Optimization methods

Summer School 2015 - Angers

Laetitia JOURDAN

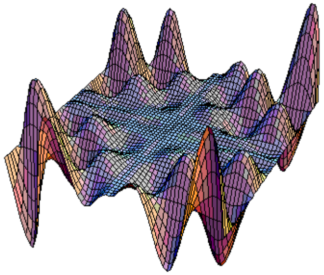
Positionning

* **Bioinformatics**
* The creation and development of advanced information and computational techniques for solving problems in biology

* **Combinatorial optimization**
* Solve instances of problems that are believed to be hard in general, by exploring the usually-large solution space of these instances

Definition combinatorial Optimization

* Wikipedia
  * Combinatorial optimization is a topic in theoretical computer science and applied mathematics that consists of finding the least-cost solution to a mathematical problem in which each solution is associated with a numerical cost.



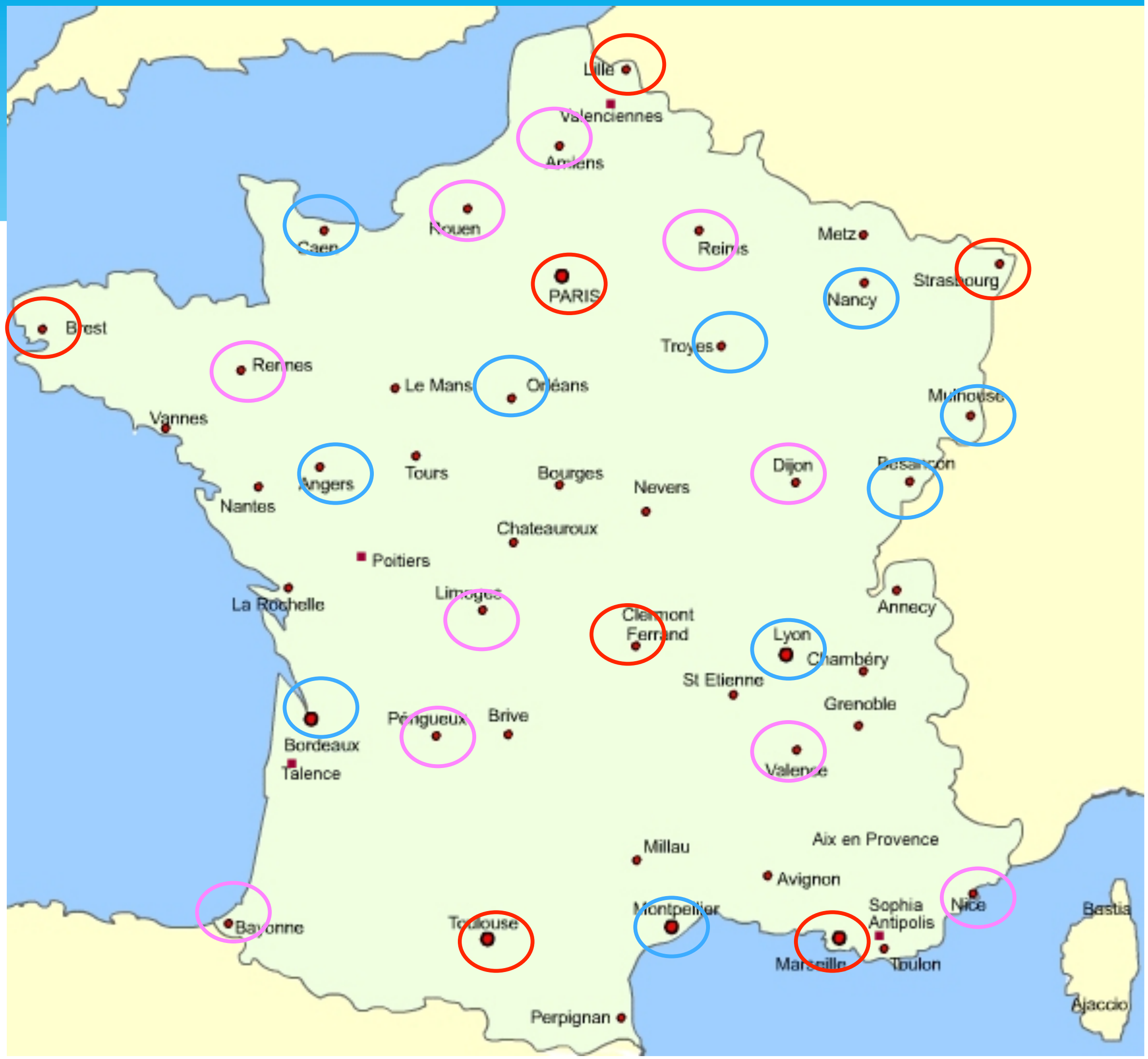
$$
\textbf{(P)} \quad \begin{array}{ll} & Opt\ F(x) \longrightarrow \text{Cost = objective function (min/max)} \\ \text{s.c.} & x \in C \longrightarrow \text{Set of feasible solutions defined using constraints} \end{array}
$$

Combinatorial problem

* Model: different elements to be defined
  * Solutions
    * How to characterize a solution?
    * How to define feasible solutions?
  * Objective function
    * What is the criterion to optimize (cost, duration…)?
    * Is there only one criterion?

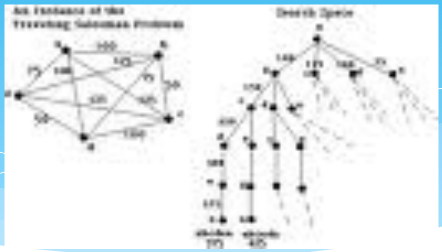
Classical problems

* Assignment problem
* Closure problem
* Constraint satisfaction problem
* Cutting stock problem
* Integer programming
* Knapsack problem
* Minimum spanning tree
* Nurse scheduling problem
* Vehicle routing problem
* Vehicle rescheduling problem
* Weapon target assignment problem



Lille
Valenciennes
Amiens
Rouen
Caen
PARIS
Reims
Metz
Strasbourg
Nancy
Brest
Troyes
Rennes
Le Mans
Orléans
Mulhouse
Vannes
Tours
Angers
Nantes
Bourges
Nevers
Dijon
Besançon
Chateauroux
Poitiers
La Rochelle
Limoges
Clermont Ferrand
Annecy
Lyon
Chambéry
St Etienne
Grenoble
Périgueux
Brive
Bordeaux
Talence
Valence
Millau
Aix en Provence
Avignon
Sophia Antipolis
Nice
Bastia
Bayonne
Toulouse
Montpellier
Marseille
Toulon
Ajaccio
Perpignan

The traveling salesman problem



* NP-hard problem
  * No efficient (polynomial) algorithm

* Simple resolution:
  Exhaustive enumeration of all solutions
  If N cities ➔ (N-1)! Possibilities

