# Optimization: an introduction

Alberto TONDA, Senior Researcher (DR)

*UMR 518 MIA-PS (Applied Mathematics and Computer Science)*
*INRAE, AgroParisTech, Université Paris-Saclay*
*Institut des Systèmes Complexes, Paris-Ile-de-France*

# Outline

- Vocabulary
- General principles
- Brainstorming
- Taxonomy (-ies)
- Intended outcome

# Vocabulary

- Objective/cost/loss/fitness function
  - Function that we aim to minimize/maximize
  - "Function" in the broadest possible sense (input, output)
- Variables
  - Inputs of the objective function; *d* variables, *d* dimensions
  - We can control them, use them to sample the objective function
- Search space/objective function landscape
  - All possible values of the input variables that we could test
  - Sampled to find best possible values of objective function

# Vocabulary

- Boundaries
  - **Limits** of variable values
  - Described for each variable, independently
  - Boundaries define the limits of the search space
- Candidate solution
  - **Point in search space** that *could* be the solution to our problem
- Constraints
  - Relationships between multiple problem variables
  - *Must* be satisfied to have an acceptable solution

# Vocabulary

- Neighborhood
  - Part of the search space “near” a given solution
  - In a **continuous search space**, small hypervolume around point

$$N(x) = \{x' \in \mathbb{R}^d, \|x - x'\|_n \leq \epsilon\}$$

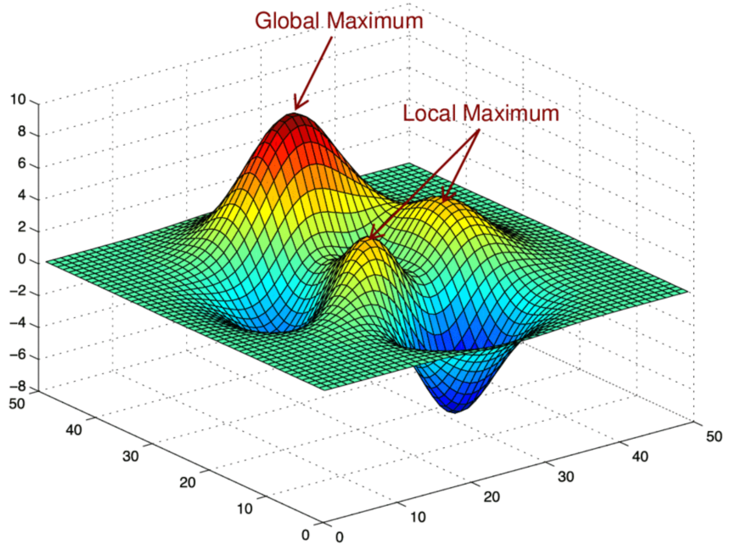
  - In a **discrete search space**, we need to define a **move operator**

$$N(x) = \{x' \epsilon\, S, x' \text{ is reachable from } x \text{ using a single move}\}$$

  - Example: for a bit string, 0101010…1 move can be “flip bit”
- Local vs global search
  - Local search moves only inside the neighborhood
  - Sometimes used loosely, definition is not precise

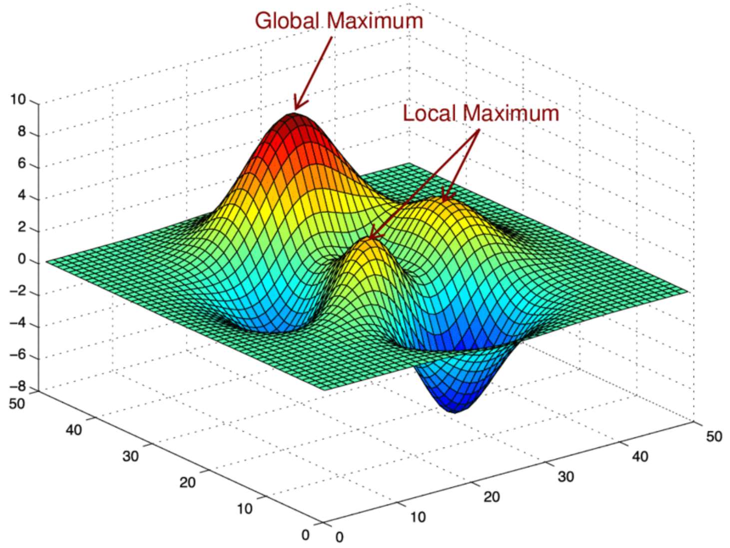
# Vocabulary

- Global optimum/optima
  - Input variables values with the best objective function value
  - Point in the search space with the best objective function value
  - There might be more than one (multi-modal function)
  - We might be satisfied with finding one, or wanting all of them



