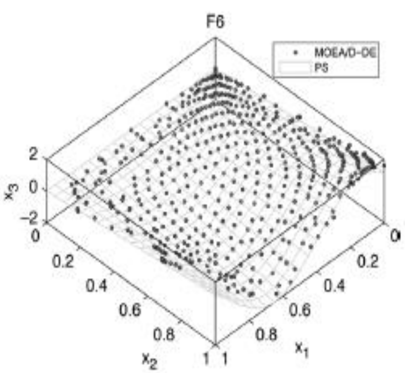
Remove from EP all the vectors dominated by $F(y')$.

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

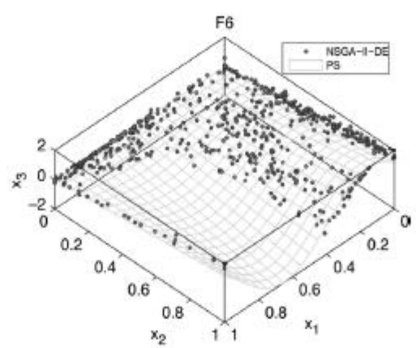
Step 3) Stopping Criteria: If stopping criteria is satisfied, then stop and output EP. Otherwise, go to **Step 2**.

MOEA/D

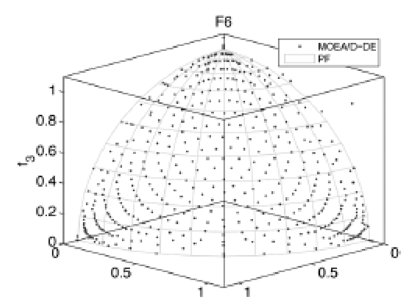
- Advantages
  - Very good in controlling search direction, can get very well distributed distributions
  - Supports parallel computing
- Issues
  - Hard to set weight vectors and Utopian points (sensitive parameters)
  - Decomposition is problem-specific (weighted sum, Tchbycheff, …)
  - Difficulty to handle discontinuous, irregular shape of Pareto front



