

**Ministry of Education and Science**

**Technical University of Moldova  
CIM Faculty**

# **R E P O R T**

**Laboratory work #1**

Done by:  
Calancea Daniel

Verified by:  
Turcanu Victor

Aunt Marry is an old lady from Chişinău. She lives in the area near the Circus and everyday she needs to take the trolleybus in order to get to the city center. From there she can go further to the *Piaţa Centrală*, where she doesn't buy anything, rather she spends her time hearing the latest news in the city.

Unfortunately, Aunt Marry is upset by the way the municipal transport is optimized especially when she returns back home. For example, from station **Ştefan cel Mare** to station **Circul** there go trolleybuses nr: 7, 10, 24, 25 (I will skip 11 and 14 as they are routed comparably few times per day). Thus, she doesn't care so much what is the number of the trolleybus she takes in order to reach the destination station. One thing she noticed is that often the trolleybuses arrive as a train (all at once) to the station, but there appear gaps of around 10 or even 15 minutes when there no trolleybuses arrive. What Aunt Marry wants, neglecting the car traffic in our city, is an evenly distributed timetable when she can wait for the minimum amount of time the transportation that will take her to the destination.

## Subproblem 1

Given the [timetables](#) of trolleybus arrival, find an algorithm to optimize the traffic from station **Ştefan cel Mare** to station **Circul**. Visualize the data output of your solution.

## Solution

In order to solve this problem, we will need an algorithm to check if the waiting time is optimized, to search for new solutions and to make sure that all rules are respected. The rules for this problem are the following:

- If you shift one trolleybus, he must be shifted on all stations.
- We can't add more trolleybuses.
- We must keep the time between arrivals of same trolleybus even.

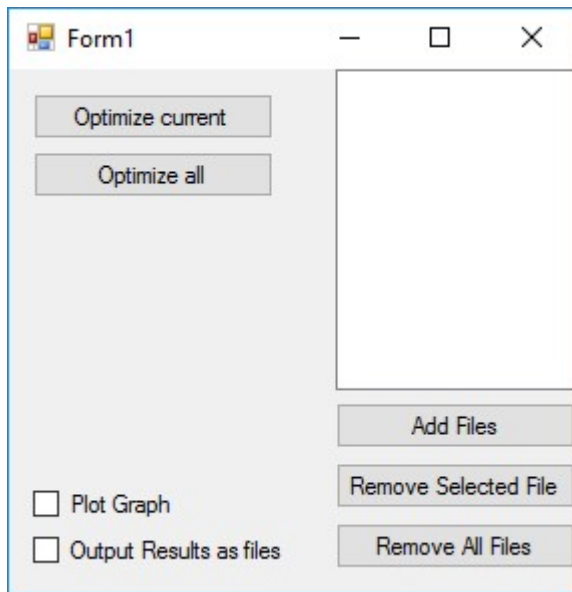
The idea I implemented is the following:

- We have a variable that checks the rate of optimality of station, the variable is calculated by summing  $e^{\text{(waiting of the next trolleybus)}}$  with all gaps we have, in this way if we have a big gap this number will become very big.
- We shift every trolleybus from -10 to 10 minutes by 1 minute and after that we check our rate of optimality if it is smaller we apply it. When we shift we shift the

trolleybus on all of our stations (implicit rule is that all imputed stations are connected).

- We plot after that the new gaps we got between our arrival of trolleybuses.
- We also have the option to save the new timetable file.

## Program preview



Here we have several options and those include:

- Add files.
- Remove the selected file from list.
- Remove all the files from list.
- Optimize the current selected timetable.
- Optimize all uploaded timetables.
- Plot graph.
- Output Results as files.

In order for the file to be parsed correctly it should be in the format:

Trolleybus.nr) HH:MM, HH:MM, HH:MM

Example:

7) 11:35, 11:53, 12:11, 12:29, 12:47, 13:05, 13:23

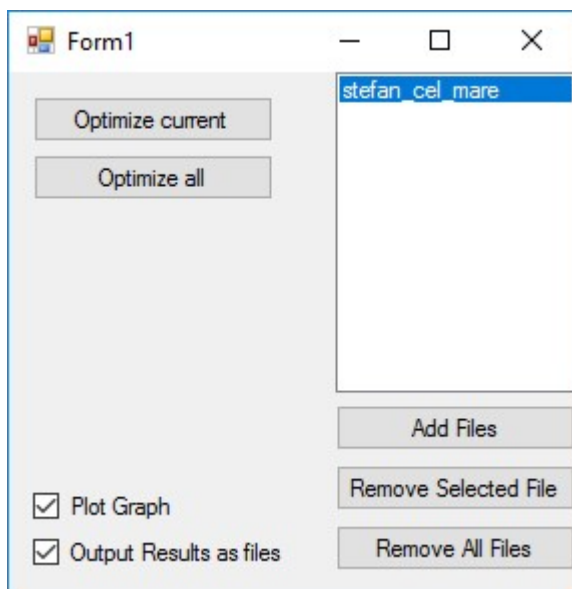
Multiple uploaded files have an implicit rule that they are connected when they are processed.