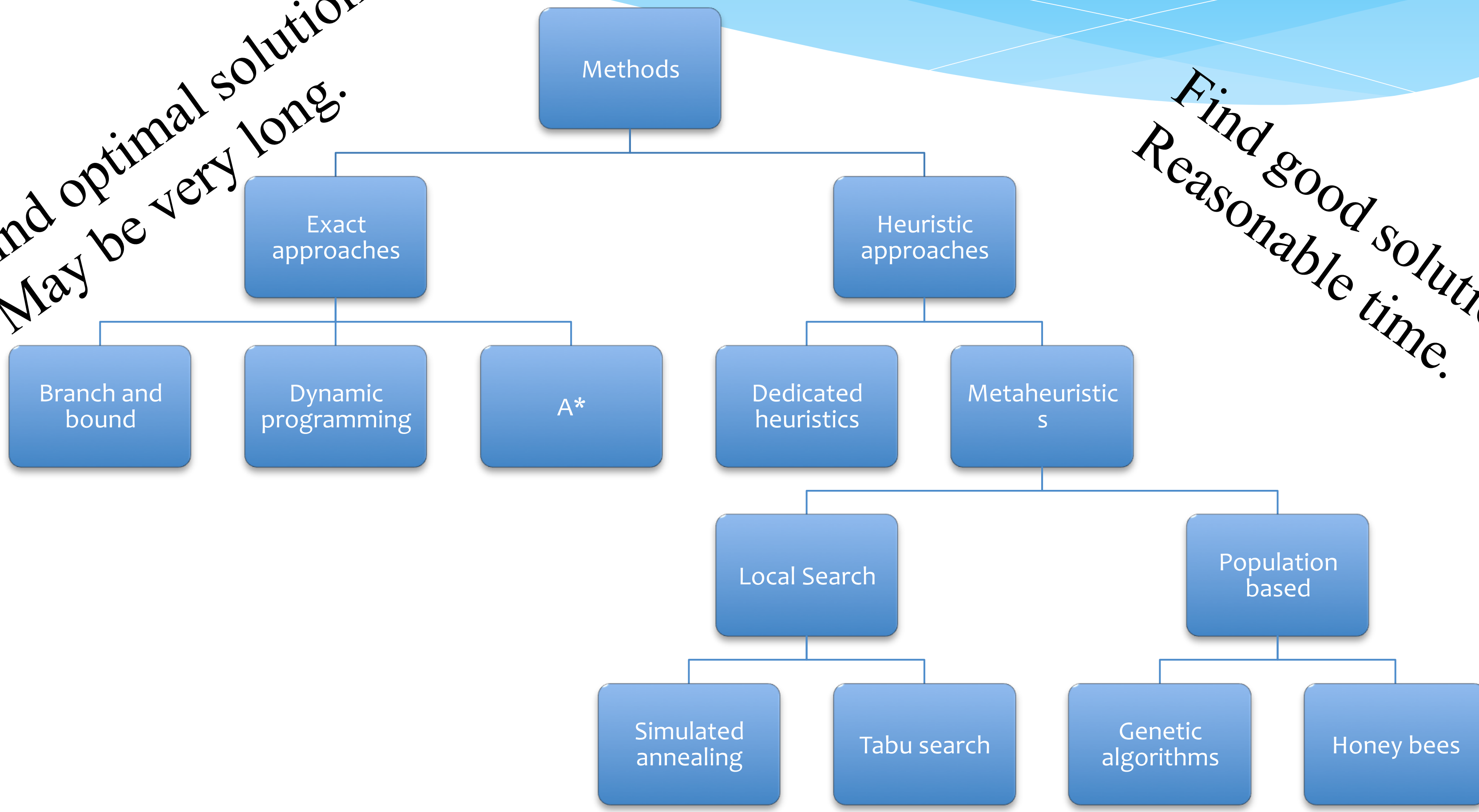
  * Ex : 5 cities ➔ 12 possibilities **6 µsec**
  * 10 cities ➔ 181 440 possibilities **0,09 sec**
  * 20 cities ➔ 60 × 1015 **964 years**

* Let’s suppose a computer requires 1/2 microsecond to evaluate a tour.

Need efficient combinatorial optimization methods

Combinatorial optimization methods

Find optimal solution.
May be very long.



Find good solution.
Reasonable time.

Metaheuristics

* Wikipedia
  * In computer science, metaheuristic designates a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality.
  * Metaheuristics make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions.
  * However, metaheuristics do not guarantee an optimal solution is ever found.
  * Many metaheuristics implement some form of stochastic optimization.
* Classical metaheuristics:
  * Descent method, Tabu search, Simulated Annealing, Genetic Algorithms

The Concept of Metaheuristic Algorithms

* Metaheuristics comes from greek words
  * Meta: beyond means higher level
  * Heuriskein (heuristics) means to find. (Glover, 1986)
* cleverness of Metaheuristics will come from two complementary notions:
  * diversification: exploration of the search space
  * intensification: exploitation of the accumulated search experience
* To solve an optimization problem we need to define three elements :
  * a search space
  * an objective function
  * a neighborhood

Encoding

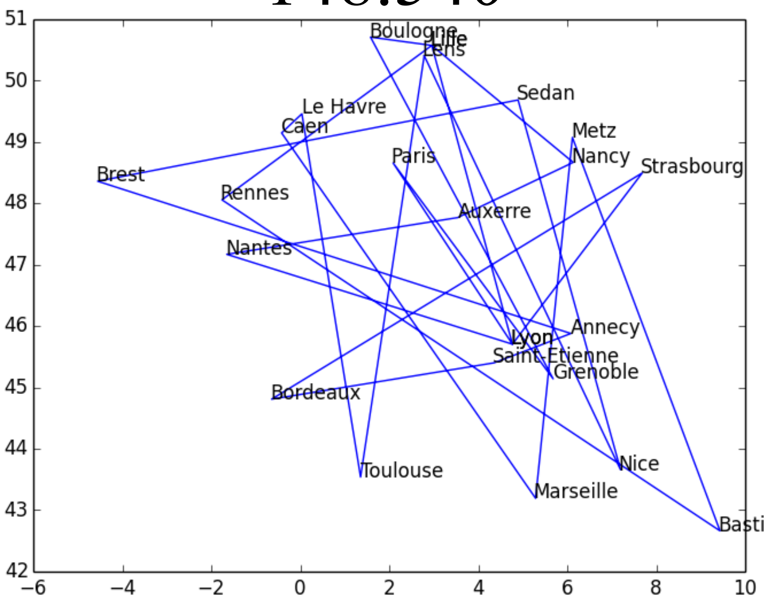
- TSP problem with n towns: permutation of numbers, from 1, …, n. 1number=1town.
  - Size of the search space= n!
  - TSP symetrical : n! / 2
  - Same start/end: (n-1)! / 2

