UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Metaheuristic for personalized trip planning

Author:
Yurii KAZAN

Supervisor:
Vasyl MYLKO

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2021

Declaration of Authorship

I, Yurii KAZAN, declare that this thesis titled, "Metaheuristic for personalized trip planning" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Metaheuristic for personalized trip planning

by Yurii KAZAN

Abstract

Lviv's second source of revenue is entertainment. The amount of different points of interest is over a few thousand. Every day Lviv is visited by a large number of tourists. Usually, they visit only the most famous places, that were suggested to them by friends. Someone visits Lviv to try traditional dishes, someone to see interesting places, someone who has a lot of money, and someone who has only a few. Also, one person could visit Lviv for five days and the other only for one. And each of them needs his own plan according to many parameters. Many services simplify planning by giving enough required information in one place, but only a few build path variants for you. What if there would be a service that will create trip plans according to your parameters: such as budget and amount of days to spend in Lviv could it make Lviv as a tourist place more effective and more widely specialized? This is main theme of this work.
Code can be found here TourGuide Github

Acknowledgements

I would like to thank Vasyl Mylko for the lots of help, that he gave me during this work and Ukrainian Catholic University for big support during the last four years. I am very grateful to my father because he taught me a lot and Java (a language which I used almost everywhere even in this project) was basically taught to me by him. Also, I would like to thank my wife, if she wouldn't support me the whole time I would gave up already at the beginning of this work.

Contents

List of Figures

List of Tables

To my wife

Chapter 1

Introduction

Today recommendation systems are everywhere. They help people create a diet, develop better writing skills, suggest courses, and much more. All of them have something in common - they use some personal data of users to create a suggestion that will fit the most. Also, the core value for the modern person is experience amount - they would like to discover as much as possible. Traveling is one of the experience sources, but trip planning is a pretty long process, which requires a lot of effort. Many services such as Curiosio try to create personal plans for the specific traveler to minimize planning time. The less time you spend on planning - the more experience you get in your life. So such services are mandatory in the world, where experience is the almost core value. They could be handy also for a person who lives in Lviv but would like to experience it more. The final important thing to mention that you could easily create extensions to this service for any additional needs to generate a lot of market value.

The main idea of my work is to create a trip plan suggestion service that will give the user a certain number of "good enough" variants to choose.

1.1 Problem statement

For a better understanding of the problem, imagine such a situation. You are traveling to some specific city with some amount of time and money. Also, you already have a place to spend nights. You don't want to spend money on some tours, or they are just not available. So what would you do?

For such cases, we can clarify the inputs and outputs of the future designed system. As input parameters, we have such values:

- Money
- Time
- Night spend location

And as output, we need some trip plan which will be the most exciting.

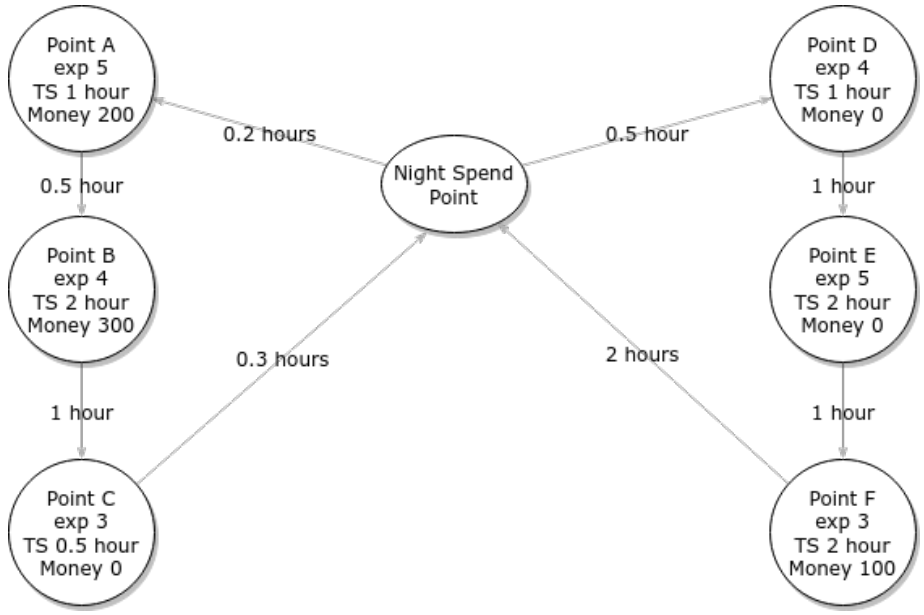

FIGURE 1.1: Simple example

Let's bring some simple example in order to understand things even better. Now let's consider two different cases:

- Traveler has over 500 money points and less than 6 hours - as the result he couldn't take the path DEF but he could take ABC.
- Traveler has more than 10 hours and less than 150 money points. In this example, he could take the path DEF but not ABC.
- Traveler has less than 6 hours and less than 100 money points. He cannot take any of this tours.

This example was shown in order to describe there is a personal part in this task. If we take a look at path ABC or DEF we can see that amount of experience is the same, and choice is different only because of special conditions.

The main conditions in this work will be money and time because all others are much harder to take into account and even if they will be counted the result won't change dramatically. So after this example, we can finally understand what the problem is and start to build some solutions.

1.2 Thesis structure and goals

In this work, the next goals are going to be reached:

- Describing existing metaheuristic algorithms
- Exploring the main milestones of this problem.
- Applying one of the described algorithms to analyze the results.

This work will contain five chapters. The second one will be about main metaheuristic algorithm groups with a brief explanation. The third one will contain information about implementation and design issues. The fourth one will show results and analysis. The last one will contains conclusions.

Chapter 2

Literature review

2.1 Combinatorial Optimization Problems

Before we get to the math model of my solution, we need to discuss some theoretical information. First of all, we will discuss what is "Combinatorial Optimization Problem (COP)." "Combinatorial optimization is a topic that consists of finding an optimal object from a finite set of objects."[from Wikipedia]. In other words, we have some number of objects where each has a definite price and value, and we need to create a subset according to some conditions. As you can see, trip planning is a combinatorial optimization problem: we have some set of possible paths, and we need to select the most optimal for a certain case. There are many different algorithms for solving such kinds of problems. All of them can be divided into categories. On the next page, you could see that classification. We will take a look only at metaheuristic methods as they are the main topic of my work. This chart was taken from [Hieu, 2011]

