Evolutionary multi-objective optimization (EMO)

Cosijopii García García
Cosijopii@inaoep.mx
https://cosijopiii.github.io/teaching/

Computer Science Lab, INAOE

April 26, 2023

The following slides take figures and information from :

- From my Master thesis : **A Cellular Evolutionary Algorithm To Tackle Constrained Multiobjective Optimization Problems**.
- Carlos A. Coello Coello, David A. Van Veldhuizen and Gary B. Lamont, **Evolutionary Algorithms for Solving Multi-Objective Problems**, Kluwer Academic Publishers, New York, March 2002, ISBN 0-3064-6762-3.
- Kalyanmoy Deb. **Multi-Objective Optimization using Evolutionary Algorithms**, UK, 2001, ISBN 0-471-87339-X.
- **Decomposition Multi-Objective Optimisation Tutorial** `https://doi.org/10.1145/3377929.3389865`
- **Coello Tutorial**: `https://www.cs.cinvestav.mx/~emooworkgroup/tutorial-slides-coello.pdf`
- **Zitzler Tutorial**: `https://www.cs.cinvestav.mx/~emooworkgroup/tutorial-slides-zitzler.pdf`
- **Sudhoff MOP slides**: `https://engineering.purdue.edu/~sudhoff/ee630/Lecture09.pdf`
- **SBX**: `http://portal.acm.org/citation.cfm?doid=1276958.1277190`
- **CHT's**: `https://www2.cs.uh.edu/~ceick/6367/Coello_CHNOPT.pdf`

Multiobjective Optimization

Why Multiobjective Optimization?

Most optimization problems naturally have several objectives to be achieved (normally conflicting with each other), but in order to simplify their solution, they are treated as if they had only one (the remaining objectives are normally handled as constraints)

Basic concepts

- **The Multiobjective Optimization Problem** can then be defined (in words) as the problem of finding:
- *"A. **vector of decision variables** which satisfies **constraints** and optimizes a **vector function** whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in **conflict** with each other. Hence, the term "optimize" means finding such a solution which would give the values of all the objective functions acceptable to the decision maker.*

Basic concepts

$$\left.\begin{aligned} \text{Minimize/Maximize } \mathbf{f}_m(\mathbf{x}), m = 1, 2, \ldots, k; \\ \text{subject to } g_j(\mathbf{x}) \geq 0, j = 1, 2, \ldots, m; \\ h_k(\mathbf{x}) = 0, k = 1, 2, \ldots, p; \\ x_i^{(L)} \leq x_i \leq x_i^{(U)}, i = 1, 2, \ldots, t; \end{aligned}\right\} \quad (1)$$

- with k objectives, m and p are the number of inequality and equality constraints. A solution $\mathbf{x} \in \mathbf{R}^n$ is a vector of n decision variables: $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, which satisfy all constraints and variable bounds.

Basic concepts

Pareto

- Having several objective functions, the notion of "*optimum*" changes, because in MOPs, we are really trying to find good compromises (or "trade-offs") rather than a single solution as in global optimization. The notion of "*optimum*" that is most commonly adopted is that originally proposed by Francis Ysidro Edgeworth in 1881.
- This notion was later generalized by Vilfredo Pareto (in 1896). Although some authors call **Edgeworth-Pareto optimum** to this notion, we will use the most commonly accepted term: **Pareto optimum.**

Basic concepts

Pareto Optimality

- A solution $\mathbf{x} \in \Omega$ is said to be **Pareto Optimal** with respect to (w.r.t.) Ω if and only if (*iff*) there is no $\mathbf{x}' \in \Omega$ for which $v = F(\mathbf{x}') = (f_1(\mathbf{x}'), ..., f_k(\mathbf{x}'))$ dominates $u = F(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_k(\mathbf{x}))$.
  The phrase Pareto Optimal is taken to mean with respect to the entire decision variable space unless otherwise specified.

Basic concepts

Pareto Dominance and Pareto Optimal Set

- A vector $\mathbf{u} = (u_1, ..., u_k)$ is said to **dominate** another vector $\mathbf{v} = (v_1, ..., v_k)$ (denoted by $u \preceq v$) if and only if $\mathbf{u}$ is partially less than $\mathbf{v}$, i.e., $\forall i \in \{1, ..., k\}, u_i \leq v_i \wedge \exists i \in \{1, ..., k\} : u_i < v_i$.

- For a given MOP, $F(\mathbf{x})$, the **Pareto Optimal Set**, $\mathcal{P}^*$, is defined as:

$$\mathcal{P}^* = \{\mathbf{x} \in \Omega \mid \neg\exists \mathbf{x}' \in \Omega \, F(\mathbf{x}') \preceq F(\mathbf{x})\} \tag{2}$$

Basic concepts

Non-dominated solutions

- In words, this definition says that $\mathbf{x}'$ is Pareto optimal if there exists no feasible vector of decision variables $\mathbf{x} \in \mathcal{F}$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion. Unfortunately, this concept almost always gives not a single solution, but rather a set of solutions called the **Pareto optimal set**. The vectors $\mathbf{x}'$ corresponding to the solutions included in the Pareto optimal set are called **non-dominated**. The plot of the objective functions whose non-dominated vectors are in the Pareto optimal set is called the **Pareto front**.

$$\mathcal{PF}^* = \{\mathbf{u} = F(\mathbf{x}) \mid \mathbf{x} \in \mathcal{P}^*\} \tag{3}$$

Decision and Objective Space

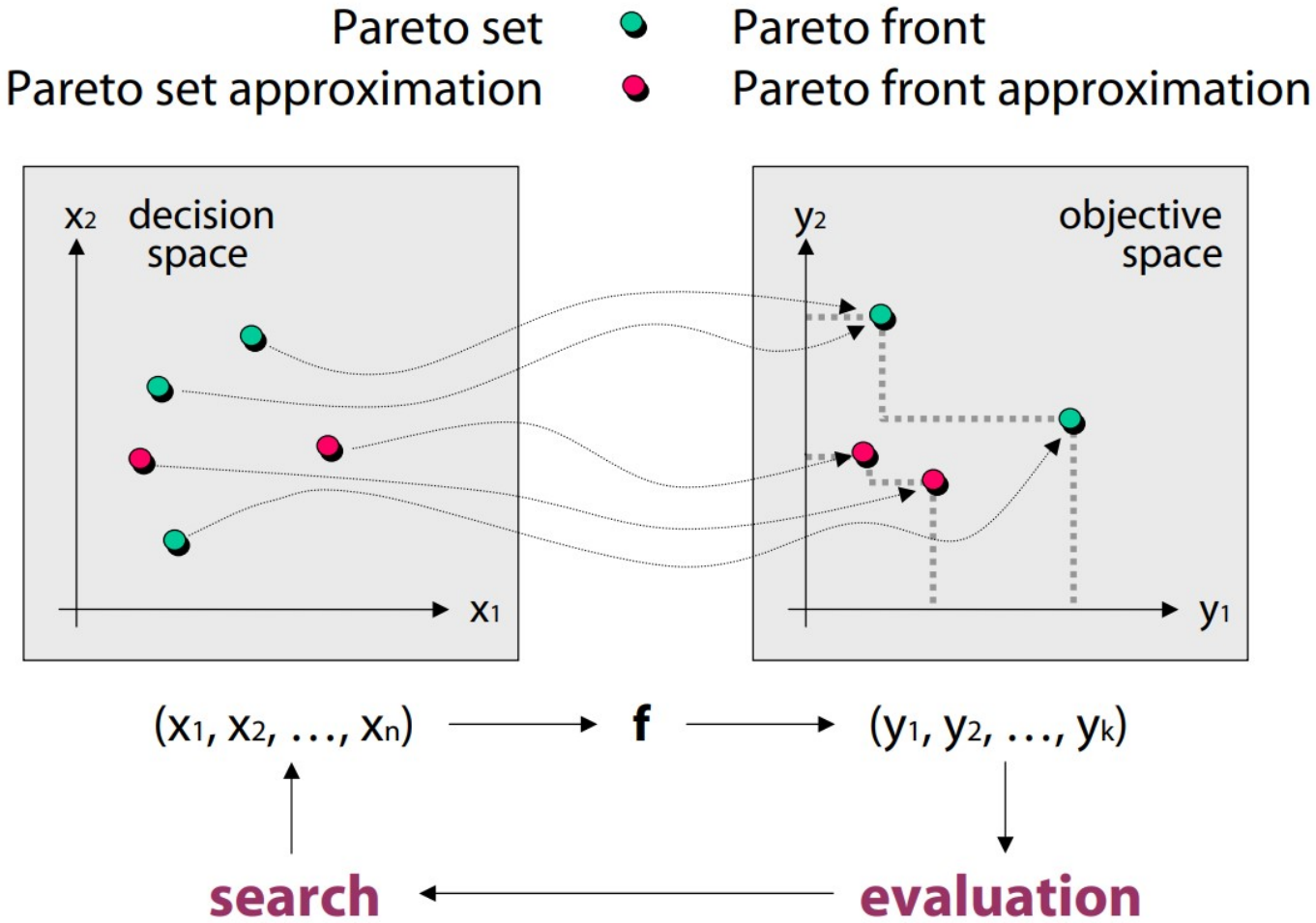


Figure: Decision variables space and objective function space.