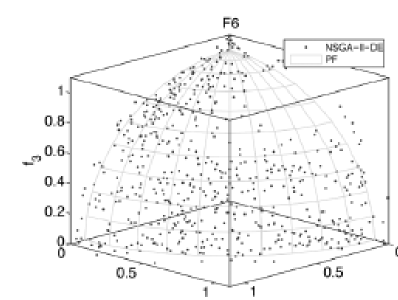
MOEA/D



NSGA-II



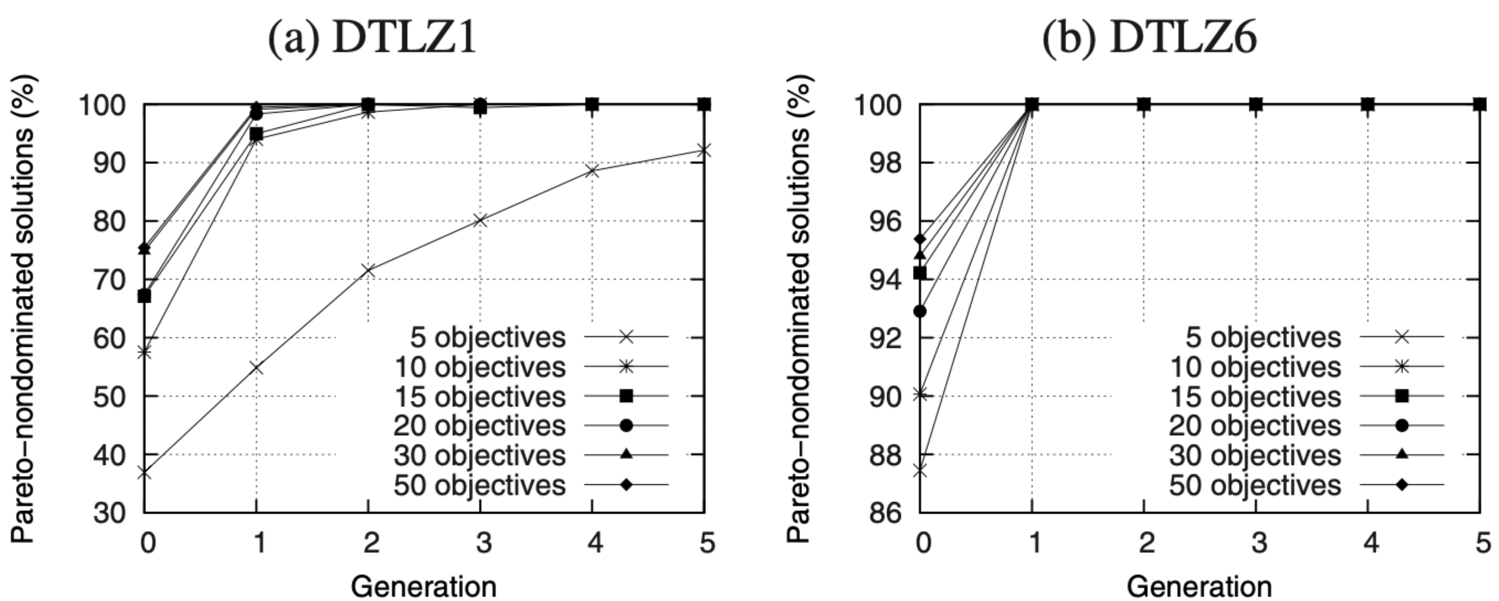
MOEA/D



NSGA-II

Many-Objective Optimisation

- Multi-objective optimisation problems can become increasingly hard when the number of objectives increases
  - More and more solutions become non-dominated
  - Calculation of diversity measures becomes time consuming
  - Crossover may become less effective (parents are more distant)



