LOCAL SEARCH

Tree Search are constructive

- **The constructive algorithms** (seen so far) **generate** a solution by adding to a starting (empty or initial) state solution components in a particular order.
- **Local search algorithms start from an initial solution and iteratively try to improve it through local moves.**
- Local moves define a neighborhood.
- Applicable when the path to reach a goal/solution is not important.
- Several problems (for example, the n queens problem) have the property that the state description contains all the information necessary for understanding if one configuration is a solution or not (the path is irrelevant).
- In this case local search algorithms often provide a valid alternative.
- For example, we can start with all the queens on the board and then move them to reduce the number of attacks.

N QUEENS PROBLEM

- Put n queens on a chessboard so that they do not attach each other.

All queens can be attacket,

This can be the first random solution
and now we can change it moving queens

In this case we are not constructive anything, we are changing something that is already given

NEIGHBORHOOD

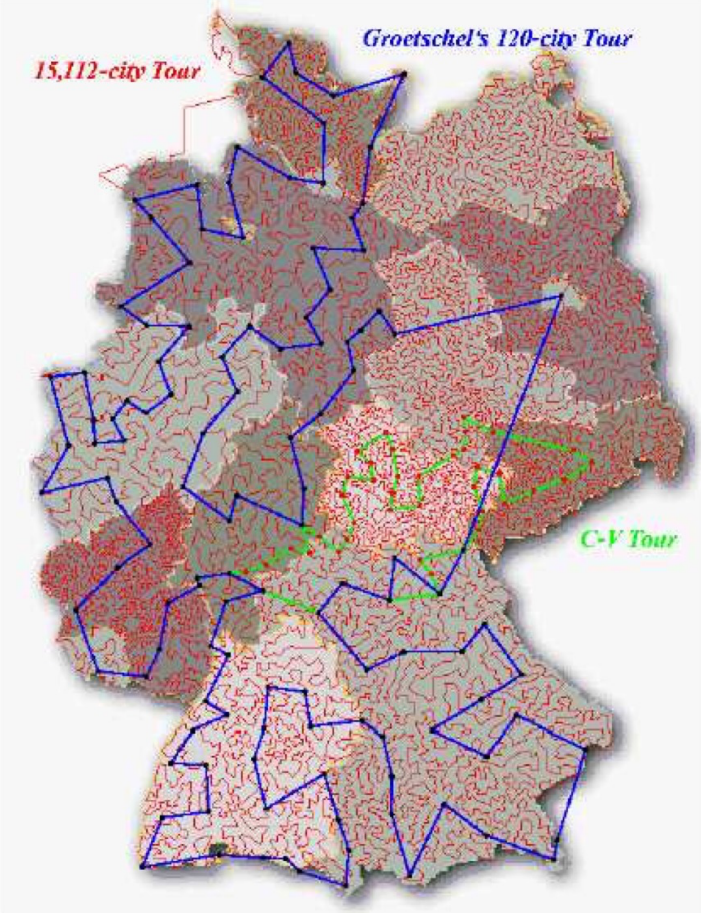
Neighborhood structure

$\mathcal{N} : \mathcal{S} \rightarrow 2^{\mathcal{S}}$

Assigns to every $s \in \mathcal{S}$ a set of neighbors $\mathcal{N}(s) \subseteq \mathcal{S}$. $\mathcal{N}(s)$ is called the neighborhood of s.

S = State

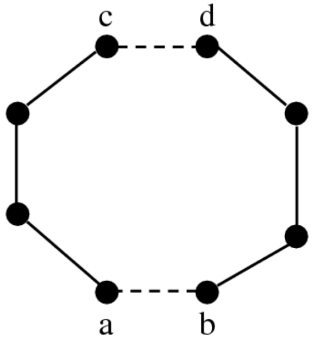
Example: TSP

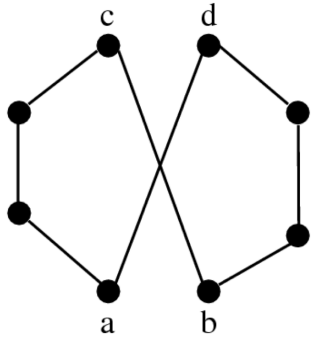


Given an undirected graph, with n nodes and each arc associated with a positive value, find the Hamiltonian tour with the minimum total cost.