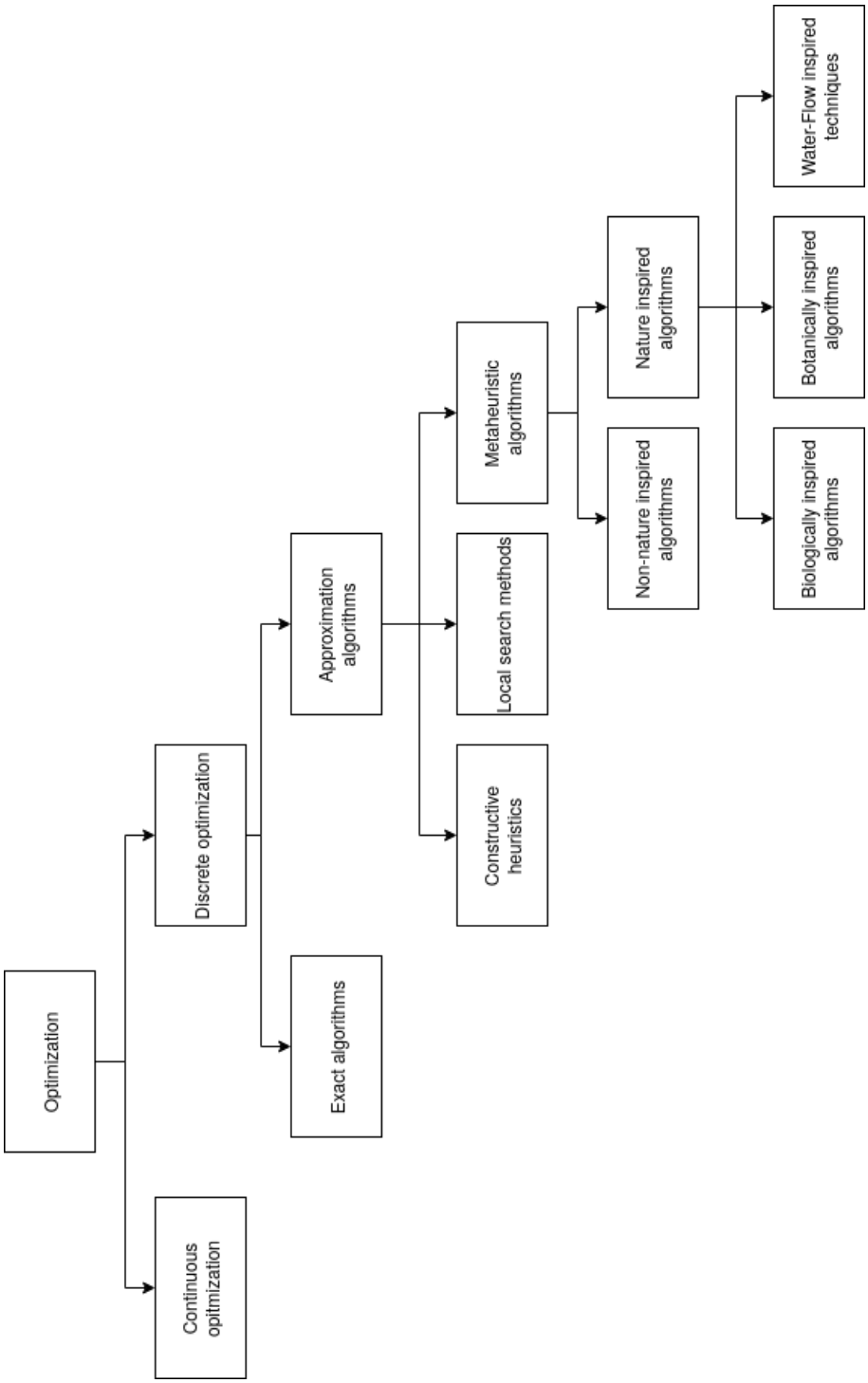

FIGURE 2.1: Classification of COP algorithms

2.2 Biologically Inspired Algorithms

"The first metaheuristic algorithms class is biologically inspired. They can also be divided into evolutionary, stigmergic optimization, and swarm-based optimization."[Hieu, 2011]. Each of these groups has something in common with others - they are based on some animals' behavior. **All of the figures shown below were taken from** [Hieu, 2011]

2.2.1 Evolutionary algorithms

Evolutionary algorithms are the algorithms that are based on the evolution of the species; in general, they are based on the main evolutionary theory of Charles Darwin. The main idea is pretty simple, we have some starting state of "creatures" then we simulate some kind of task for them and measure how good or bad these results are. The best "creatures" pass a little bit changed parameters to the next generation, and results of the next generation became better. In short, this process can be shown on the next diagram.

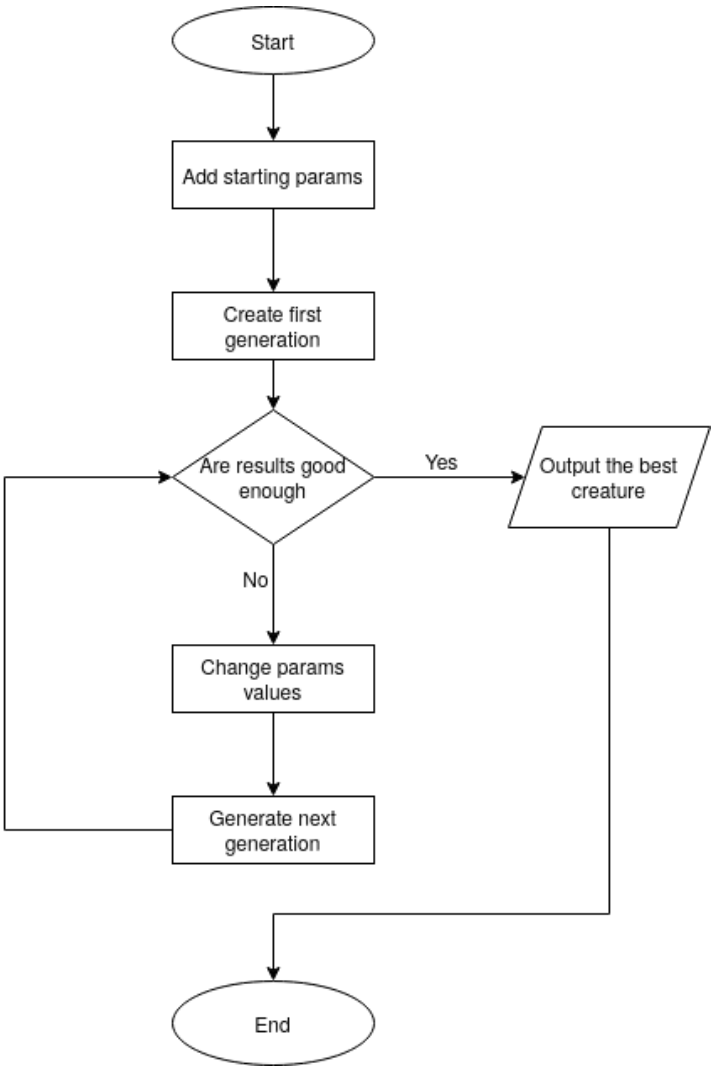

FIGURE 2.2: Generic algorithm

2.2.2 Stigmergic optimization algorithms

This group's main idea is to simulate animal interaction in order to perform some action. My selected algorithm is from this category - the ant colony algorithm. The main idea of which is to simulate the way ants build their paths from home to food. In nature, everything happens the next way. During the first attempts, ants are moving on different routes almost randomly but after each ant, some amount of pheromones stays on this route. As the result, the more ants come through a certain path - the more pheromones it has, and the fastest will have the biggest amount. After some number of iterations, all ants will move only on the route with the biggest amount of pheromones. Actually, pheromones are the main tool of communication between ants, so this example can fully describe the idea of stigmergic optimization algorithms. The next diagram will show how does this algorithm work.

