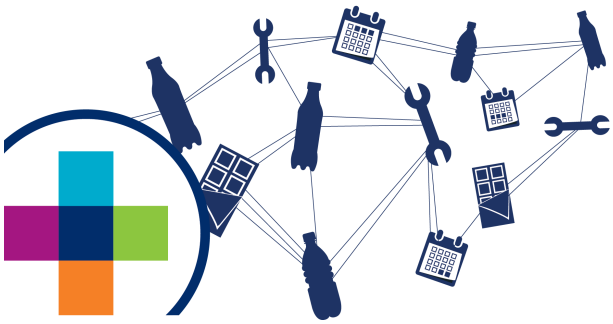
VEROLOG
VEHICLE ROUTING
AND LOGISTICS

ORTEC
OPTIMIZE YOUR WORLD

# VeRoLog Solver Challenge 2019



# Analyzing instances and solutions

*This document describes and illustrates the use of two new tools to analyze the cost function of a solution along the planning horizon and to visualize instances and solutions. These new tools can be found at the usual place* `https://verolog2019.ortec.com/downloads`.

## Contents


**1 Motivation** **1**

**2 The new tools** **2**
2.1 Visualizer . . . . . 2
2.1.1 A simple solution . . . . . 5
2.2 Analyzer . . . . . 7
2.2.1 The example instance . . . . . 7
2.2.2 Analyzing the early instances . . . . . 8
2.2.3 Analyzing and improving the simple solution . . . . . 8

**3 Dependencies** **11**
3.1 analyzerVerolog2019.py . . . . . 11
3.2 visualizerVerolog2019.py . . . . . 12


## 1 Motivation

The challenge just reached the first month of activity. The organization decided to share a pair of tools developed to analyze and visualize instances and solutions. We hope that these tools may inspire the contesting teams in finding ways to further improve their solutions.

## 2 The new tools

We will use the same example as supplied in https://verolog2019.ortec.com/pdf/VSC2019_info.pdf in order to illustrate the usage of the new tools. We list below the contents of the corresponding file in two parallel columns for the sake of compactness.

```
DATASET = VeRoLog solver challenge 2019
NAME = testInstance


DAYS                =   7
TRUCK_CAPACITY      =   6
TRUCK_MAX_DISTANCE = 250


TRUCK_DISTANCE_COST      =      1
TRUCK_DAY_COST           =    100
TRUCK_COST               = 100000


TECHNICIAN_DISTANCE_COST =      1
TECHNICIAN_DAY_COST      = 100000
TECHNICIAN_COST          =    100


MACHINES = 4
1 2 200
2 1 200
3 3 500
4 2 100
```

```
LOCATIONS = 9
1 10 50
2 20 10
3 50  5
4 33  7
5 40 40
6 70 40
7  1 35
8 10  5
9 25 60

REQUESTS = 7
1 2 1 7 1 1
2 3 3 6 3 1
3 4 2 3 4 2
4 5 5 7 1 3
5 6 1 4 2 1
6 7 1 6 4 1
7 7 3 5 2 4

TECHNICIANS = 5
1 3 100 2 0 1 1 1
2 8 100 1 1 1 0 0
3 8 100 5 0 0 1 0
4 9 100 2 1 1 0 1
5 1  50 1 1 1 1 1
```

### 2.1 Visualizer

The purpose of the visualizer is to draw the instance and, optionally, a solution along the planning horizon.

When used on an instance only – without solution – an image is drawn per day in the planning horizon.

Because the output consists of multiple images, it can be seen as an animation, and three different output formats are supported: `mp4`, `html` and multipage `pdf`.

In case a solution is also shown, multiple images per day are drawn, showing the progress of the routes. In that case each image corresponds to the situation after a given instance is traveled on a day. This is represented by 'distance@day' shown on the top left corner of the image.

The visualizer is implemented in `visualizerVerolog2019.py` and, as usual, may be called with argument `-h` or `--help` to display its usage and all supported parameters.

Doing so leads to the result below.

