MultiObjective Optimization

Trade offs?

MultiObjective Optimization (MOO)

Definition:

Multi-objective optimization or Pareto optimization (also known as multi-objective programming, vector optimization, multicriteria optimization, or multiattribute optimization) is an optimization problem that involves

more than 1 objective function to be optimized simultaneously.

→ optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives

https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/download/2198/2030/

MultiObjective Optimization (MOO)

Main message: Multiobjective optimization is somewhat of a misnomer - you actually have to have predefined weightings for each of the objectives you care about, or implement them as constraints

https://openmdao.github.io/PracticalMDO/Notebooks/Optimization/multiobjective.html

Trade off

When doing multiobjective optimization,

I strongly recommend performing multiple single-objective optimizations instead of using a multiobjective optimizer.

- Let's view aerostructural wing design for an example. In a simple trade-off, if you extend the wingspan you get more lift and aerodynamic performance, but your structural masses and costs become greater to withstand the larger load.

- This trade-off means that you cannot maximize the aerodynamic performance and minimize the structural weight - there must be some sort of a balancing act. In reality we care about the total performance of the airplane, more than any subdiscipline, so our objective function usually captures effects from multiple disciplines at once.

Multi-Objective Optimization

$$\min f_i(x), \qquad i = 1, \dots, d$$

Subject to $\quad g(x) \geq, \quad h(x) = 0$

$$F(x) = [f_1(x), \dots . f_d(d)]$$

We know how to do this: $\min f\,(x)$

Solution: $f(x) = \sum_i w_i f_i(x)$

MOO

$$\min f_i(x), \qquad i = 1, \dots, d$$

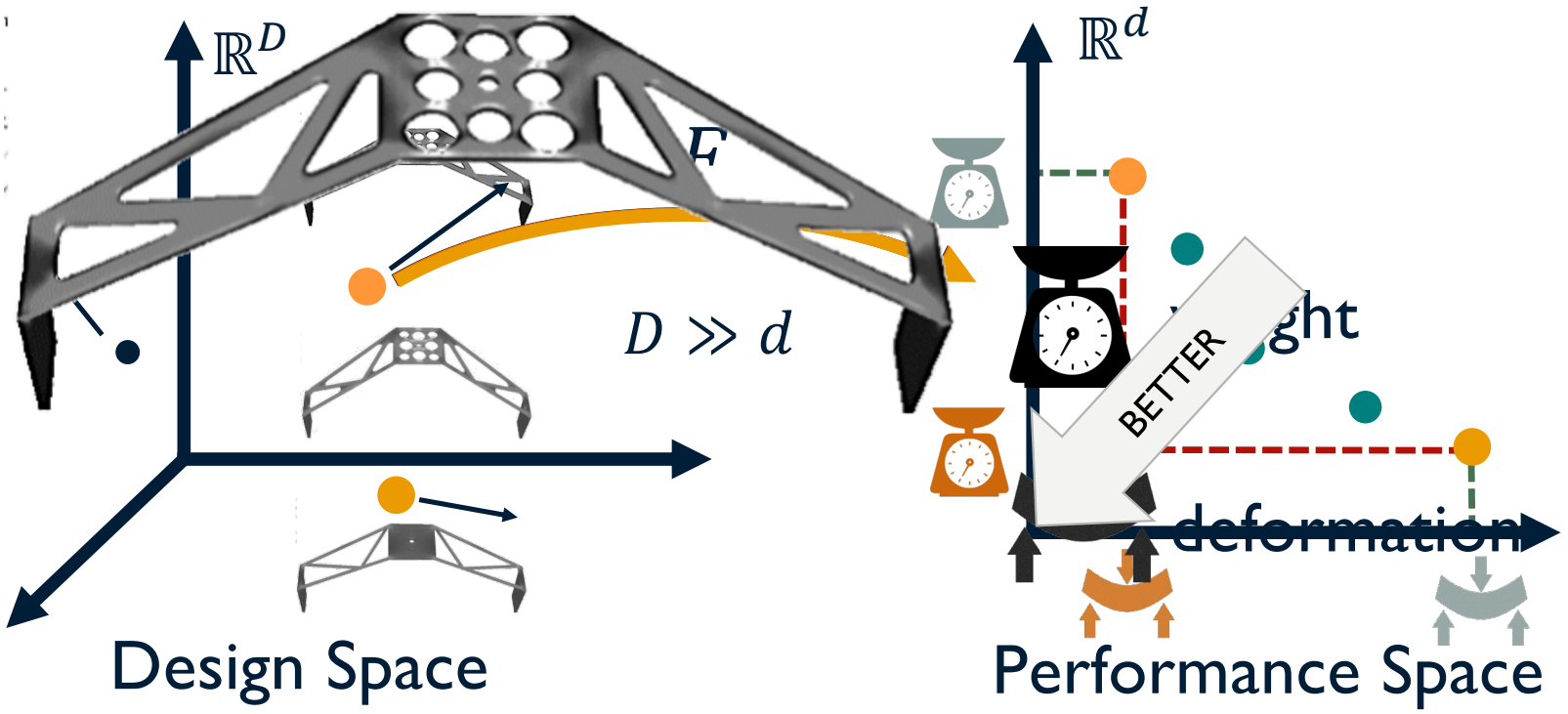
$$\text{Subject to} \quad g(x) \geq, \quad h(x) = 0$$