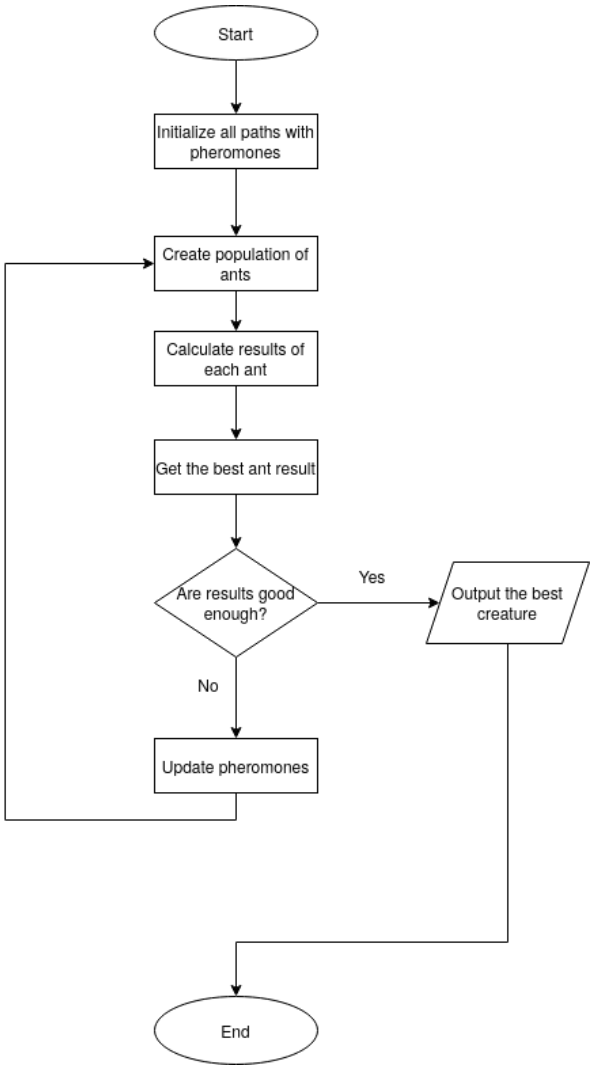

FIGURE 2.3: Ant colony algorithm

2.2.3 Swarm-based optimization algorithm

This group looks very similar to the previous one, and they are close to each other. These algorithms come from the social behavior of swarm-based animals or insects, especially those in which the property of historical information exchange among individuals is magnified. We will take a bat algorithm as an example to this group. The main idea is to use the echolocation of bats in order to solve optimization problems. We can set starting speed of the bat, its location, and loudness. After it finds prey, it will change these parameters. Search is some kind of "random walk". Selection of the best continues until certain stop criteria are met. This essentially uses a frequency-tuning technique to control the dynamic behavior of a swarm of bats, and the balance between exploration and exploitation can be controlled by tuning algorithm-dependent parameters in the bat algorithm. A detailed introduction of this and all the above algorithms is given by Yang [Yang, 2010]

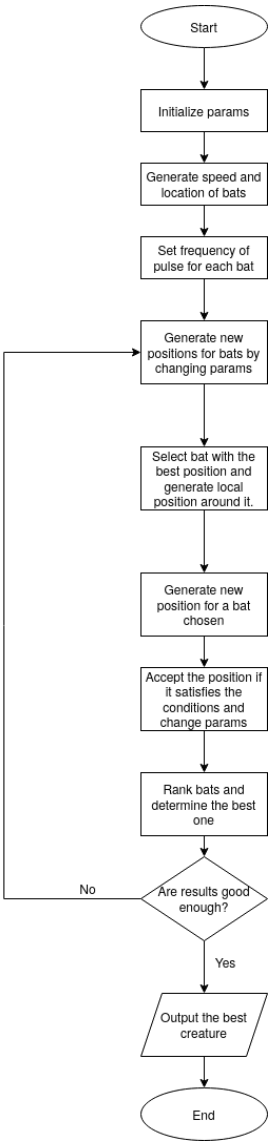

FIGURE 2.4: Bat algorithm

2.3 Botanically inspired algorithms

2.3.1 Slime Mould Algorithm

"The Slime Mould Algorithm (SMA) is one of the recent nature-inspired algorithms. It refers to the mathematical model ofsimulating the propagation wave of slime mould when forming the optimal path for connecting foods.This model adaptively simulates the process of producing negative and positive feedback during thepropagation wave. This algorithm is incorporated into different optimisation problems, includingthe engineering ones. The main two stages in the SMA algorithm are called approaching food andwarp food." [Salah L. Zubaidi and Al-Khaddar, 2020]

This approach is pretty new and already is widely used. On the first stage mould is "searching" for food. It navigate according to its odour in the air. This behaviour can be described as math system. After food was found the behaviour of the slime in conducting contraction of its venous structure. All additional information can be found in Salah L. Zubaidi and Al-Khaddar, 2020

2.4 Water-flow like algorithm

This algorithm basis is rain. The main idea is the idea of drops falling and moving afterward from the highest point to the lowest one. "Based on the property of water flow always moving to lower regions, the authors simulated with the pouring of water onto the terrain surface. In the field of image processing, the objective often considered is to extract characters from backgrounds. Thus, a threshold process is proposed to extract the valleys by the amount of filled water." [Hieu, 2011] There are many different extensions to this algorithm. The main problem with it is, that there is no "stop point" for iterations. Usage of this algorithm is also pretty hard. In order to use it, you need to set gravity, rain power, and many other physics constants (which can be different from the usual one). And after that simulate such eco-system. Extensions were made in order to solve the problem with an unknown "stop point" but they made the algorithm even harder. This is a very powerful approach, but it won't be used in this work.

Chapter 3

Design and implementation issues

3.1 Design

As was written above, the selected algorithm is the ant colony algorithm. But to complete the task, we need not only the algorithm but also the model of measurement, data model, and some other features which we will need during implementation. In this chapter, I will show the main system design issues and also describe some problems which I found during the process.

3.1.1 Data model

To begin with, we need to understand what information is needed to complete this task. As it was said, we have three main condition

- Money
- Time
- Night spend location

So for time measurement, we need to know:

- How much time it takes to move between certain points.
- How much time it takes to "experience" that point.
- How much time it takes to move from point A to the night spend location.

Almost the same questions we need for money:

- How much money it takes to move between certain points.
- How much money it takes to "experience" that point.
- How much money it takes to move from point A to the night spend location.

Next and probably the most important question: "What is the experience of the point?" The answer is pretty simple: the higher grade of the point is, the more experience it will give - so our experience measure is nothing else but point grade. This measure isn't the best possible (this will be discussed in "Problem topic"), but it is "good enough", and that's everything we need.