# Vocabulary

- Local optimum/optima
  - Point with a (relatively) high value of the objective function
  - “Surrounded” by points with worse values
  - Moving away from the point could be difficult for an algorithm
  - Generally, we don’t know if it a point is a *local* or **global** optimum

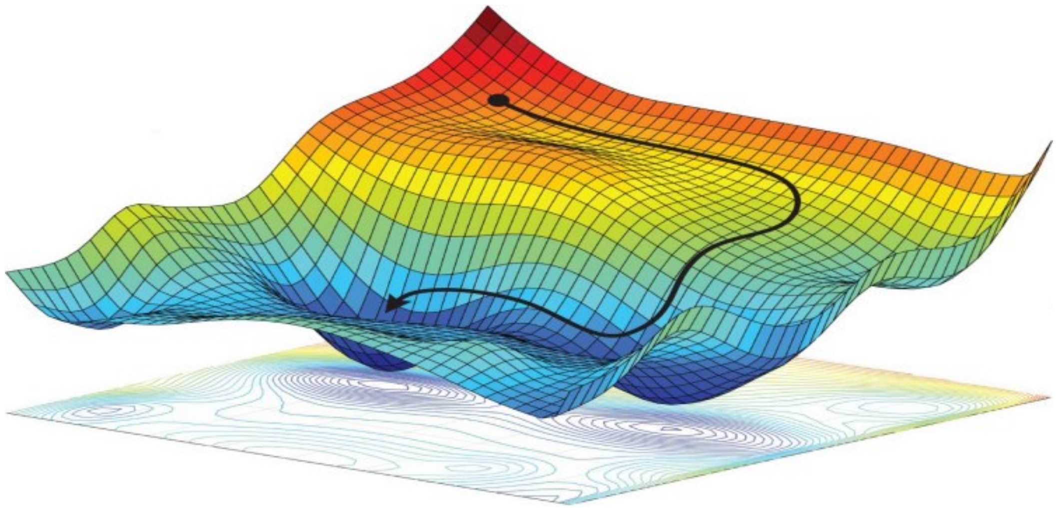


# General principles and assumptions

- We do not know much about the search space
  - Shape of the objective function (search space) is **unknown**
  - Mathematical formulation might not be possible
  - To optimize is **to explore the search space**, looking for optima
- We want to explore in an <u>efficient</u> way!
  - We cannot spend *infinite time* wandering about
  - Even a simple continuous function in one dimension has potentially *infinite points* in the search space to explore!
  - Trade-off between **quality** of solution and **time** spent

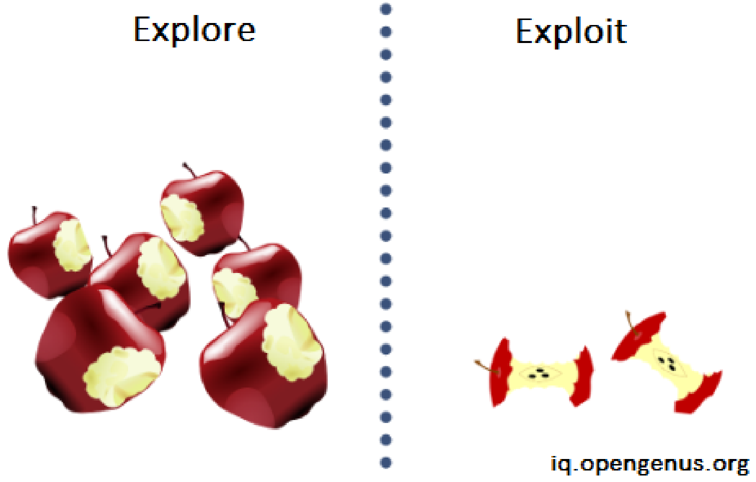
# General principles and assumptions

- Minimal requirements to be able to optimize
  - Define boundaries (min and max values of points)
  - Encode a solution in a computer (e.g. list of floating point values)
  - Describe how to move in search space (e.g. move by a small Δx in a dimension)



# General principles and assumptions

- Exploration and exploitation in iterative search algorithms
  - Initially, **explore** the search space as much as possible
  - Then, focus on/**exploit** the most promising parts found
  - Switch between exploration and exploitation is **hard to time**
  - Vocabulary: horizontal/vertical, breadth/depth, …



# Brainstorming

- Can we think about a few (simple) strategies to go through the search space and find the best possible point?