$$F(x) = [f_1(x), \dots . f_d(d)]$$

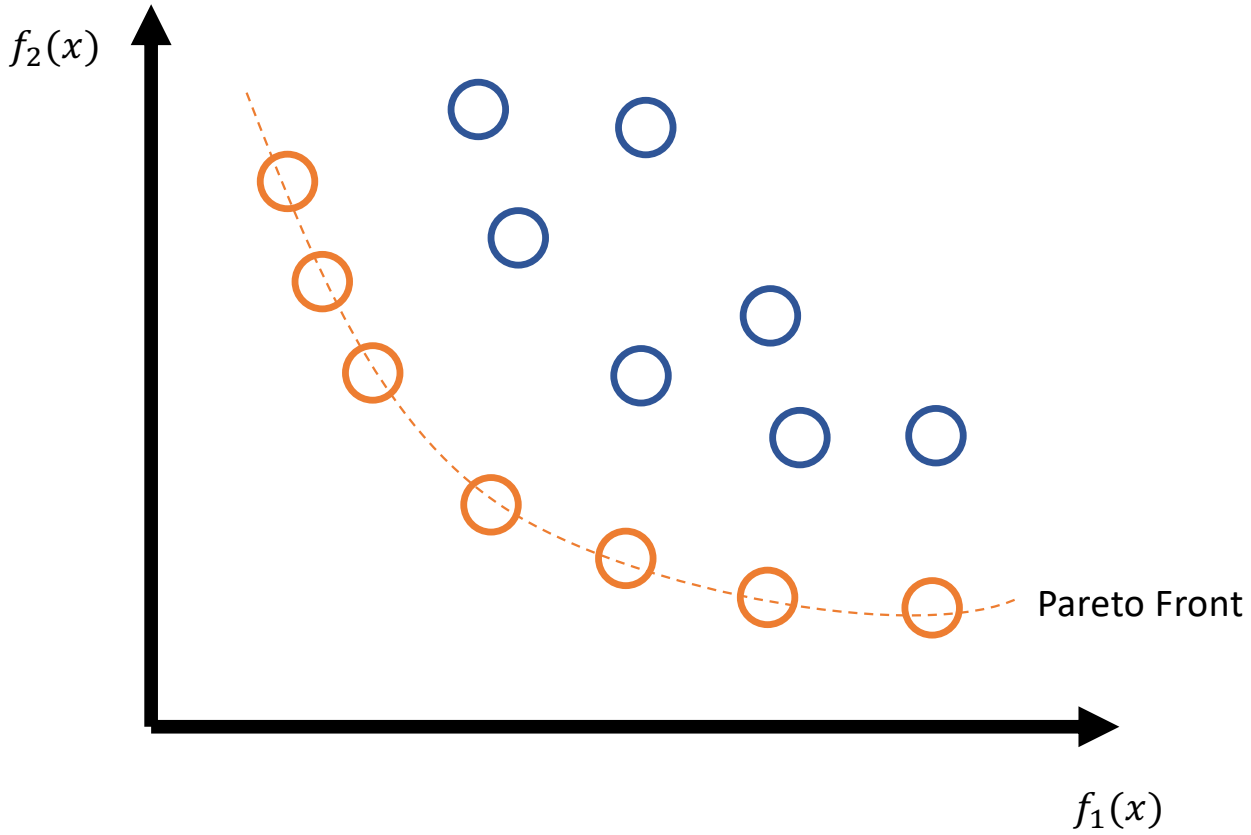
We know how to do this: $\min f(x)$

Solution: $f(x) = \sum_i w_i f_i(x)$ How do you pick the weights?

