Evolutionary multi-objective optimization (EMO)

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

Computer Science Lab, INAOE

April 19, 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`

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

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

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

- 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

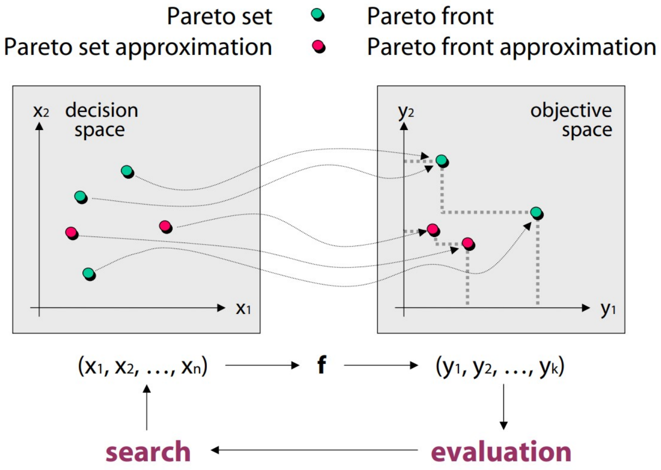


Figure: Decision variables space and objective function space.

Basic concepts

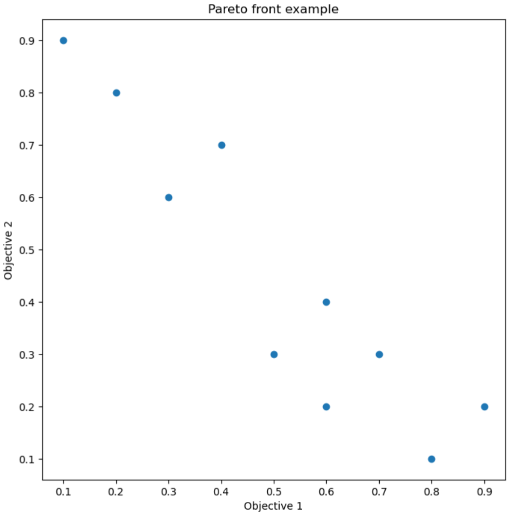
Example

Let's consider the following list of points:

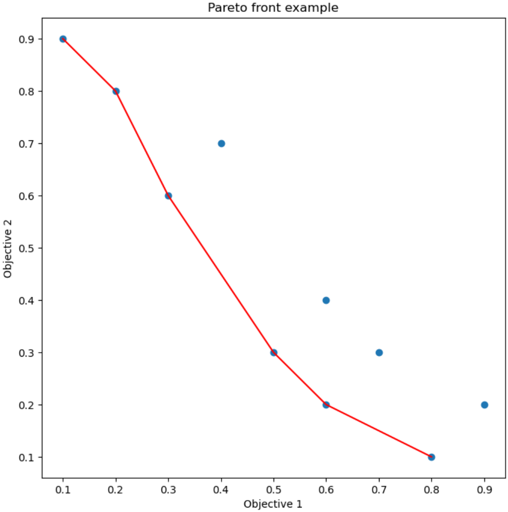
Point	Coordinates
1	$(0.2, 0.8)$
2	$(0.3, 0.6)$
3	$(0.5, 0.3)$
4	$(0.4, 0.7)$
5	$(0.6, 0.2)$
6	$(0.8, 0.1)$
7	$(0.9, 0.2)$
8	$(0.7, 0.3)$
9	$(0.6, 0.4)$
10	$(0.1, 0.9)$

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

Cont.


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

Cont.

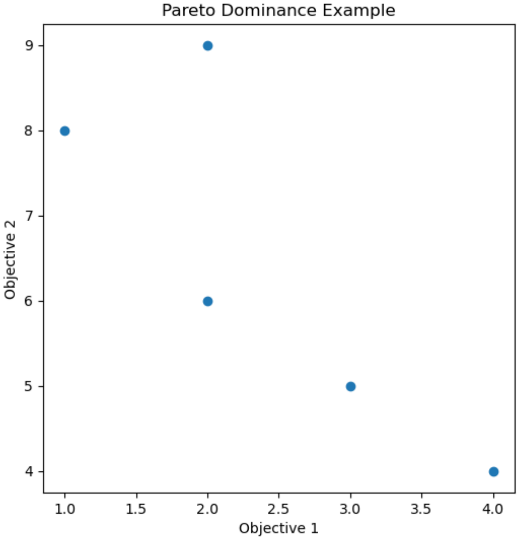

Pareto front example
Objective 2
Objective 1
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

Exercise.