1	5	2	4	3	7	6

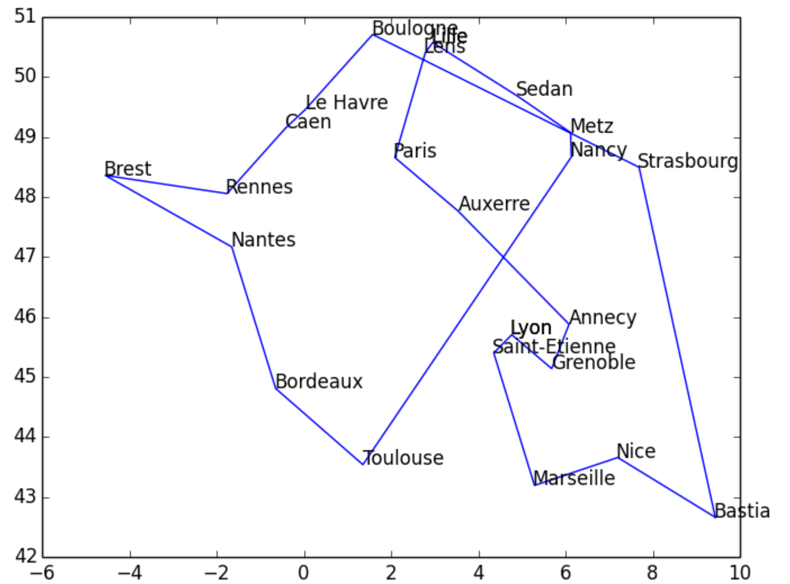
Evaluation

* Quality of the solution = length of the tour

148.540



54.877



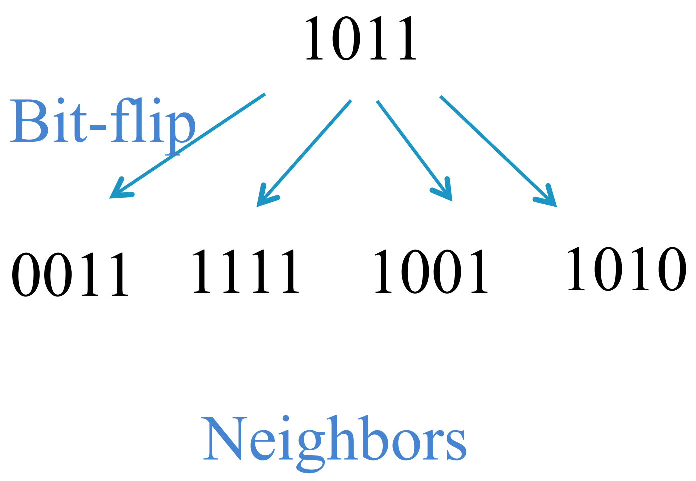
Descent method

* Hill climbing
* Gradient method

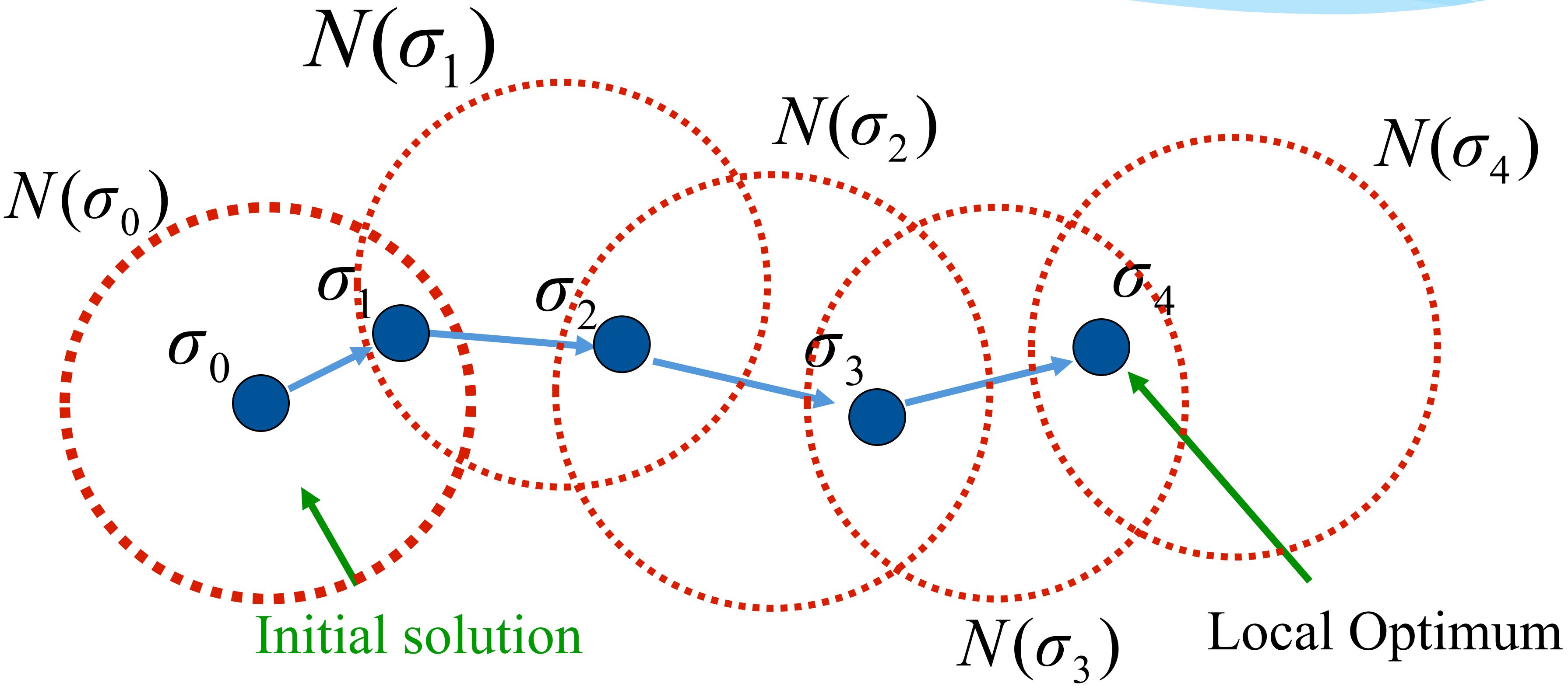
* Neighborhood notion
  * Small modification
  * Local search

* Landscape representation

* From an initial solution
  * Look for a best neighbor
  * Move to this neighbor
  * When no better neighbor → local optimum