And now, I will answer all the previous questions. First of all, about moving from point to point. To know the time, I need to build a path according to points location and type of transport (car, bus, foot, etc). In this work, we will use only two types of transport on foot and by taxi. (simply because other kinds of transport are hard to include) The same comes for money. The next point is how to know the amount of money and time spend in a certain place? For the money it is much easier, almost any tourist attraction has a price listed somewhere, but the time spend is hard to predict. Each person can spend a different time in the same attraction. For this reason, I will use some general values. For example, time spent in parks will be 2 hours and near some monument only 30 minutes. And the last question is answered pretty simple, I will add the night spent point to the rest and needed values will be counted.

So to sum up, I need such information about each point

- Name -> to show on the screen.
- Location -> longitude, and latitude for analysis of paths.
- Type -> it will be needed to set constant values and also for some further optimization.
- Grade -> main value of each point.
- Time and money spent on this point.
- Time of start working and time of stop working -> in order to not include unavailable points.

3.1.2 Math model

First I will show you the list of function which will be used in this model.

- *exist(i)* – a function which shows if i-th point is in the trip.
- *PathPoint(i,j)* – returns point with i-th index on j-th day.
- *PSTime(i)* – time when i-th point start working
- *PStTime(i)* – time when i-th point stop working
- *aTime(i)* – arive time to i-th point.
- *lTime(i)* – leave time from i-th point.
- *SpentTime(i)*– time spent on i-th point.
- *SpentMoney(i)* – money spend on i-th point.
- *TotalBudget* – total budget of the trip
- *grade(i)*– grade of point.
- *k(i)* – number of points visited on i-th day.
- *NPoint* – total number of points from database.
- *NDay* – total number of days in trip.
- *WH* – const number of hours spend on the trip per day.
- *mTS* – time spent on moving between points.
- *mBS* – money spent on moving between points.

Next, I will define the main function, which will estimate how good or bad this trip plan is. As it was said earlier main parameter here is grade.

$$\sum_{i=0}^{NPoint} exist(i) * grade(i)$$

It simply states that if the point is in our trip then the total grade includes its grade. So the total experience is nothing else than the sum of grades of each included point.

Finally, the most insane moment in this math model is constriction functions.

$$\sum_{i=0}^{NPoint} exist(i) * SpentTime(i) + mTS \in [0.8 * NDay * WH, NDay * WH]$$

This equality stated that time spent on experience must be in the range of 80-100 percent of available time. The same works for money.

$$\sum_{i=0}^{NPoint} exist(i) * SpentMoney(i) + mBS \in [0.8 * TotalBudget, TotalBudget]$$

These are the main constriction functions. According to them, each path will get some final grade. If value, got from any of this sum, lands in the range 80-100, then it got 1 point. If it lands in the 0-80 range, then 0, and finally, if it lands over 100, this path will get -1000. Now I will show some other conditions.

$$\forall i \in [0, NDay] \rightarrow k(i) < 10$$

Each day will have at most 10 points more points tire traveler.

$$mTS < 0.5 * \sum_{i=0}^{NPoint} exist(i) * SpentTime(i)$$

$$mBS < 0.5 * \sum_{i=0}^{NPoint} exist(i) * SpentMoney(i)$$

Moving must take less than half of trip time and the same for budget.

$$\forall j \in [0, NDay] \rightarrow \sum_{i=0}^{k(j)} SpentTime(PathPoint(i, j)) < 2/3WH$$

Each day time spent on points is less than 2/3 of walking hours per day (if we add here previous equality we will get that total walk time is less than walking hours).

$$\forall i \in [0, NPoint] \, if \, exist(i) \rightarrow aTime(i) > PSTime(i)$$

$$\forall i \in [0, NPoint] \, if \, exist(i) \rightarrow lTime(i) < PStTime(i)$$

To experience point we must enter after it is open and before it is closed.

3.2 Design issues

3.2.1 Data gathering

Data gathering was probably the hardest process (which is not related to the actual implementation). The actual problem is that there is no such data source which will fill all field that was described earlier. During this work, I looked at eight different sources and each of them has some kind of problems. Lack of grades, unable to parse (because the website was generated, and, as the result even the same components had different class names) or something else (terms of use) were the main ones. Here is the list of the best from used websites:

- https://www.tripadvisor.com/ –> generated and hard to parse
- https://lviv.travel/ –> lack of information
- https://virtual.ua/ua/ –> no grades.

3.2.2 Routino adapter

The next major issue was about "How would we build the path?" And the answer is OpenStreetMaps. The next part was to find a dataset with Ukrainian routes and that wasn't so hard. But how to use this dataset was the main question. I was looking for some free and easy service that works with OpenStreetMaps. And the one I choose was Routino-router. It is written on C and can be used on Ubuntu from terminal. But main code is on Java. So the idea was to use next construction:

```
ProcessBuilder processBuilder = new ProcessBuilder();
String literal_type;
switch (type){
    case FOOT:
        literal_type = "foot";
        break;
    case BUS:
        literal_type = "psv";
        break;
    case CAR:
        literal_type = "motorcar";
        break;
    default:
       throw new IllegalStateException("Unexpected␣value:␣" + type);
    }
    String command =
    "routino-router␣--dir=src/main/resources/map_data␣" +
    "--lat1=" + p1.getPoint().getLatitude() +
    "␣--lon1=" + p1.getPoint().getLongitude() +
    "␣--lat2=" + p2.getPoint().getLatitude() +
    "␣--lon2=" + p2.getPoint().getLongitude() +
    "␣--transport=" +literal_type + "\n";
    processBuilder.command("bash", "-c", command);
    try {
        Process process = processBuilder.start();
```

3.3 Implementation

In this chapter I will tell you the main components of the designed system. One of them was already described above.