Population size = 100

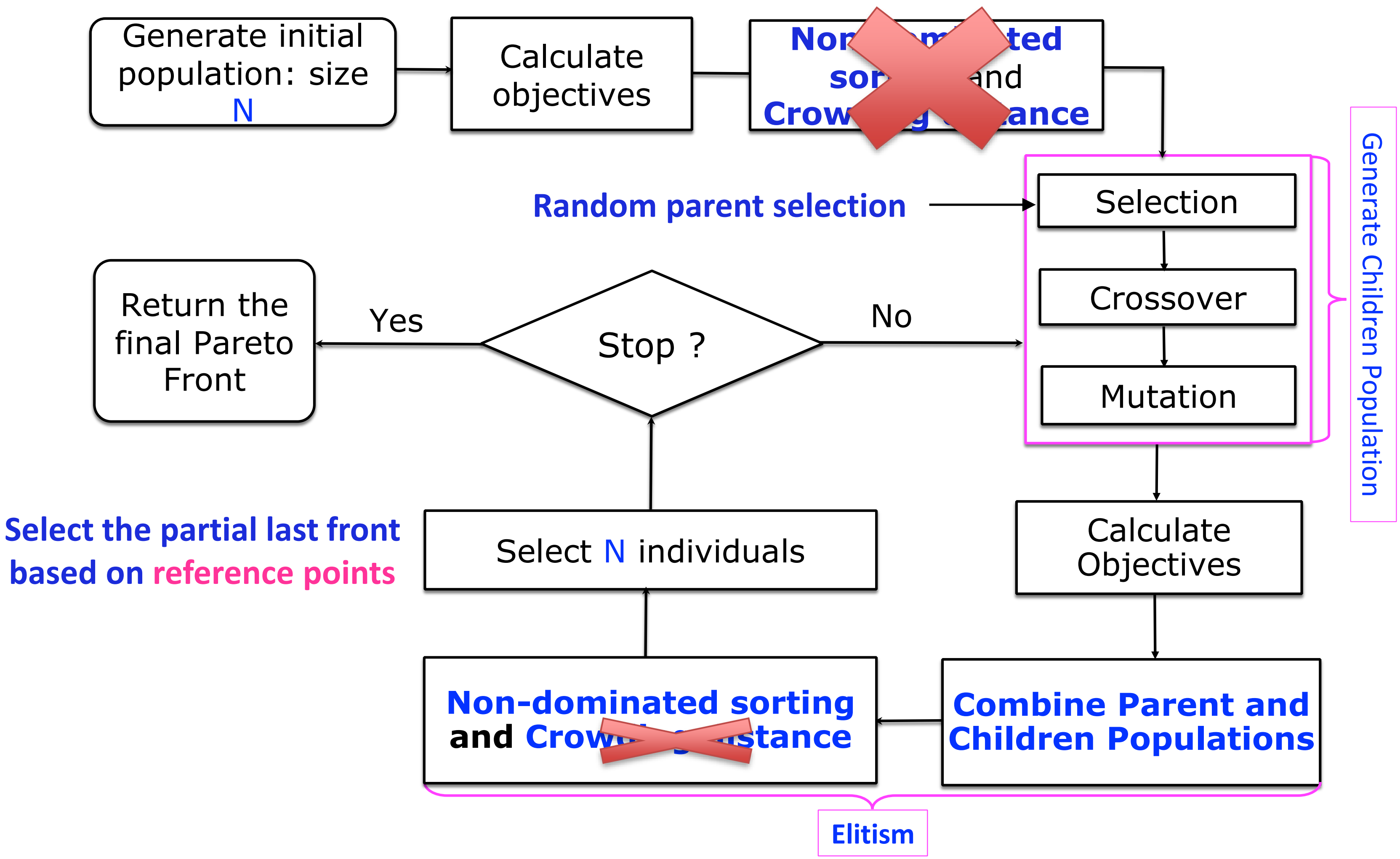
Many-Objective Optimisation

- To distinguish, many-objective optimisation problems are the multi-objective optimisation problems with four or more objectives
  - Larger than 3D, hard to visualize
  - Sufficient difficulty so that common MOEAs are less effective

- Preprocessing to reduce the number of objectives:
  - Remove redundant and less important objectives
  - Use domain knowledge to aggregate objectives
  - If still many objectives, then consider many-objective evolutionary algorithms: a well-known algorithm is NSGA-III (the version after NSGA-II)

NSGA-III

- Based on the NSGA-II framework
- Borrow MOEA/D idea on search direction, NO crowding distance



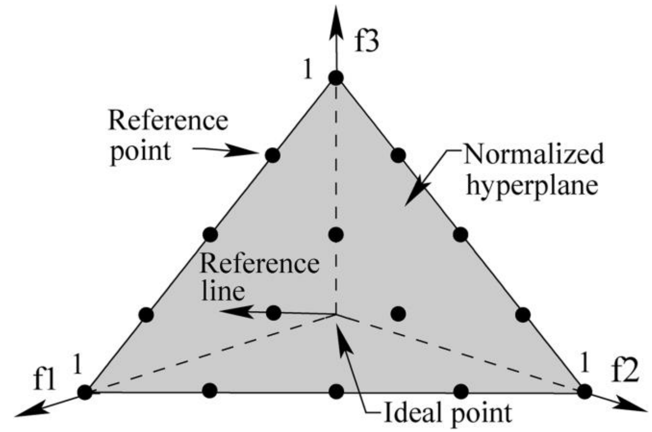
NSGA-III

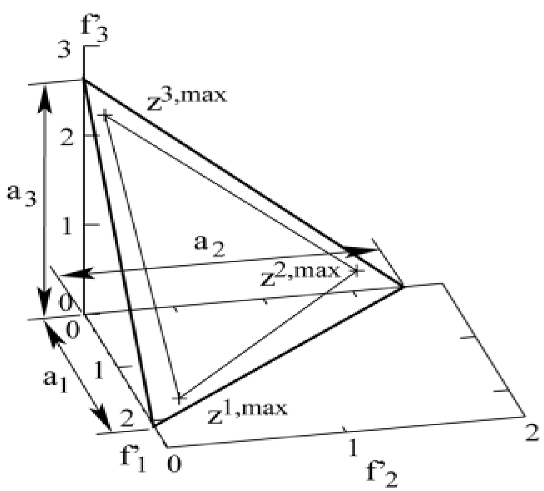
Algorithm 1 Generation t of NSGA-III procedure