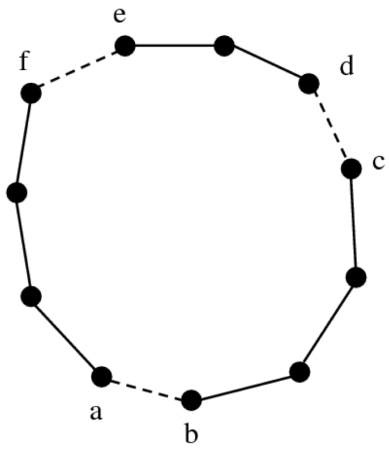
NEIGHBORHOODS FOR TSP: K-exchange

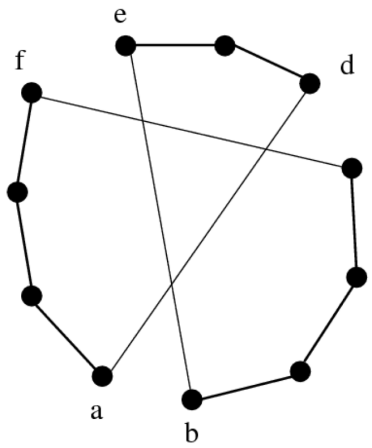
2-opt performs two exchanges





3-opt performs three exchanges





NEIGHBORHOODS FOR TSP: K-exchange

Solution represented as a permutation of numbers (1, 2 ……n)

Some neighborhoods:

- 2-swap: swap two numbers in the sequence
- k-cycle: select k positions and cycle on their numbers.
  E.g., (1, 2, 3, 4, 5, 6, 7, 8) → (1, 5, 3, 2, 4, 6, 7, 8)

ITERATIVE IMPROVEMENT – HILL CLIMBING

- Very basic local search algorithm
- A move is only performed if the solution it produces is better than the current solution (also called hill-climbing)
- The algorithm stops as soon as it finds a local minimum
- Several variants: best, first, stochastic first, etc.

ITERATIVE IMPROVEMENT

High-level algorithm:

$s \leftarrow$ GenerateInitialSolution()
repeat
    $s \leftarrow$ BestOf(s,$N(s)$)

until no improvement is possible

Main drawbacks:

- not effective in exploring the search space (e.g., it stops at the first local optimum encountered)
- Does not remember the search states already reached

"Like climbing Everest in thick fog with amnesia"

LOCAL OPTIMA

- A local maximum is a solution *s* such that for any *s'* belonging to *N(s)*, given an evaluation function *f*

$$f(s) \geq f(s')$$

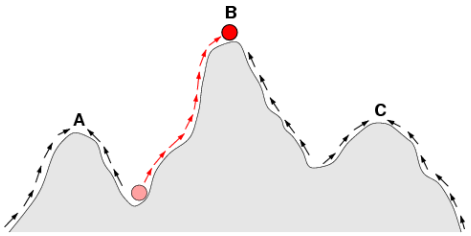
- When we solve a maximization problem we look for a global maximum s_{opt} ie such **that for any *s***,