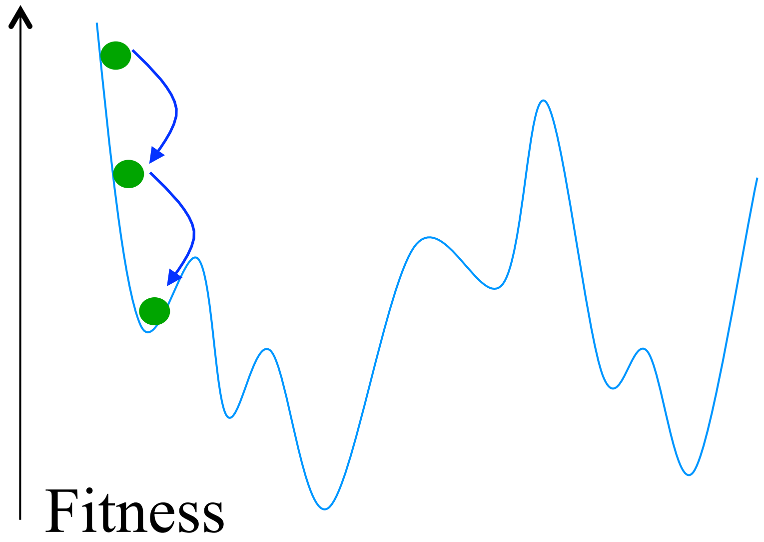


Descent method



Descent method

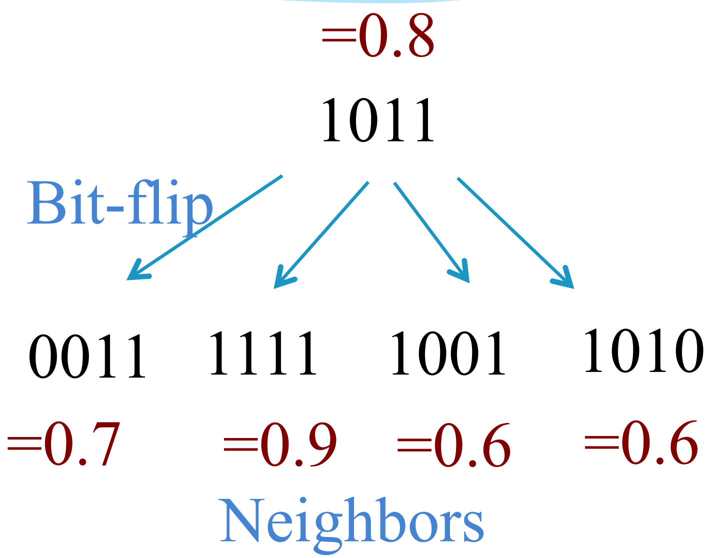
  - Hill climbing
  - Gradient method
- Neighborhood notion
  - Small modification
  - Local search
- Landscape representation
- From an initial solution
  - Look for a best neighbor
  - Move to this neighbor
  - When no better neighbor → local optimum



Minimization problem

Local search and neighbors

* Should we make a thorough search and examine all neighbors ?
  * no : less computation, best first search
  * yes : more computation,
* what to do if there are several neighbors of same quality ? (Parallelization)



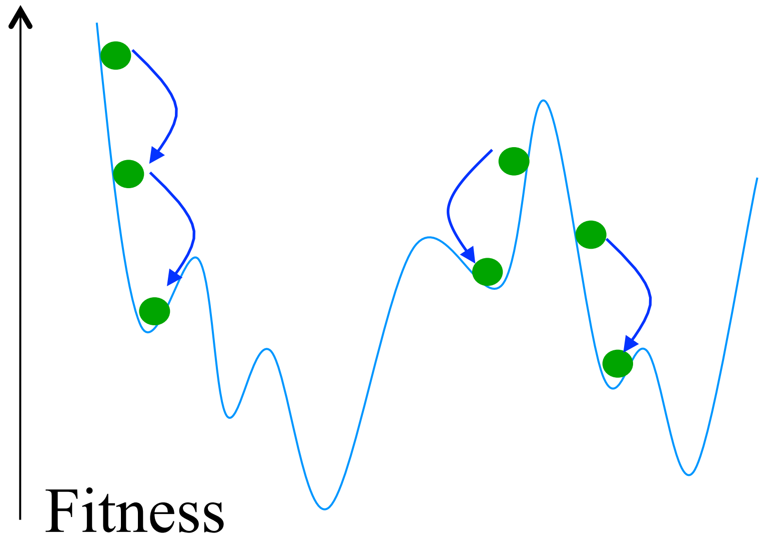
Background - OR
Local optimum

- Hill climbing
- Gradient method

- Neighborhood notion
  - Small modification
  - Local search

- Landscape representation

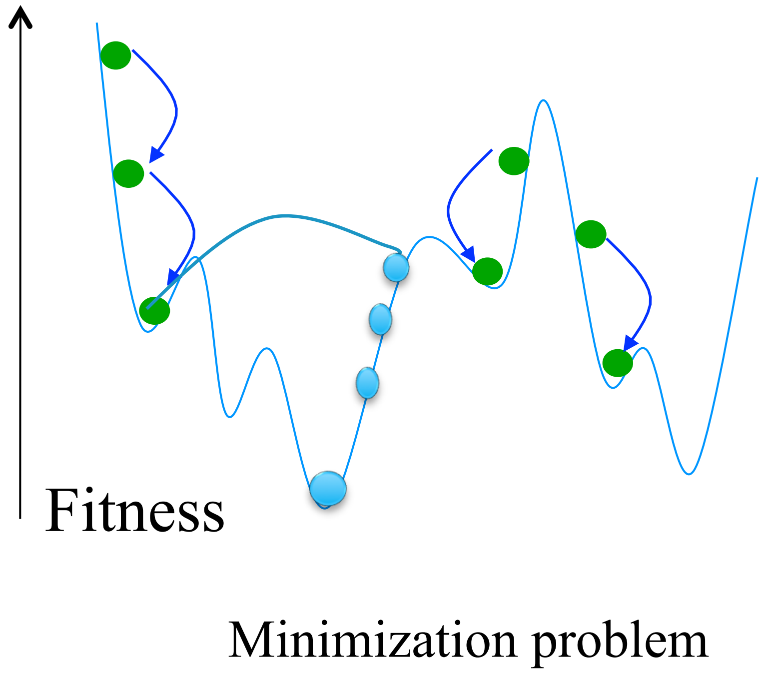
- From an initial solution
  - Look for a best neighbor
  - Move to this neighbor
  - When no better neighbor → local optimum



