Decision Aid Methodologies In Transportation
Lecture 7: Heuristics

Heuristics

Shadi SHARIF AZADEH

Transport and Mobility Laboratory TRANSP-OR

École Polytechnique Fédérale de Lausanne EPFL

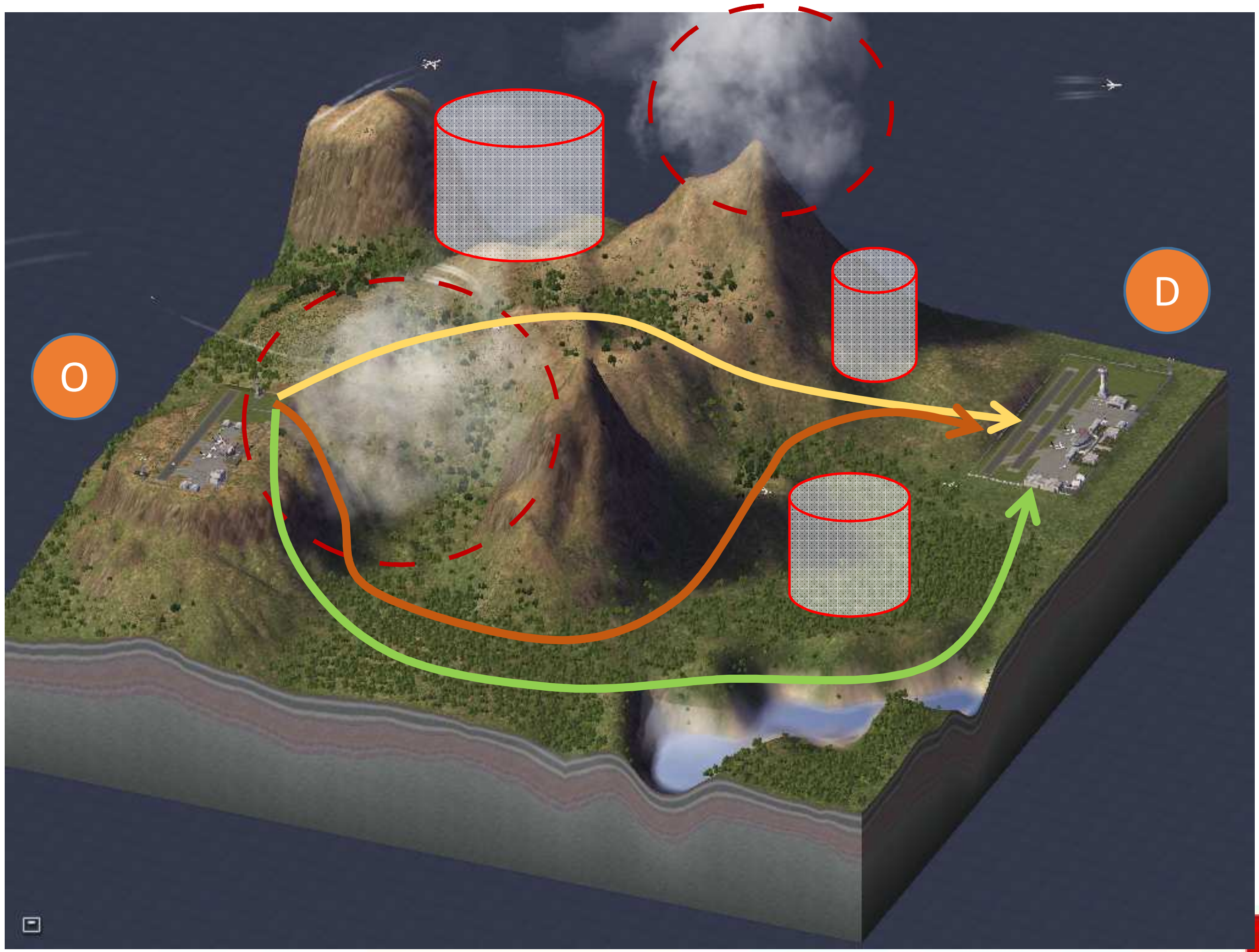
Heuristics

A heuristic is a technique designed for solving a problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find any exact solution. This is achieved by trading optimality, completeness, accuracy, and/or precision for speed.