Basic concepts

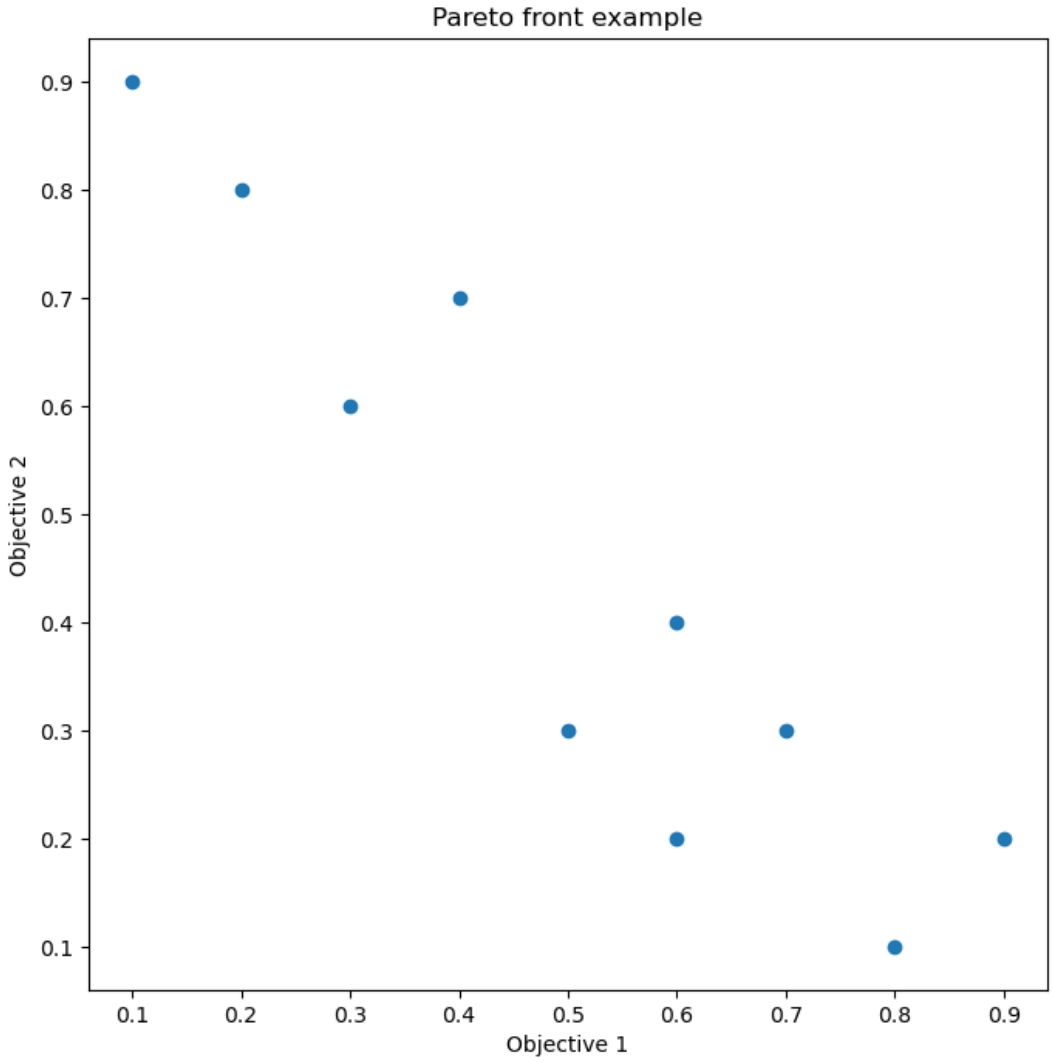
Example

Let's consider the following list of points:

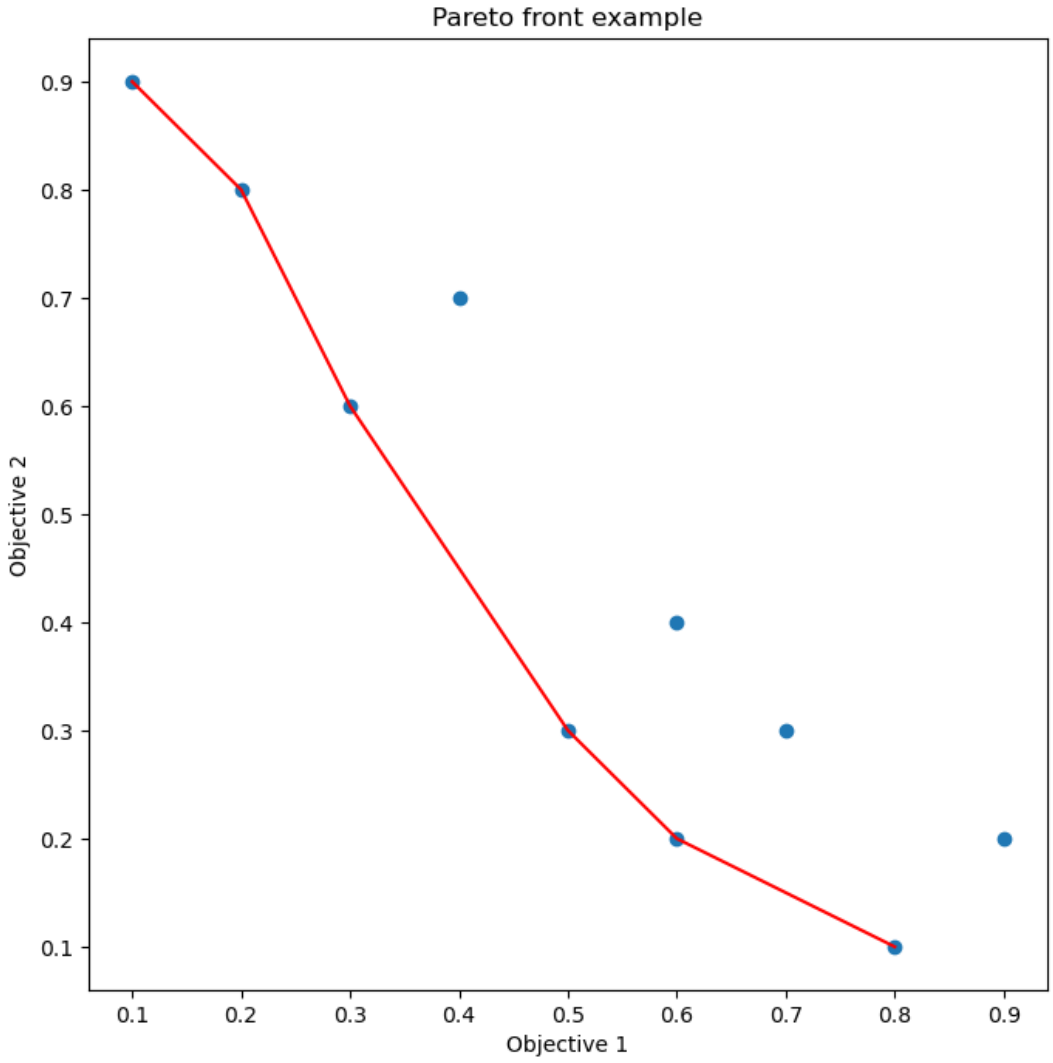
Point	Coordinates
1	(0.2, 0.8)
2	(0.3, 0.6)
3	(0.5, 0.3)
4	(0.4, 0.7)
5	(0.6, 0.2)
6	(0.8, 0.1)
7	(0.9, 0.2)
8	(0.7, 0.3)
9	(0.6, 0.4)
10	(0.1, 0.9)

In this example, points **1, 2, 3, 5, 6**and **10** are part of the Pareto front, while points **7, 8, 9,** and **4** are dominated by other points.