$$f(s_{opt}) \geq f(s)$$

- *Analogous considerations for the minimum*

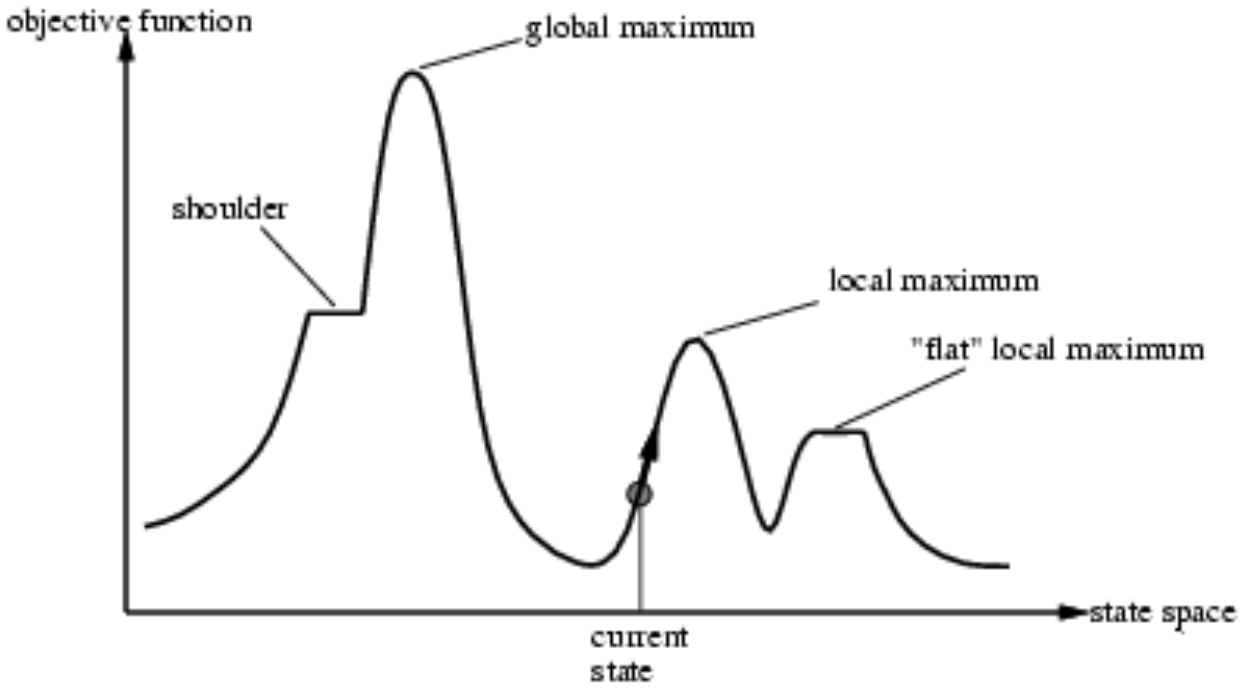
- The larger the neighborhood the more likely a local maximum is a global maximum (but obviously there is a problem of computational effort). So, the larger neighborhood the higher the quality of the solution

OTHER PROBLEMS



- Local Optima:
  - States that are better than all the neighbors, but worse than other states that are not nearby.
- Plateau:
  - flat areas of the landscape in which neighboring states have the same value. In which direction to move (random choice)?
- Ridges:
  - It is a higher area adjacent to those where we should go, but we can not go there directly. So we need to move in another direction to get there.

LANDSCAPE


objective function
global maximum
shoulder
local maximum
"flat" local maximum
state space
current
state

META HEURISTICS

Algorithms

- Local search has many pittfalls
- More effective and general search strategies are required.

- For example:
  - Accept non-improving moves
  - Change neighborhood or cost function during search
  - Use high-level search strategies

META HEURISTICS

- Local search can be seen as search processes over a graph

- Search starts from an initial node and explores the graph moving from a node to one of its neighbors, until it reaches a termination condition

we hane no guarantee to find the best solution

META HEURISTICS

- Local search can be seen as search processes over a graph

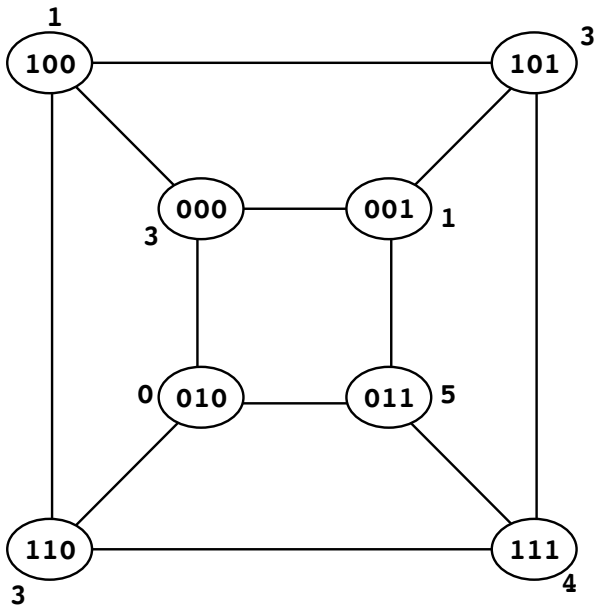
- Search starts from an initial node and explores the graph moving from a node to one of its neighbors, until it reaches a termination condition