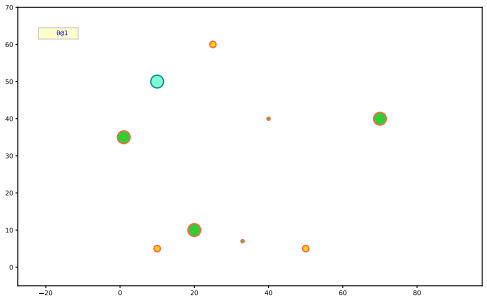
```
usage: visualizerVerolog2019.py [-h] [--solution SOLUTION_FILE]
                                [--instance INSTANCE_FILE] [--xkcd]
                                [--nofDaySteps NOFDAYSTEPS]
                                [--smallSize SMALLSIZE]
```

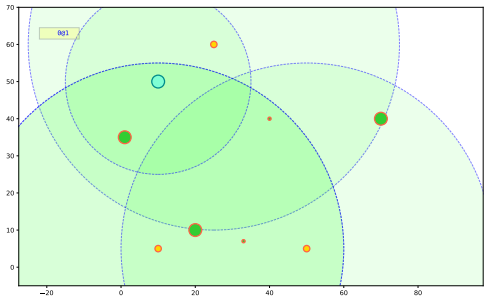
```
                                [--mediumSize MEDIUMSIZE]
                                [--largeSize LARGESIZE]
                                [--extraMargin EXTRAMARGIN] [--inchH INCHH]
                                [--inchV INCHV] [--fps FPS] [--actionRadius]
                                [--mp4] [--html] [--pdf]

Read and visualize instance, maybe with solution file.

optional arguments:
  -h, --help            show this help message and exit
  --solution SOLUTION_FILE, -s SOLUTION_FILE
                        The solution file
  --instance INSTANCE_FILE, -i INSTANCE_FILE
                        The instance file
  --xkcd                draw in xkcd style
  --nofDaySteps NOFDAYSTEPS
                        Provide number of day steps, default 10
  --smallSize SMALLSIZE
                        Provide small size of locations, default 25
  --mediumSize MEDIUMSIZE
                        Provide medium size of locations, default 100
  --largeSize LARGESIZE
                        Provide large size of locations, default 400
  --extraMargin EXTRAMARGIN
                        Provide extra margin to plot, default 10
  --inchH INCHH         Provide horizontal size in inches, default 13
  --inchV INCHV         Provide vertical size in inches, default 8
  --fps FPS             Provide number of frames per second for mp4, default 8
  --actionRadius        show action radius of technicians
  --mp4                 animate to mp4
  --html                animate to html
  --pdf                 animate to multipage pdf
```

As an example, consider calling the visualizer with arguments `-i "data/example.txt" --pdf`. Note that the specific folder and file name of the instance should match your situation, for the sake of this example `example.txt` includes the data listed above.

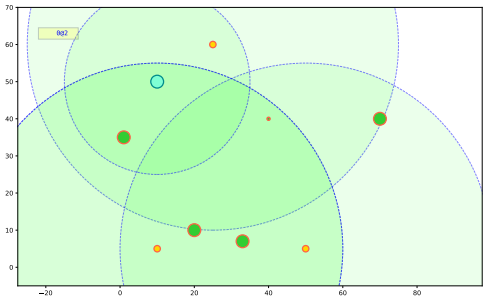
We see the several locations. Yellow locations have technicians, and they might or might not have requests. Green locations have requests, but no technicians. The location in light blue is the depot, which might have technicians. Each location is shown with different sizes (diameters). The large diameter is used when the location is open. The depot is open throughout the planning horizon and each request location is open on the days specified.

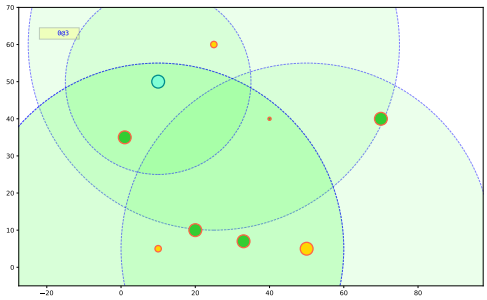
The pictures below show the beginning of day 1, as described by 0@1. The second picture was obtained by adding `--actionRadius`.