3.3.1 Architecture

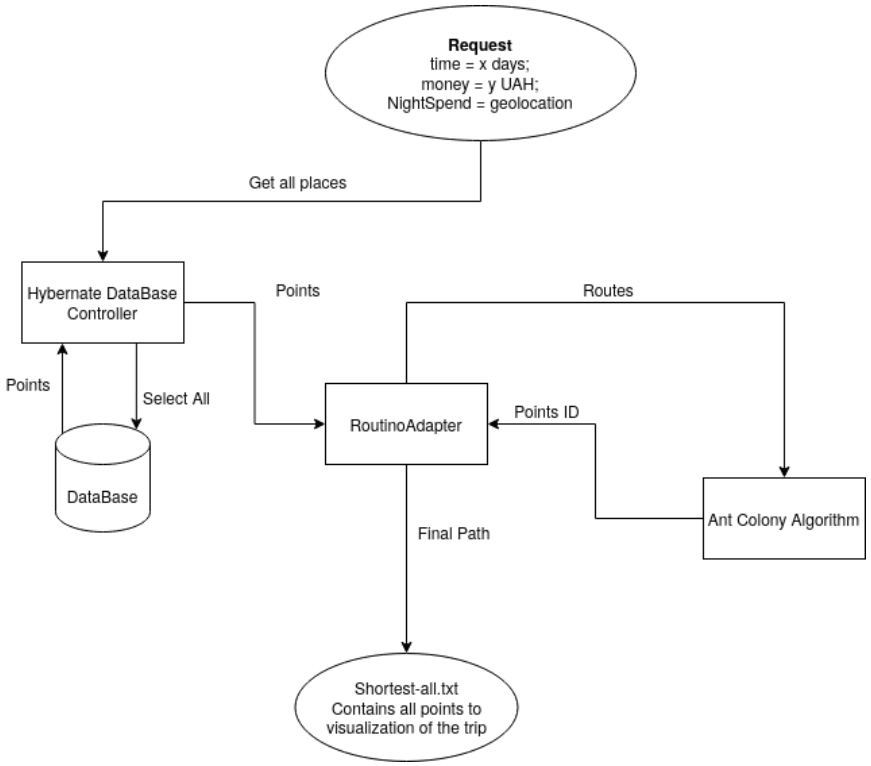

FIGURE 3.1: Project architecture

For better understanding, I will include in this chapter a corresponding image, which includes the main flow and main components of my project. As it was stated above, one of the components is the MySQL database and as the result HyberNate DataBase Connector. Also, the visualization part is not included because it is not a part of the main flow.
First of all, request - it consists of a number of days, the total amount of money and also latitude and longitude of the night spend place, the importance of each of them was already described above. The next step is to get all available places from the database. Here Data Access Object or DAO is used. Because not each column is represented in corresponding class value, we need to have some middle stage to make all systems work together.
Now we have data, but we need a connectivity matrix, so we send all this information to Routino Router in order to build paths between each pair of points.
Everything is finally prepared to be used in Ant Colony Algorithm. And this algorithm returns some number of paths (by default best 5). But it is not done yet. In order to make some visualization we need to use Routino Router again to produce the final path.
After all of these manipulations, we get a response that consists of big amount of coordinates, which can be connected on the map, and the path is done.

3.3.2 Storage

As a storage I selected MySQL database. For this project I needed only one table which will contain all information about places. So the structure is next one:

Column	Type
id	int(11)
name	varchar(255)
price	int(11)
timeStart	double
timeEnd	double
timeSpend	int(11)
stringLocation	varchar(255)
grade	double
latitude	double
longitude	double
type	int(11)

FIGURE 3.2: Database table

Chapter 4

Experiments

In this chapter, I will show few different outputs for different parameter sets. I would like to add that these pictures were made after all other code and were simplified (path is demonstrated as connected points, not as route). I made eight different cases: Traveler has one day and 0 UAH and stays in the city center. Traveler has one day and 1000 UAH and stays in the city center. Traveler has two days and 0 UAH and stays in the city center. Traveler has two days and 1000 UAH and stays in the city center. And four more with the same conditions but night spend is on Sykhiv. Also, I will show you how total experience is changed and money/time spend.

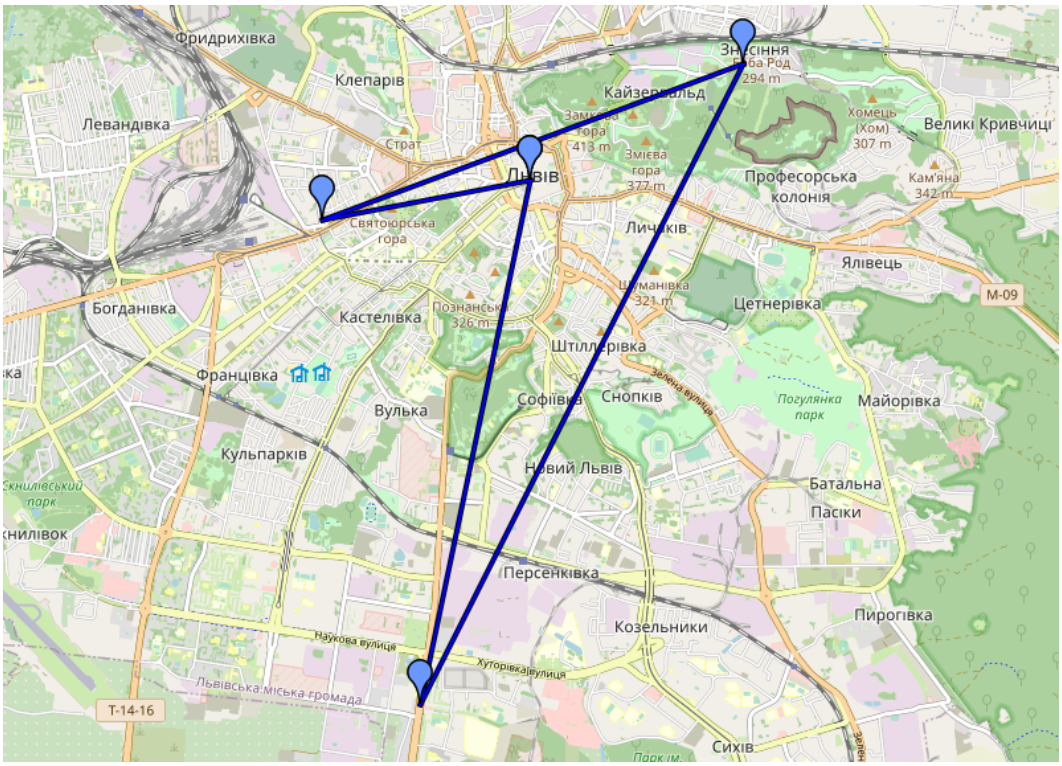

FIGURE 4.1: Experiment 1 (1 day 0 UAH City Center)

This experiment was kind of "zero point" of all set. It was made in order to see how different factors could change the result.