  - **Neighborhood graph** to represent search space topology

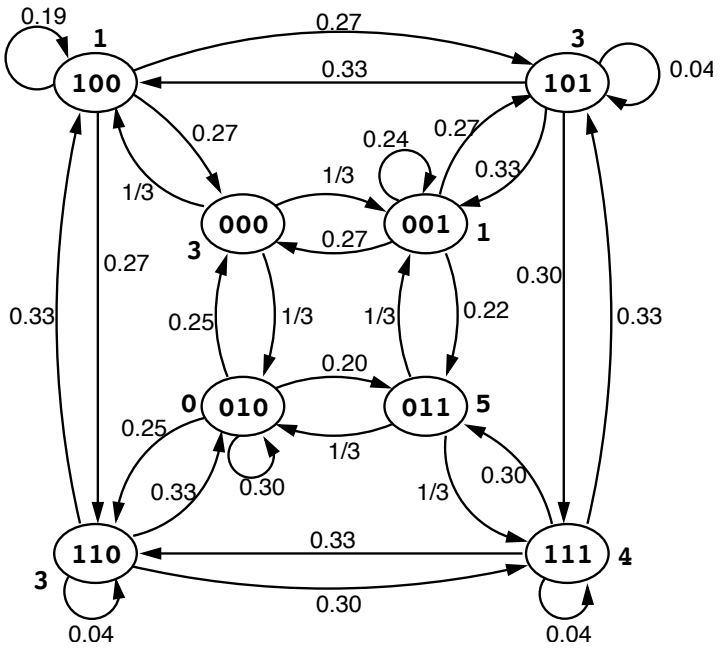
  - **Search graph** to represent the actual search space exploration

NEIGHBORHOOD GRAPH

Neighborhood of *s* := set of neighbors of *s* with Hamming distance equal to 1



SEARCH GRAPH



SIMULATED ANNEALING

is an algorithm that accept non imporving moves

- Origins in statistical mechanics (Metropolis algorithm)
- It allows moves resulting in solutions of worse quality than the current solution
- The probability of doing such a move is decreased during the search
- Usually, p(accept down-hill move s') = $\exp(-(f(s')-f(s)))/\ T$

SIMULATED ANNEALING: HIGH LEVEL ALGO

$s \leftarrow$ GenerateInitialSolution()
$T \leftarrow T0$
while termination conditions not met **do**
    $s' \leftarrow$ PickAtRandom(N (s))
    if $f(s') > f(s)$ **then**
        $s \leftarrow s'$ {s' replaces s}
    else
        Accept s' as new solution with probability $p(T, s', s)$
    end if
Update(T)
end while

SIMULATED ANNEALING

The temperature T can be varied in different ways:

- Logarithmic: $T_{k+1} = \Gamma / (\log(k + k0))$

  The algorithm is guaranteed to converge to the optimal solution with probability 1. Too slow for applications

- Geometric: $T_{k+1} = \alpha T_k$, where $\alpha \in]0, 1[$

- Non-monotonic: the temperature is decreased (intensifications is favored), then increased again (to increase diversification)

TABU SEARCH

this meta heuristic algoritm solve the problem of the memeory,
it trys to avois to explore agarin visited nodes

- Explicitly exploits the search history to dynamically change the neighborhood to explore

  Tabu list: keeps track of recent visited solutions or moves and forbids them

  ⇒ escape from local minima and no cycling

- Many important concepts developed “around” the basic TS version (e.g., general exploration strategies)

TABU SEARCH

$s \leftarrow$ GenerateInitialSolution()

$TabuList \leftarrow \emptyset$

while termination conditions not met **do**
    $s \leftarrow$ BestOf(N (s) \ *TabuList*)
    Update(*TabuList*)
end while

TABU SEARCH

- Storing a list of solutions is sometimes inefficient, therefore *moves* are stored instead.