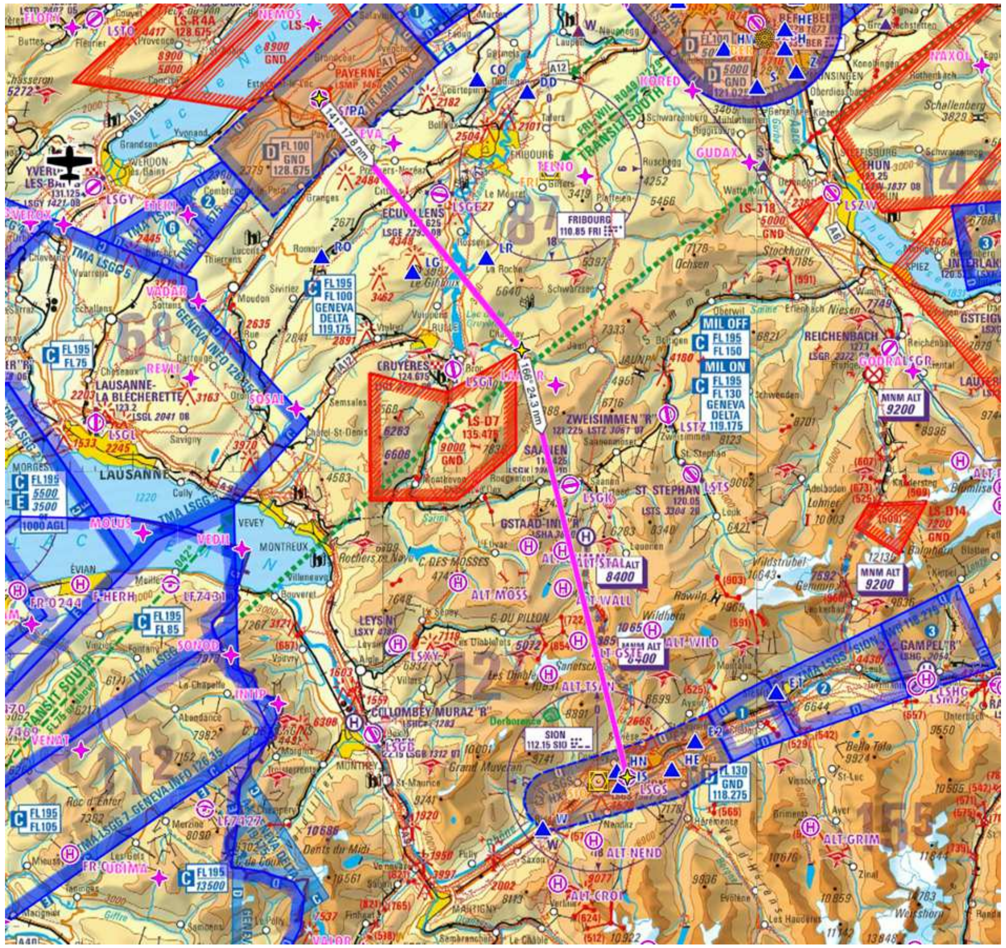
The importance of Heuristics

◌ Clouds

⌸ Airspace

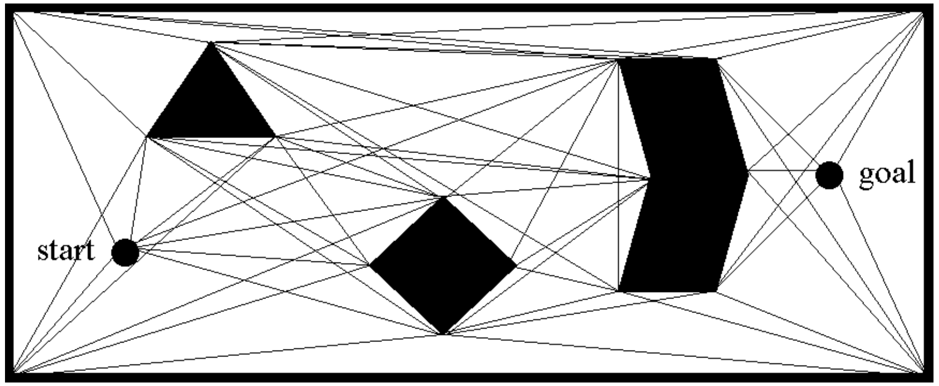


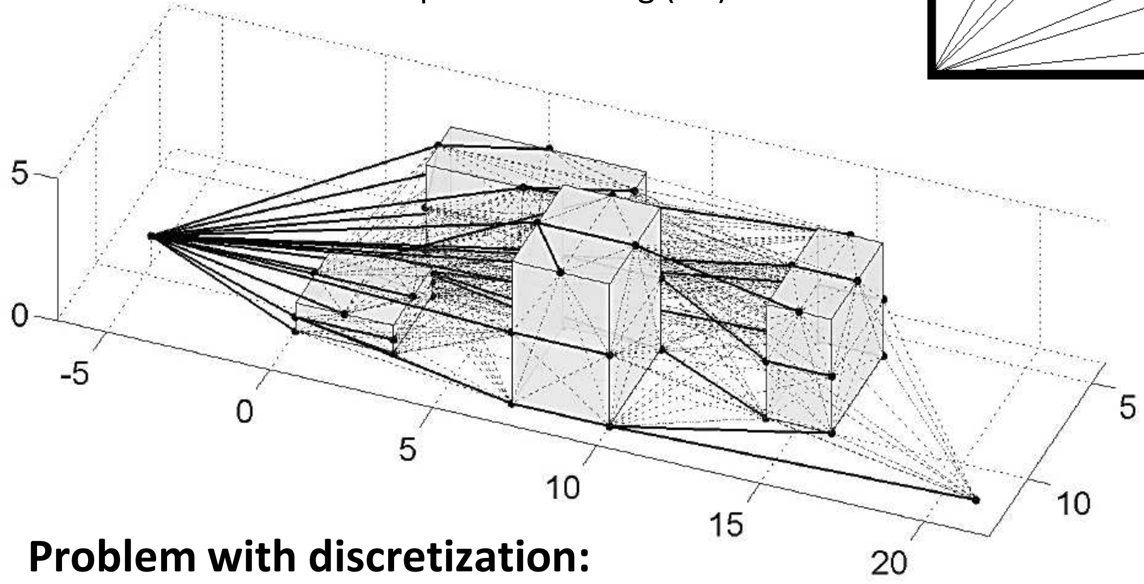
The importance of Heuristics



The importance of Heuristics

- Visibility graph
- Directly connect two nodes
- If there is an obstacle in between then the path is not visible
  - Maximum and minimum speed
  - Minimum radius turn
  - Restriction to prevent stalling (3D)





Path characteristics:

- **Visible :** avoid obstacles
- **Flyable:** respect aircraft kino dynamic limitations

Problem with discretization:

If we discretize the (x, y, z) with the size 10*10*10 then we will have 1000 nodes, 499,500 arcs and 2,977,000 trajectories to evaluate.

The importance of Heuristics

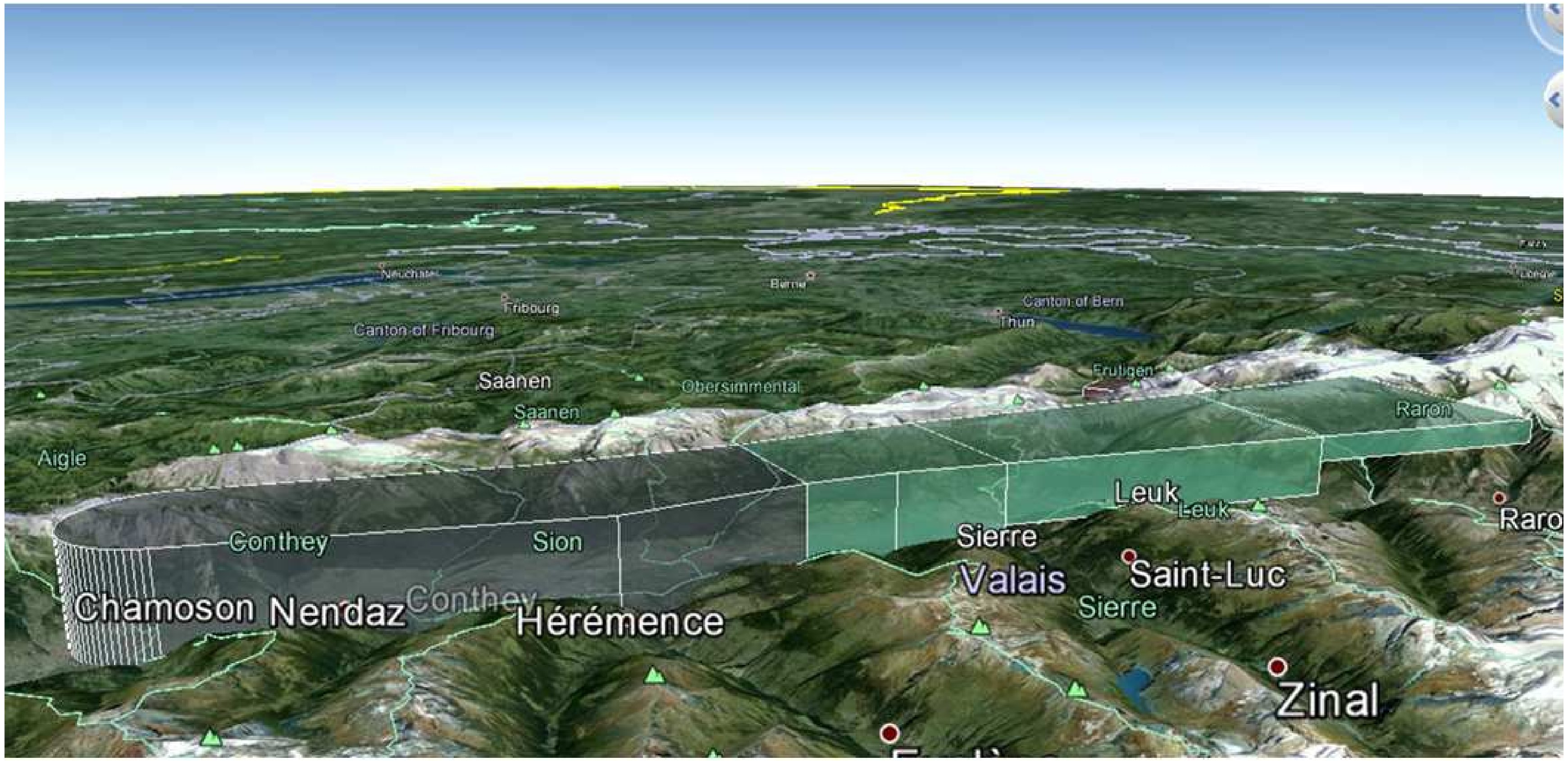
1) **Map:**
Build a GPS map based on airports locations, flyable altitude, no fly zone. All these computations can be done offline