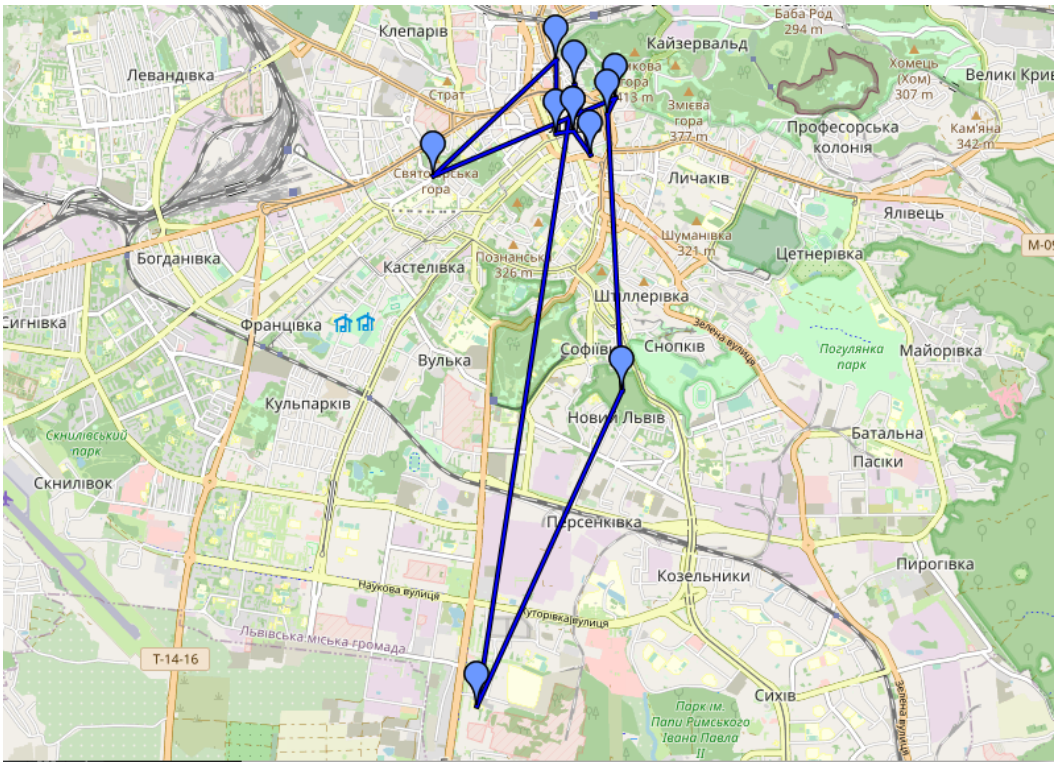

FIGURE 4.2: Experiment 2 (1 day 1000 UAH City Center)

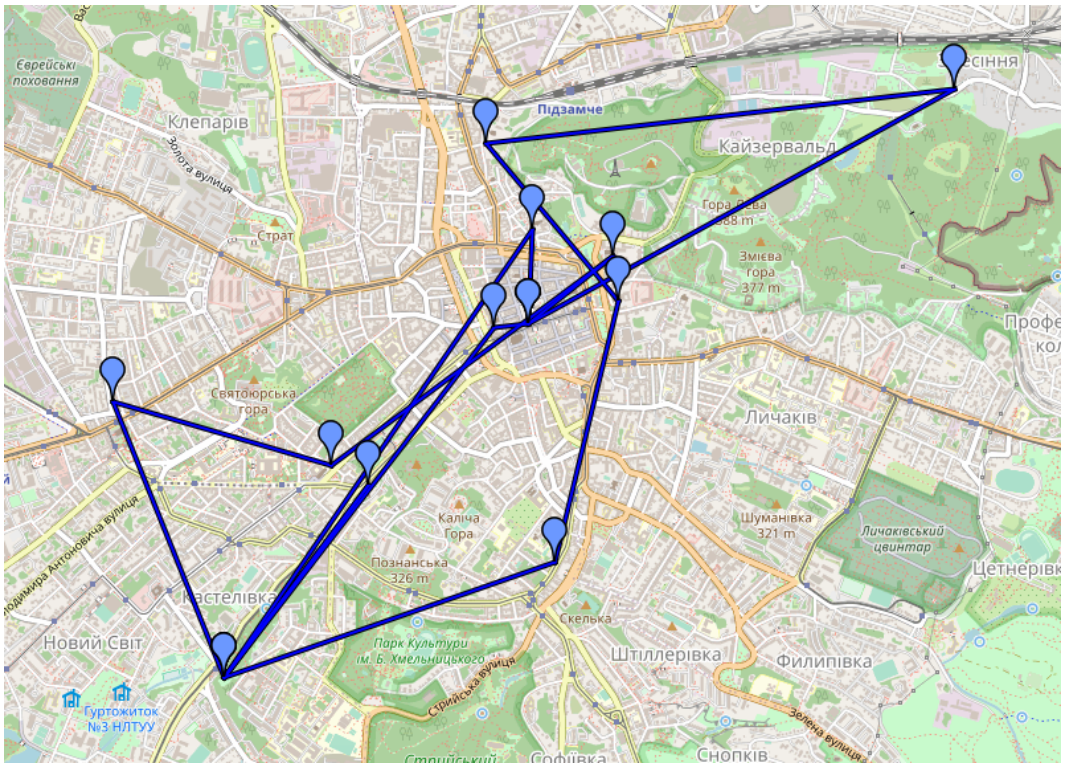

FIGURE 4.3: Experiment 3 (2 days 0 UAH City Center)

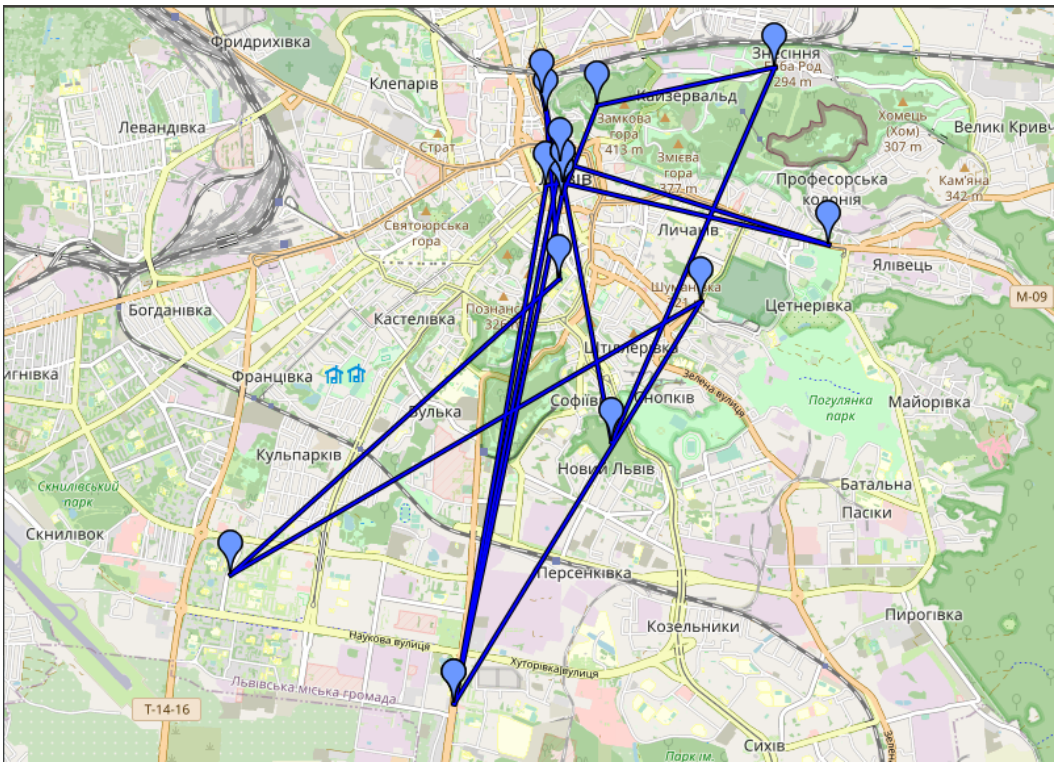

FIGURE 4.4: Experiment 4 (2 day 1000 UAH City Center)