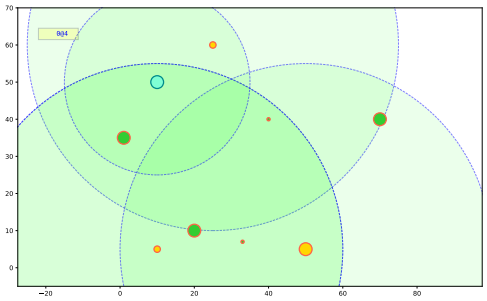


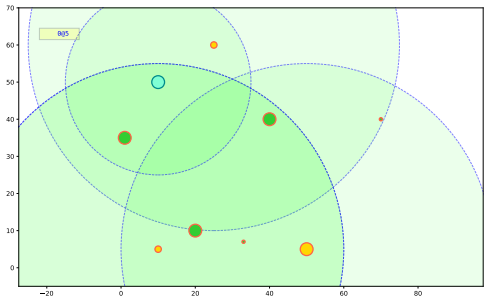



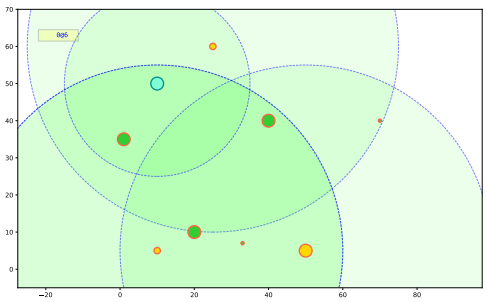

Adding `--actionRadius` show semi-transparent green circles centered on the locations with technicians. There is indeed one such action radius circle centered at the depot, showing that at least one technician is based there. The radius of these circles is equal to the half of the maximum daily distance of the corresponding technician. For the technician based at depot, as we may see in the data inside the instance file, this is technician 5 with an action radius of 50, therefore the circle has radius 25. Because these circles are semitransparent, overlap leads to color intensification, making visually clear which customer locations are within reach of few or many technicians.

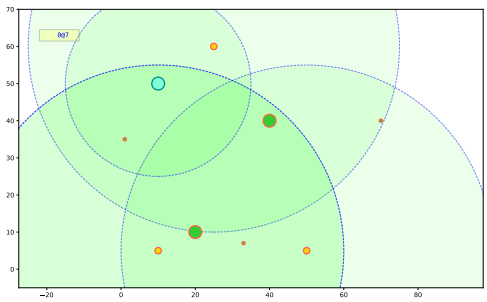


As mentioned before, one figure is drawn per day of the planning horizon. Since we have seen day 1 before, above only the days $2, \ldots, 7$ are shown, in a row-major fashion: the first row is day 2 and 3 and so forth.

The dots have three sizes (diameters):

- small green for the locations with requests that are not open for delivery
- medium yellow for locations with technicians
- large for locations that are open. These can be of any color. The depot is always open.

Notice how the dots become larger on the days that the corresponding request may be delivered.

### 2.1.1 A simple solution

A very simple solution strategy amounts to deliver each request as soon as the corresponding delivery window opens. Delivery routes are expanded in a greedy way for as long as capacity and maximum distance are respected, otherwise a new route starts. Similarly, each delivery is installed in the first subsequent day. The technicians are used in a greedy way as well. This simple strategy may lead to the solution below.

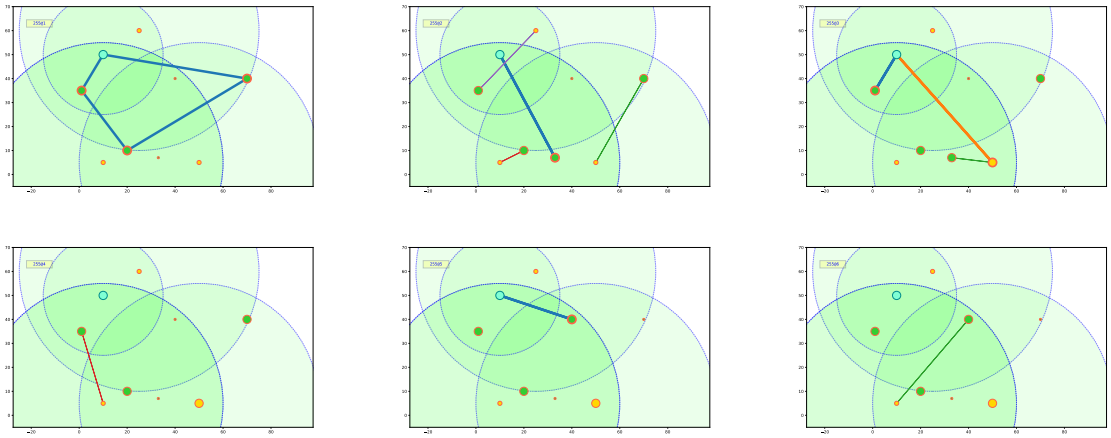
```
DATASET = VeRoLog solver challenge 2019
NAME = simple greedy solution

DAY = 1
NUMBER_OF_TRUCKS = 1
1 6 1 5
NUMBER_OF_TECHNICIANS = 0

DAY = 2
NUMBER_OF_TRUCKS = 1
1 3
NUMBER_OF_TECHNICIANS = 3
1 5
2 1
4 6

DAY = 3
NUMBER_OF_TRUCKS = 2
1 7
2 2
NUMBER_OF_TECHNICIANS = 1
1 3

DAY = 4
NUMBER_OF_TRUCKS = 0
NUMBER_OF_TECHNICIANS = 2
1 2
2 7

DAY = 5
NUMBER_OF_TRUCKS = 1
1 4
NUMBER_OF_TECHNICIANS = 0

DAY = 6
NUMBER_OF_TRUCKS = 0
NUMBER_OF_TECHNICIANS = 1
2 4

DAY = 7
NUMBER_OF_TRUCKS = 0
NUMBER_OF_TECHNICIANS = 0
```

