Introduction to discrete optimization

Laurent Lemarchand
LISyC/UBO
Laurent.Lemarchand@univ-brest.fr

Discrete optimization

- **Many choices**
  - Not obvious … many aspects to take into consideration …


Problem : How to find the best **solution** according to some **criteria** ?

- You can enumerate all of them (discrete problem)

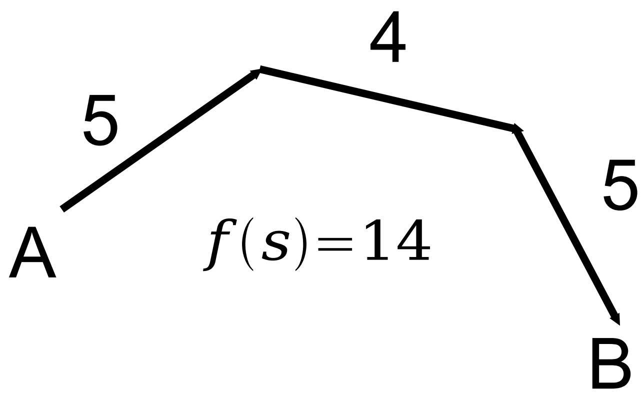
Combinatorial optimization introduction

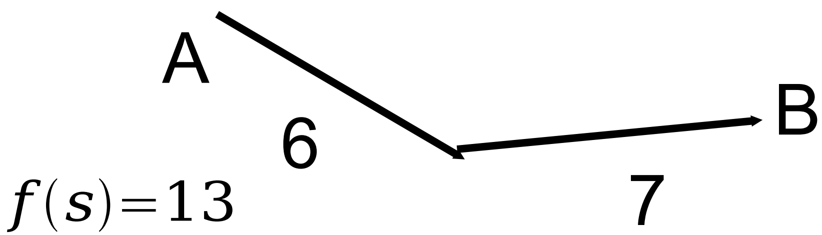
- Optimization problem : *find the best solution to a given problem among a set of feasible ones, according to some optimization criteria*
  - *S :* solution space
  - *f(S) :* function (objective function) for evaluation solution quality – can be maximized or miinimized

Oops, a single place !

Combinatorial optimization examples

- Path finding : *shortest path within a graph between couples of nodes*
  - *S :* all possible paths
  - *f(S) : path length* (to minimize !)



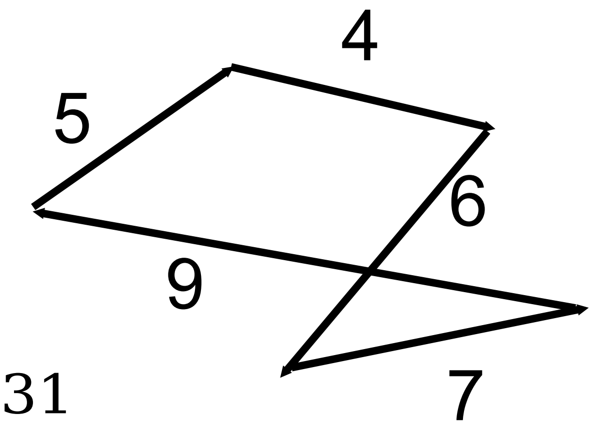


- Graph optimization

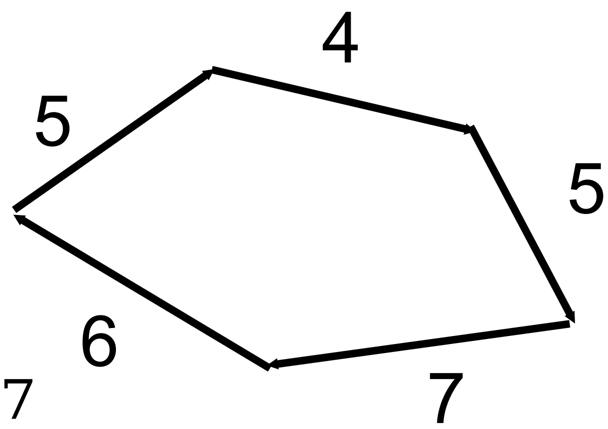
Combinatorial optimization examples

- Travelling salesman problem : *visit every town a single time and come back to the starting point*
  - *S :* all possible roundtrips (tours)
  - *f(S) : roundtrip length* (to minimize !)