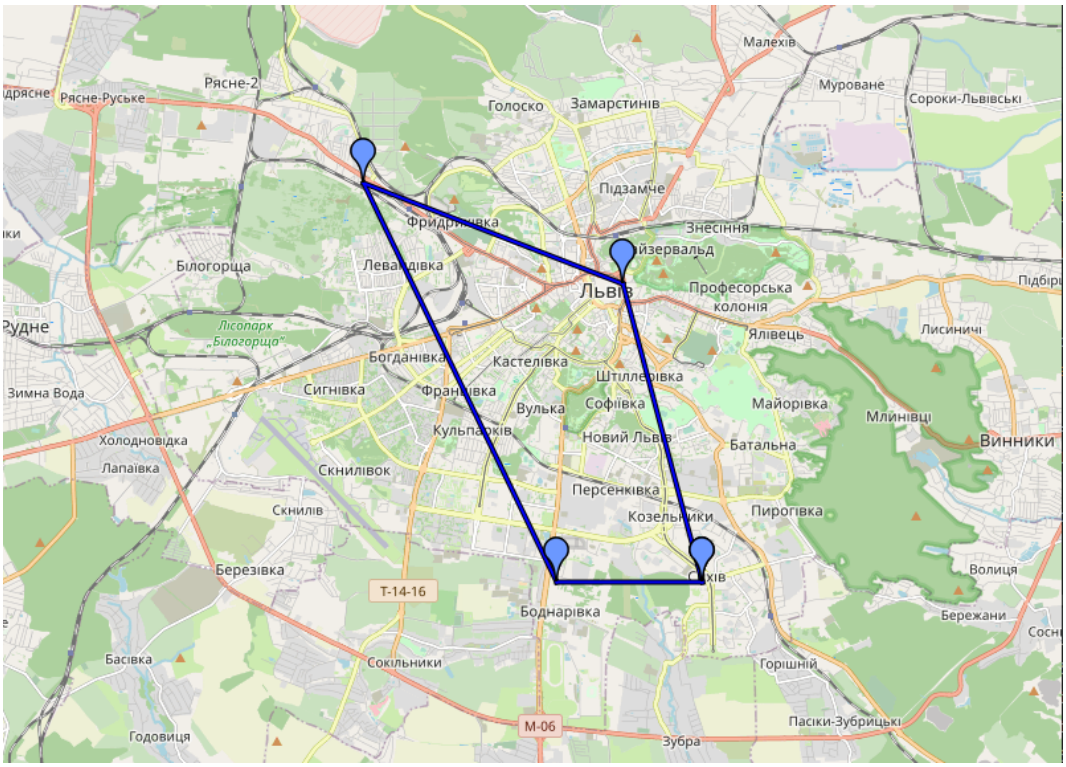

FIGURE 4.5: Experiment 5 (1 day 0 UAH Sykhiv)

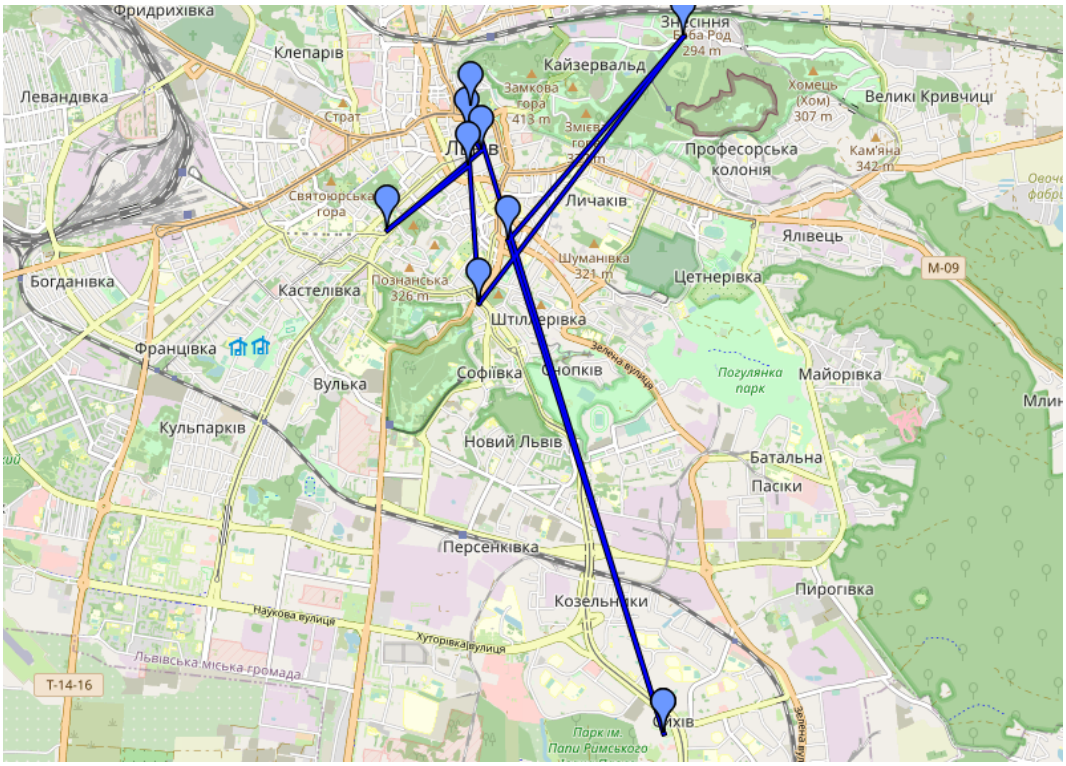

FIGURE 4.6: Experiment 6 (1 day 1000 UAH Sykhiv)

