Heuristic Optimization

Introduction and Simple Heuristics

José M PEÑA (jmpena@fi.upm.es)
(Universidad Politécnica de Madrid)

Outline

1. What are optimization problems?
2. Exhaustive vs. Heuristic approaches
3. Typical problems

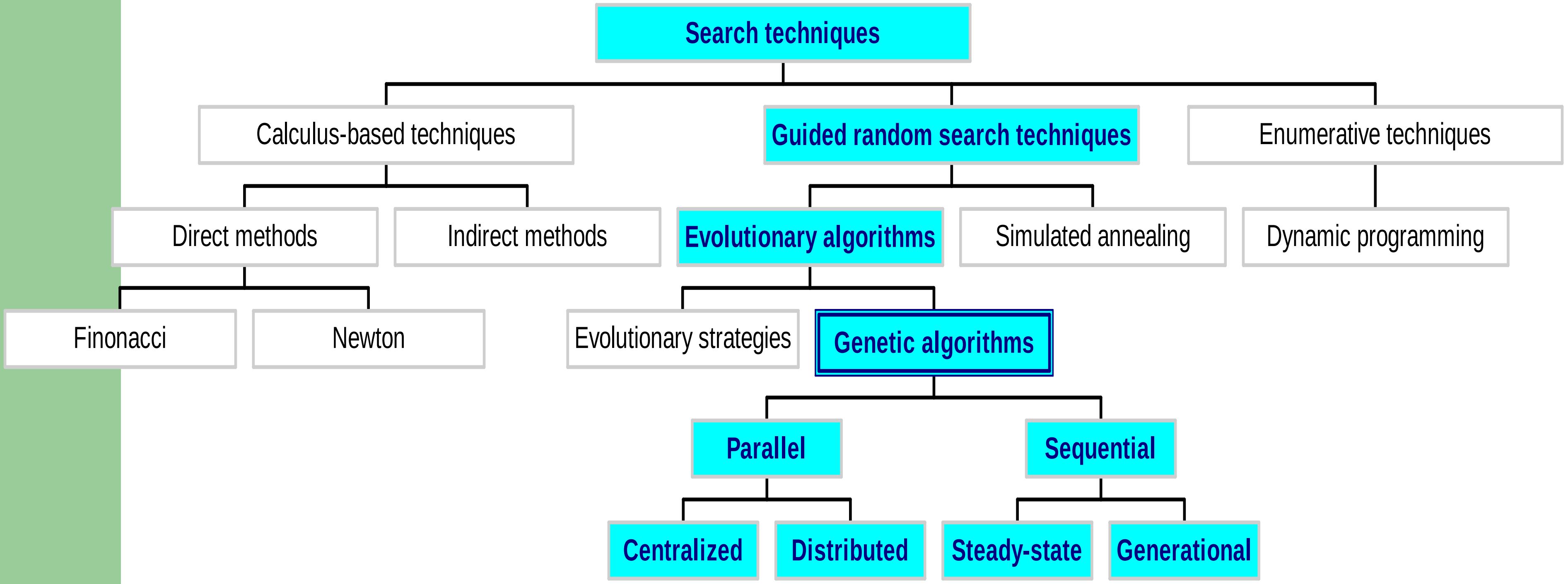
Optimization

- Search for the “best” configuration of a set of variables to achieve some goals
- Two categories:
  - Discrete variables
  - Real-valued variables
- Each optimization problem is specified defining
  - Possible solutions (states)
  - Objective

Optimization

- A **optimization problem** is a pair (X, f) being X the search space (all the possible solutions), and f a function
  - $f : X \to R$
- The solution $x_{opt} \in X$ is optime if
  - $f(x) \leq f(x_{opt}), \ \forall x \in X$

Optimization Methods



© W. Williams, '99

Heuristics vs Meta-Heuristics

- Heuristic derives from the greek verb heuriskein (ευρισκειν) which means “to find”
- Faster than mathematical optimization (branch & bound, simplex, etc)
- “Meta” means “beyond, in an upper lever”
  - Meta-heuristics are strategies that “guide” the search process
  - The goal is to explore the search space in order to find (near-)optimal solutions
  - Meta-heuristics are not problem-specific
  - The basic concepts of meta-heuristics permit an abstract level description
  - They may incorporate mechanisms to avoid getting trapped in confined areas of the search space

Classification

- Heuristic
  - Constructive algorithms (greedy)
  - Local search algorithms (hill-climbing…)
- Meta-heuristic
  - Trajectory methods: Describe a trajectory in the search space during the search process
    - Variable Neighbourhood Search
    - Iterated Local Search
    - Simulated Annealing
    - Tabu Search
  - Population-based: Perform search processes which describe the evolution of a set of points in the search space
    - Evolutionary computation (Genetic Algorithms)

Greedy

- Generate solutions from scratch by adding (to an initially empty partial solution) components, until the solution is complete
- A greedy algorithm works in phases. At each phase:
  - You take the best you can get right now, without regard for future consequences
  - You hope that by choosing a *local* optimum at each step, you will end up at a *global* optimum
- Example (Scheduling):
  - 3 processors
  - 9 jobs (3, 5, 6, 10, 11, 14, 15, 18 and 20 minutes)