Input: H structured reference points Z^s or supplied aspiration points Z^a, parent population P_t

Output: P_{t+1}

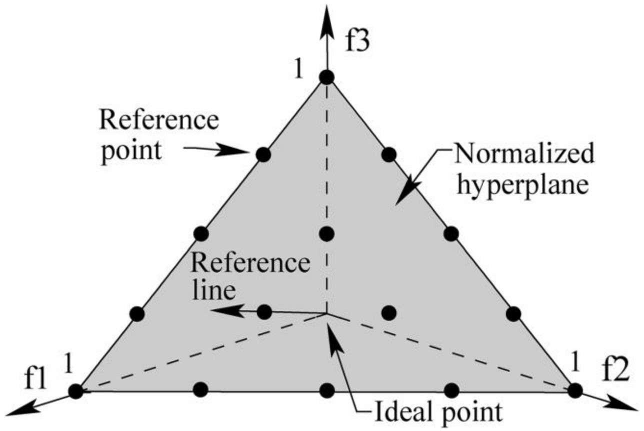
$S_t = \emptyset$, $i = 1$
Q_t = Recombination+Mutation(P_t)
$R_t = P_t \cup Q_t$
$(F_1, F_2, \ldots)$ = Non-dominated-sort(R_t)
repeat
    $S_t = S_t \cup F_i$ and $i = i + 1$
until $|S_t| \geq N$
Last front to be included: $F_l = F_i$
if $|S_t| = N$ **then**
    $P_{t+1} = S_t$, break
else
    $P_{t+1} = \cup_{j=1}^{l-1} F_j$
    Points to be chosen from F_l: $K = N - |P_{t+1}|$
    Normalize objectives and create reference set Z^r: `Normalize(`$\mathbf{f}^n, S_t, Z^r, Z^s, Z^a$`)`
    Associate each member $\mathbf{s}$ of S_t with a reference point: $[\pi(\mathbf{s}), d(\mathbf{s})]$ =`Associate(`S_t, Z^r`)`   % $\pi(\mathbf{s})$: closest reference point, d: distance between $\mathbf{s}$ and $\pi(\mathbf{s})$
    Compute niche count of reference point $j \in Z^r$: $\rho_j = \sum_{\mathbf{s} \in S_t/F_l} ((\pi(\mathbf{s}) = j) \; ? \; 1 \; : \; 0)$
    Choose K members one at a time from F_l to construct P_{t+1}: `Niching(`$K, \rho_j, \pi, d, Z^r, F_l, P_{t+1}$`)`
end if

