

Problem Solving

Local Search

Marie Pelleau

`marie.pelleau@univ-cotedazur.fr`

Greedy Algorithm

Principle

- At each step, a choice is made, the one that seems the best at that moment
- Builds a solution step by step
 - without revisiting previous decisions
 - by making at each step the choice that seems the best
 - hoping to achieve a global optimal result
- Greedy approach
 - depending on the problem, no guarantee of optimality (greedy heuristic)
 - low cost (compared to exhaustive enumeration)
 - intuitive choice

Local Search

Principle

- Start from an initial solution
- At each step, modify the solution
 - trying to improve the value of the objective function
 - hoping to achieve the global optimum
- Local approach
 - depending on the problem, no guarantee of optimality (heuristic)
 - low cost

Initial solution

- “Empty” solution
- Random solution
- Solution from a greedy algorithm

Local Search

Principle

- Start from an initial solution
- At each step, modify the solution
 - trying to improve the value of the objective function
 - hoping to achieve the global optimum
- Local approach
 - depending on the problem, no guarantee of optimality (heuristic)
 - low cost

Modifications

- Modify the value of a variable
- Swap the values of two variables

Knapsack Problem

Description

You have:

- A backpack with a weight limit
- A set of objects, each object o_i has
 - A weight: w_i
 - A value: v_i

Which objects should be taken to maximize the total value carried while respecting the weight constraint?

- The total value of the selected objects is maximized
- The total weight of the selected objects is less than or equal to the backpack's weight limit

Knapsack Problem

Variables

- We associate each item with a 0-1 variable (it only takes values 0 or 1)
- It is a membership variable for the backpack
- If the item is taken, the variable is 1, otherwise it is 0

Model

- The value and weight of an item are given data, so for item o_i , we have the value v_i and the weight w_i
- The membership variable for the backpack is x_i
- The maximum weight of the backpack is W

Knapsack Problem

Constraints

- $\max \sum_{i=1}^n v_i x_i$ the objective
- $\sum_{i=1}^n w_i x_i \leq W$ sum of weights less than or equal to the maximum weight

Knapsack

Initial solution

- “Empty” solution: empty knapsack \Rightarrow objective function 0
- Random solution : random backpack \Rightarrow must be verified as a solution
- Solution of a greedy algorithm

Modifications

- Add an item to the backpack \Rightarrow if max capacity is not exceeded
- Deletes an item from the backpack

Hitting-set: Set cover

Description

- A switch is connected to some bulbs
- When a switch is pressed, all connected bulbs are turned on
- **Question:** What is the minimum number of switches needed to turn on all bulbs?

Hitting-set: Set cover

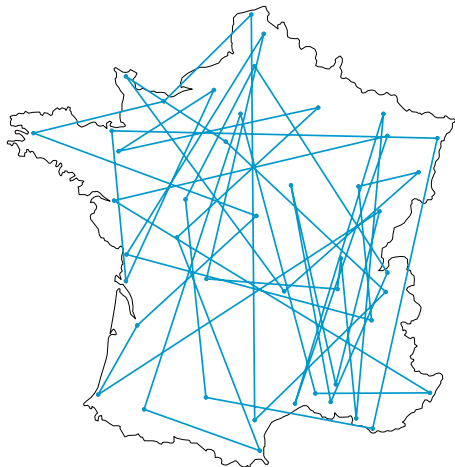
Initial solution

- “Empty” solution: all switches on \Rightarrow objective function number of switches
- Random solution: random switch positions \Rightarrow must be verified as a solution
- Solution of a greedy algorithm