To visualize this solution, day by day, add the `-s` option as in `-i "data/example.txt" -s "example-simple.sol" --pdf --actionRadius`.

We omit day 7 since nothing happens there, the corresponding figures match the seventh figure of the seventh day when visualizing the instance only.

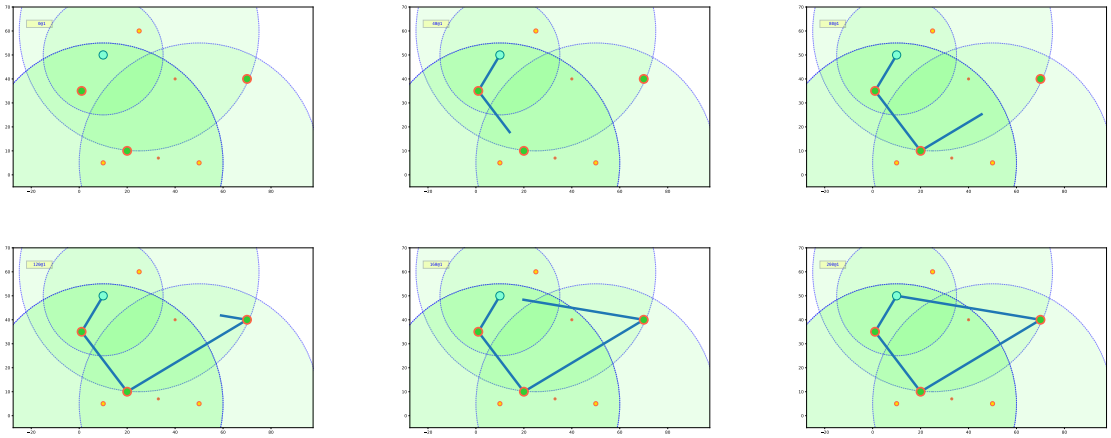
Truck routes are shown with thick lines, technician routes with thin lines.

The thickness of the location border increases for each day, including delivery, prior to installation.



We have mentioned before that during each day the routes are shown progressing stepwise. The number of steps is controlled by `--nofDaySteps`.

Below we show day 1 in the case that `--nofDaySteps 6`.



Particularly useful to know is the ability to use wildcards. These can be used on the `solution` or the `instance` arguments. In case of using wildcards in the `solution` argument, `instance` should provide the folder containing the corresponding instance files.

