

Offline 2: Local Search (Simulated Annealing)

You have a car.

A car is made of

- an engine
- a transmission
- a tire
- a roof.

Engine choice: *engines.txt* file

Tire choices: *tires.txt* file

Transmission choices: *transmissions.txt* file

Roof choices: Sunroof, Moonroof, No roof

You have a car like **START STATE** (EFI, Danlop, AT, Noroof)

You want to have a car like **GOAL STATE** (V6, Hankook, AT, Moonroof)

or **GOAL STATE** (V12, Pirelli, CVT, Sunroof)

Each year you can change only 1 component of the car which results in a way that makes a valid car model i.e **You can't change your component in a way which makes your car invalid at any point of time.** The valid car models are given in the **valid_cars.csv** file.

You want to have your dream car as soon as possible i.e Minimize the years after which you get your desired car.

How to calculate the cost between two states: Number of mismatched components.

Example: $\Delta E = 1$ for (EFI, Danlop, AT, Noroof) and (EFI, Danlop, AT, Sunroof) as only the roof state mismatches.

If at any point, you arrive at an invalid state, you can't go any further from that one, just discard that state.

Probability function for simulated annealing: $e^{(\Delta E/t)}$ **[Note: this function is required only when ΔE is negative]**

Here, $\Delta E = \text{next_node.val} - \text{current_node.val}$

Y = year passed aka BFS level

$t = 1 / Y$

Probability modeling:

`random.uniform(0, 1) <= e`

Also, Test your code for **GOAL STATE** (V12, Pirelli, CVT, Sunroof)

Simulated annealing

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
           schedule, a mapping from time to “temperature”

  current  $\leftarrow$  MAKE-NODE(problem.INITIAL-STATE)
  for  $t = 1$  to  $\infty$  do
     $T \leftarrow$  schedule( $t$ )
    if  $T = 0$  then return current
    next  $\leftarrow$  a randomly selected successor of current
     $\Delta E \leftarrow$  next.VALUE – current.VALUE
    if  $\Delta E > 0$  then current  $\leftarrow$  next
    else current  $\leftarrow$  next only with probability  $e^{\Delta E/T}$ 
```

- Memory requirement $O(1)$
- If you use heuristic cost instead of heuristic value, then ΔE should be *current.cost* - *next.cost*.

Instruction:

- Upload the (.py) or (.ipynb) file in the eLMS
- You have to make sure your code runs.
- You can name your file as your ID. (112....345.ipynb) and upload it
- **Bonus: Print the model you find in each year. (i.e., print the path)**