Minimization problem

How to escape

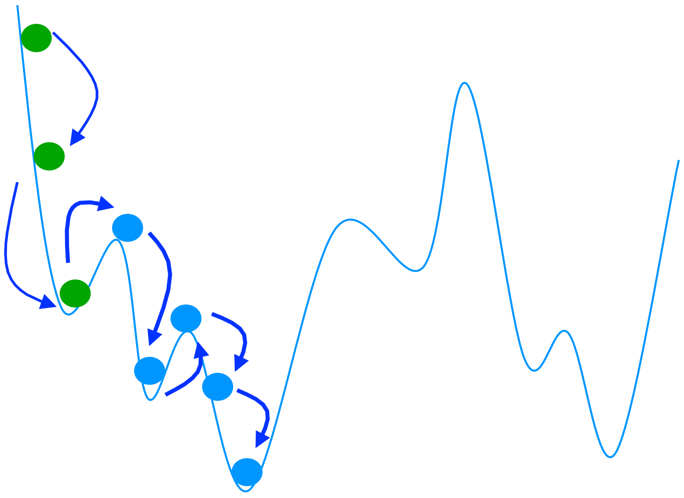
* accept neighbors of same quality
* accept neighbors of lower quality (SA)
* start search from new configuration
* Iterated Local search
    * perturbation of current configuration followed by LS
* change the neighborhood (VNS)
* change the objective function



Tabu search [Glover, 1986]

- From an initial solution
  - Look for a best neighbor
  - Move to this neighbor
  - When no better neighbor
    - May degrade the solution
    - Interdiction to come back to recently visited solution (tabu solutions)

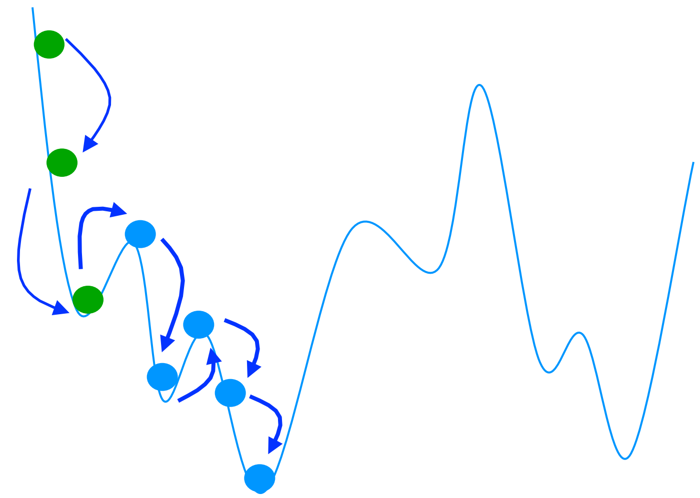
  - Parameters:
    - Tabu move
    - Size of the Tabu list (short term memory)



Simulated annealing [Kirkpatrick, 1983]

* Name inspired from annealing in metallurgy

* From an initial solution
  * Look for a neighbor
  * If better solution
    * Move to this neighbor
  * If not
    * Accept to move to this neigbhor according to a probability that depends
      on a temperature T
  * Parameters:
    * Management of temperature T

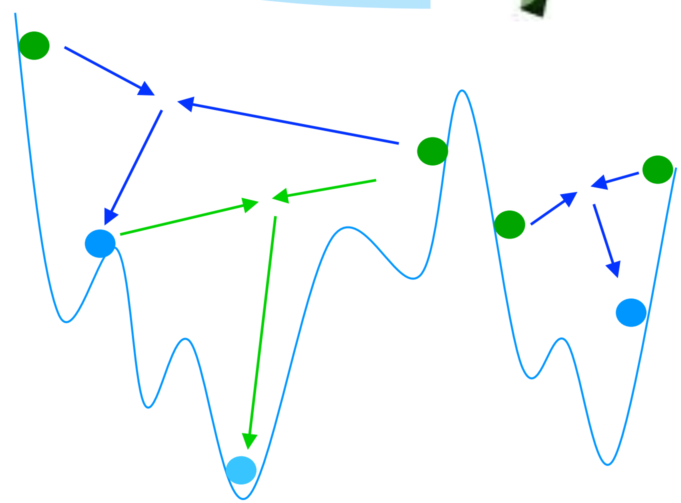


Evolutionary algorithms

* Apply principles of Darwin and Lamark to resolution of problems in computer science :
  * evolution is based on competition and selection
  * that leads to survival of the fittests,
  * transmission of acquired characters to descendants
  * mutation of genes
* Main approaches
  * Evolutionary Programming
  * Evolution Strategy
  * Genetic Algorithms (GA)

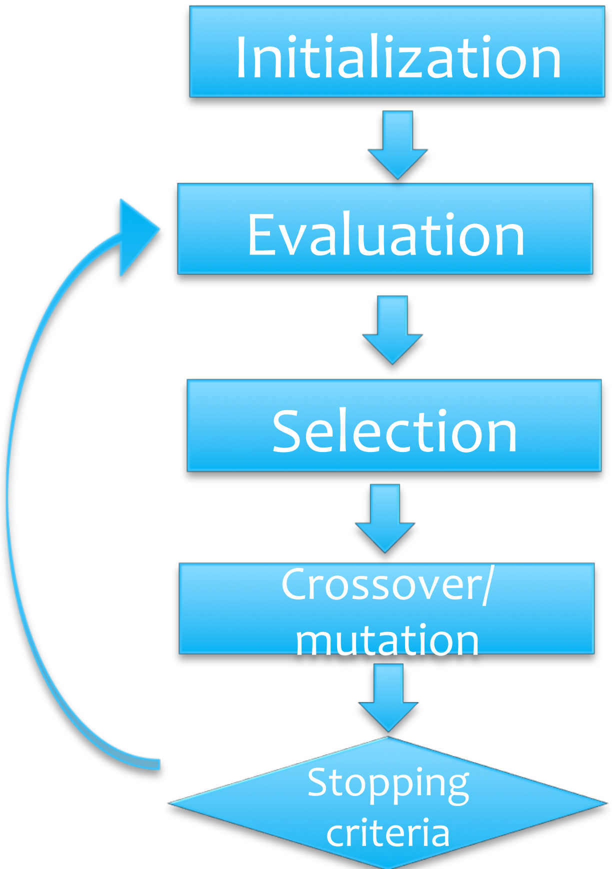
Genetic algorithm [Holland, 1975]

- Population based (set of solutions)
- Inspired by natural evolution
  - Inheritance
  - Selection
  - Mutation …
- Global improvement
- Components
  - Encoding technique (gene, chromosome)
  - Initialization procedure (creation)
  - Evaluation function (environment)
  - Selection of parents (reproduction)
  - Genetic operators (mutation, recombination)
  - Parameter settings (practice and art)



Simple GA

```
initialize population;
evaluate population;
while TerminationCriteriaNotSatisfied
{
    select parents for reproduction;
    perform recombination and mutation;
    evaluate population;
    Maintain population
}
}
```