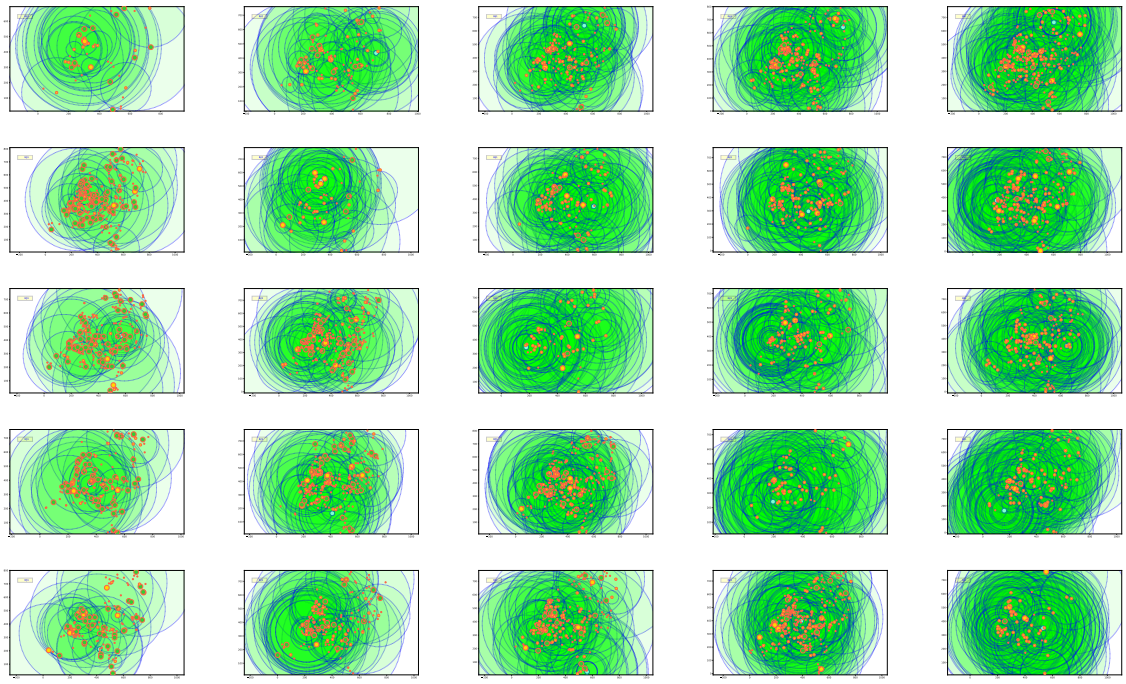
For example, suppose that your solutions to the `VSC2019_ORTEC_early` instances are located in `D:/VSC2019/solutions` and named `VSC2019_ORTEC_early_01.sol` and so on. When using wildcards it is important that the name of the solution and the instance match, only the extensions differ. Then, if the instances are located in `D:/VSC2019/data` you may visualize all solutions with the arguments `-i "d:/vsc2019/data/" -s "d:/vsc2019/solutions/*.sol"`. Notice how `instance` is used to pass the folder name only.

You may have subdirectories as well, as in `best`, `test` for example. Then you may use more than one wildcard as in `-i "d:/vsc2019/data/" -s "d:/vsc2019/solutions/*/*.sol"`.

Wildcards may be used for the `instance` parameter as well, but in such a case the `solution` parameter **must** be omitted.

Calling the visualizer on `-i "d:/vsc2019/data/*.txt"` allows us to *see* all early instances. Please find them below in a matrix of $5 \times 5$ pictures, shown in row major order, i.e. the first row is $1, \ldots, 5$ and so on.

You may notice, for instance, how some instances are more limitative in terms of number and action radius of technicians.



The instances may also differ substantially in terms of the structure of the cost function. The following tool is particularly useful in that case.

## 2.2 Analyzer

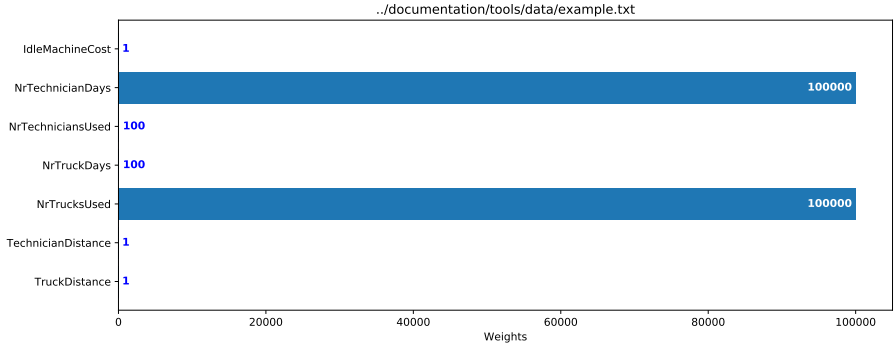
The purpose of the analyzer, implemented in `analyzerVerolog2019.py`, is to show the cost function weights of an instance and, in case of a solution, the evolution of the cost function during the planning horizon.