The optimization algorithm is the same for one station as for all. In case when we have multiple stations we sum the rate for every station. The algorithm is located in the class **OptimizeStations.cs** where we have the methods:

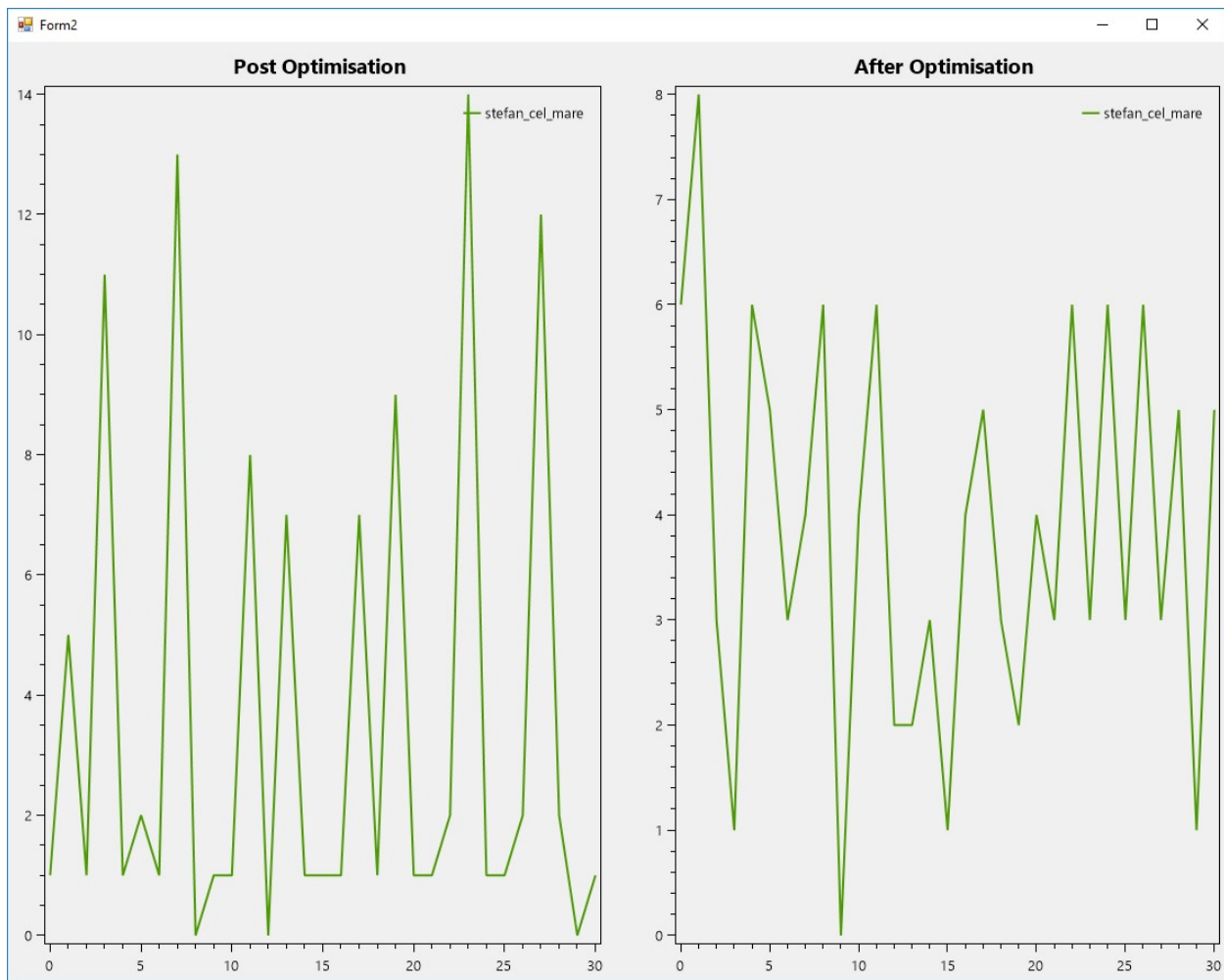
- **fileParse**(parses the files that were passed to the constructor).
- **getInitialOptimalValue** (calculates the rate of optimality for the originally parsed data)
- **getShiftedTimesBus** (has 2 parameters: the value of shift if negative we shift to left otherwise to right, the name of the trolleybus we want to shift. It generates a dictionary that contains the trolleybus name as key and a list of shifted timetables as value)
- **getShiftedDict** (has same parameters as getShiftedTimesBus however it returns a complete dictionary with shift changes applied to those trolleybuses).
- **optimiseTime** (the main algorithm that performs shifts and checks for optimality)
- **getPlotData** (used to get the data for plotting library, outputs a dictionary containing gaps between arrival for every station before and after optimization)
- **generateOutputFiles** (generates optimized output timetables files for every input file, files are stored in optimised/ folder with the same name as original)