2) **Flyable Map:**
A truncated map that is used for computation. In this map, it is possible to add some preferred locations as waypoints (Online). Some preferences can also be added to the map.

3) **Path-finder:**
Finding a set of alternative paths to fly based on the main criteria.

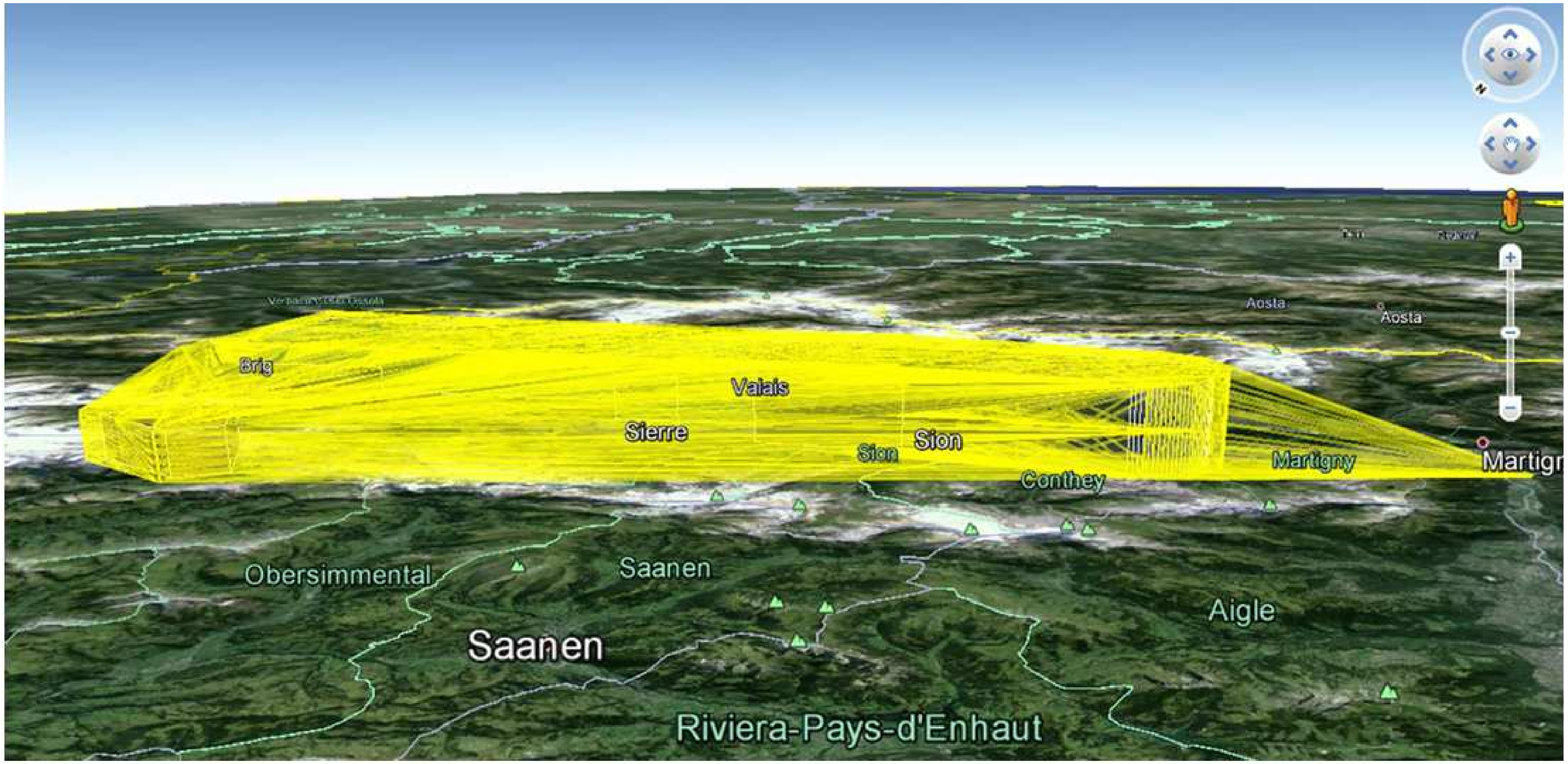
3) **Detailed trajectory:**
Use the set of paths to construct an airplane trajectory based on its physical characteristics (pitch angle, speed, velocity etc.)