- BUT: storing moves we could cut good not yet visited solutions

- we use *ASPIRATION CRITERIA* (e.g., accept a forbidden move toward a solution better than the current one)

TABU SEARCH: more in detail

$s \leftarrow$ GenerateInitialSolution()
InitializeTabuLists($TL1, \ldots, TLr$)
$k \leftarrow 0$
while termination conditions not met **do**
$AllowedSet(s,k) \leftarrow \{z \in N(s) \mid$ no tabu condition is violated or at least one aspiration condition is satisfied$\}$
$s \leftarrow$ ChooseBestOf($AllowedSet(s,k)$)
UpdateTabuListsAndAspirationConditions()
$k \leftarrow k+1$
end while

ITERATED LOCAL SEARCH

this is out third meta-heuristic algorithm

Uses two types of SLS steps: Local Sesrch

Subsidiary local search steps for reaching local optima as efficiently as possible (intensification)

Perturbation steps for effectively escaping from local optima (diversification)

Also: acceptance criterion to control diversification vs intensification behaviour.

ITERATED LOCAL SEARCH

$s_0 \leftarrow$ GenerateInitialSolution()
$s_* \leftarrow$ LocalSearch(s_0)

while termination conditions not met **do**
    $s' \leftarrow$ Perturbation(s_*, *history*)
    $s_{*'} \leftarrow$LocalSearch(s')
    $s_* \leftarrow$ApplyAcceptanceCriterion(s_*,$s_{*'}$,*history*)
end while

LESSON LEARNED

- Modeling is an issue
- High level search strategies have to be applied to effectively explore the search space (intensification/diversification)

  Intensification is a deep search around a solution

  Diserfication is jumping away from a solution
- Importance of search history
- Parameter tuning often critical

POPULATION BASED METAHEURISTICS

- Evolution of a set of points in the search space
- Some are inspired by natural processes, such as natural evolution and social insects foraging behavior
- **Basic principle**: learning correlations between “good” solution components

GENETIC ALGORITHM

- Genetic algorithms are inspired by **evolution**. Three key components:

  - *Adaptation*: organisms are suited to their habitats
  - *Inheritance*: offspring resemble their parents
  - *Natural selection*: new, adapted types of organisms emerge and those that fail to change adequately are subject to extinction

KEY CONCEPTS

- The *fittest* individuals have a high chance of having a numerous offspring.
- The children are similar, but not equal, to the parents.
- The traits characterizing the fittest individuals spread across the population, generation by generation.

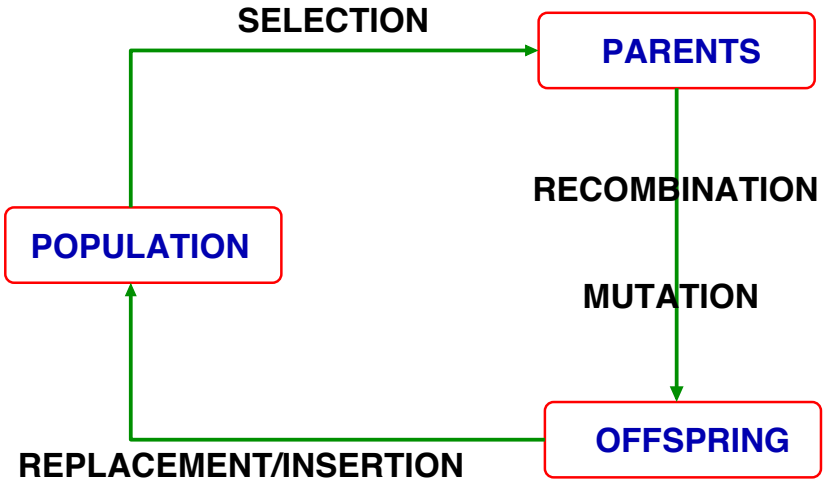
GAs are not meant to simulate the biological evolutionary processes, but rather aimed at exploiting these key concepts for problem solving.