Cont.


Pareto front example
Objective 2
Objective 1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

Cont.

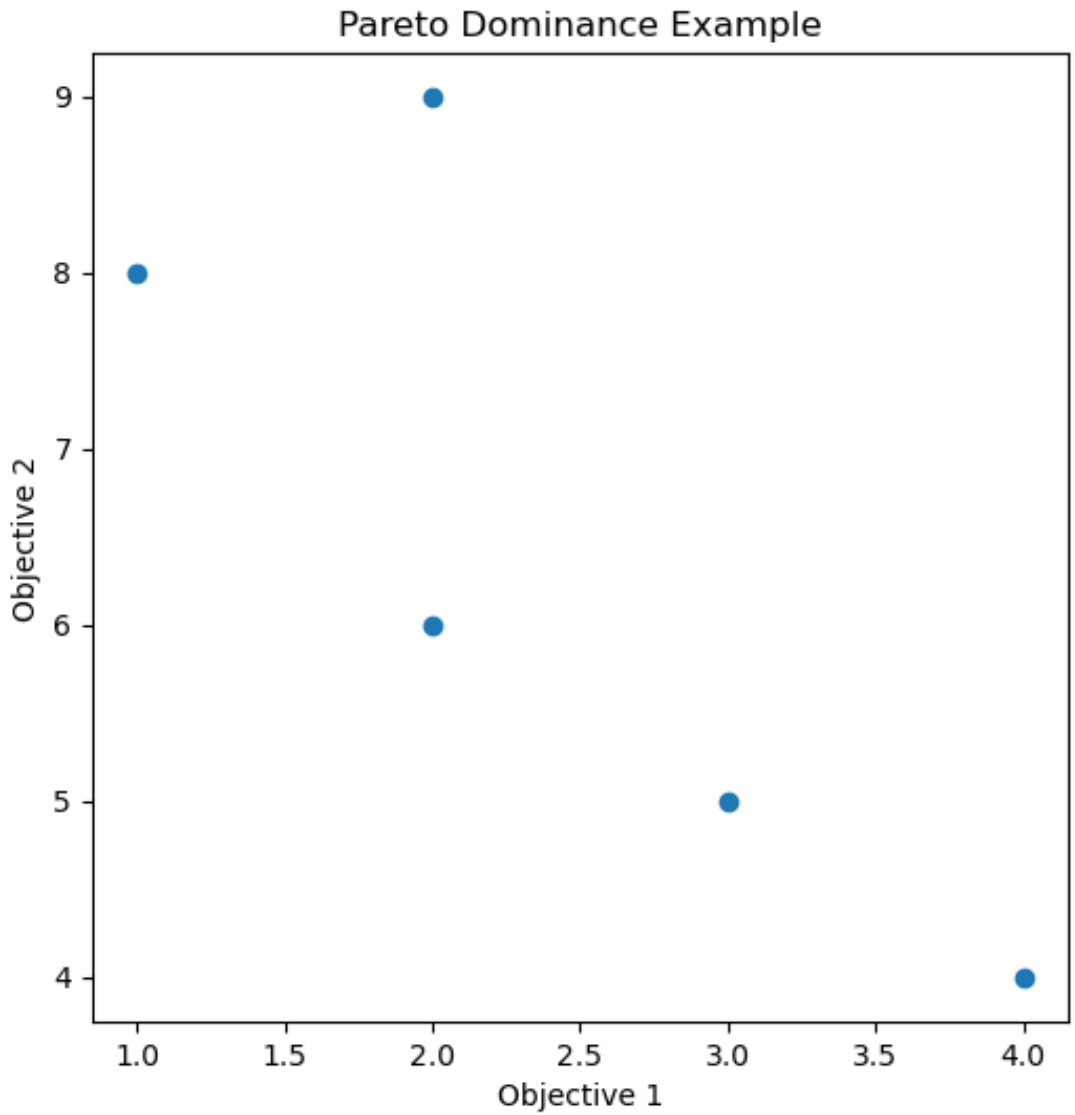

Pareto front example
Objective 2
Objective 1
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

Exercise.

Given the following points, what is the dominated solution?

$$A = [2, 6]$$
$$B = [1, 8]$$
$$C = [3, 5]$$
$$D = [4, 4]$$
$$E = [2, 9]$$

Cont.


Pareto Dominance Example
Objective 2
Objective 1
9
8
7
6
5
4
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Basic concepts

Goals in MOO

- Find set of solutions as close as possible to Pareto optimal front
- To find a set of solutions as diverse as possible

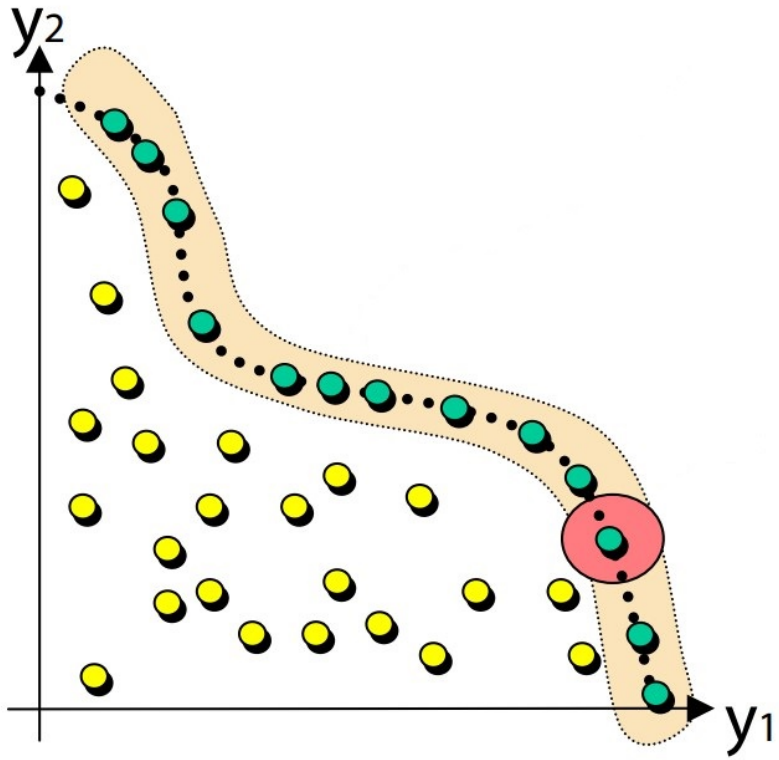


Figure: Pareto Front.

Special solutions

Three types of special solutions widely used in multiobjective optimization algorithms are explained. These are **ideal**, **utopian**, and **nadir** objective vectors

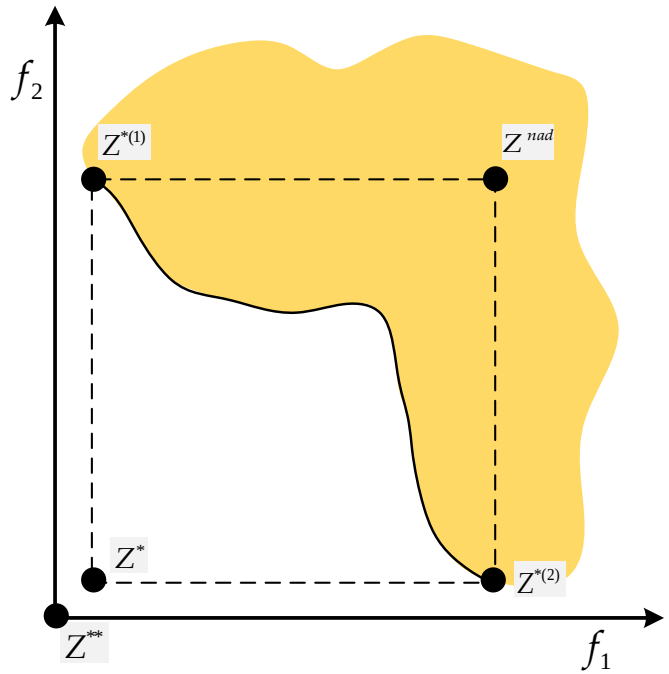


Figure: Representation of an objective function space with ideal (Z^*), utopian (Z^{**}), and nadir (Z^{nad}) objective vectors.