# Simple strategies

- Exhaustive search
  - Evaluate all possible variable values in search space
  - In practice, **impossible**; but a systematic (grid) search could be
- Random search
  - Randomly sample objective function in points within boundaries
  - **Does not take into account the feedback** from objective function
- Greedy search
  - Start from a point, explore neighborhood and take best point
  - Keep going until no improvement is found

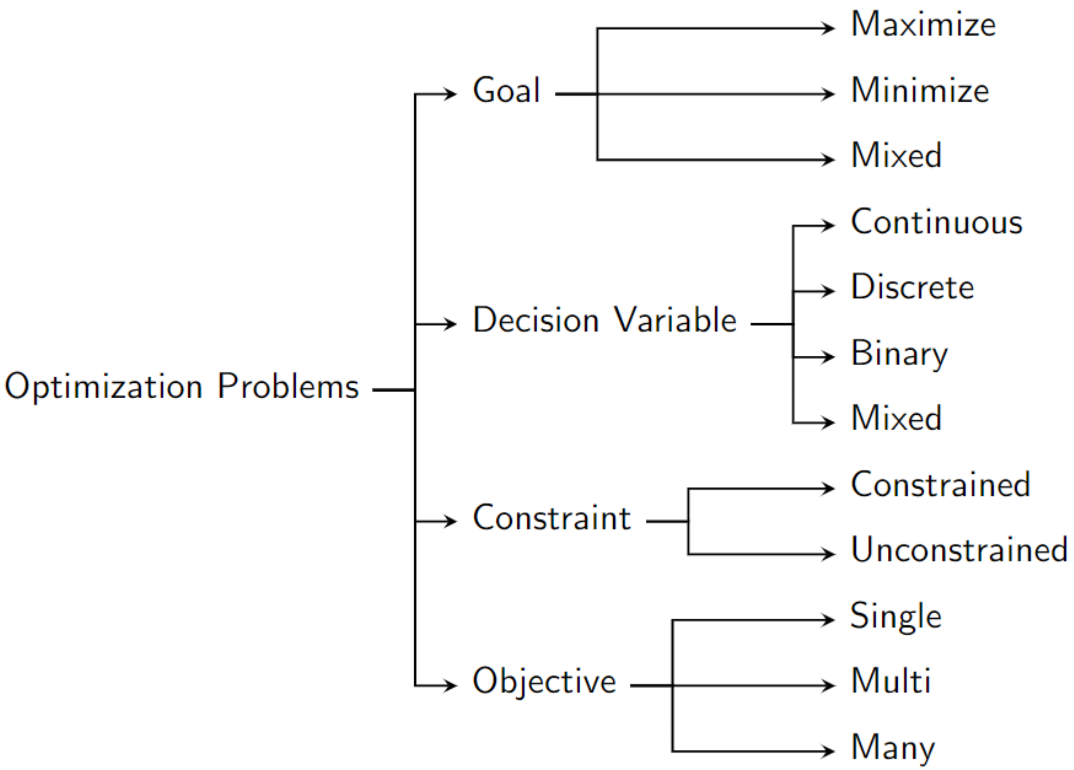
# Taxonomy of optimization methods

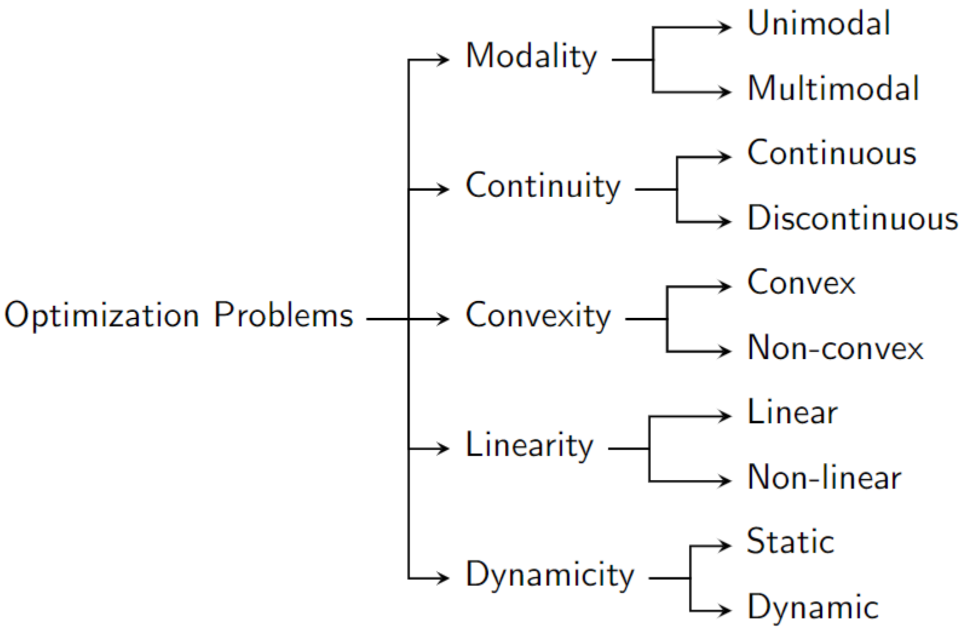
- Continuous vs Discrete
  - For discrete optimization, it becomes “choose one among many”
  - Domain is “combinatorial optimization”
  - Mixed discrete/continuous problems exist; also complex structures
- Exact vs Stochastic/Approximate
  - Exact methods guarantee convergence on a global optimum