GENETIC ALGORITHM

The metaphor

BIOLOGICAL EVOLUTION		ARTIFICIAL SYSTEMS
Individual	$\leftrightarrow$	A possible solution
Fitness	$\leftrightarrow$	Quality
Environment	$\leftrightarrow$	Problem

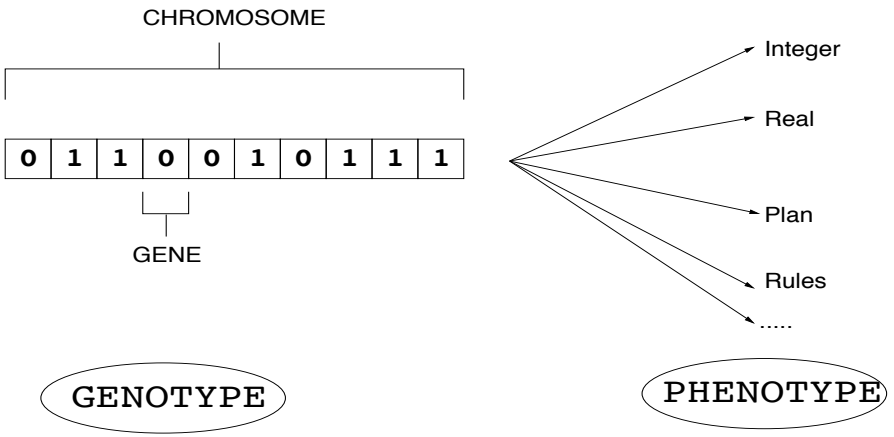
THE EVOLUTIONARY CYCLE



Recombination: combines the genetic material of the parents.
Mutation: introduce variability in the genotypes.
Selection: acts in the choice of parents whose genetic material is then reproduced with variations.
Replacement/insertion: defines the new population from the new and the old one.

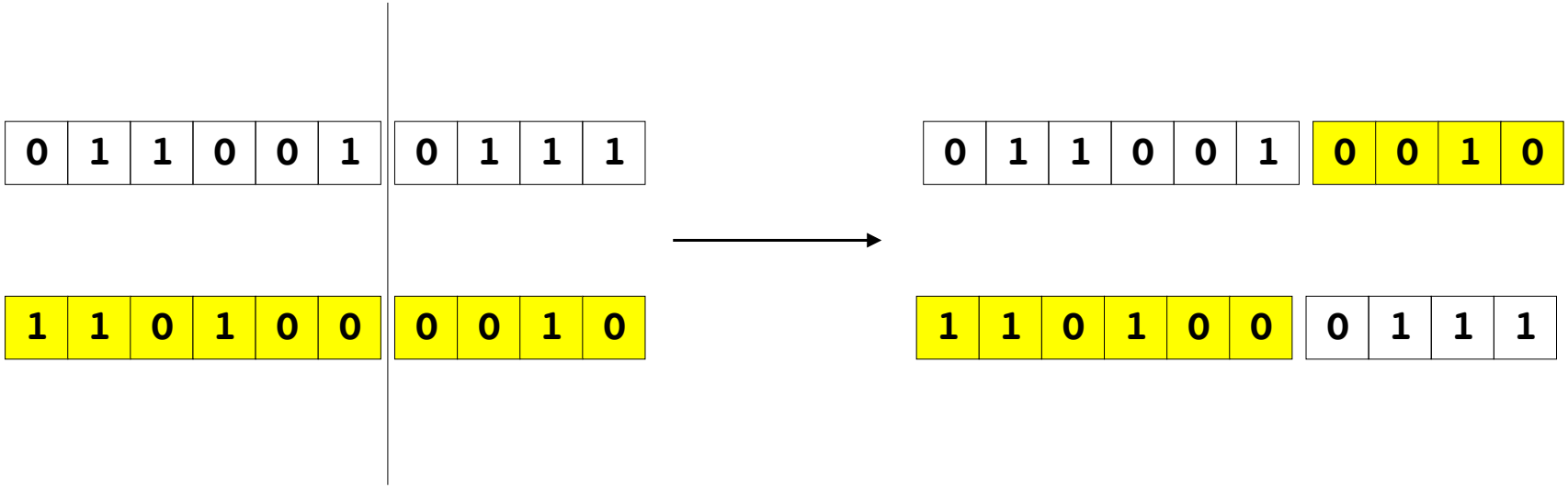
TERMINOLOGY

- A **population** is the set of individuals (solutions)
- Individuals are also called **genotypes**
- Chromosomes are made of units called **genes**
- The domain of values of a gene is composed of **alleles** (e.g., binary variable ↔ gene with two alleles)