V Rodin et. all



$f(s)=31$



$f(s)=27$

Combinatorial optimization examples

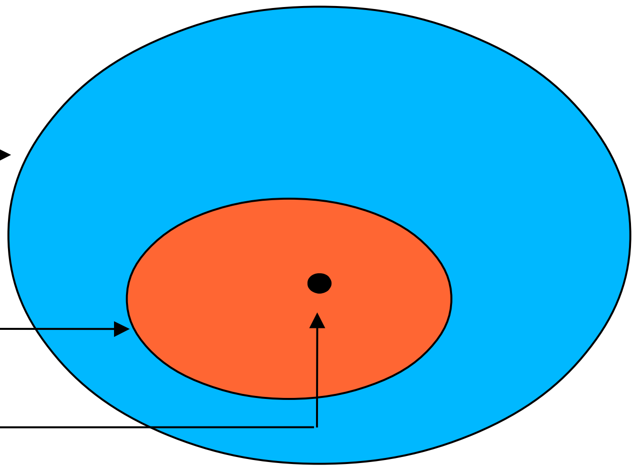
- A carpenter can make at most 6 seats and 3 tables by day (8 hours of work)
  - He sells a table \$90 (working 1h15)
  - A seat, \$50 (working 45mn)
- How to maximize his benefit ?

$$\begin{cases} 90t & + & 50c & = & f(s) \\ 75t & + & 45c & \leq & 480 \\ 0 & \leq & t & \leq & 3 \\ 0 & \leq & c & \leq & 6 \end{cases}$$

- *Linear programming :* simplex method with $O(2^n)$ complexity

Combinatorial optimization framework

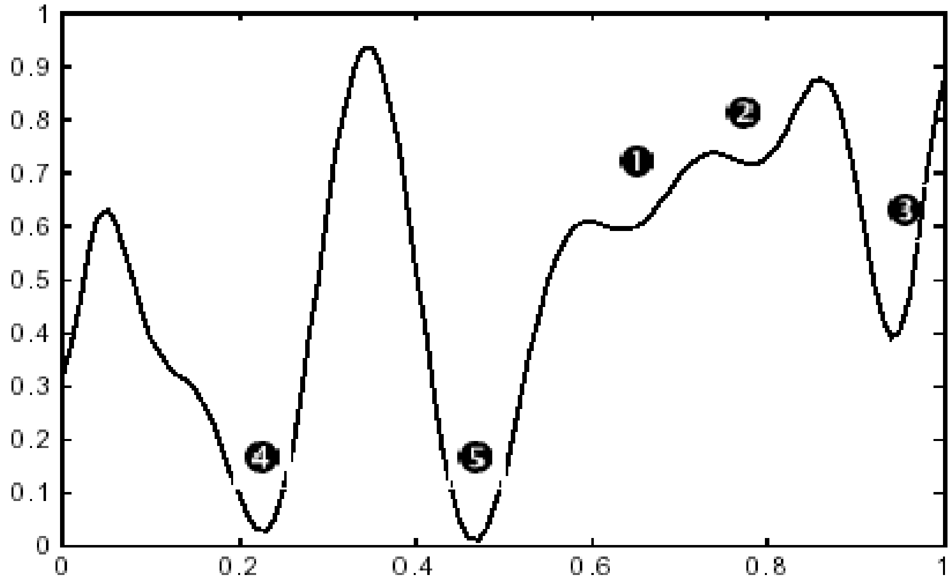
- Solution space $S \subseteq X$
- Objective function (e.g. min) $f : X \to \mathbb{R}$
- Find $s^* \in S$ s.t. $\forall s \in S \;\; f(s^*) \leq f(s)$
- X , solution space
- S , **feasible** solution space
- s^* , optimal solution



Combinatorial optimization
local sub optimal solutions

- Can fall into a local minimum as 1, 2, 3, 4, 5 (5 is the best :-)
- Must explore the whole solution space
- Not only neighbourhood
- Example : minimize a continous function on a single variable

f(x)