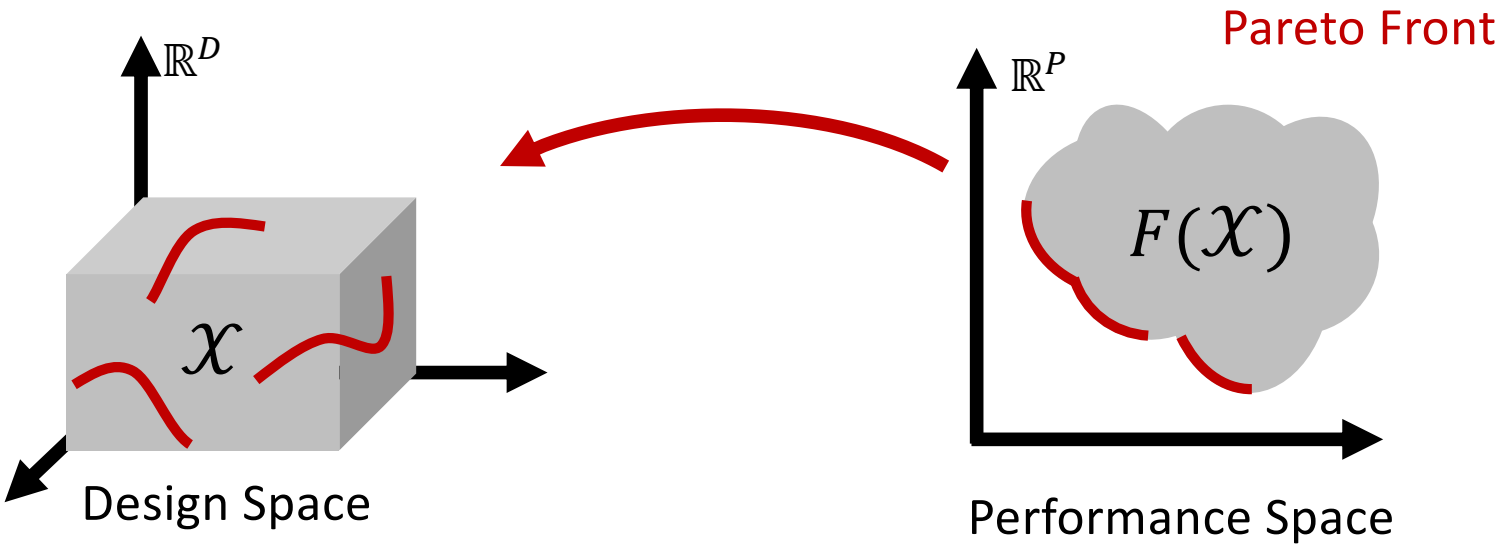
When Objectives are Conflicting



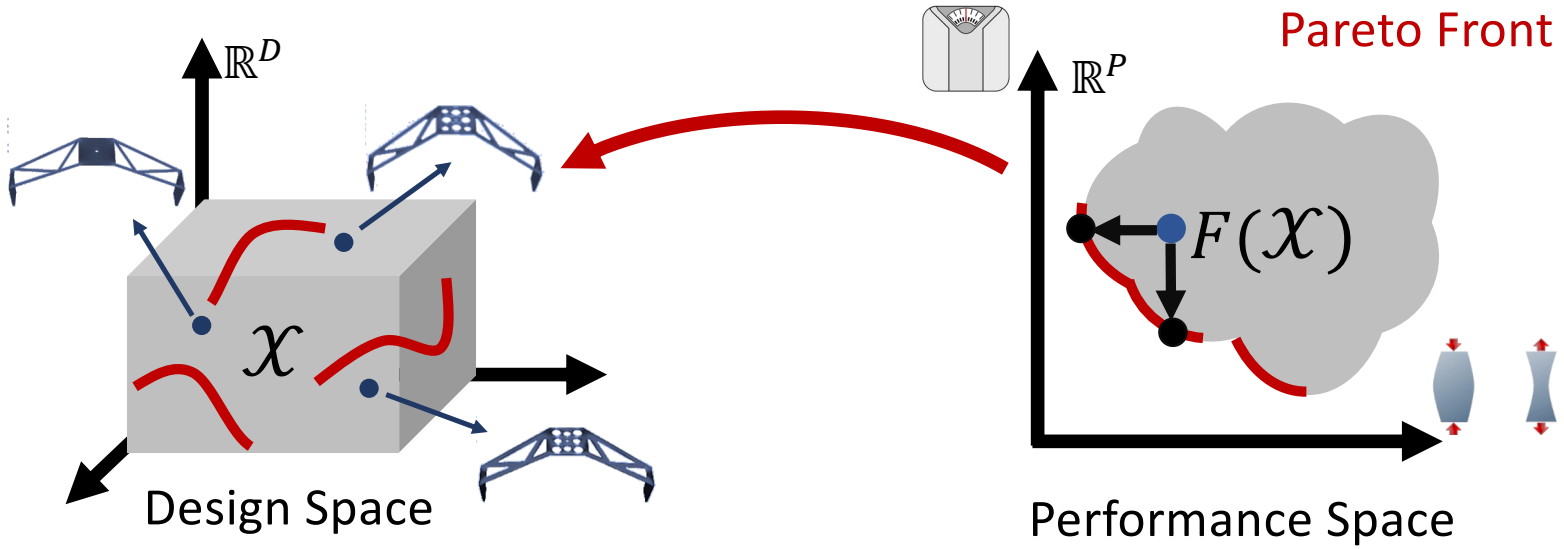
For Minimization



Space of Optimal Solutions



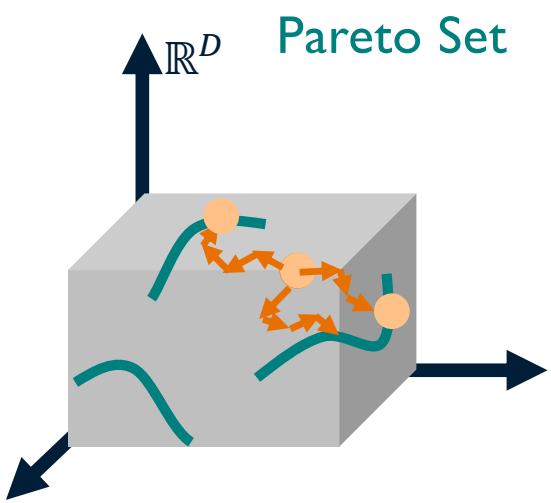
Space of Optimal Solutions

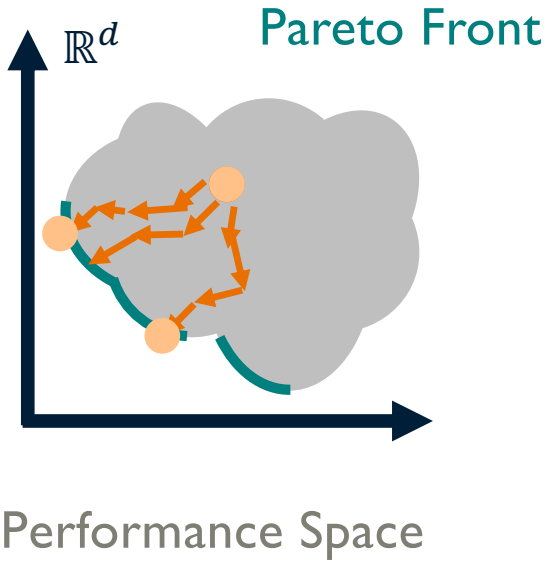


Pareto Front Discovery

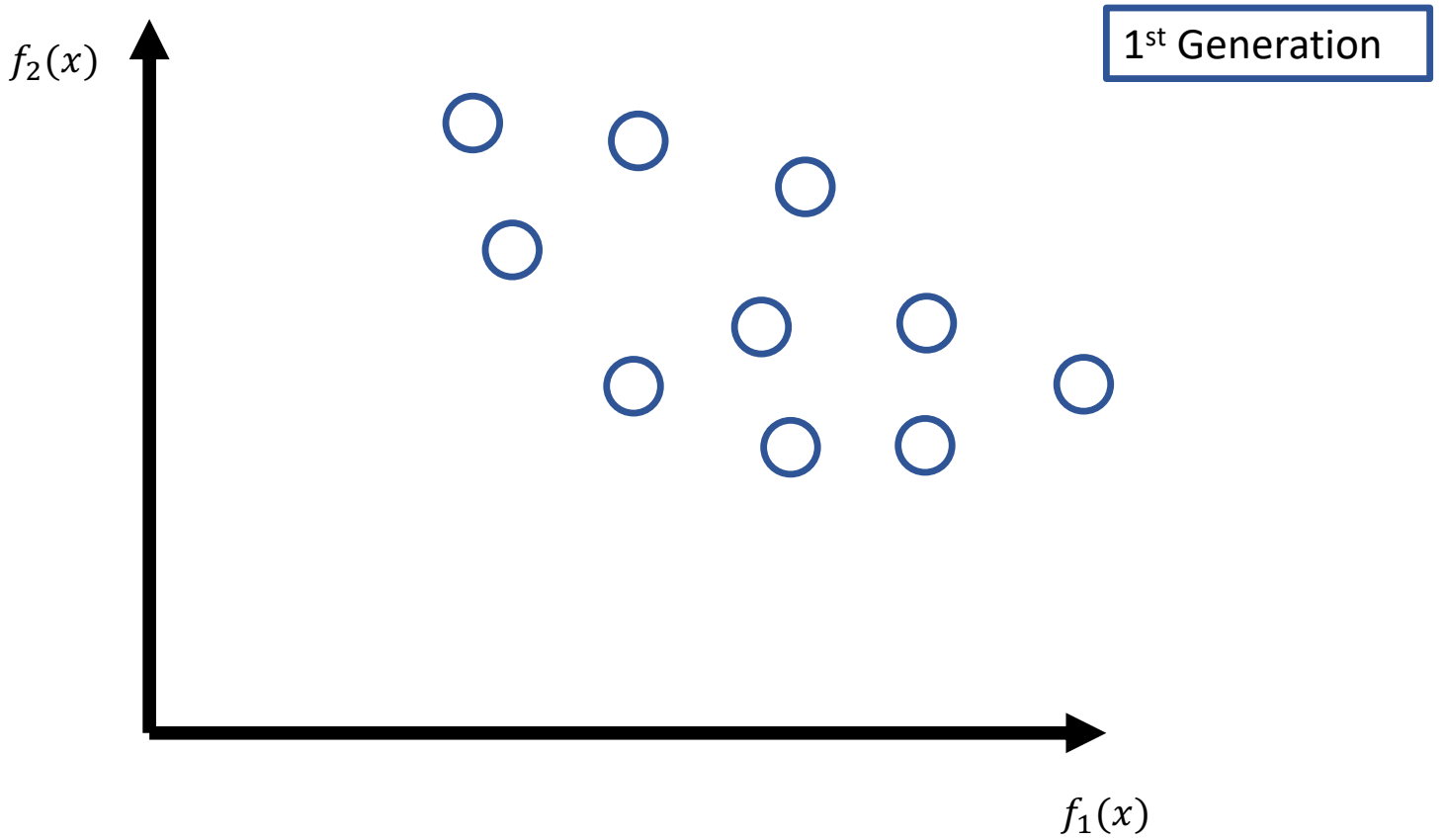
Main Challenge:

- Converge to optimal solutions
- Diverse set that describes the full front

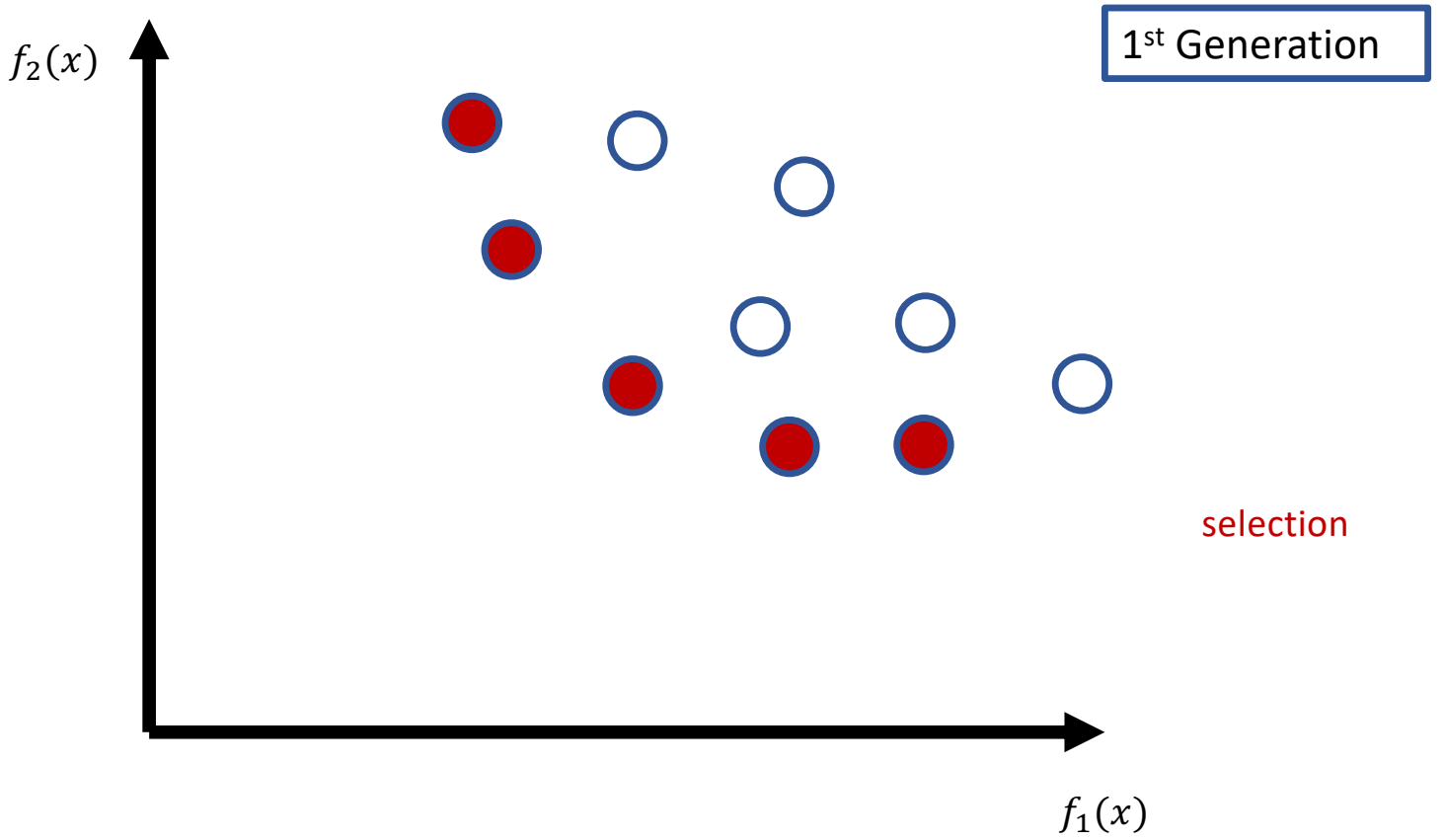




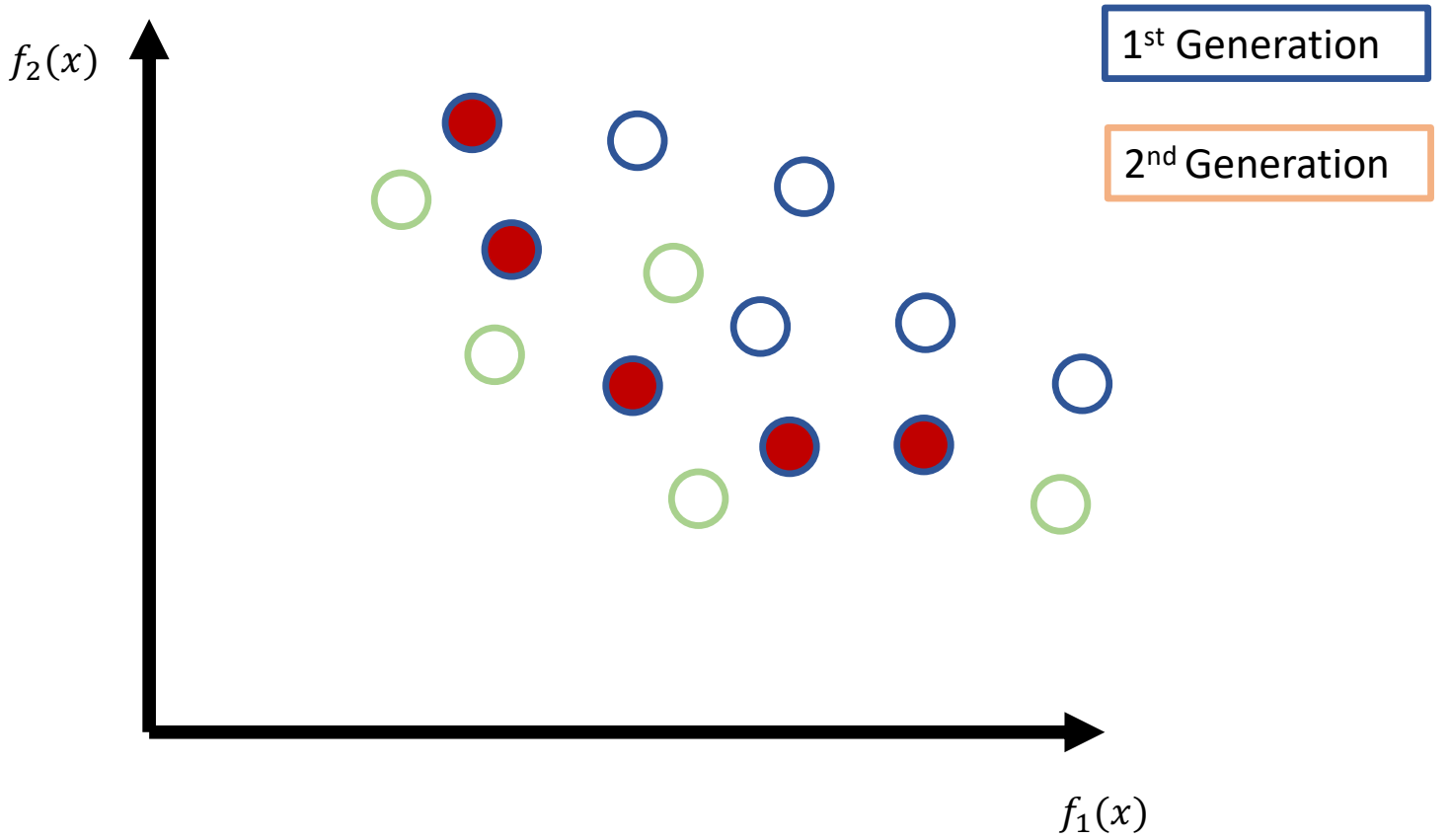
Evolutionary Algorithms



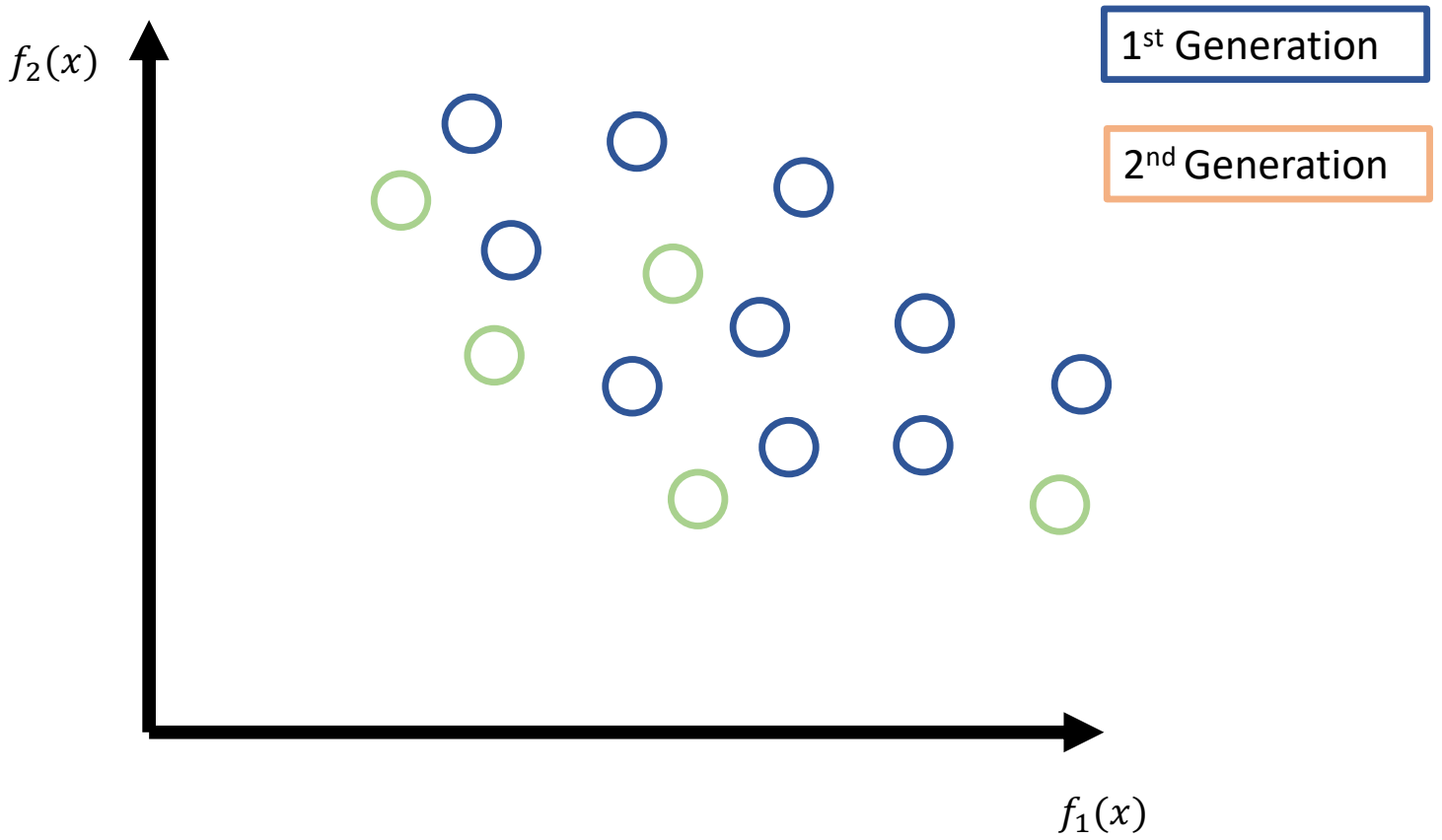
Evolutionary Algorithms



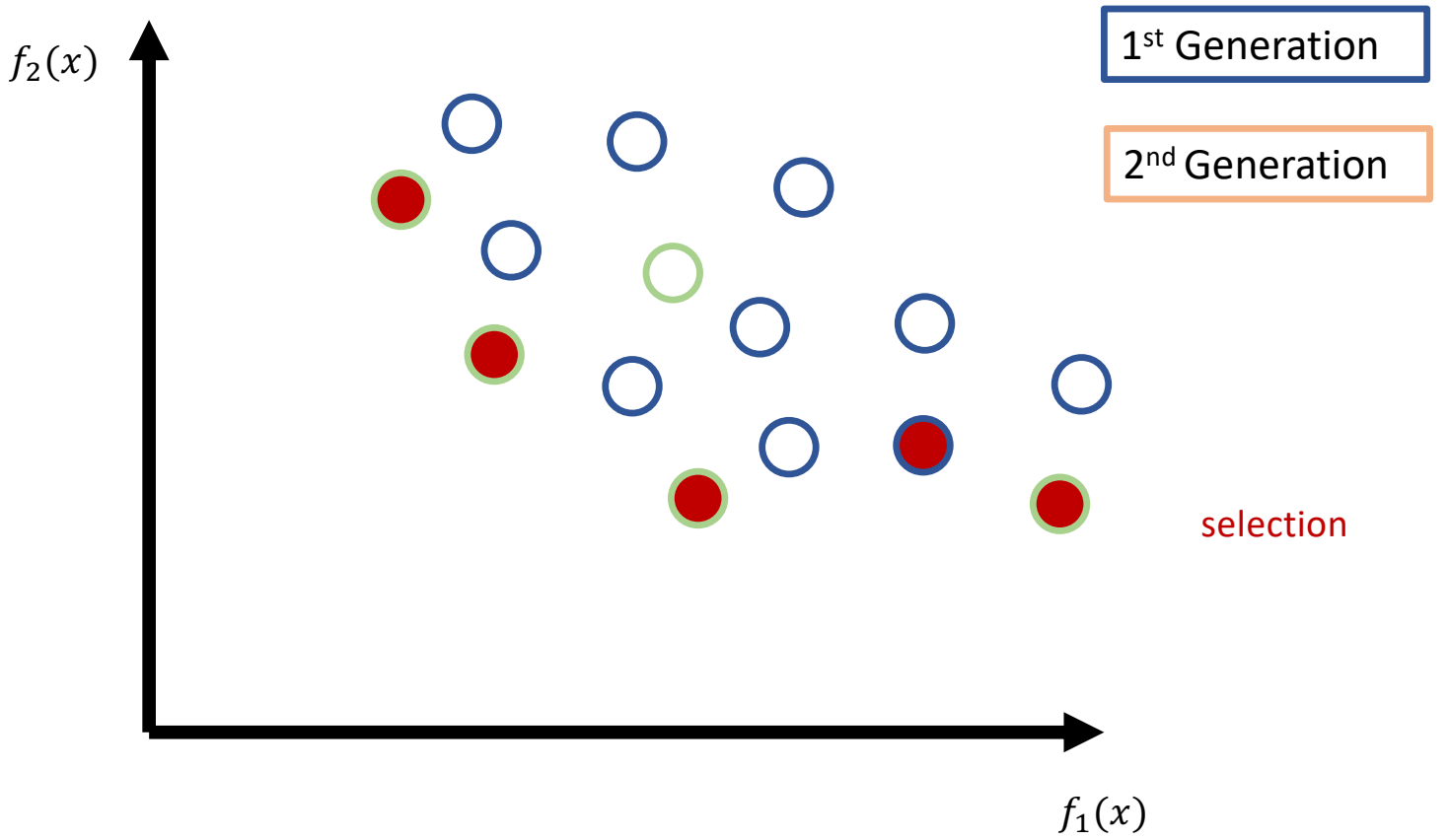
Evolutionary Algorithms



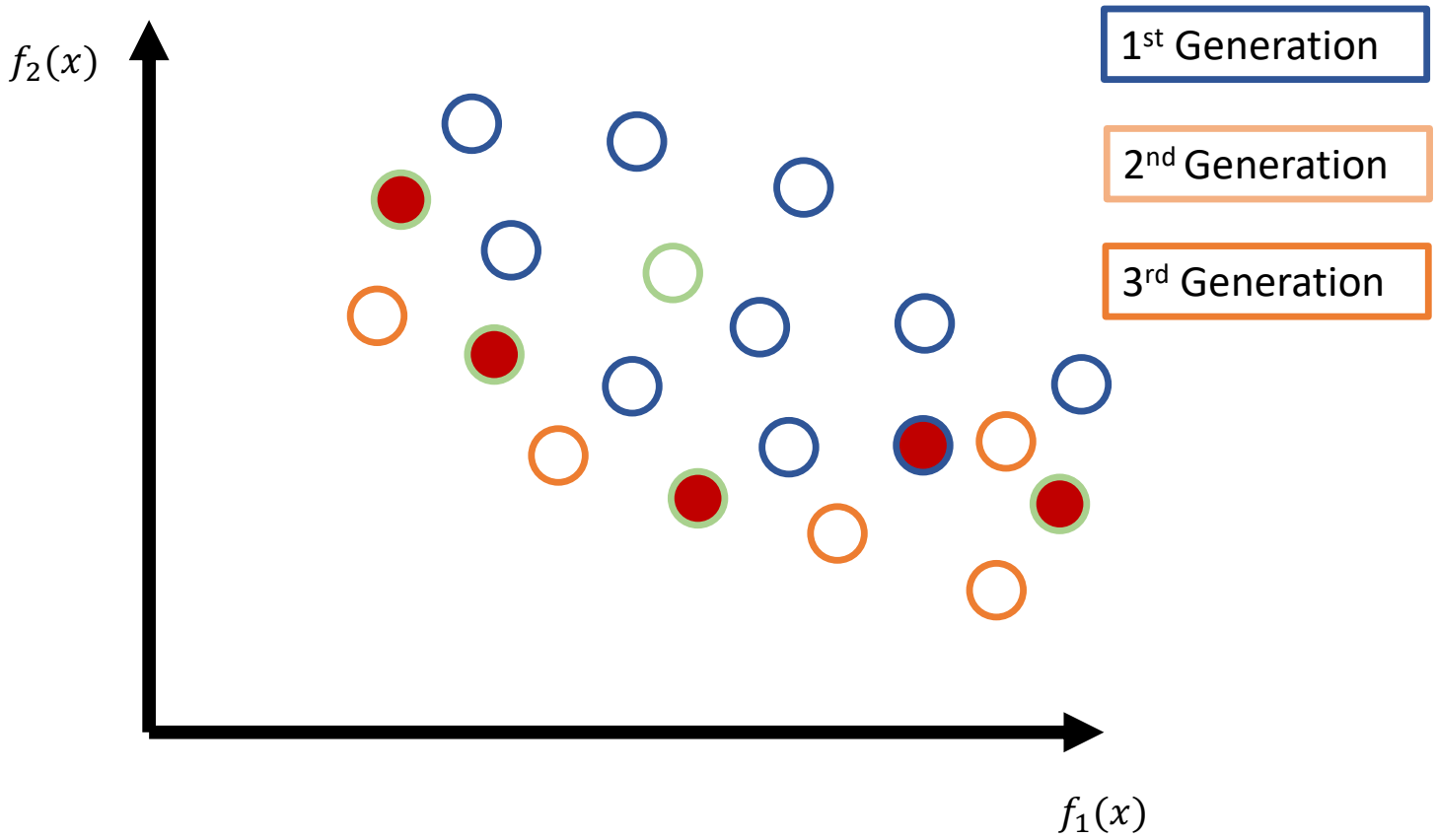
Evolutionary Algorithms



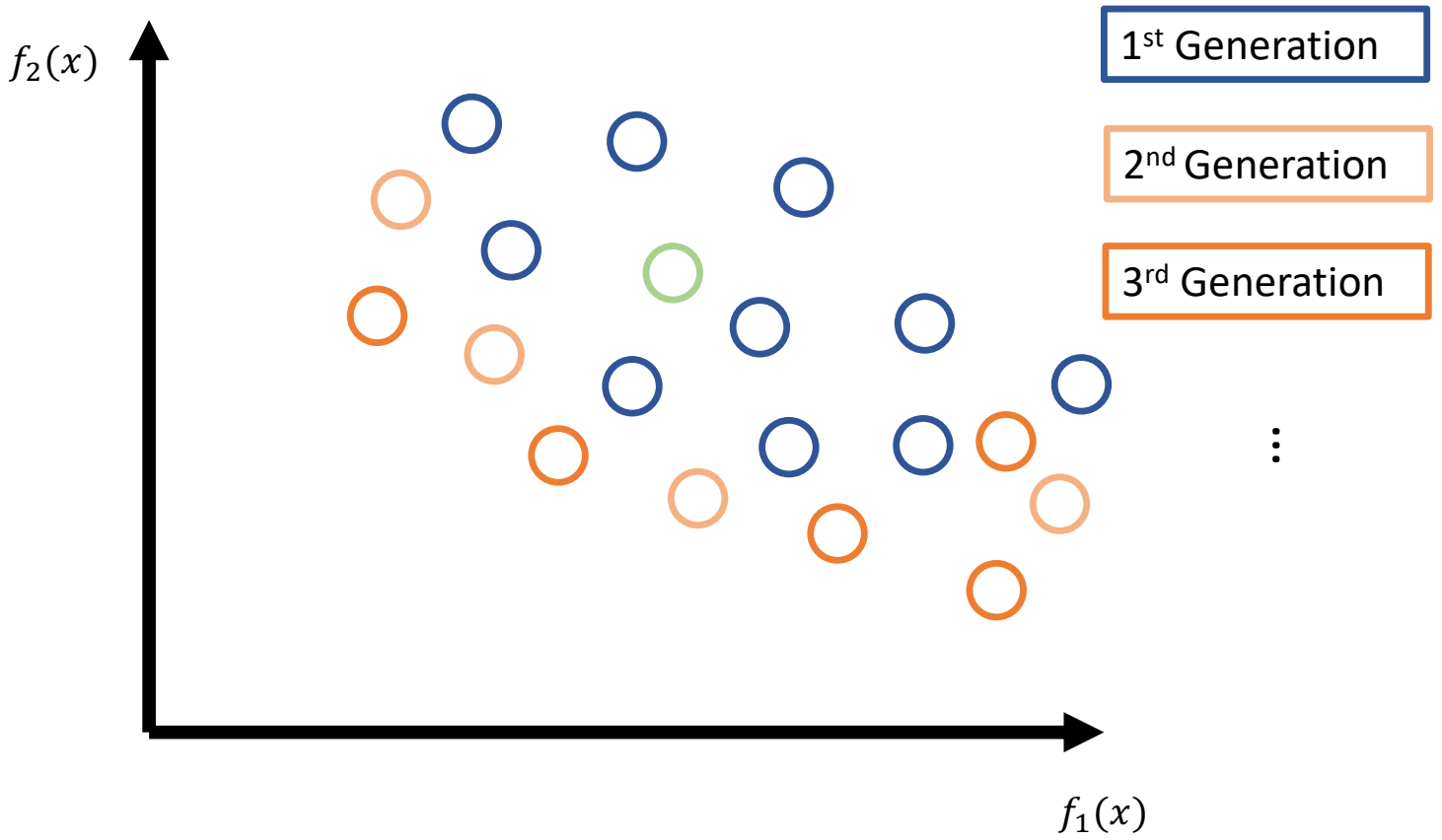
Evolutionary Algorithms



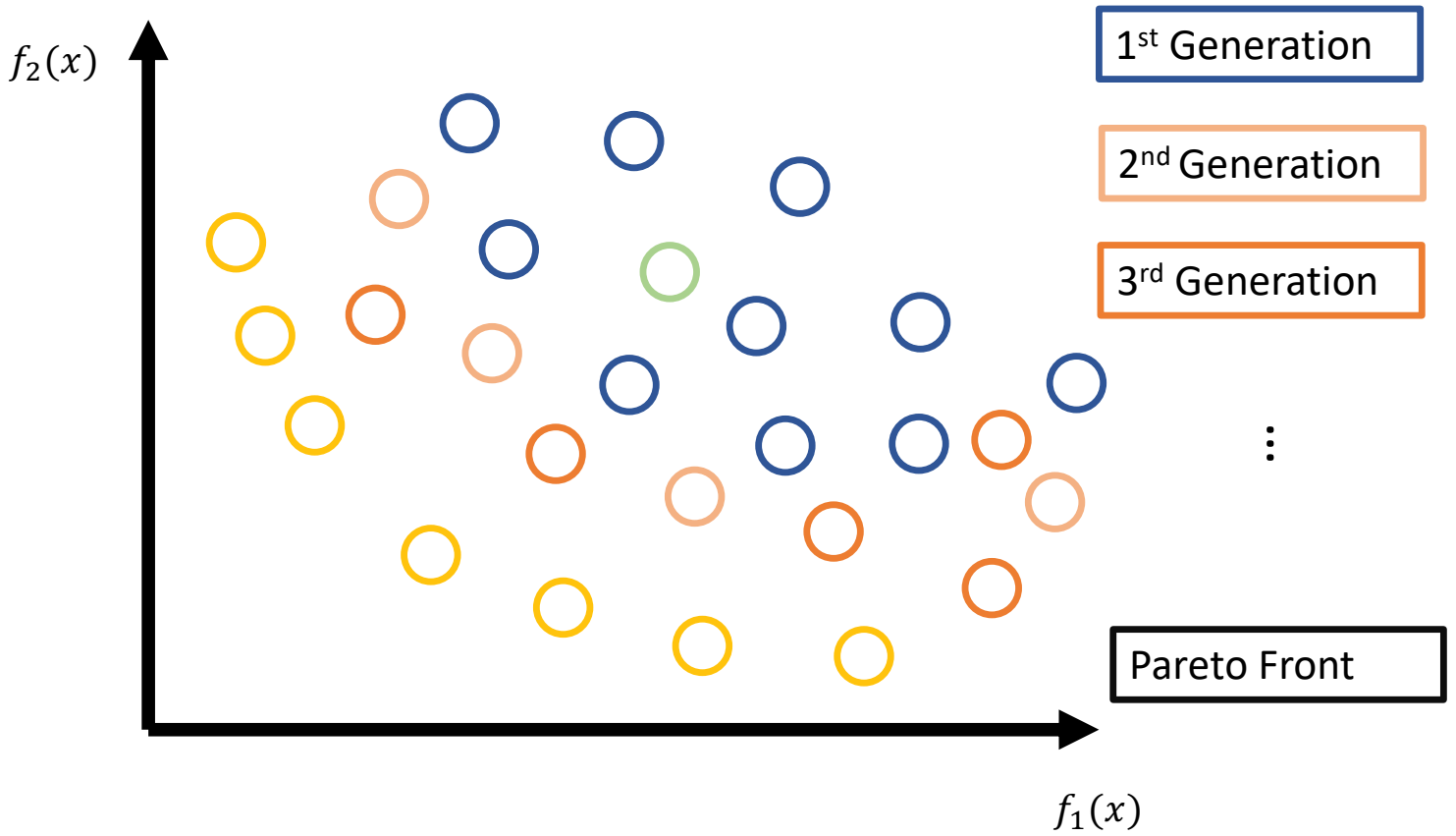
Evolutionary Algorithms



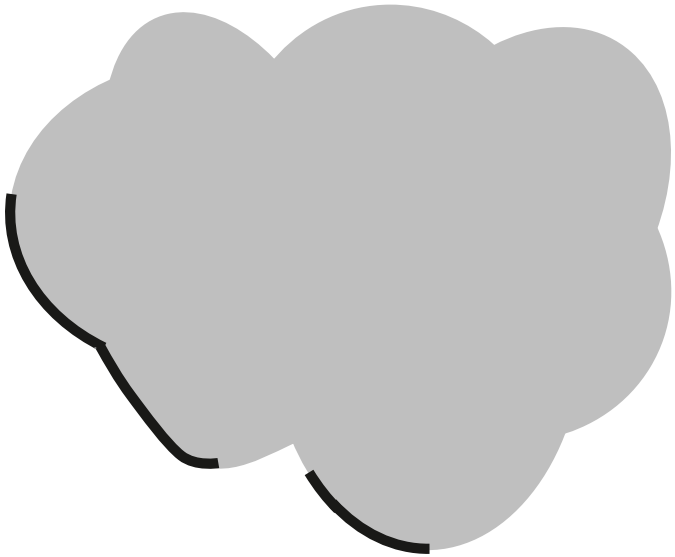
Evolutionary Algorithms



Evolutionary Algorithms (Try NSGA-II)



The Geometry of the Front



Not a straight line!

Solution: $f(x) = \sum_i w_i f_i(x)$ ☹

Pymoo example