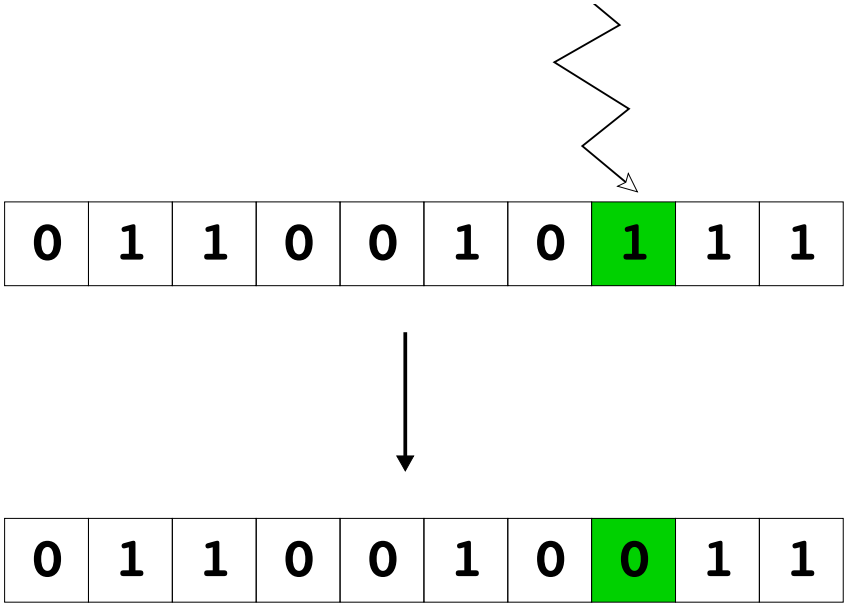
GENETIC OPERATORS

- Recombination or **Crossover**: cross-combination of two chromosomes (loosely resembling biological crossover)



GENETIC OPERATORS

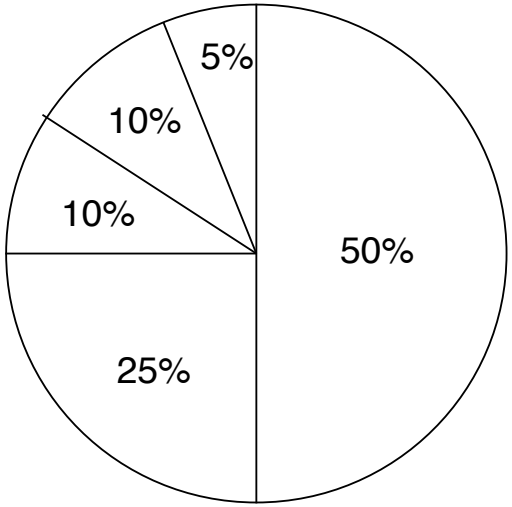
- **Mutation**: each gene has probability *pM* of being modified (’flipped’)



GENETIC OPERATORS

Proportional selection: the probability for an individual to be chosen is proportional to its fitness.

Usually represented as a roulette wheel.



I_1	50
I_2	25
I_3	10
I_4	10
I_5	5

GENETIC OPERATORS

Generational replacement: The new generation replaces entirely the old one.

- Advantage: very simple, computationally not expensive, easier theoretical analysis.
- Disadvantage: it might be that good solutions are not maintained in the new population.

Alternatively, one can keep the best *n* individuals from the old and the new population (or choose any variant of this scheme).

GENETIC OPERATORS

Real valued Variables

- Solution: $x \in [a,b]$, $a,b \in \mathrm{R}$
- Mutation: random perturbation $x \rightarrow x \pm \delta$, accepted if $x \pm \delta \in [a, b]$
- Crossover: linear combination $z = \lambda 1 x + \lambda 2 y$, with $\lambda 1$, $\lambda 2$ such that $a \leq z \leq b$.