Evaluation

* 1 Individual = a solution = a chromosome = genotype

1	0	1	1	0	1	1

gene

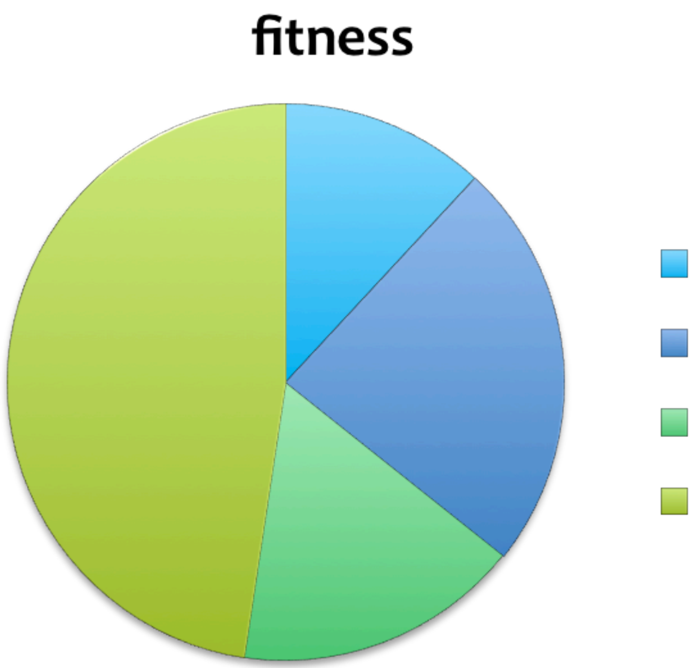
* 1 individual quality = phenotype = fitness

* Evaluation = problem dependent

Selection

* a proportion of the existing population is selected to breed a new generation.

GA Parent Selection - Roulette Wheel

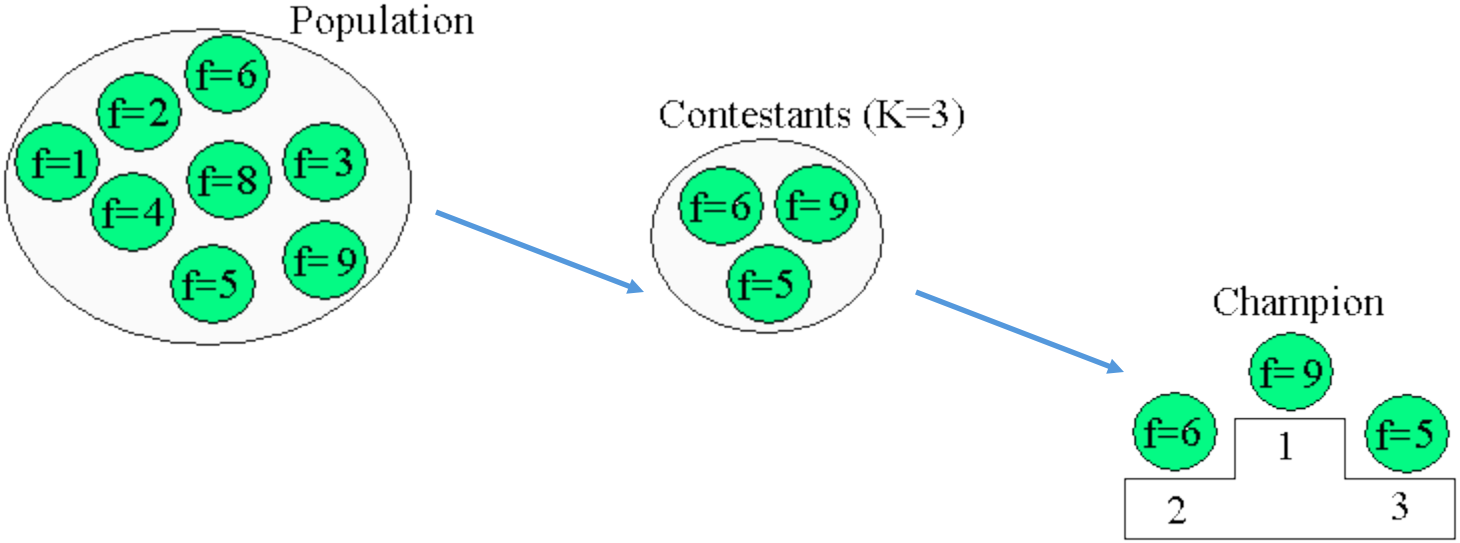


Roulette Wheel Selection

* Sum the fitnesses of all the population members, TF
* Generate a random number, m, between 0 and TF
* Return the first population member whose fitness added to the preceding population members is greater than or equal to m

GA Parent Selection - Tournament

* Select k individuals at random. The individual with the highest evaluation becomes the parent. Repeat to find a second parent



GA - Mutation

* Local modification (as neighbor)
* Causes movement in the search space (local or global)
* Restores lost information to the population

1	0	1	1	0	1	1

Bit flip mutation

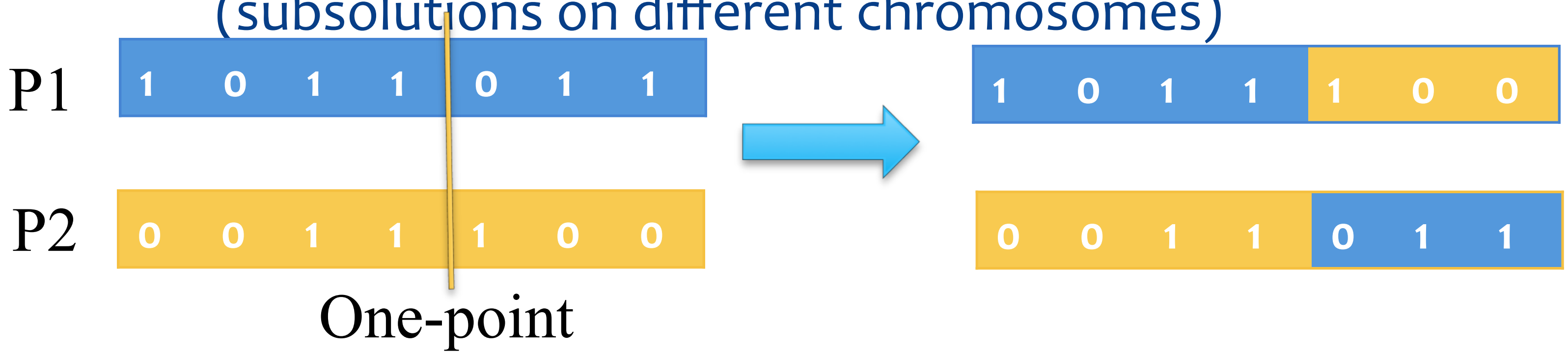
1	0	1	1	1	1	1

GA – maintain population

* Deletion
  * Delete-All : Deletes all the members of the current population and replaces them with the same number of chromosomes that have just been created
  * Steady-State : Deletes n old members and replaces them with n new members; n is a parameter
    But do you delete the worst individuals, pick them at random or delete the chromosomes that you used as parents?
  * Steady-State-No-Duplicates : Same as steady-state but checks that no duplicate chromosomes are added to the population. This adds to the computational overhead but can mean that more of the search space is explored