Special Solutions

Cont.

- **Ideal** objective vector:
  $Z^* = (Z_1^*, \ldots, Z_m^*)$ where $Z_i^* = \min f_i(\mathbf{x}) | \mathbf{x} \in P$

- In a **Utopian** objective vector Z^{**} each component is sightly smaller than the ideal objective vector, or $Z_i^{**} = Z_i^* - \epsilon_i$ with $\epsilon > 0$ for all $i = 1, 2, \ldots, M$

- **Nadir** objective vector:
  $Z^{nad} = (z_1^*, \ldots, z_m^*)$ where $n_i^* = \max f_i(\mathbf{x}) | \mathbf{x} \in P$

Why Use Evolutionary Algorithms?

- **Population approach** suits well to find multiple solutions
- **Niche-preservation methods** can be exploited to find diverse solutions
- **Implicit parallelism** helps provide a parallel search

Classifying Techniques

We will use the following simple classification of Evolutionary Multi-Objective Optimization (EMO) approaches:

- Classical Techniques (***Homework***: *read about Classical Techniques like Goal programming or weight sum method*)
- Pareto-based Techniques
- Decomposition-based Techniques
- Indicator-based Techniques

Before MOEAs

new evolutionary operators

- Simulated Binary Crossover (SBX)
- Polynomial Mutation (PM)

EVOPs

Simulated Binary Crossover (SBX)

- As the name suggests, the SBX operator, simulates the working principle of the single-point crossover operator on binary strings.
- The procedure of computing the offspring $x_i^{(1,t+1)}$ and $x_i^{(2,t+1)}$ from the parent solutions $x_i^{(1,t)}$ and $x_i^{(2,t)}$:
  - Choose a random number $u \in [0, 1)$.
  - Calculate β_q using:

$$\beta_{qi} = \begin{cases} (2u_i)^{\frac{1}{\eta+1}} & \text{if } u_i < 0.5 \\ \left(\frac{1}{2(1-u_i)}\right)^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} \tag{4}$$

where η is the distribution index which should be a non-negative number. Large η values increase the probability of creating near-parent solutions and small values allow to select distant values to generate the offspring.

EVOPs

Simulated Binary Crossover (SBX)

- Finally, offspring are calculated using the following equations:

$$x_i^{(1,t+1)} = 0.5 \left[\left(1 + \beta_{qi}\right) x_i^{(1,t)} + \left(1 - \beta_{qi}\right) x_i^{(2,t)} \right] \tag{5}$$

$$x_i^{(2,t+1)} = 0.5 \left[\left(1 + \beta_{qi}\right) x_i^{(1,t)} + \left(1 - \beta_{qi}\right) x_i^{(2,t)} \right] \tag{6}$$

EVOPs

Polynomial Mutation (PM)

- Similar to SBX recombination operator, in polynomial mutation the probability distribution can be a polynomial function instead of a normal distribution.

1. First, choose a random number $u \in [0, 1]$.
2. Finally, the following equation applies:

$$p' = \begin{cases} p + \overline{\delta_L}\left(P - x_i^{(L)}\right) & \text{for } u \leq 0.5 \\ p + \overline{\delta_R}\left(x_i^{(U)} - P\right) & \text{for } u > 0.5 \end{cases} \tag{7}$$

where $\overline{\delta_L}$ and $\overline{\delta_R}$ are calculated as follow:

$$\overline{\delta_L} = (2u)^{\frac{1}{(1+\eta_m)}} - 1, \text{for } u \leq 0.5 \tag{8}$$

$$\overline{\delta_R} = 1 - (2(1-u))\frac{1}{1+\eta_m}, \text{for } u > 0.5 \tag{9}$$

Pareto-based Multi-Objective evolutionary algorithms (MOEAs)

- Pareto-based MOEAs use a dominance based ranking scheme and combine elitist strategies such those that converge to a global optimal in some problems.
- Pareto and elitist strategies lead the way or set the basis for one of the most important algorithmic approaches in the area: **NSGA-II** algorithm proposed by Deb et. al in 2002.
- Pareto based MOEAs have in common the use of Pareto dominance with some diversity criteria based on secondary ranking, some algorithms of this class are MOGA, PAES, SPEA and SPEA-2

NSGA-II

- Non-dominated sorting Genetic Algorithm (NSGA-II) was proposed by Deb et. al. in 2002 among its main characteristics is the use of elitism, as well as the use of a mechanism of diversity and focus on non-dominated solutions.
- Important points: **Non-dominated sorting** and **crowding distance** (Niche-preservation)

NSGA-II

Algorithm 1: NSGA-II

Create initial population P_t;
Evaluate fitness of each solution;
Apply non-dominated sorting to rank the solutions;
while *Termination condition not satisfied* **do**
    Offspring population $Q_t = \emptyset$;
    for *each solution in* P_t **do**
        Select two parents using binary tournament;
        Recombine the parents using SBX and generate a child r;
        Apply mutation on r generating q;
        $Q_t = Q_t \cup q$;
    $R_t = P_t \cup Q_t$;
    Apply Algorithm NDS to rank R_t population: $F = NDS(R_t)$ obtaining F fronts;
    $P_{t+1} = \emptyset$ and $i = 1$;
    Until $|P_{t+1}| + |F_i| \leq N$
        Apply Algorithm Crowding distance to F_i: CD(F_i);
        $P_{t+1} = P_{t+1} \cup F_i$;
        $i = i + 1$;
    Sort($F_i, \prec_n$);
    $P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$;