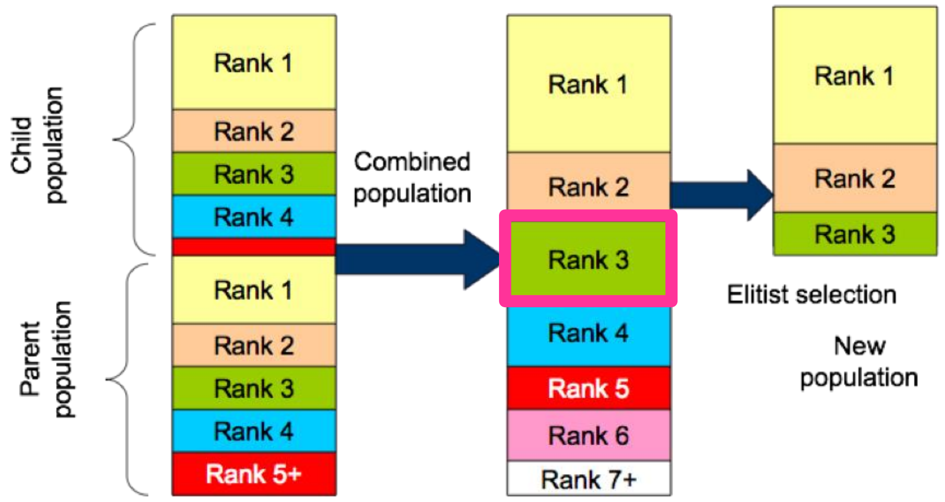


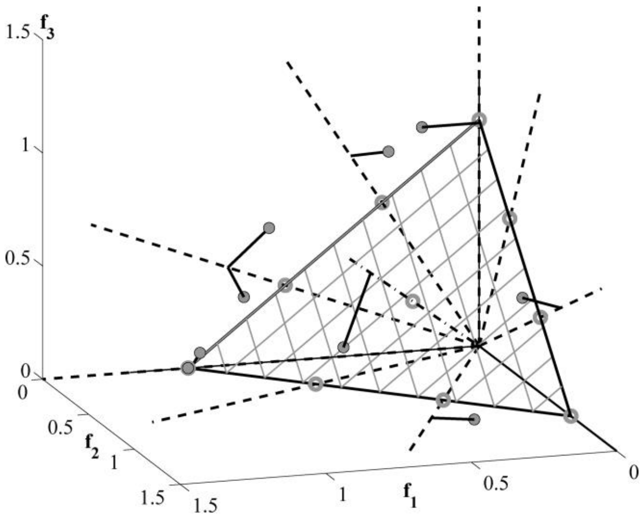


Partial Last Front Selection

- Normalise solution objectives and generate reference points (search directions)
  - Ideal point, uniformly distributed reference points
- Associate each previously selected solution to the closest reference points
  - Perpendicular distance to the reference line
- Select the partial last front by niching



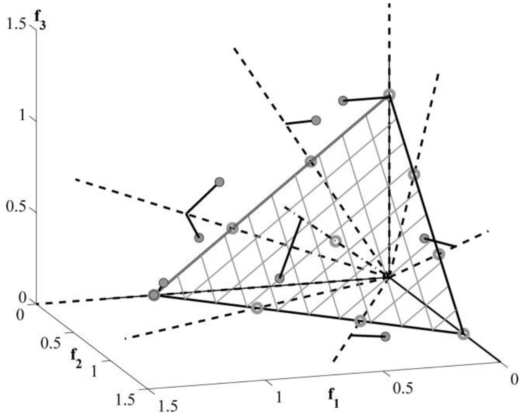




Partial Last Front Selection

- Select the partial last front by niching
- **Repeat K times** (K is the number of individuals needed)
  - Select the reference point with the least associated individuals
  - If some individuals in the last front associated to this reference point
    - If there is NO current associated individuals to this ref point, then add the closest individual in the last front associated to the ref point
    - Otherwise, randomly add an individual in the last front associated to it
  - If NO individual in the last front associated to this reference point
    - Remove this reference point

- #reference points can change during NSGA-III



- NSGA-III vs NSGA-II?
- NSGA-III vs MOEA/D?

Other EMO Topics

- Multi-objective Constrained Optimisation
- Preference-base EMO
- EMO + Decision Making, Interactive MOEAs
- Dynamic multi-objective optimisation
- Large scale multi-objective optimisation
- Irregular Pareto front
- Different objectives have very different distributions

Summary

- MOEA/D
  - Decompose into single-objective sub-problems, each searching for one point on the Pareto front
  - Collaborate with each other
  - Can control the distribution well if the Pareto front is well shaped
  - But not good if the Pareto front has irregular shape
- Many-objective optimisation: more than 3 objectives
- NSGA-III: combine NSGA-II and MOEA/D
  - Make distribution more well spread and uniform
  - Can incorporate preference by adjusting the reference points