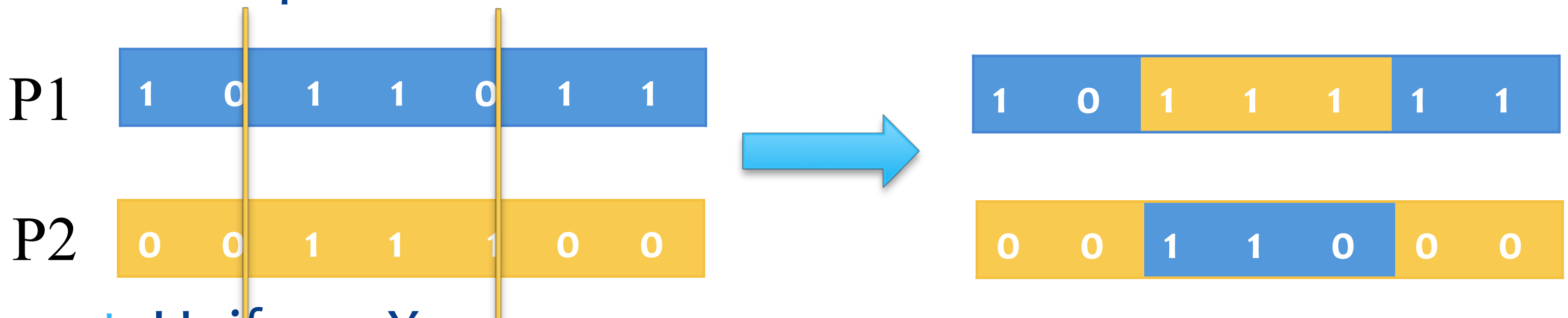
Crossover/recombination

* combine two or more individuals to get one or a set of children
* is a critical feature of genetic algorithms:
  * It greatly accelerates search early in evolution of a population
  * It leads to effective combination of schemata (subsolutions on different chromosomes)

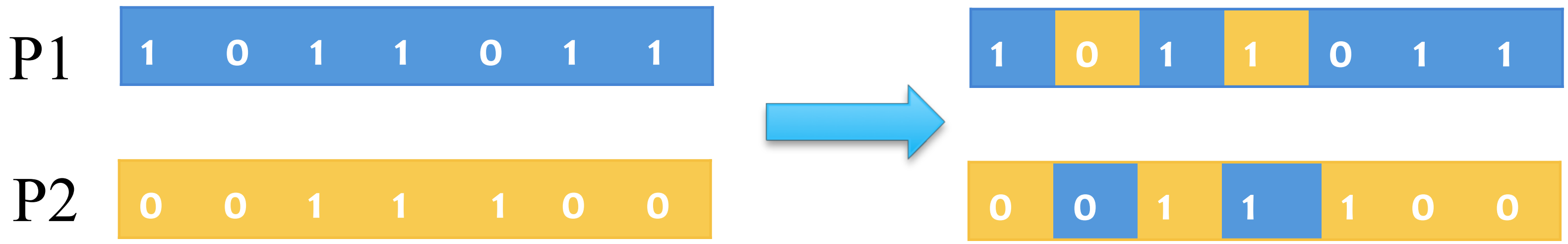


Crossover/recombination

* Two points Xover



* Uniform Xover



Other paradigms

* ANTS based metaheuristics (ACO): behavior of ants seeking a path between their colony and a source of food
* PSO based metaheuristics (particle swarm, birds, fishes)
* Bees: mimics the food foraging behaviour of swarms of honey bees
* Firefly Algorithm : attractiveness is proportional to the light intensity seen by adjacent fireflies
* Cuckoo Search : based on the brood parasitism of some

cuckoo species
* Monkey Search : inspired by the behavior of a monkey climbing trees looking for food
* Harmony Search : inspired by the improvisation process of musicians

Background - OR
Multi-objective optimisation:
Motivations

* Many real world problems are multi-objective by nature
* Objectives may be in conflict
* Not always possible to construct a single criterion

Background - OR
Multi-objective Main concepts

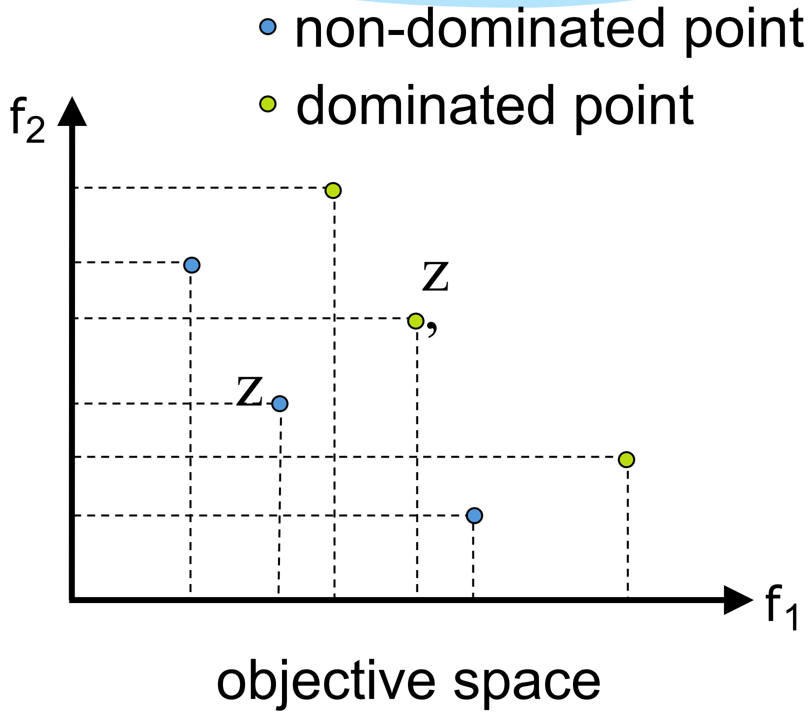
* Multi-objective Optimization Problem (MOP):

$$(MOP) = \begin{cases} \min \text{ (or max) } f(x) = (f_1(x), f_2(x), \ldots, f_n(x)) \\ \text{Subject to } x \in X \end{cases}$$

  * $n \geq 2$ objective functions $f_1(x), f_2(x), \ldots, f_n(x)$
  * $x \in X$ is a decision vector $(x_1, x_2, \ldots, x_k)$
  * X is the set of feasible solutions in the decision space
  * Z is the set of feasible points in the objective space

Dealing with multiple objectives

- Definitions:
  - z ∈ Z dominates z' ∈ Z iff ∀i ∈ [1..n], zi ≤ zi' and ∃j ∈ [1..n], zj < zj' .
  - z ∈ Z is a non-dominated vector if there does not exist another z' ∈ Z such that z' dominates z.
  - The Pareto frontier is the set of all non-dominated points.
  - The efficient set is the set of all efficient solution.