Given the following points, what is the non-dominated solution?

$$
\begin{aligned}
A &= [2, 6] \\
B &= [1, 8] \\
C &= [3, 5] \\
D &= [4, 4] \\
E &= [2, 9]
\end{aligned}
$$

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

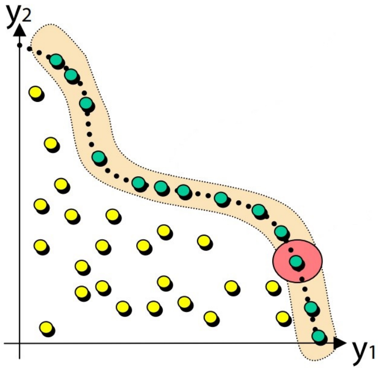


Figure: Pareto Front.

Special solutions

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

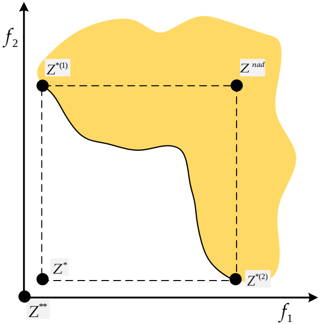


Figure: Representation of an objective function space with ideal (Z^*), utopian (Z^{**}), and nadir (Z^{nad}) objective vectors.

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

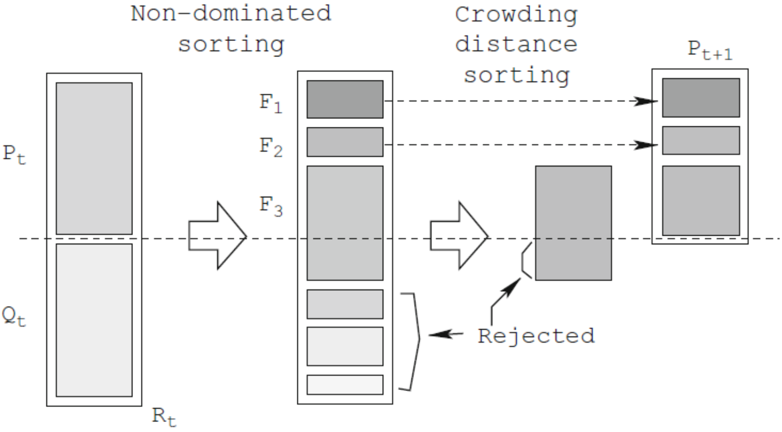


Figure: NSGA-II, process

NSGA-II, NDS

Algorithm 2: Non-dominated sorting

```
for each p ∈ Population 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;
        if n_p = 0 then
            p_rank = 1;
            F_1 = F_1 ∪ {p};
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};
    i = i + 1;
    F_i = Q;
```

NSGA-II, CD

Algorithm 3: Crowding Distance

$\alpha = |D|$; `// number of solutions in` D

for *each* i **do**

  $D[i]_{distance} = 0$

for *each objective* m **do**

  $D = sort(D, m)$; `// sort using each objective value`

  $D[1]_{distance} = D[\alpha]_{distance} = \infty$; `// so that boundary points are always selected`

  $i = 2$ to $(\alpha - 1)$ D[i]$_{distance} = D[i]_{distance} + (D[i+1].m - D[i-1].m)/(f_m^{max} - f_m^{min})$;

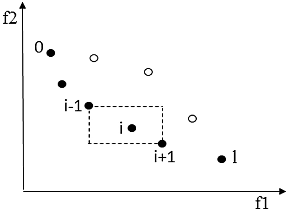


Figure: Cuboid, Crowding-distance calculation.

NSGA-II, example

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

Constraint handling techniques

- Penalty solutions
- Stochastic Ranking
- Constraint dominance principle (CDP), a.k.a Deb's rules.
- $\epsilon-$Constrained

MOEA/D

Metrics / Indicator - based MOEAs

Many Objetives.

References I