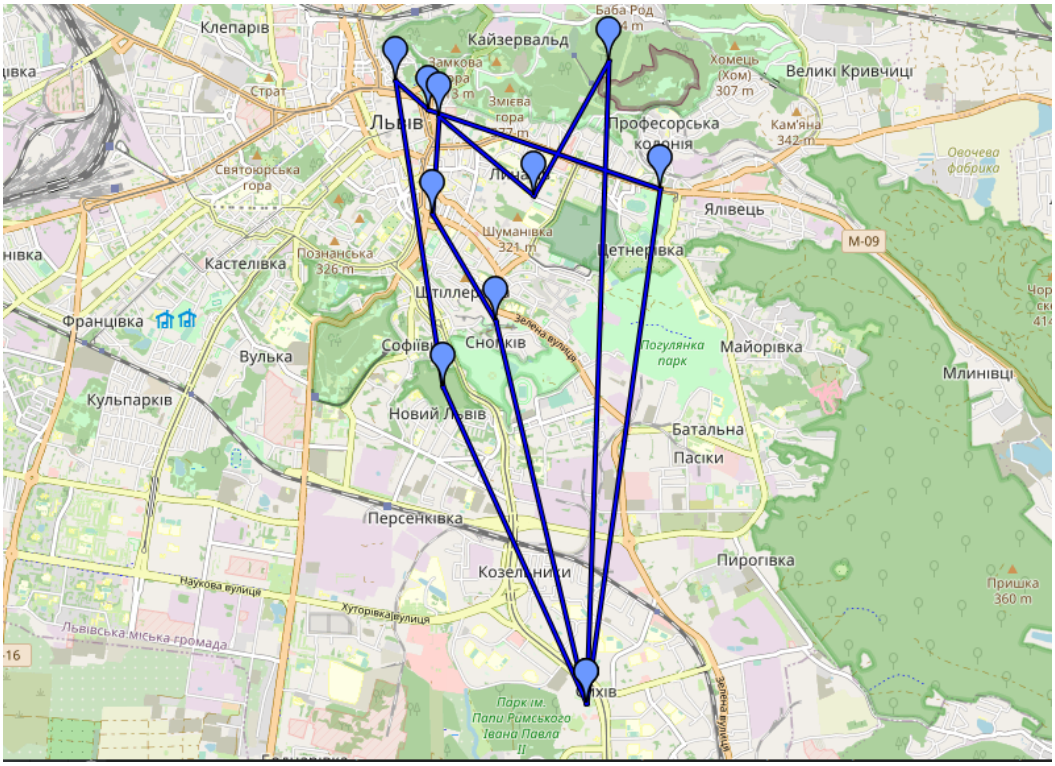

FIGURE 4.7: Experiment 7 (2 day 0 UAH Sykhiv)

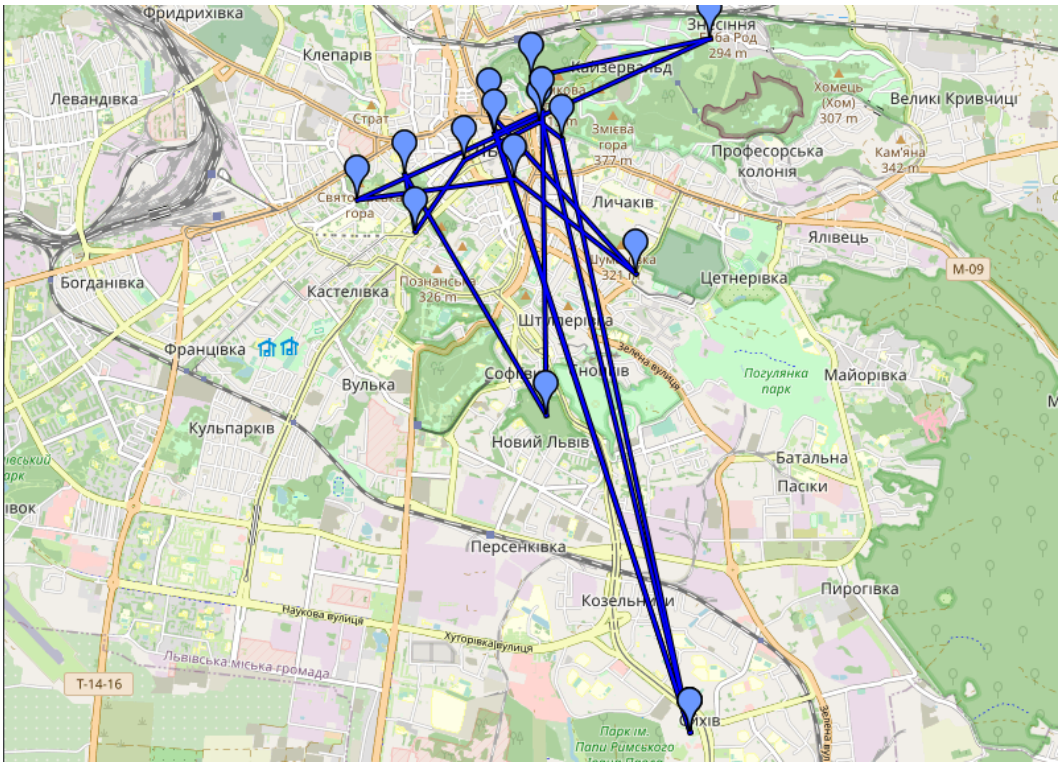

FIGURE 4.8: Experiment 8 (2 day 1000 UAH Sykhiv)

Now I would like to show you a table with some numeric data. From this table, we could make some conclusions.

#	Total experience	Total time spend	Total money spend
Experiment 1	2.8	6.7 hours	0 UAH
Experiment 2	7.9	9.7 hours	495 UAH
Experiment 3	10.9	18.3 hours	0 UAH
Experiment 4	13.1	18.6 hours	981.2 UAH
Experiment 5	2.2	8 hours	0 UAH
Experiment 6	6.7	9.8 hours	655 UAH
Experiment 7	7.7	16.1 hours	0 UAH
Experiment 8	10.6	17 hours	765.8 UAH

TABLE 4.1: Results numeric data

#	Time fit 80-100% range	Money fit 80-100% range
Experiment 1	No	Yes
Experiment 2	Yes	No
Experiment 3	Yes	Yes
Experiment 4	Yes	Yes
Experiment 5	Yes	Yes
Experiment 6	Yes	No
Experiment 7	Yes	Yes
Experiment 8	Yes	No

TABLE 4.2: Results fit best range

We can see the next thing:

- Time is the most important factor if we double the time the experience grows more than three times.
- Money also affects experience, and it makes the traveler spend more time on traveling (he could faster get to "night spend" using the car).
- Night spend point has some impact on the total experience. This impact is even bigger than my expectation. (I was thinking - that if we had enough money then the difference will be almost 0).
- Almost all routes get the best time range and one day trip missed money range (I think it is really hard to spend 1000 UAH in one day just on a trip without shopping)

Chapter 5

Conclusions

After all of these experiments, we can conclude that this system could build some sorts of optimal paths. Also, we can see that Ant Colony Algorithm efficiency is quite good even with a small number of extensions. There are many problems with paths generated by this system.

- It takes too much time to generate anything.
- It takes too many resources (on my machine nothing else can run side by side with this project).
- Results are not so accurate (they can be used only as of the first step of trip creation).

The next extensions could be applied to solve the listed problems. The first and the most efficient is to divide points by some kind of classification algorithm. This will decrease the number of unnecessary calculations. (Not every point is accessible according to the possibilities of the corresponding user).
The second important issue is the update of the dataset. This dataset was enough to create such a simple model, but the better data is - the better the result is.
Finally, we can add many other extensions - such as weather and even mood, but it will increase the complexity of the task. After these extensions, the next major one is to run everything on some kind of server. (maybe to add Java Spring and change the structure a little bit).
As we can see, this field of study has many directions to develop, many other algorithms to use, and dataset to explore.

Bibliography

Hieu, Tran Trung (2011). *A Water flow algorithm for optimization problems*. National University of Singapore.

Salah L. Zubaidi Iqbal H. Abdulkareem, Khalid S. Hashim Hussein Al-Bugharbee Hussein Mohammed Ridha Sadik Kamel Gharghan Fuod F. Al-Qaim Magomed Muradov Patryk Kot and Rafid Al-Khaddar (2020). *Hybridised Artificial Neural Network Model with Slime Mould Algorithm: A Novel Methodology for Prediction of Urban Stochastic Water Demand*.

Yang, Xin-She (2010). *Nature-Inspired Metaheuristic Algorithms Second Edition*. Luniver Press.