  - Too much time or incorrect assumptions on objective function
  - Stochastic methods deliver reasonable solution in short(er) time…
  - …but they have no guarantees on whether it’s the global optimum
  - Terminology: “Stochastic’’ might also refer to stochastic variables

# Taxonomy of optimization methods

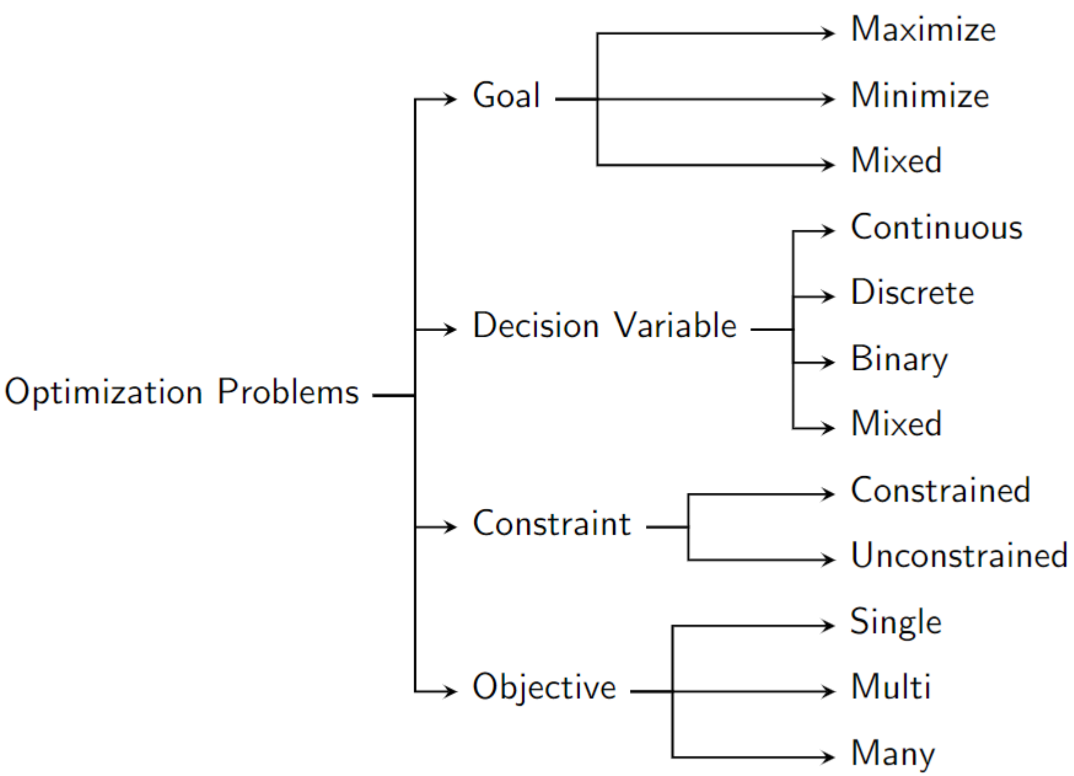
- Archive/Population vs No-Memory
  - Keep in memory a set of candidate solutions
  - Representing current “knowledge” of the search space
  - Use this knowledge to take decisions on next exploration
  - Lots of function evaluations! Also, memory occupation
- Single-objective vs Multi-objective
  - Conflicting objectives: improve one, deteriorate other(s)
  - Not searching for a single solution, but several compromises
  - “Many”-objective: 10 or more objectives (…)