NSGA-II

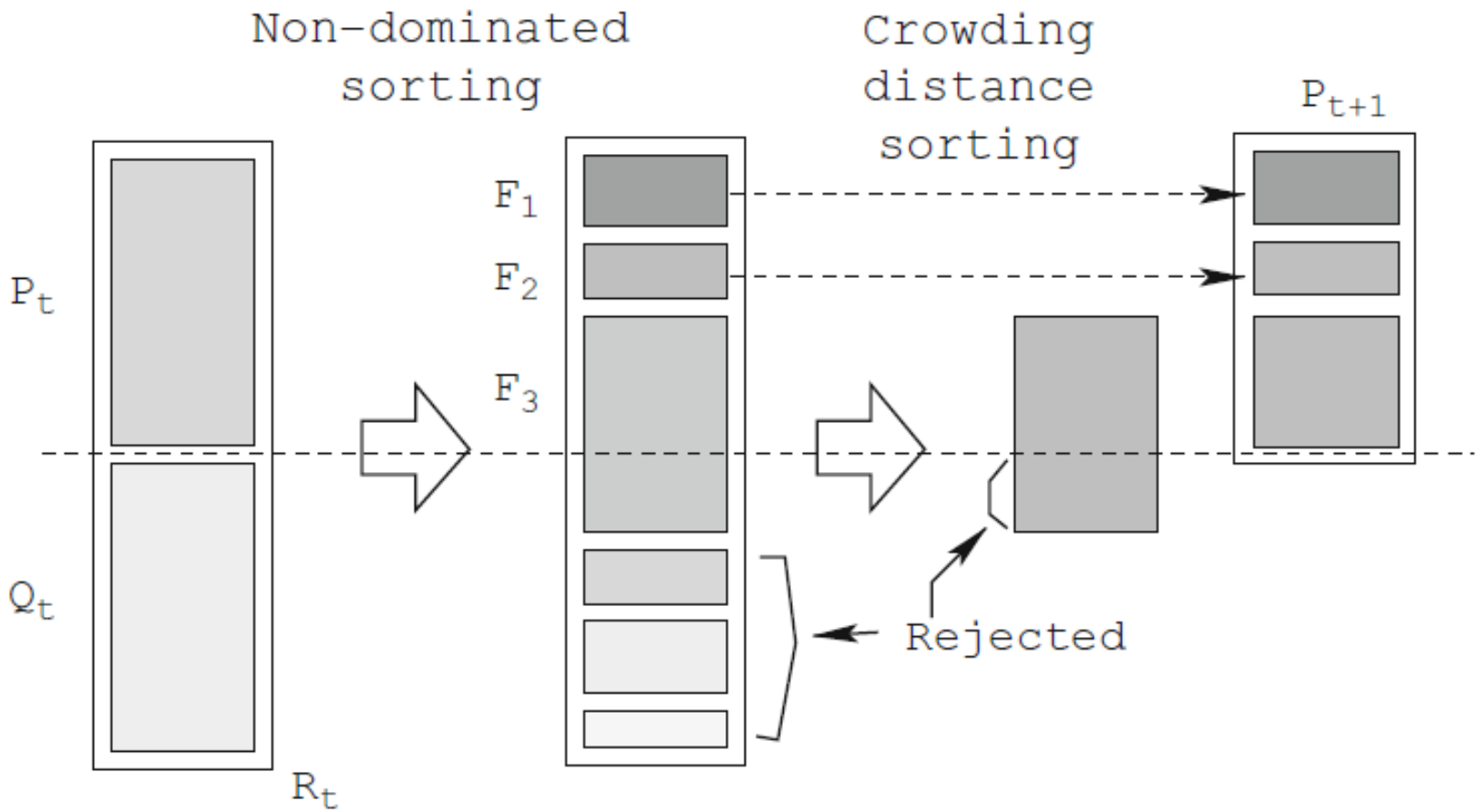


Figure: NSGA-II, process

NSGA-II, NDS

Algorithm 2: Non-dominated sorting

```
1  for each p ∈ Population P do
2      for each q ∈ P do
3          if p ≺ q then
4              S_p = S_p ∪ {q};
5          else
6              if q ≺ p then
7                  n_p = n_p + 1;
8          if n_p = 0 then
9              p_rank = 1;
10             F_1 = F_1 ∪ {p};
11 while F_i ≠ ∅ do
12     Q = ∅;
13     for each p ∈ F_i do
14         for each q ∈ S_p do
15             n_q = n_q − 1;
16             if n_q = 0 then
17                 q_rank = i + 1;
18                 Q = Q ∪ {q};
19     i = i + 1;
20     F_i = Q;
```

NSGA-II, CD

Algorithm 3: Crowding Distance

1 $\alpha = |D|$; // number of solutions in D
2 **for** *each i* **do**
3 $D[i]_{distance} = 0$
4 **for** *each objective m* **do**
5 $D = sort(D, m)$; // sort using each objective value
6 $D[1]_{distance} = D[\alpha]_{distance} = \infty$; // so that boundary points are always selected
7 $i = 2$ to $(\alpha - 1)$ $D[i]_{distance} = D[i]_{distance} + (D[i+1].m - D[i-1].m)/(f_m^{max} - f_m^{min})$;

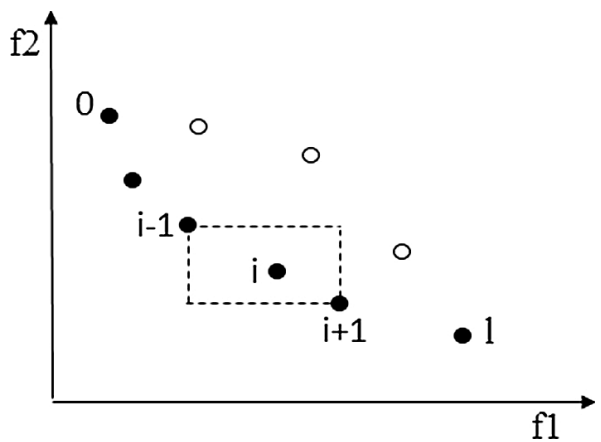


Figure: Cuboid, Crowding-distance calculation.

NSGA-II, example

https://github.com/Cosijopiii/SENIAC-2022-CE/tree/main/MOO

Constraint handling techniques

- Constraint dominance principle (CDP)
- Penalty solutions
- Stochastic Ranking
- $\epsilon-$Constrained

Constraint handling techniques

Constraint dominance principle

- Constraint dominance principle (CDP) is proposed by Deb[2000]. It is a popular and widely used constraint-handling approach. The constraint-handling approach of CDP is defined as follows:
- A solution x_i is said to constrained-dominate a solution x_j , if any of the following conditions is true.
  1. Solution x_i is feasible and solution x_j is infeasible.
  2. Solution x_i and x_j are both infeasible, but solution x_i has a smaller constraint violation than solution x_j.
  3. Solution x_i and x_j are both feasible and solution x_i dominates x_j .

Constraint handling techniques

Penalty solutions

- The most common approach in the EA community to handle constraints (particularly, inequality constraints) is to use penalties. Penalty functions were originally proposed by Richard Courant in the 1940s and were later expanded by Carroll and Fiacco & McCormick.
- The idea of penalty functions is to transform a constrained optimization problem into an uncontrained one by adding (or subtracting) a certain value to/from the objective function based on the amount of constraint violation present in a certain solution.

Constraint handling techniques

Penalty solutions

- EAs normally adopt external penalty functions of the form:

$$\phi(x) = f(x) \pm \left[\sum_{i=1}^{n} r_i \times G_i + \sum_{j=1}^{p} c_j \times L_j \right] \tag{10}$$

- where $\phi(x)$ is the new (expanded) objective function to be optimized, G_i and L_j are functions of the constraints $g_i(x)$ and $h_j(x)$, respectively, and r_i and c_j are positive constants normally called "penalty factors".

Constraint handling techniques

Penalty solutions

- The most common form of G_i and L_j is:

$$G_i = max[0, g_i(x)] \tag{11}$$

$$L_i = |h_j(x)| \tag{12}$$

- Penalty functions can deal both with equality and inequality constraints, and the normal approach is to transform an equality to an inequality of the form:

$$|h_j(x)|\epsilon \leq 0 \tag{13}$$

- where ϵ is the tolerance allowed (a very small value)

Constraint handling techniques

Static Penalty

- The approach proposed by Homaifar, Lai and Qi in 1994, which they define levels of violation of the constraints (and penalty factors associated to them):

$$fitness(x) = f(x) + \sum_{i=1}^{m} (R_{k,i} \times max[0, g_i(x)]^2) \tag{14}$$

- where $R_{k,i}$ are the penalty coefficients used, m is total the number of constraints, $f(x)$ is the unpenalized objective function, and $k = 1, 2, ..., l$, where l is the number of levels of violation defined by the user.

Constraint handling techniques

Criticism to Static Penalty

- It may not be a good idea to keep the same penalty factors along the entire evolutionary process.
- Penalty factors are, in general, problem-dependent.
- Approach is simple, although in some cases, the user may need to set up a high number of penalty factors.

Constraint handling techniques

Adaptive Penalty

- Bean and Hadj-Alouane[1992,1997] developed a method that uses a penalty function which takes a feedback from the search process. Each individual is evaluated by the formula:

$$fitness(x) = f(x) + \lambda(t) \left[\sum_{i=1}^{n} G_i^2(x) + \sum_{j=1}^{p} |h_j(x)| \right] \quad (15)$$

- where $\lambda(t)$ is updated at every generation t.

Constraint handling techniques

Adaptive Penalty

- $\lambda(t)$ is updated in the following way:

$$\lambda(t+1) = \begin{cases} (1/\beta_1) \cdot \lambda(t) & \text{if } \textit{case}\#1 \\ \beta_2 \cdot \lambda(t) & \text{if } \textit{case}\#2 \\ \lambda(t) & \text{if } \textit{otherwise} \end{cases} \tag{16}$$

- where cases #1 and #2 denote situations where the best individual in the last k generations was always (case #1) or was never (case #2) feasible, $\beta_1, \beta_2 > 1, \beta_1 > \beta_2$, and $\beta_1 = \beta_2$ (to avoid cycling).

Constraint handling techniques

Adaptive Penalty

- In other words, the penalty component $\lambda(t+1)$ for the generation $t+1$ is decreased if all the best individuals in the last k generations were feasible or is increased if they were all infeasible. If there are some feasible and infeasible individuals tied as best in the population, then the penalty does not change.