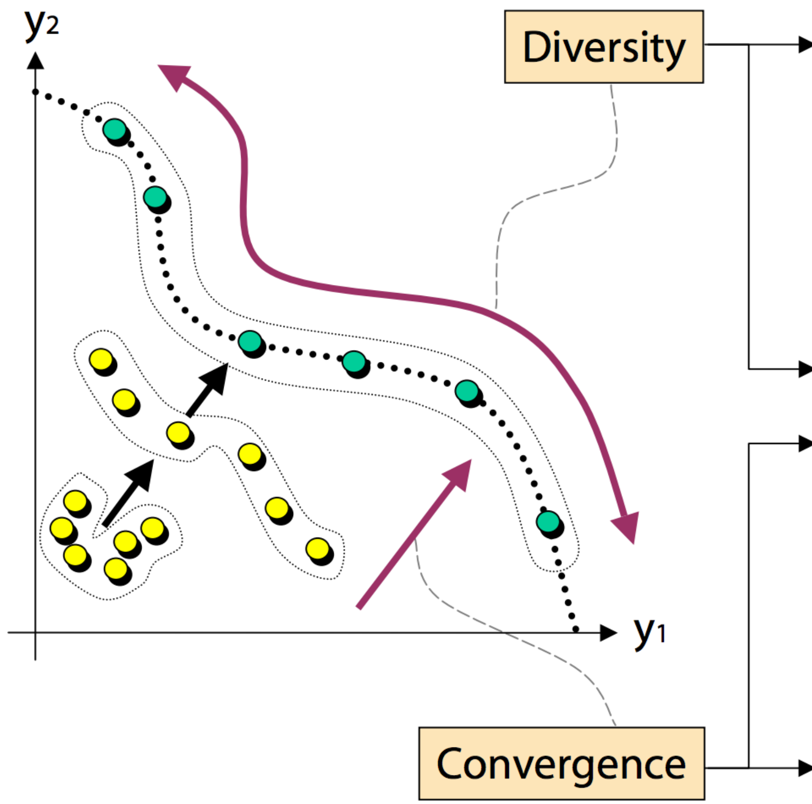


Difficulties of MOP

* Definition of the optimality: partial order relation, final choice depend on the decision
* Number of Pareto solutions grows with the problem size and the number of criteria
* For non convex MOP, solutions are not all located on the domain boundary but also in the convex hull → difficulty to find them.
* Performance assessment is difficult (comparisons of methods = comparisons of sets of solutions)

Population based algorithms are well fitted to solve Multi-objective problems

Issues in EMO [Ziztler'02]



- How to maintain a diverse Pareto set approximation?

  ❷ **density estimation**

- How to prevent nondominated solutions from being lost?

  ❸ **environmental selection**

- How to guide the population towards the Pareto set?

  ❶ **mating selection**

Non-dominated Sorting GA (NSGA-II) [Deb et al. 2002]

Pareto based

- **Initialization** of population P
- **Fitness assignment** non-dominated sorting
  - Population divided into fronts
  - Fitness (x) = index of the front x belongs to
- **Diversity** preservation ⇔ crowding distance.
- **Selection** ⇔ Binary tournament
- Recombination and mutation operators
- **Replacement** ⇔ N worst individuals are removed
- **Elitism** ⇔ Archive A of potentially efficient solutions

Indicator-Based EA (IBEA) [Zitzler et al. 2004]

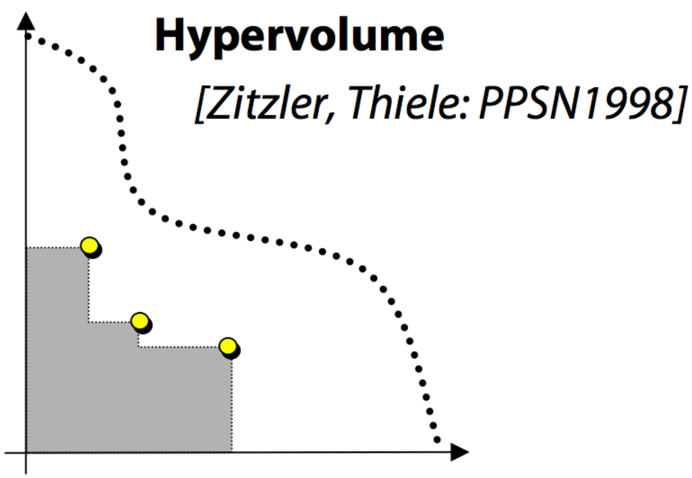
Indicator based

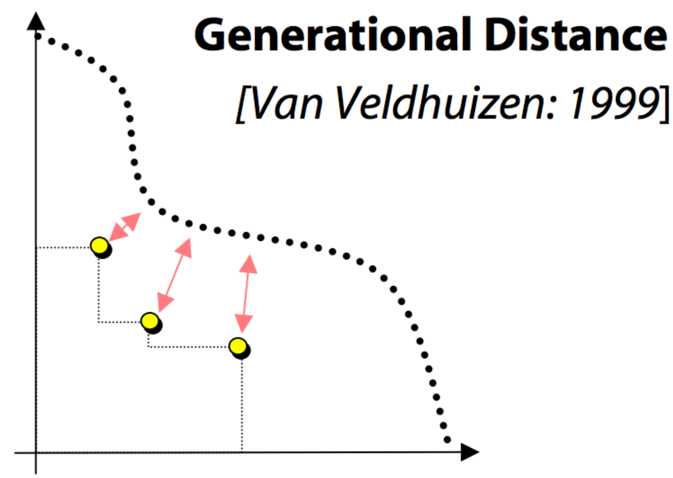
* Initialization of population P
* Fitness assignment quality indicator Qi :
  * Fitness (x) = Qi (x , P\{x})
* Diversity preservation ⇔ none
* Selection ⇔ binary tournament
* Recombination and mutation operators
* Replacement ⇔ remove the worst individual and update fitness values until |P| = N
* Elitism ⇔ Archive A of potentially efficient solutions

Performance assessment

Issues: quality measures, statistical testing, benchmark problems, visualization, …

Popular approach: unary quality measures





- Assign each outcome a *real number*
- Outcomes are compared by comparing the corresponding *values*

Conclusion

* In general, there is no method always better than the others on all types of problems, it depends on the problem instance (No Free Lunch).
* Metaheuristics are not problem-specific
* the basic concepts of metaheuristics permits an abstract level description
* Adapted to bioinformatics
  * Adapted to discover near optimal solutions
  * Can cope with difficult problems with black box evaluation