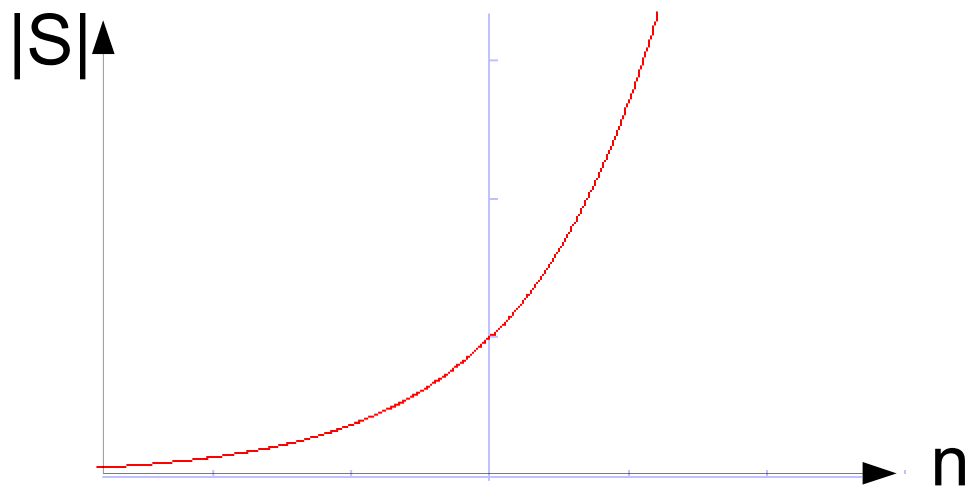


Minimize $f(x) : X \rightarrow [0..1]$

V Rodin et. all

Combinatorial optimization combinatorial explosion

- Problem of the size of S related to the size of data
  - TSP $(n-1)! / 2$
  - Bi partitioning 2^n
  - 0-1 Integer Programming 2^n



- S size is exponential

Combinatorial optimization
combinatorial explosion

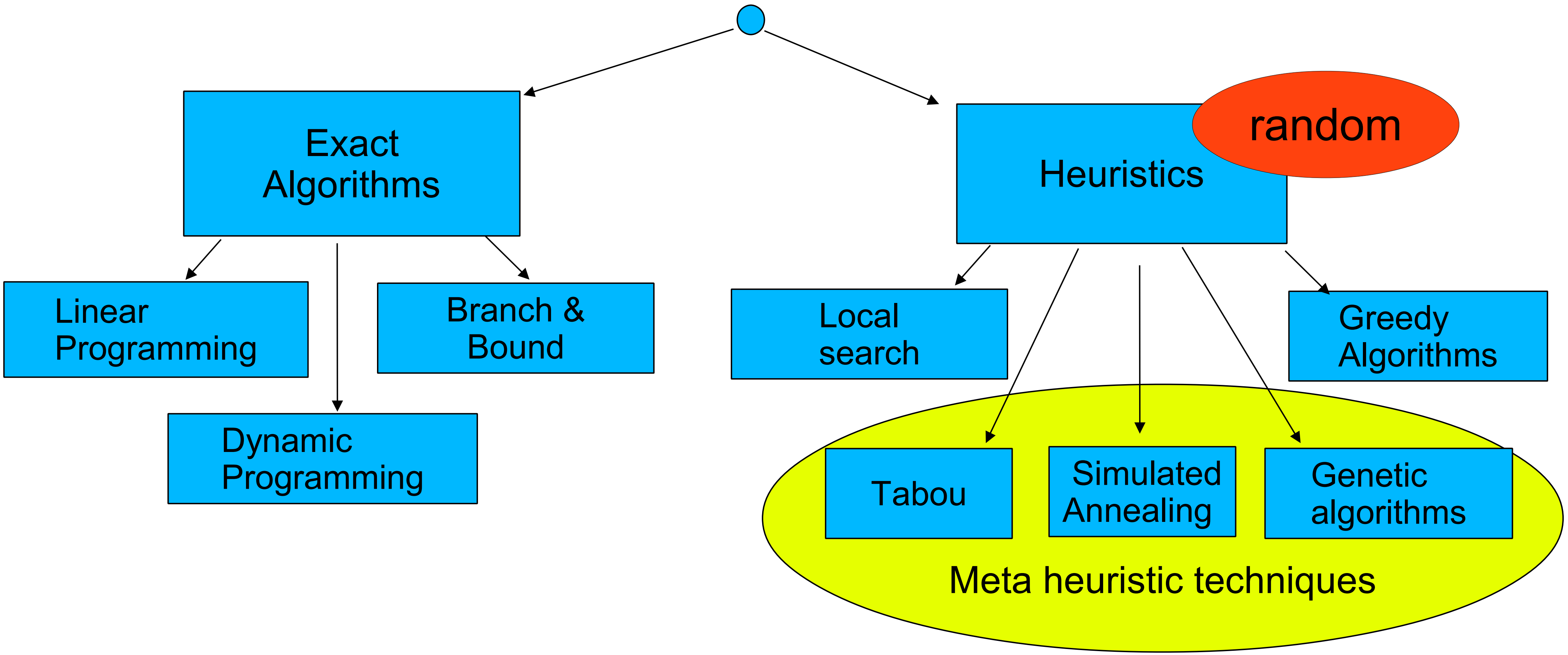
- Enumerate all solutions : often impossible