Constraint handling techniques

Criticism to Adaptive Penalty

- Setting the parameters of this type of approach may be difficult (e.g., what generational gap (k) is appropriate?).
- This sort of approach regulates in a more "intelligent" way th e penalty factors.

Constraint handling techniques

Penalty Functions: Central Issues

- The main problem with penalty functions is that the "ideal" penalty factor to be adopted in a penalty function cannot be known a priori for an arbitrary problem. If the penalty adopted is too high or too low, then there can be problems.
- If the penalty is too high and the optimum lies at the boundary of the feasible region, the EA will be pushed inside the feasible region very quickly and will not be able to move back towards the boundary with the infeasible region
- On the other hand, if the penalty is too low, a lot of the search time will be spent exploring the infeasible region because the penalty will be negligible with respect to the objective function.

Constraint handling techniques

Stochastic Ranking

- This approach was proposed by Runarsson and Yao [2000], and it consists of a multimembered evolution strategy that uses a penalty function and a selection based on a ranking process. The idea of the approach is try to balance the influence of the objective function and the penalty function when assigning fitness to an individual.
- An interesting aspect of the approach is that it doesn't require the definition of a penalty factor. Instead, the approach requires a user-defined parameter called P_f , which determines the balance between the objective function and the penalty function.

Constraint handling techniques

Stochastic Ranking

Algorithm 4: Stochastic Ranking

```
1  for i = 1 to N do
2      for j = 1 to P − 1 do
3          u = random(0, 1)
4          if φ(I_j) = φ(I_{j+1}) = 0) or (u < P_f) then
5              if f(I_j) > f(I_{j+1}) then
6                  Swap(I_j, I_{j+1})
7          else
8              if φ(I_j) > φ(I_{j+1}) then
9                  Swap(I_j, I_{j+1})
10     if no swap is performed then
11         Break
```

Constraint handling techniques

Stochastic Ranking

- The population is sorted using an algorithm similar to bubble-sort (which sorts a list based on pairwise comparisons). Based on the value of P_f , the comparison of two adjacent individuals is performed based only on the objective function.
- The remainder of the comparisons take place based on the sum of constraint violation. Thus, P_f introduces the "stochastic" component to the ranking process, so that some solutions may get a good rank even if they are infeasible.
- The value of P_f certainly impacts the performance of the approach. The authors empirically found that $0.4 < P_f < 0.5$ produces the best results

Constraint handling techniques

$\epsilon-$ Constrained

- The ϵ level comparisons are defined as an order relation on a pair of objective function value and constraint violation $(f(x), \phi(x))$. If the constraint violation of a point is greater than 0, the point is not feasible and its worth is low. The ϵ level comparisons are defined basically as a lexicographic order in which $\phi(x)$ precedes $f(x)$, because the feasibility of x is more important than the minimization of $f(x)$. This precedence can be adjusted by the parameter ϵ

Constraint handling techniques

$\epsilon-$ Constrained

- Let f_1, f_2 and ϕ_1, ϕ_2 be the function values and the constraint violation at a point $x1, x2$, respectively. Then, for any ϵ satisfying $\epsilon \geq 0$, ϵ level comparisons $<_\epsilon$ and $\leq_\epsilon$ between (f_1, ϕ_1) and (f_2, ϕ_2) are defined as follows:

$$(f_1, \phi_1) <_\epsilon (f_2, \phi_2) \iff \begin{cases} f_1 < f_2 & \text{if } \phi_1, \phi_2 \leq \epsilon \\ f_1 < f_2 & \text{if } \phi_1 = \phi_2 \\ \phi_1 < \phi_2 & \text{if } \textit{otherwise} \end{cases} \tag{17}$$

Constraint handling techniques

$\epsilon-$ Constrained

$$(f_1, \phi_1) \leq_\epsilon (f_2, \phi_2) \iff \begin{cases} f_1 \leq f_2 & \text{if } \phi_1, \phi_2 \leq \epsilon \\ f_1 \leq f_2 & \text{if } \phi_1 = \phi_2 \\ \phi_1 < \phi_2 & \text{if } \textit{otherwise} \end{cases} \tag{18}$$

In case of $\epsilon = \infty$, the ϵ level comparisons $< \infty$ and $\leq \infty$ are equivalent to the ordinary comparisons $<$ and $\leq$ between function values. Also, in case of $\epsilon = 0, < 0$ and ≤ 0 are equivalent to the lexicographic orders in which the constraint violation $\phi(x)$ precedes the function value $f(x)$.

Constraint handling techniques

Example

https://github.com/Cosijopiii/SENIAC-2022-CE/tree/main/knapsack

MOEA/D

- Decomposition-based evolutionary multi-objective optimization algorithms decompose a multi-objective optimization problem into subproblems using a set of predefined reference points. The convergence is guaranteed by optimizing the single-objective or simplified multi-objective subproblems while the diversity is handled by the evenly distributed reference points.
- Nevertheless, studies have shown that the performance of decomposition-based algorithms is strongly dependent on the Pareto front shapes due to unadaptable reference points and subproblem formulation

MOEA/D

Basic idea

- Decomposition (from traditional optimisation)
  - Decompose the task of approximating the PF into N subtasks, i.e. MOP to subproblems.
  - Each subproblem can be either single objective or multi-objective
- Collaboration (from EC)
  - Population-based technique: N agents for N subproblems.
  - Subproblems are related to each other while N agents solve these subproblems in a collaborative manner