Permutation

- Solution: $x = (x_1, x_2, \ldots, x_n)$ is a permutation of $(1,2,\ldots,n)$.
- Mutation: random exchange of two elements in the n-ple.
- Crossover: like 2-point crossover, but avoiding value repetition.

GENETIC ALGORITHM

Execution time limit reached.
Satisfactory solution(s) have been obtained.
Stagnation (limit case: the population converged to the same individual)

Initialize Population
Evaluate Population
while Termination conditions not met **do**
    while New population not completed **do**
        Select two parents for mating
        Apply crossover
        Apply mutation to each new individual
    end while
Population ← New population
Evaluate Population
end while

GENETIC ALGORITHM

Intuition:

- Crossover combines good parts from good solutions (but it might achieve the opposite effect).
- Mutation introduces diversity.
- Selection drives the population toward high fitness.

GENETIC ALGORITHM

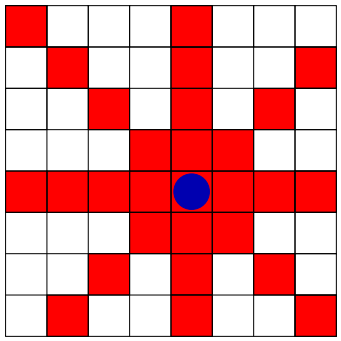
Pros:

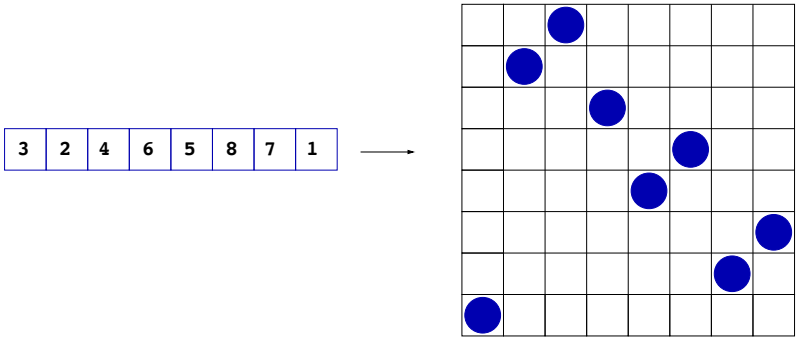
- Extremely simple.
- General purpose.
- Tractable theoretical models.

Cons:

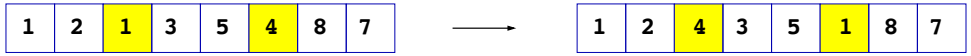
- Coding is crucial.
- Too simple genetic operators.

EXAMPLE: 8 QUEENS

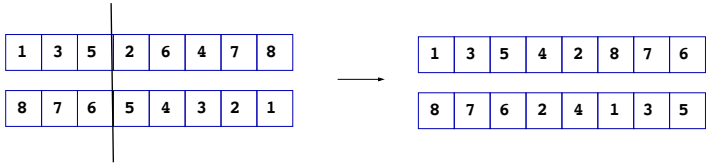




Mutation: swap two numbers



Crossover: combine two parents



Fitness: penalty of a queen is the number of queens it can check.
The fitness of the configuration is the sum of the single penalties.

DESIGN GUIDELINES

Local search/metaheuristics approach preferable when:

- neighborhood structures create a *correlated* search graph
- computational cost of moves is low
- 'inventing' moves is *easy*

Population-based approach preferable when:

- solutions can be encoded as composition of good building blocks
- computational cost of moves in local search is high
- it is difficult to design effective neighborhood structures
- coarse grained exploration is preferable (e.g., huge search spaces)

SOFTWARE FRAMEWORKS

Comet (by *Dynadec*)

- Constraint-based metaheuristics framework
- Proprietary programming language (free use for academics)
- Efficient data structures

www.dynadec.com

EASYLOCAL++ (by Luca Di Gaspero and Andrea Schaerf)

- C++ stochastic local search framework (GPL license)
- User should instantiate abstract classe
- Analysis tool also available (upon request)

tabu.diegm.uniud.it/EasyLocal++