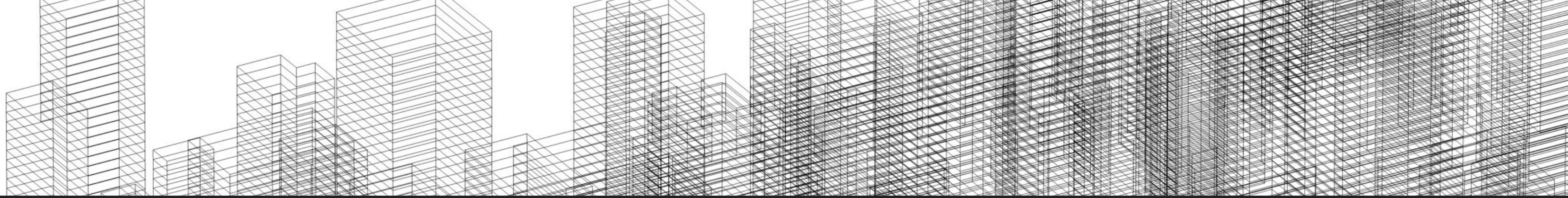


Router Placement

IART - Checkpoint 1
Ana Barros - up201806593
João Costa - up201806560
João Martins - up201806436

Problem Specification

Task: Given a building plan, decide where to put wireless router sand how to connect them to the fiber backbone to maximize coverage and minimize cost. More Info at: https://storage.googleapis.com/coding-competitions.appspot.com/HC/2017/hashcode2017_final_task.pdf

Building:
. **H** rows and **W** columns
. Coords **[r, c]**, starting at 0
. [0, 0] is the upper left corner of the grid
. wall is **'#'**
. target is **'.'** - these cells need **wireless coverage**
. void is **'-'** - these cells **don't** need wireless coverage

Routers:
. Each router covers at most **(2 * R + 1)^2** cells around itself, where **R** is the router's **range**.
. Signals are stopped by walls
. Signals are stopped by walls
  . |a - x| <= R,
  . |b - y| <= R,
  . there is no wall [w, v] where **min(a, x) <= w <= max(a, x) && min(b, y) <= v <= max(b, y)**
. described as: there are no walls in the smallest enclosing rectangle of [a, b] and [x, y]

Backbone:
. Routers can only be placed in cells connected to the backbone.
. In the beginning only 1 cell is connected to the backbone.
. Cells of any type can be connected to the backbone (one of its eight neighboring cells must already be connected to the backbone).

Budget:
. Placing a router costs **Pr**
. Connecting a cell to the backbone costs **Pb**
. The maximum budget is **B**

Input:
.1st line - **H W R**
.2nd line - **Pb Pr B**
.3rd Line - **br bc**
.Rest of lines - Cells

(1 <= H <= 1000) - number of rows on the grid
(1 <= W <= 1000) - number of columns on the grid
(1 <= Pb <= 5) - price of connecting one cell to the backbone
(5 <= Pr <= 100) - price of one wireless router
(1 <= B <= 10^9) - maximum budget
(0 <= br < H) - row of the initial cell connected to the backbone
(0 <= bc < W) - column of the initial cell connected to the backbone

```
12 22 3
1 100 220
4 6
----------------------
-####################-
-#....#......#.#....#-
-#....#.########....#-
-#............#.....#-
-######.............#-
-#............#######-
-#....###...........#-
-#....#.#.....#.....#-
-#....#.#.....#.....#-
-####################-
----------------------
```

Bibliographic Search

Router placement with genetic algorithm:
https://github.com/tasosxak/Router-Placement/blob/master/routers.py

Steiner tree problem:
https://en.wikipedia.org/wiki/Steiner_tree_problem
http://www.cs.ucr.edu/~michalis/COURSES/240-08/steiner.html

Simulated annealing:
https://machinelearningmastery.com/simulated-annealing-from-scratch-in-python/

Kruskals minimum spanning tree:
https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/