The importance of Heuristics

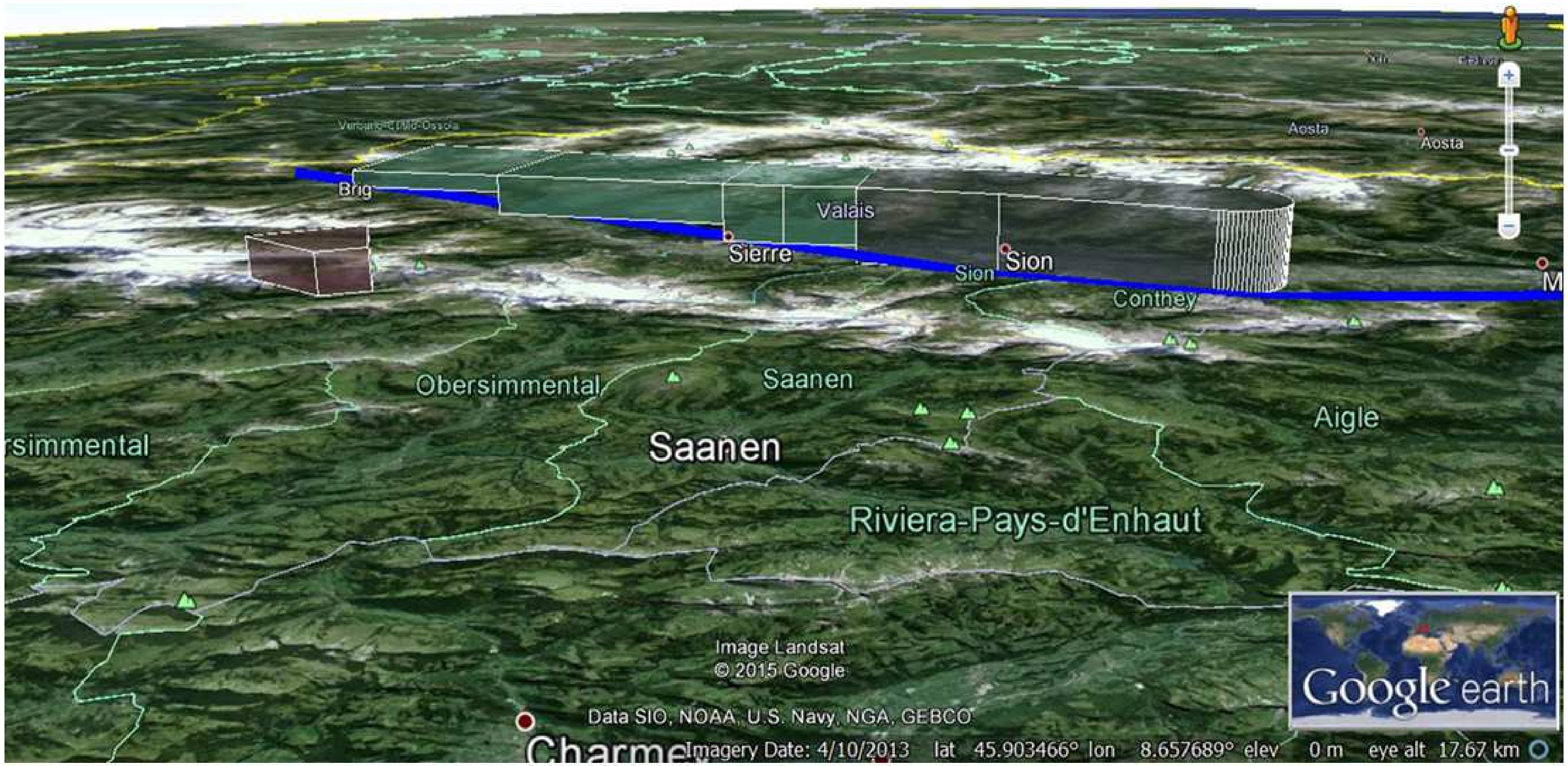


Origin: Vernayaz
Destination: Blatten

The importance of Heuristics



The importance of Heuristics



Heuristics-Importance

Why do we use these heuristic methods?