- Complete Pymoo_example.ipynb

In Practice

To finish Complementary materials online

CHALLENGE #4 Could you use PYMOO?

https://pymoo.org/getting_started/preface.html

https://colab.research.google.com/drive/1EjHgXfbP21WvgWmGPGE7CRh2w42ejCJz

Weighted sum method

The weighted sum method simply combines multiple objective functions by adding them together with some weights on each function.

An example is shown here,

$$f_{\text{obj}}(x) = \alpha g(x) + \beta h(x)$$

where $g(x)$ and $h(x)$ are two objectives we're trying to optimize simultaneously

and α and β are weighting variables for each of the objective functions.

Weighted sum method

If you cared much more about the value of $\mathbf{g(x)}$, you would set $\boldsymbol{\alpha}$ to be larger than $\boldsymbol{\beta}$.

However, it's not that simple as $\mathbf{g(x)}$ and $\mathbf{h(x)}$ might have radically different magnitudes.

If $\mathbf{g(x)}$ is on the order 5,000 while $\mathbf{h(x)}$ is on the order 0.1 ,

this would cause the optimizer to prioritize changing the design to make $\mathbf{g(x)}$ smaller.

Weighted sum method

An example of this is if we're trying to optimize the structural weight (large magnitude) and the coefficient of drag C_D (small value) for an aircraft:

- $f_{obj}(x) = f_{\text{structural weight}_{(x)}}[kg] + f_{C_D}$

If you care about both objectives approximately equally, you should scale both objective functions by an appropriate amount to make them approximately the same magnitude. The general form looks like this:

- $f_{obj}(x) = \alpha\frac{g(x)}{g_0} + (1-\alpha)\frac{h(x)}{h_0}$

where g_0 and h_0 are those magnitude scalars.

In the previously mentioned example, this might look like:

- $f_{obj}(x) = \alpha\frac{f_{\text{structural weight}_{(x)}}[kg]}{2700[\ kg]} + (1-\alpha)\frac{f_{C_D}}{0.025}$

Epsilon constraint method

- The epsilon-constrained (ϵ-constrained) method is another way to perform multiobjective optimization where one objective function is minimized by the optimizer while the other objective functions are constrained to specific values. This ensures that a given design is optimal is one objective for a given value in the other objective functions.

- The ϵ-constrained method is the preferred way to create Pareto fronts because it produces a more robust curve and avoids rare situations where the Pareto front is non-convex that may be troublesome if using the weighted-sum method.

Epsilon constraint method

This method also allows you to easily control the spacing for a Pareto front. Given a spread of $g(x)$ values for constraints, you can directly set the spacing produced by a series of optimizations.

For a bi-objective optimization, a good way to know what bounds to use for the constraint values are to run two optimizations first: one unconstrained using the first objective and the other unconstrained with the second objective only. From those two optimizations, the resulting values of the second and first objectives, respectively, could be your constraint limits.