Chebyshev distance:
https://en.wikipedia.org/wiki/Chebyshev_distance

Genetic Algorithm:

https://en.wikipedia.org/wiki/Genetic_algorithm

Formulation of the problem as a search problem

Solution Representation

```
Router price: 100
Backbone price: 1
Max budget: 220
Steps taken: 3
Board 12x22. Router range is 3. Backbone at (4, 6)
Value is 41014. There are 41 cells covered by 2 routers. The budget spent was 206
----------------------
-####################-
-#....#......#.#....#-
-#....#.########....#-
-#....B..::::.#.....#-
-#####bb.::::.......#-
-#:::b.b::::.#######-
-#:::b###R::........#-
-#::b:#.#::::.#.....#-
-#:R::#.#::::.#.....#-
-####################-
----------------------
```

PNG:

- Orange: Covered Cell
- Blue: Uncovered cell
- Pink: Backbone
- Yellow: Wall
- Black: Void

Neighborhood/Mutation: Add router.

Crossover Function: Merge all odd routers of one parent and all even routers of the other parent into a new child node.

Rigid Constraints:

- We can’t go over the max budget.
- Routers have to be placed next to a backbone.
- Routers can’t be placed on top of walls, other routers, or void cells.

Evaluation Function: The **cost** is given by **(R * Pr + Bb * Pb)**, where:

- **R** is the number of placed routers,
- **Pr** is the price of a router,
- **Bb** is the numbers of placed backbones (besides the initial one),
- **Pb** is the price of each backbone.

If this cost is higher than the budget, **B**, the value of the solution is 0. Otherwise, the value of a node follows the formula **C * 1000 + (B - Cost)**, where **C** is the numbered of cells covered by at least one router.

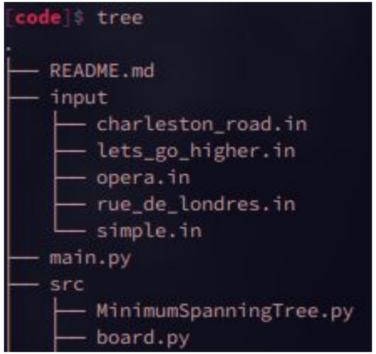
Current work status

Programming Language: Python

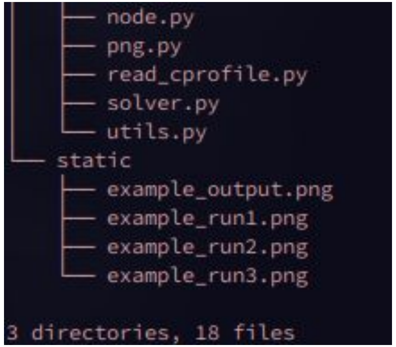
Development Environment: NeoVim and VSCode

File structure:

- **input/** - contains the input files.
- **src/** - contains the source code files.
- **static/** - contains images used in **README.md** file
- **main.py** - is the menu interface.
- **README.md** - is the problem and project description



```
[code]$ tree
.
├── README.md
├── input
│   ├── charleston_road.in
│   ├── lets_go_higher.in
│   ├── opera.in
│   ├── rue_de_londres.in
│   └── simple.in
├── main.py
├── src
│   ├── MinimumSpanningTree.py
│   ├── board.py
```



```
│   ├── node.py
│   ├── png.py
│   ├── read_cprofile.py
│   ├── solver.py
│   └── utils.py
└── static
    ├── example_output.png
    ├── example_run1.png
    ├── example_run2.png
    └── example_run3.png

3 directories, 18 files
```

Data Structures:

- **Classes**
  - **Board** - Holds a problem representation
  - **Node** - Represents a possible solution (routers' and backbones' placement in the board)
  - **Graph** - Tree representation of the router's placement which is used to calculate the best backbone path
  - **png** - Used to export the solution to png
  - **Solver** - Finds a solution to a problem using different algorithms
- **Sets**
  - **CoveredCells** - Keep track of the cells covered by a router(s)
  - **Walls** - Store the coordinates of all the walls of a given board
  - **Backbones** - Store the coordinates of all the backbones placed for a solution
- **List**
  - **Routers** - Contains the positions of all placed routers
  - **AvailablePositions** - is a shuffled list of all positions where a router can be placed

Group 03

Members:

- Ana Inês Oliveira de Barros, up201806593@fe.up.pt

- João de Jesus Costa, up201806560@fe.up.pt

- João Lucas Silva Martins, up201806436@fe.up.pt