- The number of possible solutions in the search space is so large as to forbid an exhaustive search for the best answer.
- The possible solutions are so heavily constrained that constructing even one feasible answer is difficult , let alone searching for an optimum solution.

Heuristics-TSP as an example large sized space

For example, consider a traveling salesman problem (TSP). Conceptually, it 's very simple: the traveling salesman must visit every city in his territory exactly once and then return home covering the shortest distance.

Some closely related problems require slightly different criteria, such as finding a tour of the cities that yields minimum traveling time, or minimum fuel cost , or a number of other possibilities , but the idea is the same.

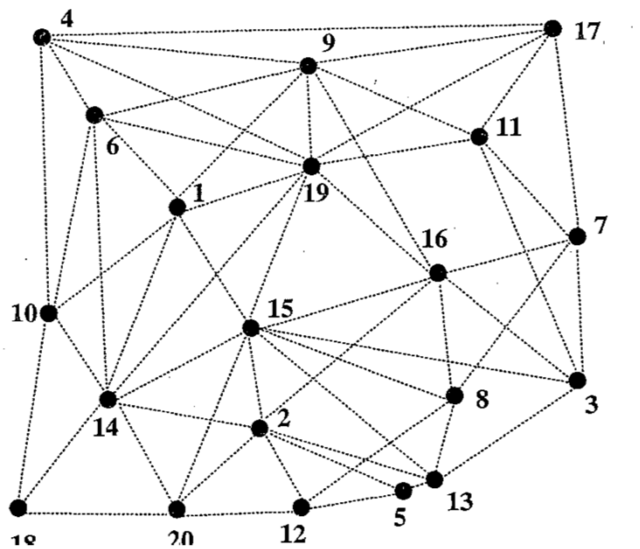
Question:
Given the cost of traveling between each pair of cities, how should the salesman plan his itinerary to achieve a minimum cost tour?

Heuristics-Size of the feasible space

For n = 6, there are 5! /2 = 60 different solutions to the TSP. But for n = 7 these numbers are 360.

Rate of growth of (n - 1)! /2 , consider the following numbers:

- 10-city TSP has about 181 ,000 possible solutions .
- 20-city TSP has about 1 0,000,000,000,000,000 possible solutions.
- 50-city TSP has about 100 ,000 ,000,000,000 ,000,000 ,000 ,000 ,000, 000,000,000 ,000 ,000 ,000 ,000 ,000 ,000,000 ,000 possible solutions .



Heuristics-Evaluation Function

EF is a mapping from the space of possible candidate solutions under the chosen representation to a set of numbers, where each element from the space of possible solutions is assigned a numeric value that indicates its quality. The evaluation function allows you to compare the worth of alternative solutions to your problem as you've modeled it .

Example (TSP):
suppose we want to discover a good solution to a TSP:

The objective is to minimize the sum of the distances between each of the cities along the route while satisfying the problem's constraints . One evaluation function might map each tour to its corresponding total distance. We could then compare alternative routes and not only say that one is better than another, but exactly how much better.

Evaluation function could be directly inspired by the objective function.

Heuristics-Search space and neighborhoods

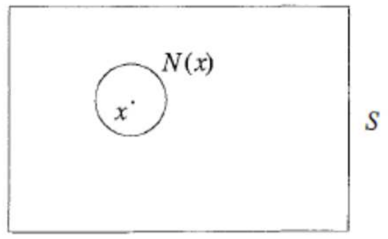
Given a **search space S** together with its feasible part $F \subseteq S$ find $x \in F$ such that

$$eval(x) \leq eval(y)$$

For all $y \in F$.

Note that here we're using an evaluation function for which solutions that return smaller values are considered better (i.e. , the problem is one of minimization).

If we concentrate on a region of the search space that 's "near" some particular point in that space, we can describe this as looking at the neighborhood of that point . Graphically, consider some abstract search space S together with a single point $x \in S$. The intuition is that a neighborhood $N(x)$ of x is a set of all points of the search space S that are close in some measurable sense to the given point x.



Heuristics-Search space and neighborhoods

We can define the nearness between points in the search space in many different ways. Two important possibilities include:

1- We can define a distance function distance on the search space S

$$dist: S \times S \longrightarrow R$$

Then we define the neighborhood $N(x)$: this called $\mathcal{E}$-*neighborhood* of x

$$N(x) = \{y \in S: dist(x, y) \leq \mathcal{E}\},$$
$$\mathcal{E} \leq 0$$

2- We can define a mapping m on the search space S as

$$m: S \longrightarrow 2^S$$

and this mapping defines a neighborhood for any point $x \in S$. we can define a 2-swap mapping for the TSP that generates a new set of potential solutions from any potential solution x . Every solution that can be generated by swapping two cities in a chosen tour can be said to be in the neighborhood of that tour under the 2-swap operator.