Also the analyzer reacts to `-h` or to `--help` displaying its usage.

```
usage: analyzerVerolog2019.py [-h] [--solution SOLUTION_FILE]
                              [--instance INSTANCE_FILE] [--pdf] [--show]
                              [--verbose]

Read and analyze a instance or solution file.

optional arguments:
  -h, --help            show this help message and exit
  --solution SOLUTION_FILE, -s SOLUTION_FILE
                        The solution file or mask of several files
  --instance INSTANCE_FILE, -i INSTANCE_FILE
                        The single instance file of folder in case of several
                        solutions
  --pdf                 save figure to pdf
  --show                show figure in console
  --verbose             show commulative costs as text in the console
```

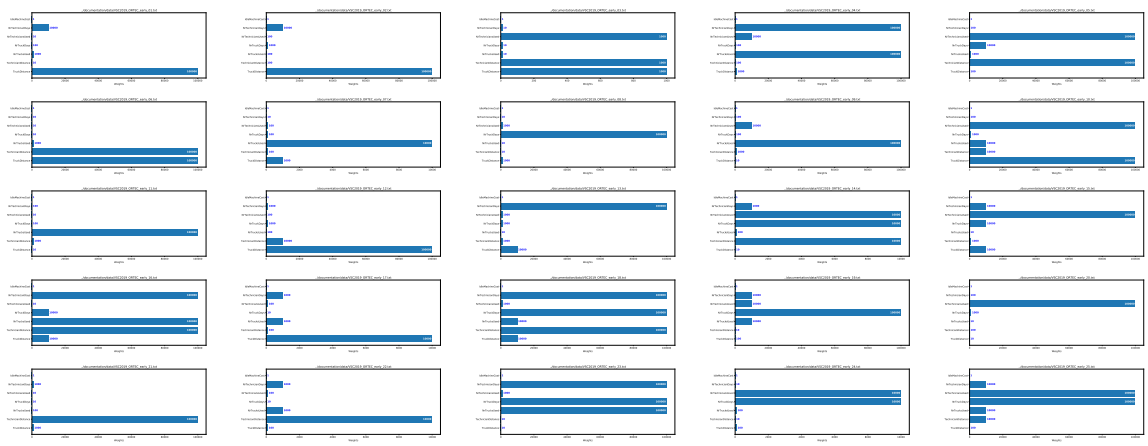
### 2.2.1 The example instance

Visualizing the weights shows their real importance.




### 2.2.2 Analyzing the early instances

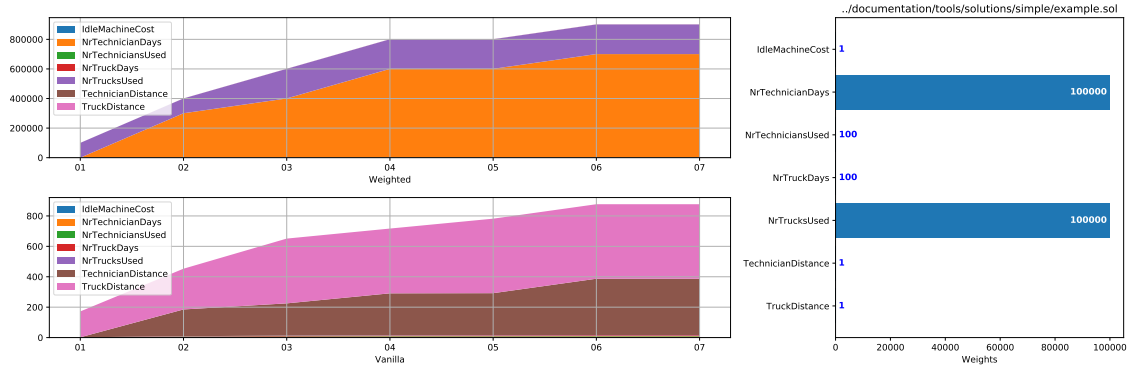
The analyzer uses the same wildcard logic as the visualizer, therefore we may also analyze all instances in the same way, obtaining the following.