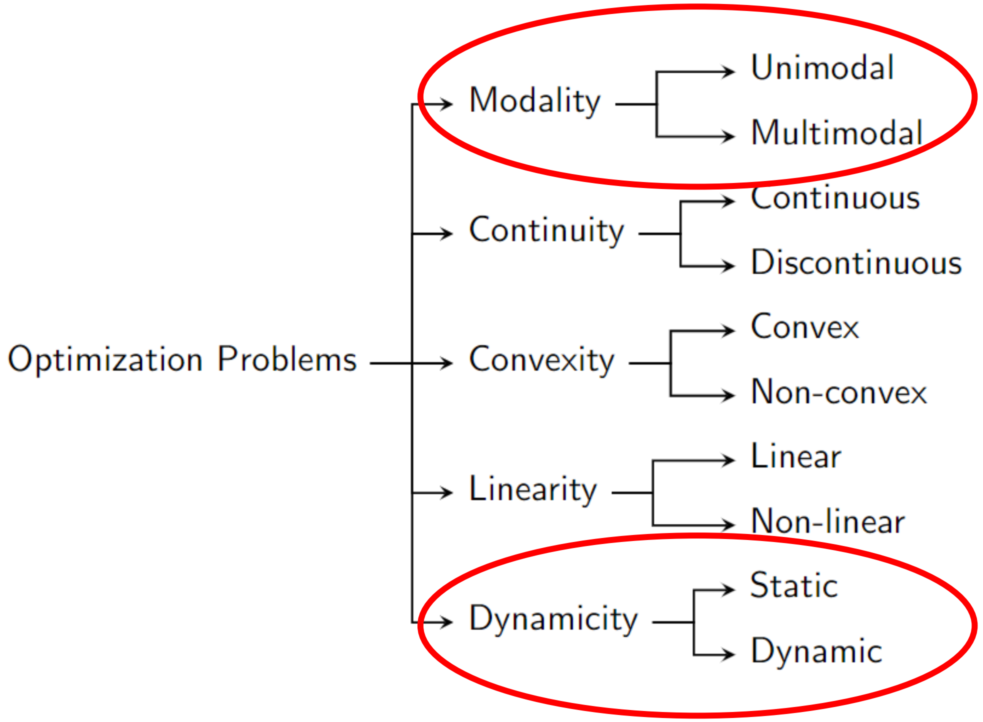
# Taxonomy of optimization problems



# Taxonomy of optimization problems



# Taxonomy of optimization problems

- Modality
  - Unimodal: there is only one global optimum, find ONE solution
  - Multi-modal: there are multiple global optima, or local optima close in value to the global optimum
  - Multi-modal: we are interested in finding ALL (or more) optima
- Dynamicity
  - Static: a regular optimization problem
  - Dynamic: the objective function CHANGES WITH TIME!
  - Objective function: $y = f(\boldsymbol{x}, t)$

# Taxonomy of optimization problems

- Computational expensiveness of objective function
  - Not expensive: extensive sampling possible
  - Expensive: surrogate models, Bayesian optimization, store list of all solutions previously evaluated…
- Objective function’s search space is deceptive/flat
  - Assumption: “good solutions are close to other good solutions”
  - If this is not true, most algorithms don’t work
  - Better off with a completely random sampling
  - Flat search space has no clues on where to move

# Real-world applications can be *weird*

- Mix of continuous and discrete variables
- Optimize graphs, trees, ensembles of trees…
- Search space can be hard to characterize
  - E.g. “optimize the shape of a car to minimize wind resistance”
  - E.g. “optimize order of visit of towns, to minimize traveling time”
  - E.g. “optimize an Assembly language program that is able to set all bits in the *ax* computer registry to zero (maximize number of bits set to zero)”

# Intended outcome



- You will have optimization problems to solve
  - Identify the typology (linear, non-linear, dynamic, static…)
  - Match with the best algorithm for the problem
  - Or get some ideas on how to design an optimization algorithm

- Very often, the best optimization algorithm is HEURISTIC
  - Heuristic is developed **ad-hoc for the target problem**
  - Employs **domain knowledge** of the problem inside algorithm

# Questions?

Bibliography

- Kochenderfer & Wheeler, *Algorithms for Optimization*, MIT Press, 2019

Images: unless otherwise stated, I stole them from the Internet. I hope they are not copyrighted, or that their use falls under the Fair Use clause, and if not, I am sorry. Please don't sue me.