To plot the graph we used the library OXYPlot and as input we gave X and Y points.

So let's input now our station and see the graph and output file:



The screenshot shows a Windows application window titled "Form1". On the left side, there are two buttons: "Optimize current" and "Optimize all". Below these buttons are two checked checkboxes: "Plot Graph" and "Output Results as files". On the right side, there is a list box containing the text "stefan cel mare". Below the list box are three buttons: "Add Files", "Remove Selected File", and "Remove All Files".



We can observe an improvement, we normalized the waiting time so that now we don't have gaps so big. We can also see the numeric value in debug mode.

So we have the starting rate that is: 1881148

After optimization we have: 6805

Big difference huh?

Also we can't see so clear on the graph because X axis is just an incremented number for every gap not actual time.

The output file with timetables contains:

7) 11:24, 11:42, 12:00, 12:18, 12:36, 12:54, 13:12

10) 11:41, 11:56, 12:10, 12:20, 12:28, 12:38, 12:51, 13:09, 13:23

24) 11:38, 11:53, 12:06, 12:16, 12:23, 12:33, 12:45, 13:03, 13:17

25) 11:30, 11:48, 12:06, 12:24, 12:42, 13:00, 13:18

Trolleybus 7 was shifted 8 minutes to the left.

Trolleybus 10 was shifted 4 minutes to the right.

Trolleybus 25 was shifted 1 minute to the left.

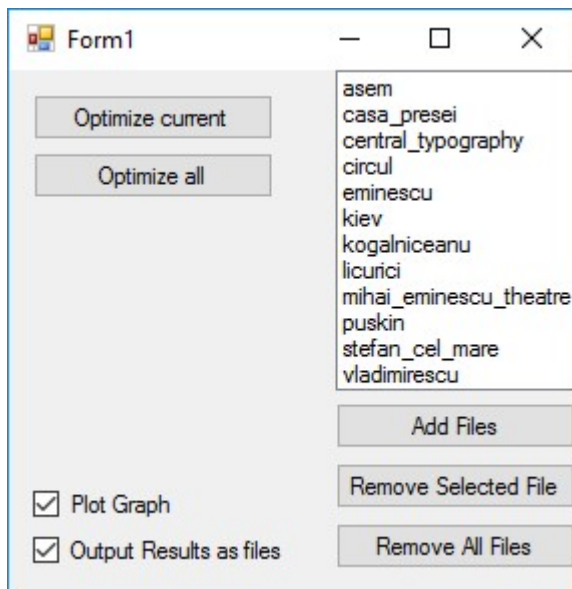
## Subproblem 2

Given the [timetables](#) of trolleybus arrival, find an algorithm to optimize the traffic among all given stations.

## Solution

Since our algorithm works with multiple stations, we can use it easily in the same way. Also the parse, plot and file output are made so that they can handle multiple files at once if we input them.