### 2.2.3 Analyzing and improving the simple solution

If we run the analyzer with the arguments `-s "example-simple.sol" -i "data/example.txt" --pdf --verbose` then we obtain the output below.

```
example-simple.sol
        TruckDistance: 490
        NrTruckDays: 5
        NrTrucksUsed: 2
        TechnicianDistance: 370
        NrTechnicianDays: 7
        NrTechniciansUsed: 3
        IdleMachineCost: 0
        Cost: 901660
[[100270 400944 601338 801402 801566 901660 901660]
 [      0      0      0      0      0      0      0]
 [      0      3      4      6      6      7      7]
 [      0      3      3      3      3      3      3]
 [      1      2      4      4      5      5      5]
 [      1      1      2      2      2      2      2]
 [      0    176    212    276    276    370    370]
 [    170    268    426    426    490    490    490]]
```



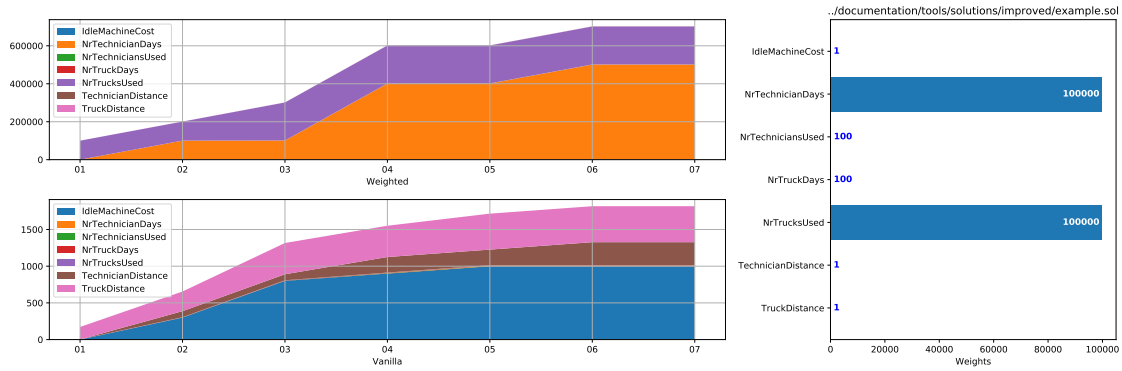

We see that the total cost of 901660 is obtained mainly as a function of the number of trucks used and of the number of technician days. The weighted graph clearly shows that lowering the number of technician days improves the solution. A bit of puzzling shows that each of the following technician routes are feasible, provided that the group on each column occurs on a different day.

| | | |
|---|---|---|
| 1 5 | 1 3 2 | 4 6 4 |
| | 2 1 | |
| | 4 7 | |

By planning the technician routes listed above on the first day that the corresponding requests are available for installation we obtain the solution below. Notice that this solution increases the idle machine costs, but reduces the number of technician days.

```
DATASET = VeRoLog solver challenge 2019
NAME = improved solution

DAY = 1
NUMBER_OF_TRUCKS = 1
1 6 1 5
NUMBER_OF_TECHNICIANS = 0

DAY = 2
NUMBER_OF_TRUCKS = 1
1 3
NUMBER_OF_TECHNICIANS = 1
1 5

DAY = 3
NUMBER_OF_TRUCKS = 2
1 7
2 2
NUMBER_OF_TECHNICIANS = 0

DAY = 4
NUMBER_OF_TRUCKS = 0
NUMBER_OF_TECHNICIANS = 3
1 3 2
2 1
4 7

DAY = 5
NUMBER_OF_TRUCKS = 1
1 4
NUMBER_OF_TECHNICIANS = 0

DAY = 6
NUMBER_OF_TRUCKS = 0
NUMBER_OF_TECHNICIANS = 1
4 6 4

DAY = 7
NUMBER_OF_TRUCKS = 0
NUMBER_OF_TECHNICIANS = 0
```

We can now analyze this solution, as shown below.