Modifications

- Turn on a switch
- Turns off a switch \Rightarrow if all bulbs remain on

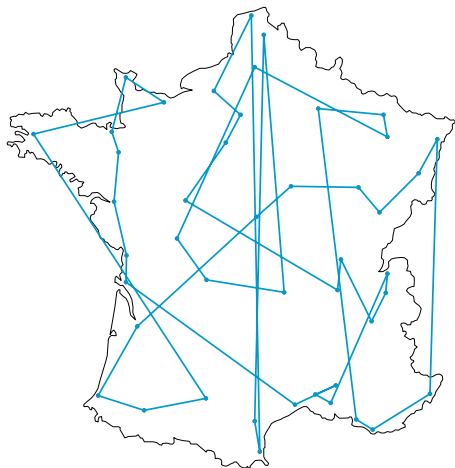
TSP



Solution initiale

- Cities in alphabetical order

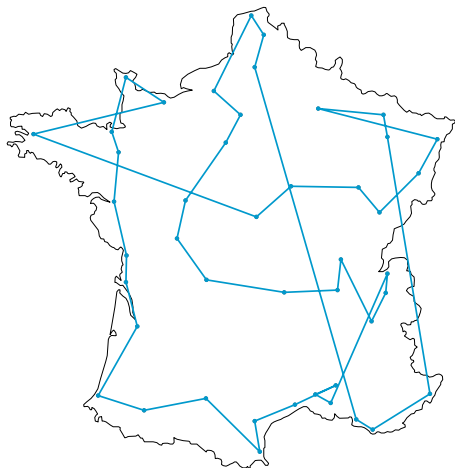
TSP



Solution initiale

- Cities in alphabetical order
- Cities in random order

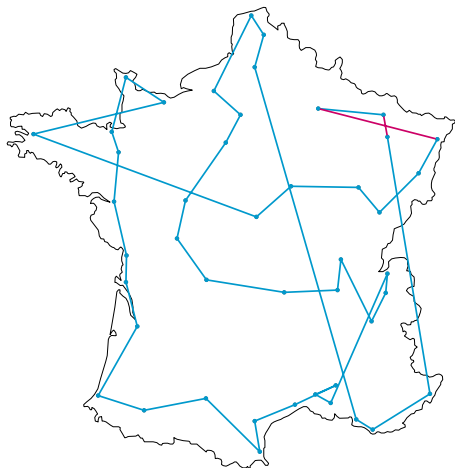
TSP



Solution initiale

- Cities in alphabetical order
- Cities in random order
- Solution of a greedy algorithm

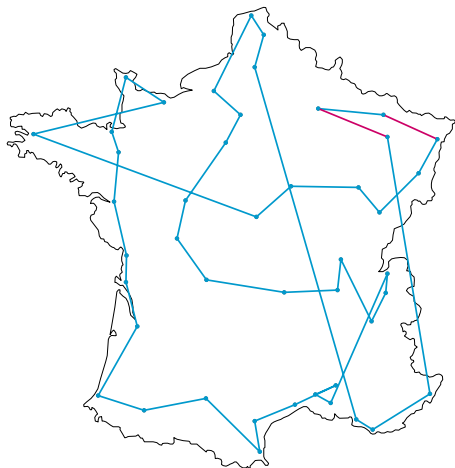
TSP



Modifications

- k -opt
 - $k = 2$
 - $k = 3$

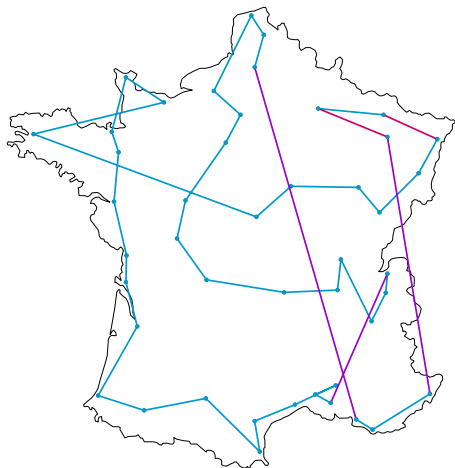
TSP



Modifications

- k -opt
 - $k = 2$
 - $k = 3$

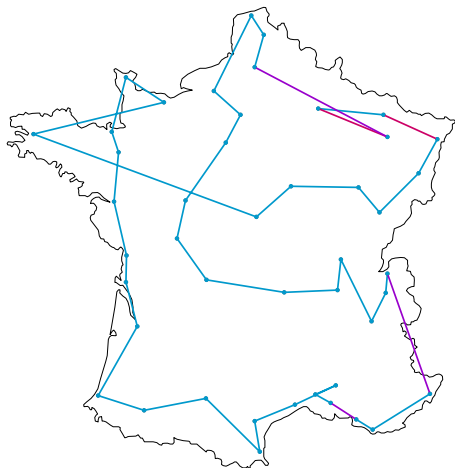
TSP



Modifications

- k -opt
 - $k = 2$
 - $k = 3$

TSP



Modifications

- k -opt
 - $k = 2$
 - $k = 3$

Local search

Principle

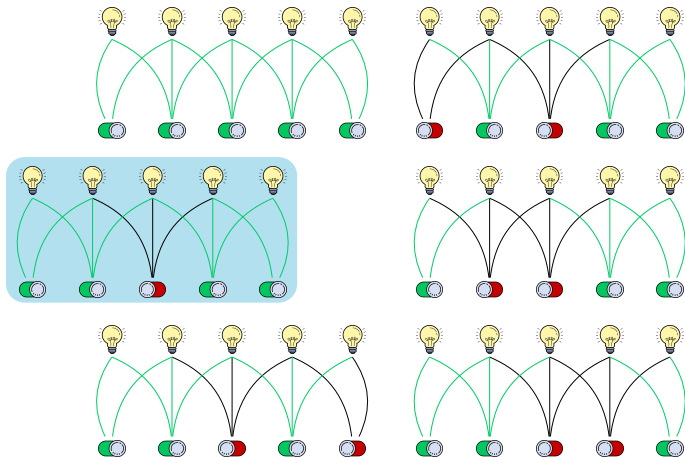
- We start with an initial solution
- At each step, we modify the solution \Rightarrow notion of neighborhood