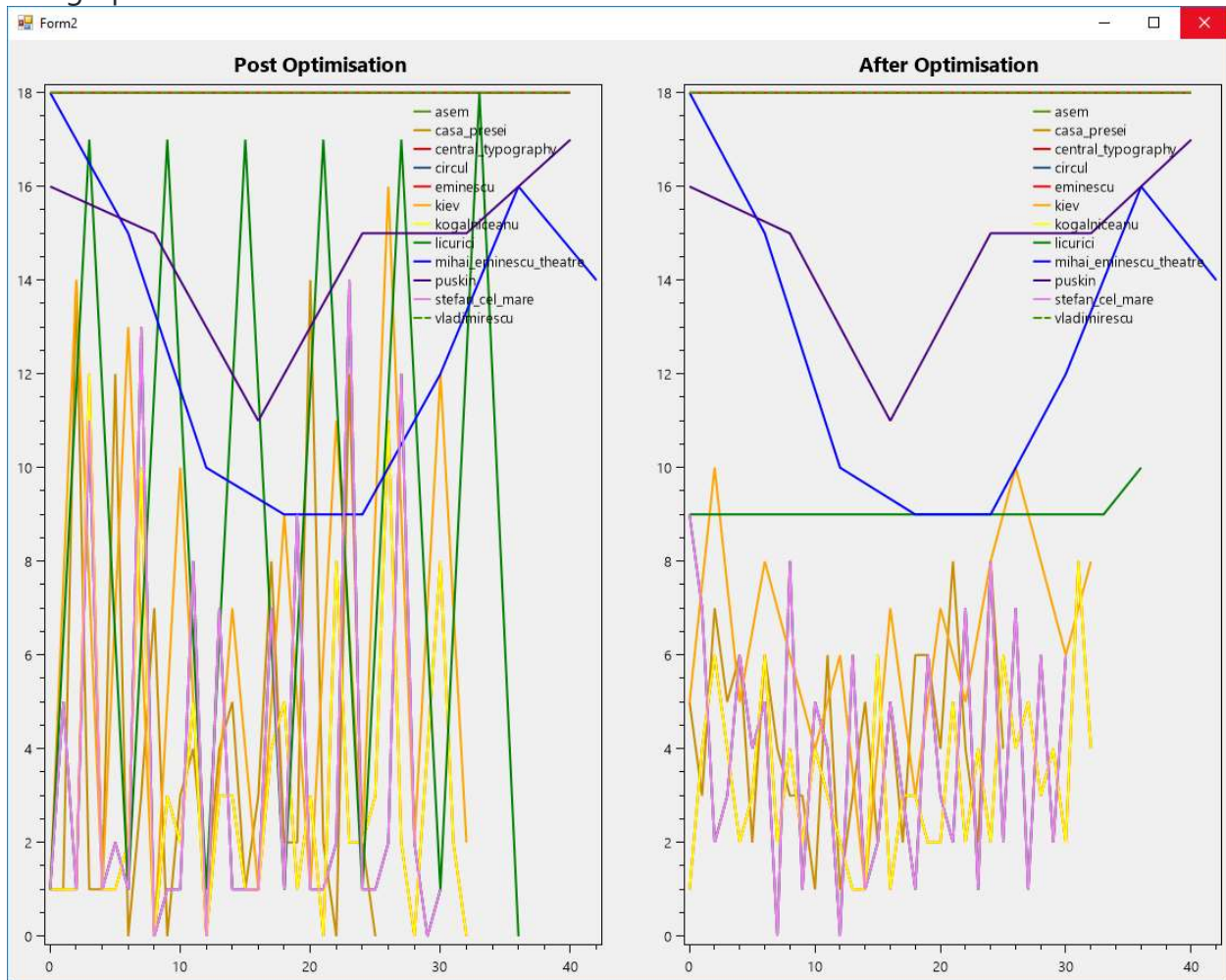
So we upload all of our stations:



The starting rate is: 1.116.147.072

The final rate is: 910.306.587

The graph:



The output is so ugly because some stations have less trolleybuses.

The output files were generated for all stations. For example, for station Stefan cel Mare timetable looks like this:

- 7) 11:23, 11:41, 11:59, 12:17, 12:35, 12:53, 13:11
- 10) 11:44, 11:59, 12:13, 12:23, 12:31, 12:41, 12:54, 13:12, 13:26
- 24) 11:39, 11:54, 12:07, 12:17, 12:24, 12:34, 12:46, 13:04, 13:18
- 25) 11:32, 11:50, 12:08, 12:26, 12:44, 13:02, 13:20

It is different from previous one because now we took into considerations the other stations too, and the order in what they are processed can also influence the output.



## Conclusion

In this laboratory work we learned that I am stupid and I can't understand how to solve this kind of problems. Also we learned that this algorithm can compute only a local optimization since the output will vary if we change the input order. If we would apply this algorithm at city level, we won't get a better result I suppose because we need to add more conditions like traffic, times when a lot of people use trolleybuses (mornings and after work in evening).