MOEA/D

Basic idea

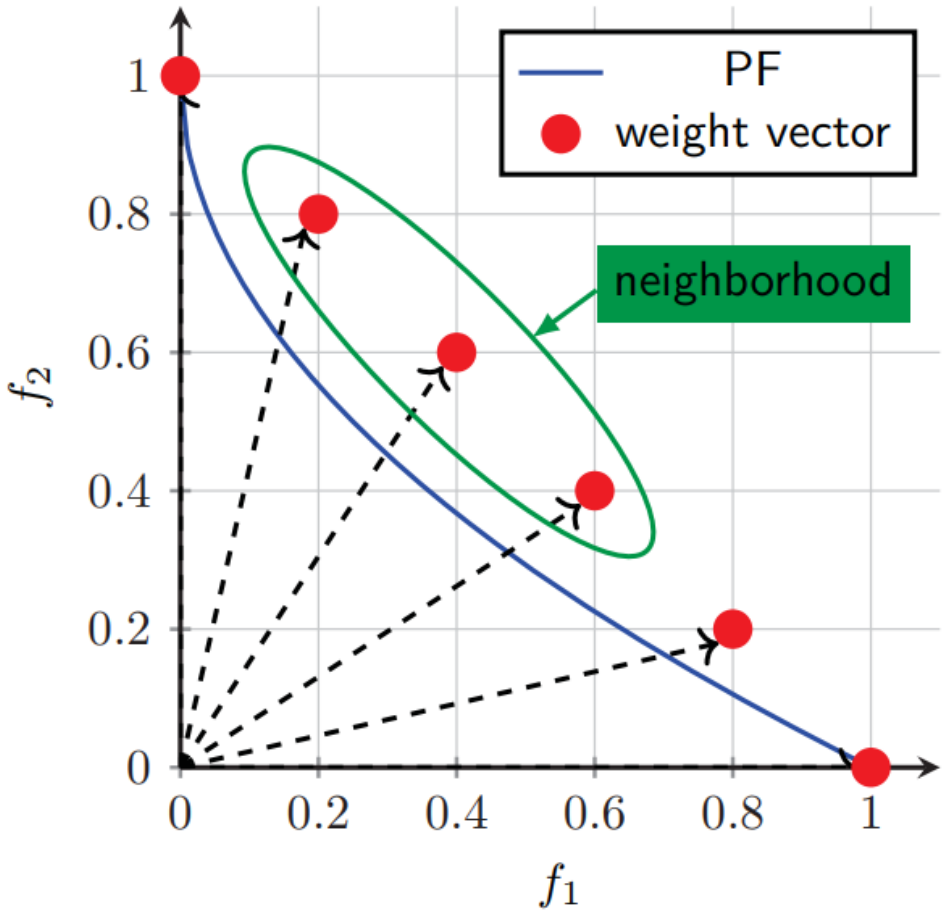


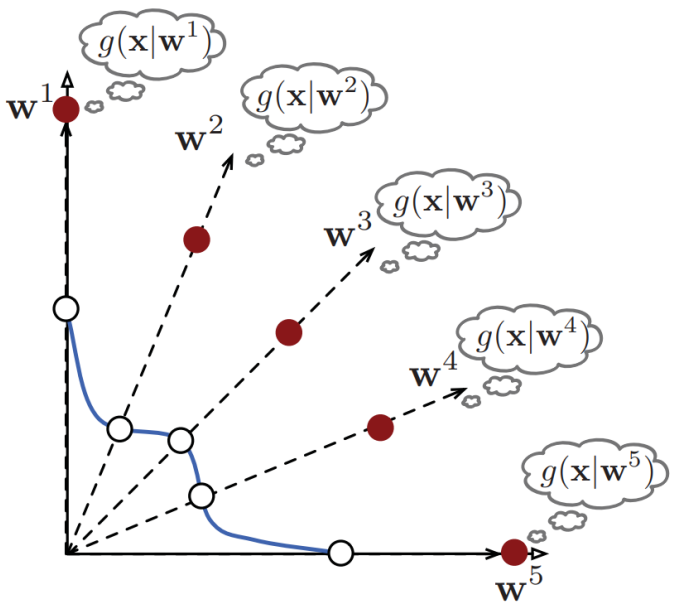
Figure: MOEA/D

MOEA/D

Basic idea

- Subproblem formulation:
- multiple objectives $\rightarrow$ Parameters $\rightarrow$ scalarizing function

$$F(X) = (f_1(x), \ldots, f_m(x)) \in \mathcal{R} \rightarrow \textbf{ Transformation } \rightarrow g(x|\cdot) \tag{19}$$



$$\begin{array}{ll} \text{minimize} & g^{ws}(\mathbf{x}|\mathbf{w}) = \sum_{i=1}^{m} w_i f_i(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \mathbf{\Omega} \end{array}$$

☞ NOTE: It works for convex PF!

Figure: scalarizing function example

MOEA/D

scalarizing function

The Canonical MOEA/D algorithm use Tchebycheff decomposition defined as follows:

$$\min g^{te}(\mathbf{x}|\lambda^j, z^*) = \max_{1\leq i\leq m} \frac{1}{\lambda_i^j}|f_i(\mathbf{x}) - z_i^*| \tag{20}$$

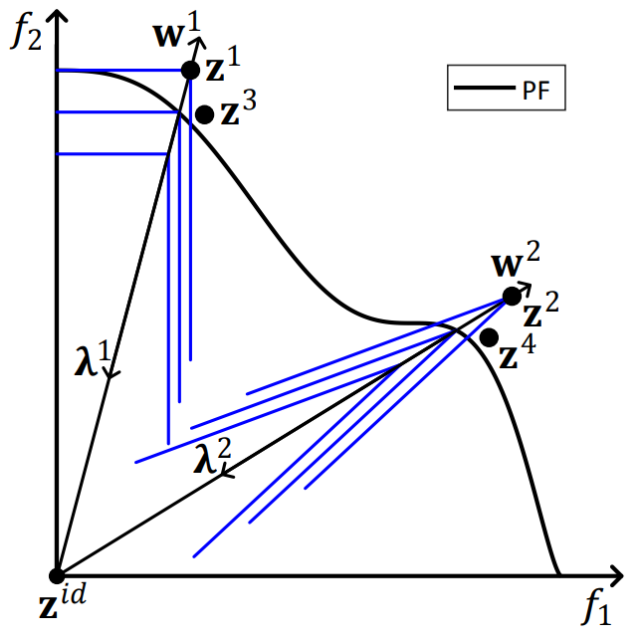


Figure: Contours of TCH

MOEA/D

Subproblem Settings

- Weight vector/Reference point Setting
- Sample a set of evenly distributed weight vectors from a unit simplex

$$\mathbf{W} = (w_1, \ldots, w_m)^T \text{ where } \sum_{i=1}^{m} w_i = 1, \mathbf{W} \in \mathbb{R}^m \tag{21}$$

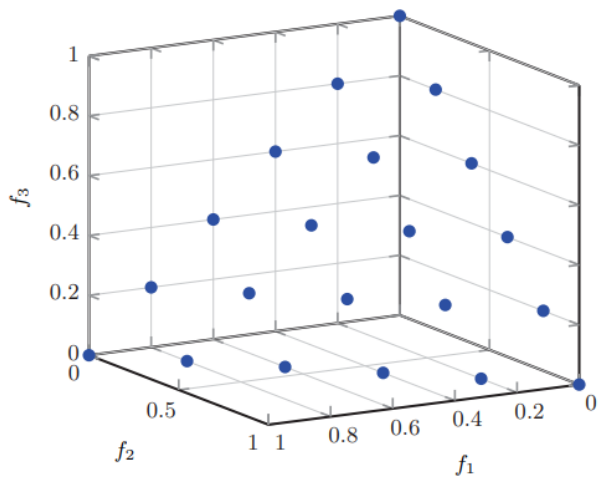


Figure: Reference points

MOEA/D

Subproblem Settings

- Neighbourhood structure:
- Two subproblems are neighbours if their weight vectors are close
- Neighbouring subproblems are more likely to have similar properties (e.g. similar objective function and/or optimal solution).

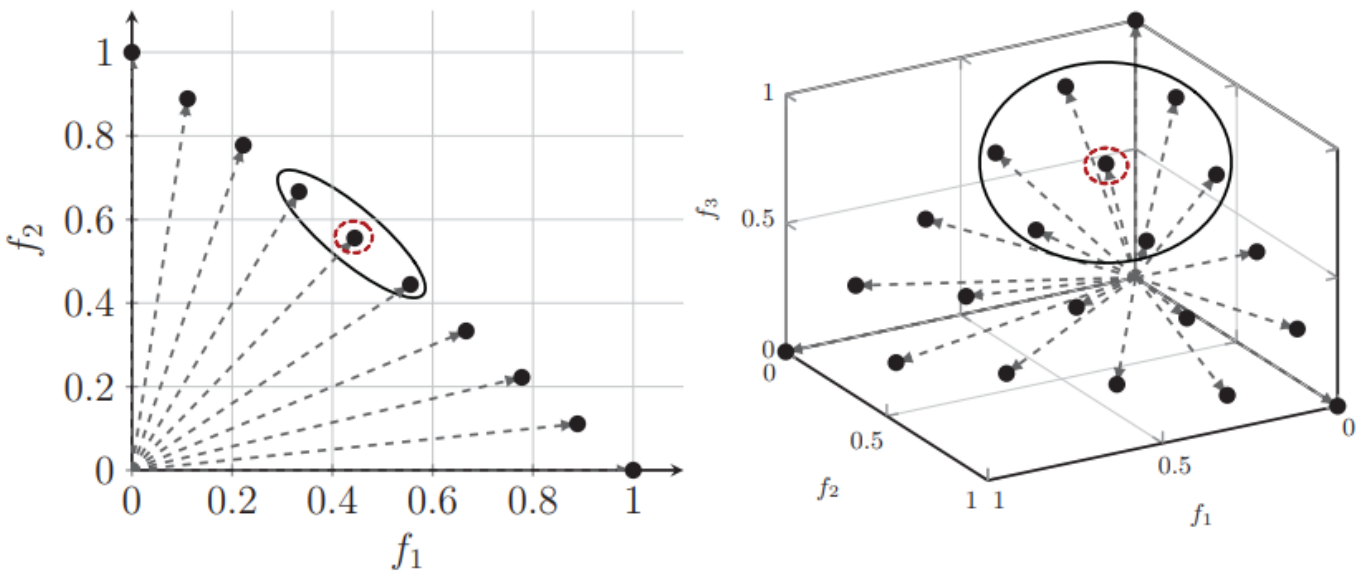


Figure: Neighbourhood structure

MOEA/D

- At each iteration, each agent does the following:
  - Mating selection (local selection): borrows solutions from its neighbours.
  - Reproduction: reproduce a new solution by applying reproduction operators on its own solutions and borrowed solutions
  - Replacement (local competition):
    - Replace the old solution by the new one if the new one is better than old one for its objective