Neighborhood

For a solution, the set of solutions with one modification

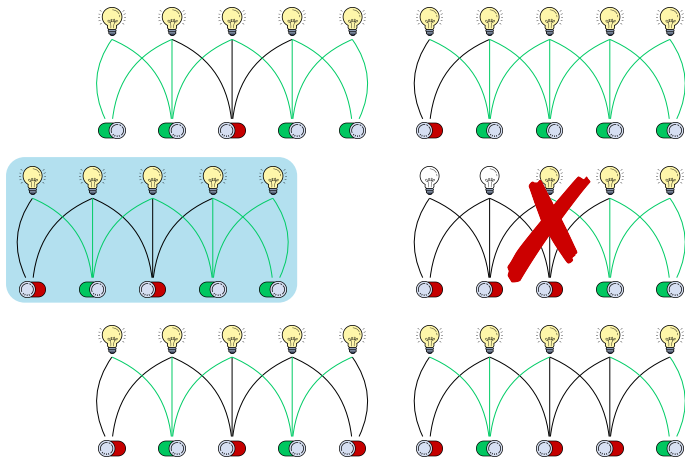
Hitting-set: Set cover

Neighborhood



Hitting-set: Set cover

Neighborhood



Local search

Principle

- We start with an initial solution
- At each step, we modify the solution \Rightarrow notion of neighborhood

Which neighbor to choose?

- Randomly
- The best
- One of the best

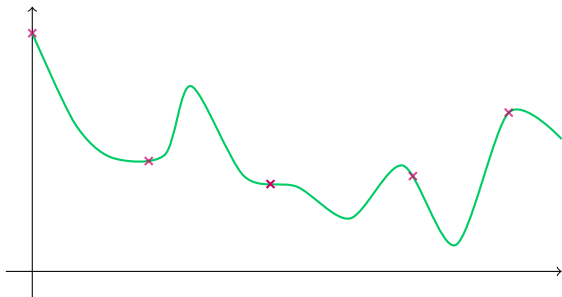
Contents

- 1 Random walk
- 2 Gradient descent
- 3 Restarts
- 4 Tabu Search
- 5 Constraint Based Local Search

Random walk

Principle

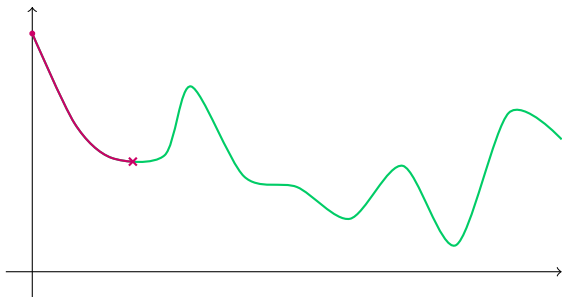
- We start with an initial solution
- At each step, the solution is **randomly** modified



Gradient descent

Principle

- We start with an initial solution
- At each step, we move towards a solution in the neighborhood **strictly improving** the objective



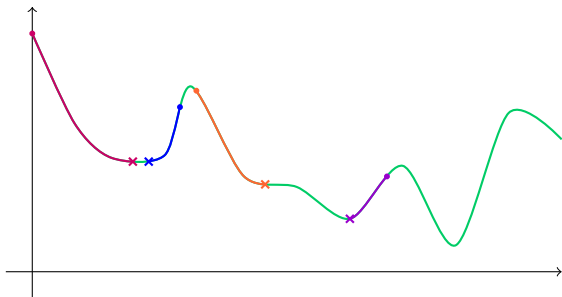
Drawbacks

You can get stuck in local minima

Gradient descent

Principle

- We start with an initial solution
- At each step, we move towards a solution in the neighborhood **strictly improving** the objective



Restarts

Start again from another solution

Local search

Restarts

- Random solution
- “Empty” solution, in which a certain percentage of variables is fixed as in the best solution found so far
 - 5%, 10%, 20%

Large Neighborhood Search (LNS) [Shaw, 1998]