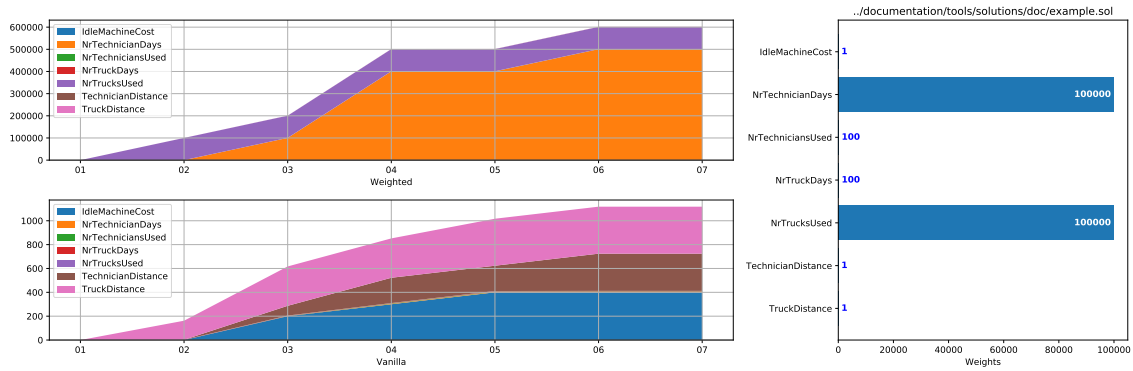



```
improved/example.sol
        TruckDistance: 490
        NrTruckDays: 5
        NrTrucksUsed 2
        TechnicianDistance: 312
        NrTechnicianDays: 5
        NrTechniciansUsed: 3
        IdleMachineCost: 1000
        Cost: 702602

[[100270 200950 301808 602238 602502 702602 702602]
 [     0    300    800    900   1000   1000   1000]
 [     0      1      1      4      4      5      5]
 [     0      1      1      3      3      3      3]
 [     1      2      4      4      5      5      5]
 [     1      1      2      2      2      2      2]
 [     0     82     82    212    212    312    312]
 [   170    268    426    426    490    490    490]]
```

We could indeed improve the solution. A bit more puzzling leads to the following solution. This is the solution shown in https://verolog2019.ortec.com/pdf/VSC2019_info.pdf.

```
DATASET = VeRoLog solver challenge 2019
NAME = beter solution

DAY = 1
NUMBER_OF_TRUCKS = 0
NUMBER_OF_TECHNICIANS = 0

DAY = 2
NUMBER_OF_TRUCKS = 1
1 5 3
NUMBER_OF_TECHNICIANS = 0

DAY = 3
NUMBER_OF_TRUCKS = 1
1 6 7 0 1 2
NUMBER_OF_TECHNICIANS = 1
1 5

DAY = 4
NUMBER_OF_TRUCKS = 0
NUMBER_OF_TECHNICIANS = 3
1 3 2
2 1
4 7

DAY = 5
NUMBER_OF_TRUCKS = 1
1 4
NUMBER_OF_TECHNICIANS = 0

DAY = 6
NUMBER_OF_TRUCKS = 0
NUMBER_OF_TECHNICIANS = 1
4 6 4

DAY = 7
NUMBER_OF_TRUCKS = 0
NUMBER_OF_TECHNICIANS = 0
```

```
doc/example.sol
        TruckDistance: 394
        NrTruckDays: 3
        NrTrucksUsed 1
        TechnicianDistance: 312
        NrTechnicianDays: 5
        NrTechniciansUsed: 3
        IdleMachineCost: 400
        Cost: 601706
[[     0 100260 200912 501342 501606 601706 601706]
 [     0      0    200    300    400    400    400]
 [     0      0      1      4      4      5      5]
 [     0      0      1      3      3      3      3]
 [     0      1      2      2      3      3      3]
 [     0      1      1      1      1      1      1]
 [     0      0     82    212    212    312    312]
 [     0    160    330    330    394    394    394]]
```




# 3 Dependencies

The tools provided depend upon `InstanceVerolog2019.py` and `SolutionVerolog2019.py`. Furthermore, `visualizerVerolog2019.py` depends upon `analyzerVerolog2019.py`. Below you find the dependencies on standard or freely available libraries and packages.

## 3.1 analyzerVerolog2019.py

Depends on

- `numpy`
- `matplotlib.pyplot`
- `matplotlib.backends.backend_pdf`
- `glob`
- `operator`
- `os.path`
- `pathlib`

### 3.2 visualizerVerolog2019.py

Depends on

- numpy
- matplotlib
- matplotlib.pyplot
- matplotlib.animation
- matplotlib.collections
- matplotlib.patches
- matplotlib.backends.backend_pdf
- itertools
- scipy.spatial
- math
- copy

We hope that you may find these tools useful and that using them may inspire you to improve your algorithms. You may take a moment to play with the parameters and try different output formats. In order to support --mp4 you may need to install http://www.imagemagick.org/script/download.php#windows. The archive containing this document and the tools also includes animations in mp4 of the three solutions discussed here.