Complexity	*N = 1*	*N = 10*	*N = 100*	*N = 1000*	*N = 10000*
log N	0 ms	1 ms	2 ms	3 ms	4 ms
N	1 ms	10 ms	0.1 s	1 s	10 s
N^2	1 ms	0.1 s	10 s	17 min	28 hours
N^3	1 ms	1 s	17 min	12 days	32 years
e^N	3 ms	22 s	$9\ 10^{32}$ years !	Long time	Very long time

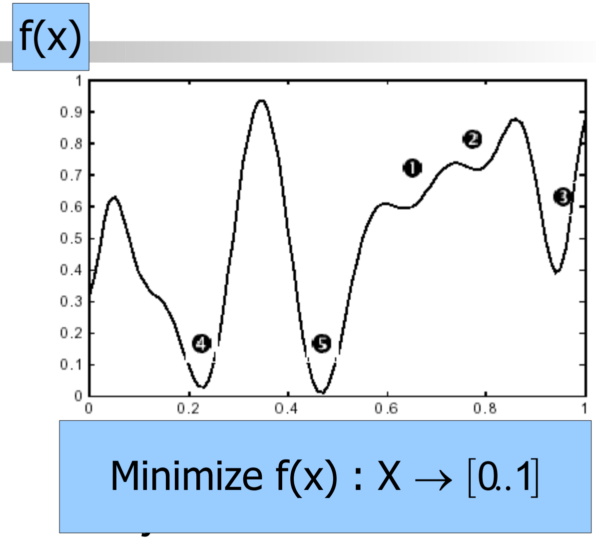
If 1000 solutions evaluated per sec

V Rodin et. all

Combinatorial Optimisation
search techniques



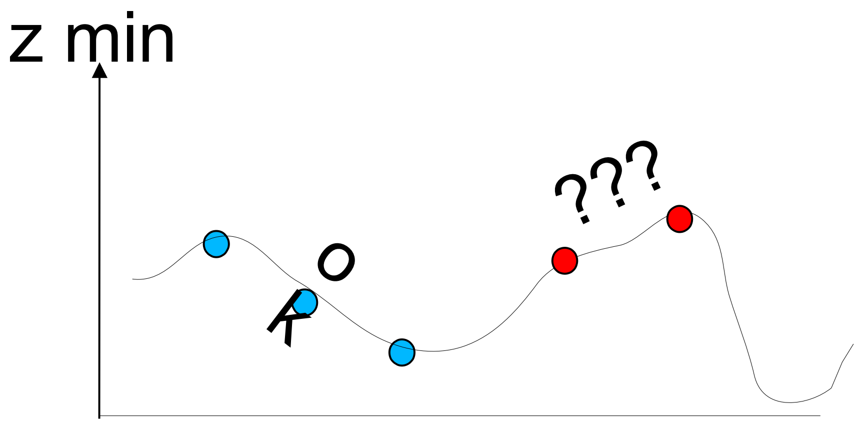
Combinatorial Optimisation
local vs. global techniques



- Remember local minimum problem
- Choice between
  - Improve current solution
  - Exploring the whole search space

Exploitation ⟷ Exploration

- Trade off politics design



Combinatorial Optimisation
exact vs. approximative techniques

- Practically speaking :
  - *Don't always need* the best solution
  - *but* have a *good* solution and eventually a guarantee on the quality loss

- If exact solution, exact method (sometimes impractical or too much time consuming)
- If appproximation
  - Heuristics (allowing discovery based on random mechanism)
  - Meta-heuristics (Frameworks for derivating specialized heuristics)

V Rodin et. all

Large scale problems
Heuristics useful

- Approximative result, but
  - Sometimes only available method (*e.g* program optimization)
  - Or exact methods for approximative model only (*e.g* circuit testing)
- Usefullness
  - Combinatorial explosion
  - Multiple or fuzzy objectives
  - Variability (robustness)
  - Fast runtimes more important than performance

Large scale problems
Parallelism

- Too large problems or need for faster runtimes
- Availability of parallel computers (multicore, NOW)

- Possible parallelization
  - Search space partitioning : positive or negative anomalies favorables ou défavorables depending on search strategy and fitness function
  - Centralized or distributed implementation
  - Z update problem