No improvement

- We move towards a solution in the neighborhood **without improving** the objective
emph⇒ Don't be a goldfish

Tabu Search [Glover, 1986]

Principle

- We start from a solution s .
- We move towards **the best** solution in the neighbourhood which is not **forbidden**
- Add s to the forbidden solutions for the next m iterations

Memory

- Prohibiting solutions can be memory-intensive
- Instead we forbid movements

Aspiration criterion

A tabu movement can be accepted if it leads to a **better** solution than the best solution known so far

Size of tabu list

- If m too small, **intensification** too strong \Rightarrow blocking search around a local optimum
- If m too large, **diversification** too strong \Rightarrow risk of missing solutions

Optimal list length varies

- from one problem to another
- from one instance to another of the same problem
- during the resolution of the same instance

[Battiti, Protasi 2001]: adapt this length dynamically

- Need for diversification \Rightarrow increase m
- Need for intensification \Rightarrow decrease m

Local search

Principle

- We start with an initial solution
- At each step, we modify the solution
 - trying to improve the value of the objective function
 - in the hope of obtaining the global optimum
- Local approach
 - depending on the problem no guarantee of optimality (heuristic)
 - low-cost

Note

- This assumes the existence of an objective function
- What if there isn't one?

Constraint Based Local Search

Principle

- Given a problem of the form
 - $\mathcal{V} = \{v_1, \dots, v_n\}$: variables
 - $\mathcal{D} = \{D_1, \dots, D_n\}$: domains
 - $\mathcal{C} = \{C_1, \dots, C_p\}$: constraints
- Objective function to minimize: number of unsatisfied constraints

Intuition

- Search guided by problem structure
 - constraints give structure to the problem and variables link them together
- any type of constraint can be used

Constraint Based Local Search

N-queens

- on a $n \times n$ chessboard
- Place n queens so that no queen can capture another one

Formulation

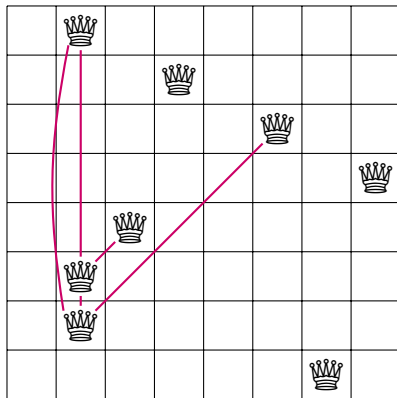
- l_i : queen's column on line i
- $l_i \neq l_j$
- $l_i + i \neq l_j + j$ (upward diagonal)
- $l_i - i \neq l_j - j$ (downward diagonal)

Objective function

- Number of unsatisfied constraints

Constraint Based Local Search

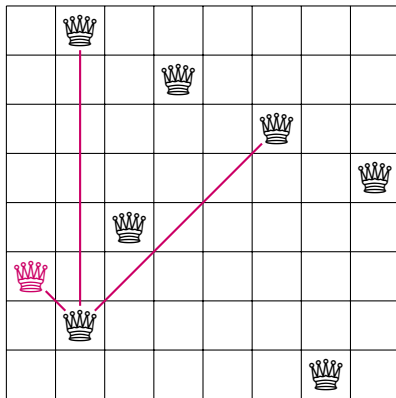
Example



Objective function: 5

Constraint Based Local Search

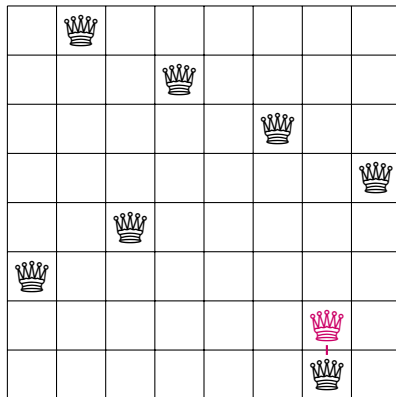
Example



Objective function: 3

Constraint Based Local Search

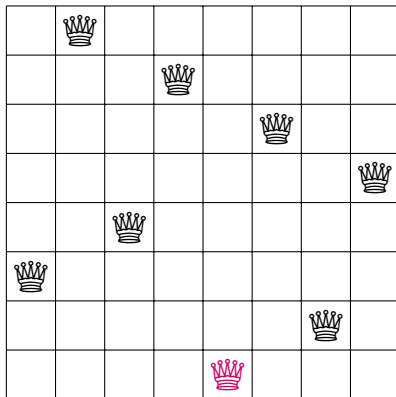
Example



Objective function: 1

Constraint Based Local Search

Example



Objective function: 0