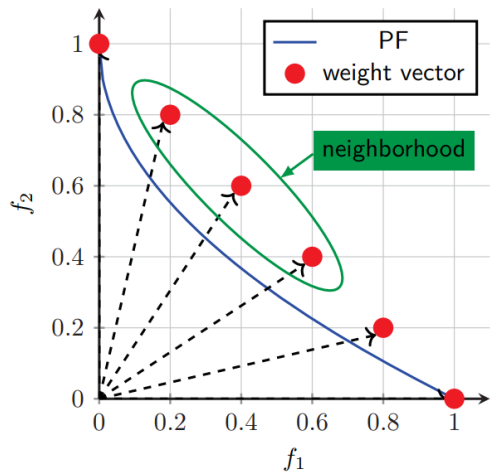


Figure: Each agent records the best-so-far solution found for its subproblem (memory)

MOEA/D

Algorithm 5: MOEA/D

$\lambda^i = (\lambda_1^i, \ldots, \lambda_m^i)^T, i = 1, \ldots, N_p$;
$B(i) = \{i_1, \ldots, i_t\}$, where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the T closest weight vectors to λ^i.;
$P = \{\mathbf{x}^1, \ldots, \mathbf{x}^{N_p}\}$;
Set Z as a ideal point Z^*;
while $k \leq T_{max}$ **do**
    for $i = 1$ **to** $size(P)$ **do**
        $P = B(i, randperm(B))$;
        Generate y from $\mathbf{P}(\mathbf{1})$ and $\mathbf{P}(\mathbf{2})$ by SBX operator;
        Polynomial mutation on y to new solution y^i;
        Update Z using a ideal point Z^*;
        $g_{pop} = g^{te}(P|\lambda^P, z^*)$;
        $g_y = g^{te}(y^i|\lambda^P, z^*)$;
    $Population(P(g_{pop} \geq g_y)) = y^i$;

MOEA/D

Conclusions

- In recent years, decomposition-based EMO algorithms have become the most popular EMO algorithms thanks to their strengthened convergence pressure by optimizing the subproblems and well-maintained population diversity by the predefined reference points. Nevertheless, when the PFs are not in line with the unit simplex, on which the reference points are evenly distributed, e.g., PFs with disparate scales, discontinuous segments or other complex shapes , they suffer from inappropriate decomposition due to unadaptable reference points and subproblem formulation.

Performance assessment of MOEAs

- Different from single-objective optimization, where the quality of a solution can be defined using the objective function values: the smaller (to minimize) or the larger (to maximize) value corresponds to a better solution, in multiobjective optimization, other aspects should be considered to evaluate the performance of MOEAs
- To evaluate the performance of different MOEAs on a given problem, the algorithm is executed a number of times, and resulting solutions known as Pareto front approximations ($\mathcal{PF}_{known}$), are compared in two aspects: (i) Solution accuracy determines how similar an evolved solution is to the true Pareto front ($\mathcal{PF}_{true}$) and (ii) Solution diversity, e.g. to evaluate how well the solution is distributed in the solution space

Performance assessment of MOEAs

- Therefore, to assess the performance of the MOEAs, several performance measures have been proposed which considered the two above issues.
- **Generational Distance**. This metric (GD) reports how far, on average the $\mathcal{PF}_{known}$ is from $\mathcal{PF}_{true}$. It is defined as

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{22}$$

where n is the number of elements in the $\mathcal{PF}_{known}$ and d is the Euclidean phenotypic distance between each member

Performance assessment of MOEAs

- **Inverted Generational Distance**. This metric (IGD) evaluates the performance related to convergence and diversity simultaneously, this metric represents the average distance from $\mathcal{PF}_{true}$ to $\mathcal{PF}_{known}$. It is defined as:

$$IGD = \frac{\sum_{y^* \in \mathcal{PF}_{true}} d(y^*, \mathcal{PF}_{known})}{n}$$
$$d(y^*, \mathcal{PF}_{known}) = min_{y \in \mathcal{PF}_{true}} \sqrt{\sum_{i=1}^{m} (y^* - y_i)^2} \tag{23}$$

Performance assessment of MOEAs

- **Hypervolume**. This metric (HV) reflects the closeness between $\mathcal{PF}_{known}$ and $\mathcal{PF}_{true}$. A large HV means that the $\mathcal{PF}_{known}$ set is closer to the $\mathcal{PF}_{true}$. HV corresponds to the non-overlapping volume of all hypercubes formed by reference point z and every vector in the $\mathcal{PF}_{known}$. HV with a larger value represents better performance with respect to both diversity and convergence. It is defined as follows:

$$HV = \bigcup_{i=1}^{Q} \{vol_i | vec_i \in \mathcal{PF}_{true}\} \tag{24}$$

where vec_i is a non-dominated vector from $\mathcal{PF}_{known}$, Q is a set of $\mathcal{PF}_{known}$ solutions and vol_i is the hypercube's volume formed by the reference point and the non-dominated vector vec_i

Performance assessment of MOEAs

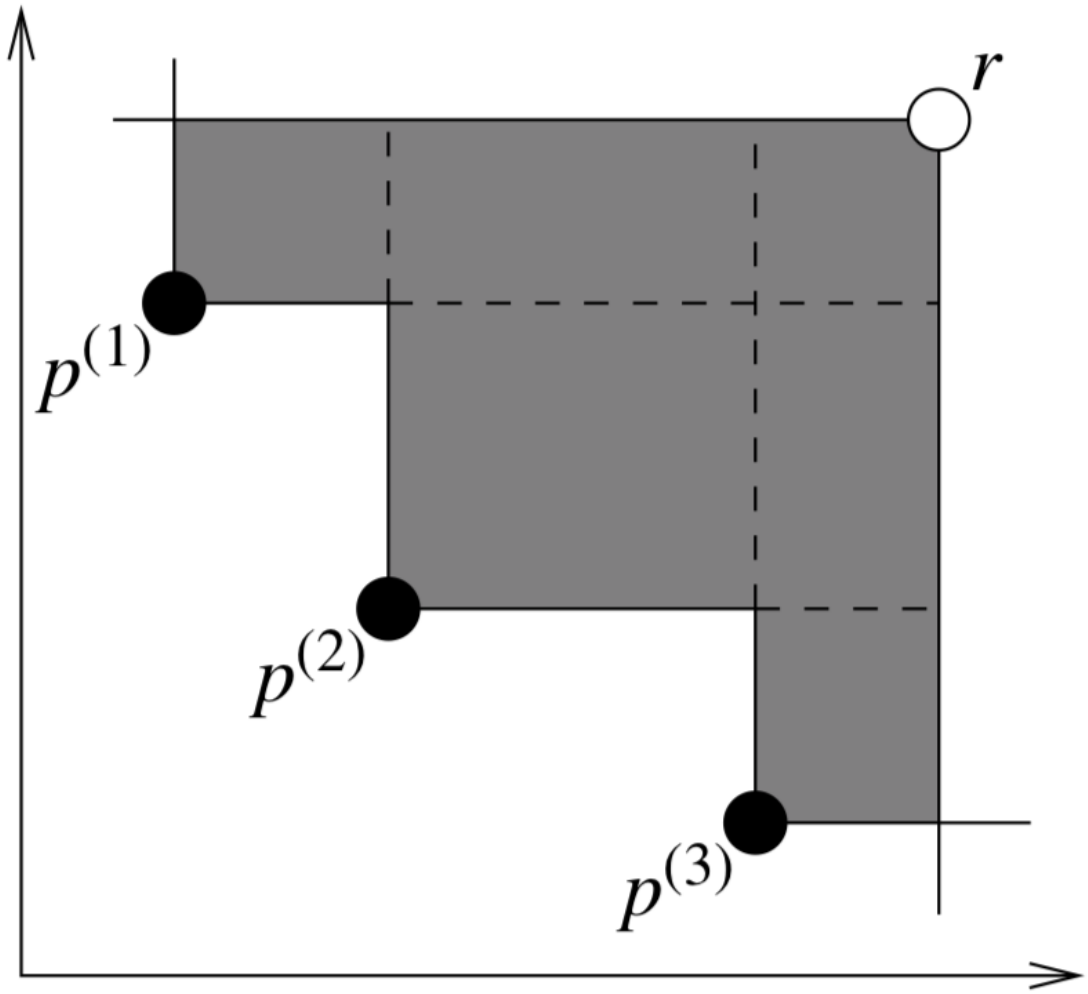


Figure: HV