Local Search

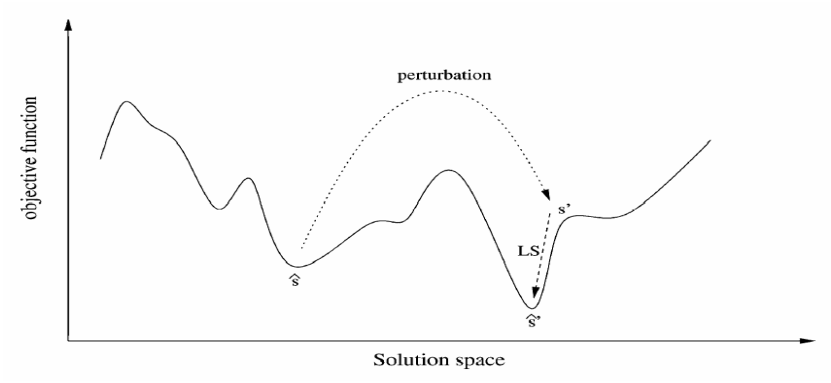
- Iterative algorithm:
  - Start from some initial solution
  - Explore the neighbourhood of the current solution
  - Replace the current solution by a better solution
- Different procedures depending on choice criteria and termination criteria
  - *Stochastic*: choose at random
  - *Hill climbing*: only permit moves to neighbours that improve the current
    - *Greedy* – the best neighbour
    - *Anxious* – the first neighbour improving
    - *Sideways moves* – allows moves with same fitness

Local Search: Problems

- Success of hill-climbing depends on shape of landscape
  - Shape of landscape depends on problem formulation and fitness function
  - Landscapes for realistic problems often look like a worst-case scenario
  - NP-hard problems typically have exponential number of local-minima
- Failing on neighbourhood search
  - Propensity to deliver solutions which are only local optima
  - Solutions depend on the initial solution

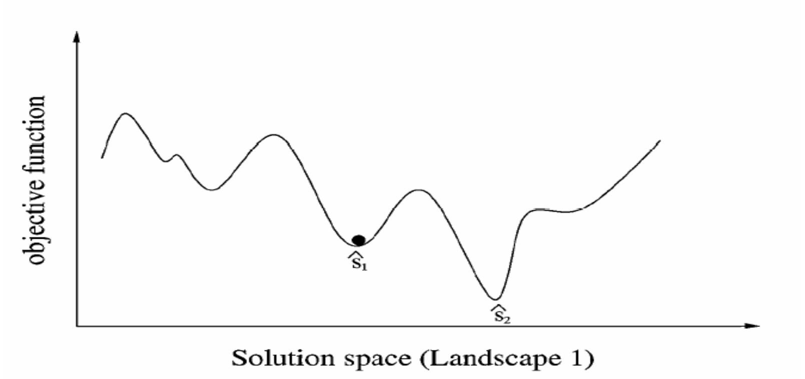
Local Search: Solutions

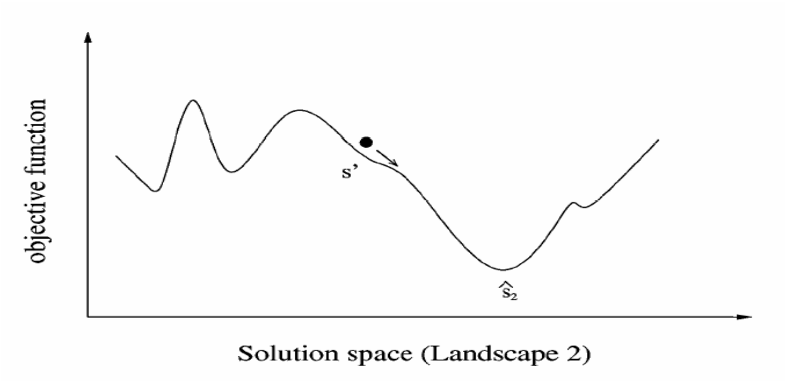
- Applies local search to an initial solution until it finds the local optimum; then it perturbs the solution and it restarts local search



Local Search: Solutions

- Strategy based on dynamically changing neighborhood structure.
- VNS: Different coding schemas represent different neighborhood relations





Tabu Search

- Steepest descent with *memory*
  - Moves through solution space
  - Uses memory techniques to avoid cycling
- “The overall approach is to avoid entrainment in cycles by forbidding or penalizing moves which take the solution, in the next iteration, to points in the solution space previously visited” (Fred Glover, *Computers and Operations Research*, 1986)

Tabu Search

- To avoid revelsal moves the last moves are maked as Tabu
  - Change value of x from false to true:
    [x = false] is tabu
  - Swap elements i and j:
    [Change j,i] is tabu
  - Drop i and add j:
    [add i] and [drop j] are tabu

Tabu Search: Pros and cons

- Pros:
  - Tabu Search yields relatively good solutions to previously intractable problems
  - Tabu Search provides comparable or superior solutions to other optimization techniques
- Cons:
  - Tabu Search does not guarantee optimality
  - Tabu Search is awkward for problems with continuous variables
  - Tabu Search assumes fast performance evaluation
  - The construction of tabu list is heuristic

Simulated Annealing

- Based on the cooling of material in a heat bath
- Local Search but we allow moves resulting in solutions of worse quality than the current solution
  - Avoid from local solutions

Simulated Annealing

```
s=GenerateInitialSolution()
e=Energy(s)
t=t(0)
while Termination criteria not met
    s’=PickAtRandom( N(x) ); e’=Energy(s’);
    If random(0,1) < P(e,e’,T) then
        s=s’
    Endif
    Update(T)
end while
```

If e’>e (worse)

  $P(e,e',T)>0$

When $T \rightarrow 0$

  $P(e,e',T) \rightarrow 0$