Heuristics-Hill Climbing method

Hill-climbing methods , just like all local search methods, use an iterative improvement technique. The technique is applied to a single point - the current point - in the search space.

- A new point is selected from the neighborhood of the current point .
- If new point provides a better value in light of the evaluation function

  the new point becomes the current point .

  Otherwise, some other neighbor is selected and tested against the current point

- The method terminates if no further improvement is possible, or we run out of time or patience.

Heuristics-Hill Climbing method cons and pros

Disadvantages of hill climbing method:

- They usually terminate at solutions that are only locally optimal.
- There is no information as to the amount by which the discovered local optimum deviates from the global optimum, or perhaps even other local optima.
- The optimum depends on the initial configuration.
- In general, it is not possible to provide an upper bound for the computation time.

They're very easy to apply! All that's needed is the representation, the evaluation function, and a measure that defines the neighborhood around a given solution.

Exhaustive search

Definition: exhaustive search checks each and every solution in the search space until the best global solution has been found. These algorithms are very simple to implement.

Some classic optimization algorithms that construct a complete solution from partial solutions (e.g. , Dijkstra or the A* algorithm) are based on an exhaustive search.

We have already explained Dijkstra algorithm, we will explain A*.

Exhaustive search-A star

A* needs two lists:

- One to write down all the squares that are being considered to find the shortest path (called the open list)
- One to write down the square that does not have to consider it again (called the closed list. It begins by adding current position (we'll call this starting position point "A") to the closed list. Then, it adds all walkable tiles adjacent to the current position to the open list.

Path Scoring:

We will give each square a score G + H where:

G is the total movement cost from the start point A to the current square.
H is the estimated movement cost from the current square to the destination point.

The G cost function can be computed but the H cost function can just be estimated. That's why this cost function is called heuristic cost function.

Exhaustive search-A star algorithm-pros and cons

Among all shortest-path algorithms, A* algorithm expands the fewest number of nodes.

The main drawback of A* algorithm and indeed of any best-first search is its memory requirement.

Local Search

1. Pick a solution from the search space and evaluate its merit . Define this as the current solution.

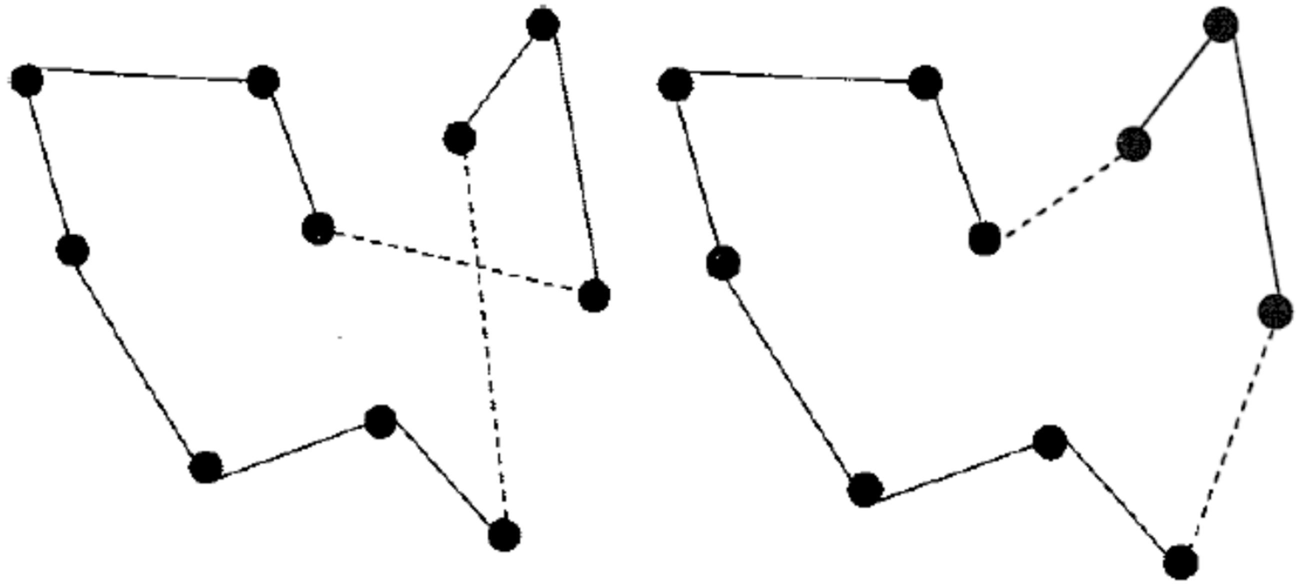
2. Apply a transformation to the current solution to generate a new solution and evaluate its merit.

3. If the new solution is better than the current solution then exchange it with the current solution; otherwise discard the new solution.

4. Repeat steps 2 and 3 until no transformation in the given set improves the current solution.

Local Search-TPS-2 Opt

The simplest is called 2-opt. It starts with a random permutation of cities (call this tour T) and tries to improve it . The neighborhood of T is defined as the set of all tours that can be reached by changing two nonadjacent edges in T. This move is called a 2-interchange



Local Search-TPS-2 Opt

The simplest is called $2 - opt$. It starts with a random permutation of cities (call this tour T) and tries to improve it . The neighborhood of T is defined as the set of all tours that can be reached by changing two nonadjacent edges in T. This move is called a $2 - interchange$.

A new tour T' replaces T if it is better. If none of the tours in the neighborhood of T is better than T, then the tour T is called $2 - optima$ and the algorithm terminates .

The $2 - opt\ algorithm$ can be generalized easily to a $k - opt$ procedure, where either k ; or up to k , edges are selected for removal and are replaced by lower cost connections

Traditional Search-Greedy algorithm

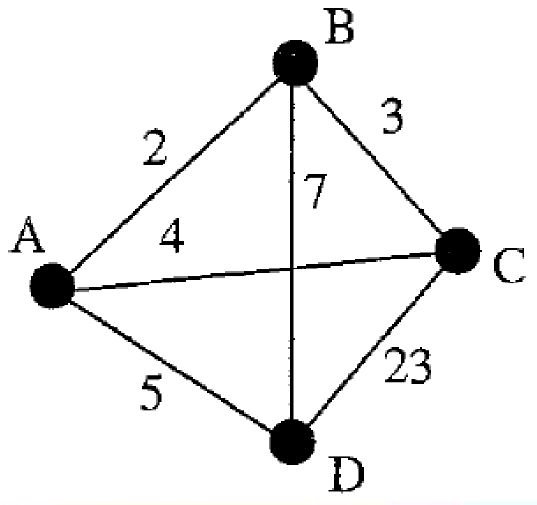
Greedy algorithms attack a problem by constructing the complete solution in a series of steps. They are easy to implement.

We assign the values for all of the decision variables one by one and at every step make the best available decision. This approach is short-sighted since taking the optimum decisions at each separate step doesn't always return the optimum solution overall.

Traditional Search-TPS-Greedy algorithm

Nearest-neighbor heuristic:

Starting from a random city, proceed to the nearest unvisited city and continue until every city has been visited, at which point return to the first city. Such a tour can be far from perfect . There's often a high price to pay for making greedy choices at the beginning of the process.
Starting from city A, this greedy heuristic constructs a tour A - B - C - D - A, with the total cost of 2 + 3 + 23 + 5 = 33, but the tour A - C - B - D - A costs only 4 + 3 + 7 + 5 = 19.



Escaping a local optima

Either heuristics guarantee discovering the global solution, but are too expensive (i.e. , too time consuming) for solving typical real-world problems, or else they have a tendency of "getting stuck" in local optima.

Since there is almost no chance to speed up algorithms that guarantee finding the global solution, i.e. , there is almost no chance of finding polynomial-time algorithms for most real problems (as they tend to be NP-hard) , the other remaining option aims at designing algorithms that are capable of escaping local optima.

Simulated annealing: An additional parameter (called temperature) that changes the probability of moving from one point of the search space to another.

Escaping a local optima-Simulated Annealing

Differences between simulated annealing and local search:

1. Stopping criterion: Simulated annealing is executed until some external "termination condition" is satisfied as opposed to the requirement of local search to find an improvement.

2. the function "improve(x , T)" doesn't have to return a better point from the neighborhood of x . It just returns an accepted solution y from the neighborhood of x, where the acceptance is based on the current temperature T .

3. In simulated annealing, the parameter T is updated periodically, and the value of this parameter influences the outcome of the procedure "improve" . This feature doesn't appear in local search.

Escaping a local optima-Simulated Annealing

- Choose $y \in N(x_k)$
- If $f(y) \leq f(x_k)$ then $x_{k+1} = y$
- If not $x_{k+1} = y$ with a probability

$$e^{-\frac{f(y)-f(x)}{T}}$$

With $T > 0$

- Choose a random value (r) between 0 and 1
- Accept y if

$$e^{-\frac{f(y)-f(x)}{T}} > r$$

http://en.wikipedia.org/wiki/Simulated_annealing

Illustration in class: for TSP on Matlab